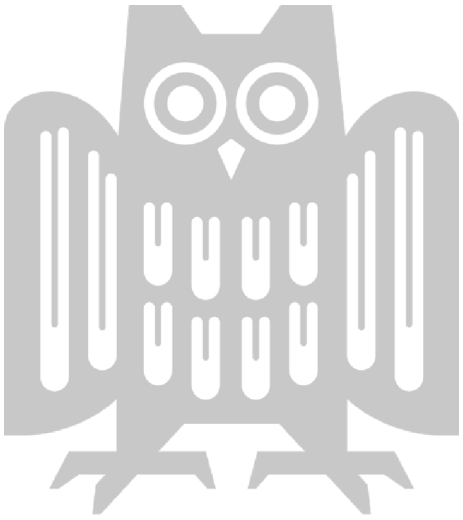
Understanding and Improving Robustness and Uncertainty Estimation in Deep Learning

David Stutz

Dissertation zur Erlangung des Grades des
Doktors der Ingenieurwissenschaften (Dr.-Ing.)
der Fakultät für Mathematik und Informatik
der Universität des Saarlandes

Saarbrücken, 2022.



Date of Colloquium:	July 22, 2022
Dean of the Faculty:	Prof. Dr. Jürgen Steimle
Chair of the Committee:	Prof. Dr. Eddy Ilg
Reviewers:	Prof. Dr. Bernt Schiele
	Prof. Dr. Matthias Hein
	Dr. M. Pawan Kumar
	Prof. Dr. Mario Fritz
Academic Assistant:	Dr. Jan Eric Lenssen

Dedicated to Hilmar, Doris, Reiner, Cedric, Vanessa

Abstract

Deep learning is becoming increasingly relevant for many *high-stakes* applications such as autonomous driving or medical diagnosis where wrong decisions can have massive impact on human lives. Unfortunately, deep neural networks are typically assessed solely based on generalization, e.g., *accuracy* on a fixed test set. However, this is clearly insufficient for safe deployment as potential malicious actors and distribution shifts or the effects of quantization and unreliable hardware are disregarded. Thus, recent work additionally evaluates performance on potentially manipulated or corrupted inputs as well as after quantization and deployment on specialized hardware. In such settings, it is also important to obtain reasonable estimates of the model's confidence alongside its predictions. This thesis studies *robustness* and *uncertainty estimation* in deep learning along three main directions: First, we consider so-called *adversarial examples*, slightly perturbed inputs causing severe drops in accuracy. Second, we study *weight* perturbations, focusing particularly on *bit errors* in quantized weights. This is relevant for deploying models on special-purpose hardware for efficient inference, so-called *accelerators*. Finally, we address uncertainty estimation to improve robustness and provide meaningful statistical performance guarantees for safe deployment.

In detail, we study the existence of adversarial examples with respect to the underlying data manifold. In this context, we also investigate *adversarial training* which improves robustness by augmenting training with adversarial examples at the cost of reduced accuracy. We show that regular adversarial examples leave the data manifold in an almost orthogonal direction. While we find no inherent trade-off between robustness and accuracy, this contributes to a higher sample complexity as well as severe overfitting of adversarial training. Using a novel measure of flatness in the *robust* loss landscape with respect to weight changes, we also show that robust overfitting is caused by converging to particularly sharp minima. In fact, we find a clear correlation between flatness and good robust generalization.

Further, we study random and adversarial *bit errors* in quantized weights. In accelerators, random bit errors occur in the memory when reducing voltage with the goal of improving energy-efficiency. Here, we consider a *robust quantization* scheme, use *weight clipping* as regularization and perform *random bit error training* to improve bit error robustness, allowing considerable energy savings without requiring hardware changes. In contrast, adversarial bit errors are maliciously introduced through hardware- or software-based attacks on the memory, with severe consequences on performance. We propose a novel *adversarial bit error attack* to study this threat and use *adversarial bit error training* to improve robustness and thereby also the accelerator's security.

Finally, we view robustness in the context of uncertainty estimation. By encouraging low-confidence predictions on adversarial examples, our *confidence-calibrated adversarial training* successfully rejects adversarial, corrupted as well as out-of-distribution examples at test time. Thereby, we are also able to improve the robustness-accuracy trade-off compared to regular adversarial training. However, even robust models do not provide any guarantee for safe deployment. To address this problem, *conformal prediction* allows the model to predict *confidence sets* with user-specified guarantee of including the true label. Unfortunately, as conformal prediction is usually applied after training, the model is trained without taking this calibration step into account. To address this limitation, we propose *conformal training* which allows training conformal predictors end-to-end with the underlying model. This not only improves

the obtained uncertainty estimates but also enables optimizing application-specific objectives without losing the provided guarantee.

Besides our work on robustness or uncertainty, we also address the problem of 3D shape completion of partially observed point clouds. Specifically, we consider an autonomous driving or robotics setting where vehicles are commonly equipped with LiDAR or depth sensors and obtaining a complete 3D representation of the environment is crucial. However, ground truth shapes that are essential for applying deep learning techniques are extremely difficult to obtain. Thus, we propose a *weakly-supervised* approach that can be trained on the incomplete point clouds while offering efficient inference.

In summary, this thesis contributes to our understanding of robustness against both input and weight perturbations. To this end, we also develop methods to improve robustness alongside uncertainty estimation for safe deployment of deep learning methods in high-stakes applications. In the particular context of autonomous driving, we also address 3D shape completion of sparse point clouds.

Zusammenfassung

Moderne Methoden des maschinellen Lernens, vor allem basierend auf tiefen neuronalen Netzen, werden zunehmend in sicherheitskritischen Anwendungen wie dem autonomen Fahren oder der medizinischen Diagnose eingesetzt, bei denen falsche Entscheidungen einen signifikanten Einfluss auf das Leben von Menschen haben können. Leider werden neuronale Netze typischerweise nur im Hinblick auf ihre Generalisierungsfähigkeit bewertet, zum Beispiel anhand ihrer Genauigkeit auf einem festen Datensatz. Insbesondere im Hinblick auf potenziell bösartiger Nutzer, eine veränderte Verteilung der Eingaben, sowie Quantisierung und Operation auf unzuverlässiger Hardware ist dies unzureichend, um den sicheren Einsatz zu gewährleisten. Daher wird in aktuellen Arbeiten zusätzlich die Genauigkeit dieser Modelle auf potenziell manipulierten oder verrauschten Eingaben sowie nach entsprechender Quantisierung für anwendungsspezifische Hardware in Betracht gezogen. In solchen Szenarien ist es außerdem wichtig, zuverlässige Schätzungen der Unsicherheit dieser Modelle zu erhalten. Die folgende Arbeit befasst sich mit drei Aspekten der *Robustheit* sowie der *Unsicherheitsschätzung* von tiefen neuronalen Netzen: Zunächst befassen wir uns mit sogenannten *Adversarial Examples*, welche die Genauigkeit von Modellen durch kleine Änderungen der Eingaben erheblich reduzieren können. Daraufhin schauen wir uns Störungen der Gewichte von neuronalen Netzen an, insbesondere bezogen auf Bitfehler in quantisierten Gewichten. Dies ist relevant für den Einsatz dieser Modelle auf spezialisierter Hardware, sogenannten *Beschleunigern*. Letztendlich nutzen wir Unsicherheitsschätzungen, um die Robustheit solcher Modelle zu verbessern und statistische Garantien für deren sicheren Einsatz zu erhalten.

Im Einzelnen studieren wir die Existenz von Adversarial Examples im Bezug auf die zugrunde liegende Mannigfaltigkeit der Daten. In diesem Kontext untersuchen wir zudem das Lernen auf Adversarial Examples, sogenanntes *Adversarial Training*, was in der Regel zu Genauigkeitsverlust führt. Wir zeigen, dass Adversarial Examples die Mannigfaltigkeit in meist orthogonaler Richtung verlassen. Während wir keinen direkten Zielkonflikt zwischen Genauigkeit und Robustheit gegen Adversarial Examples finden, trägt dies zu einer höheren Beispielkomplexität und Überanpassung von Adversarial Training bei. Mithilfe einer neuen Metrik für die Flachheit der Fehlerfunktion bezüglich Störungen in den Gewichten des Modells zeigen wir, dass Überanpassung durch ein zu scharfes Minimum in der Fehlerfunktion herbeigeführt wird. Tatsächlich finden wir eine klare Korrelation zwischen der gemessenen Flachheit und der Robustheit.

Ferner untersuchen wir zufällige sowie zielgerichtete Bitfehler in quantisierten Gewichten von neuronalen Netzen. In Beschleunigern treten Bitfehler zufällig im Speicher auf, wenn die Spannung reduziert wird, um Energie zu sparen. Wir benutzen eine robuste Quantisierungsmethode, beschränken die Größe der Gewichte und injizieren Bitfehler während des Trainings, um die Robustheit gegen derartige Bitfehler zu verbessern und somit die Energieeffizienz ohne Hardwareänderungen zu erhöhen. Im Gegensatz dazu können Bitfehler auch zielgerichtet durch einen Angreifer provoziert werden. Derartige Bitfehler können die Genauigkeit empfindlich reduzieren. Wir entwickeln eine neue Methode, um besonders bösartige Bitfehler zu berechnen und diese Bedrohung besser studieren zu können. Daraufhin verwenden wir diese Attacke während des Lernens, um die Robustheit zu erhöhen und damit die Sicherheit von Beschleunigern zu verbessern.

Schließlich betrachten wir die Robustheit von neuronalen Netzen im Kontext von Un-

sicherheitsquantifizierung. Wir entwickeln ein kalibriertes Adversarial Training, indem wir neuronale Netze dazu zwingen, Adversarial Examples mit höherer Unsicherheit zu klassifizieren. Dadurch können diese sowie verrauschte Eingaben oder Beispiele außerhalb des gelernten Konzepts anhand der assoziierten Unsicherheit detektiert und abgelehnt werden. Das erlaubt es uns, die Robustheit zu erhöhen, ohne signifikante Verluste der Genauigkeit hinnehmen zu müssen. Allerdings bieten auch derart robuste Modelle keine Garantie für entsprechende Genauigkeit in der Praxis. Arbeiten zu *Conformal Prediction* begegnen diesem Problem durch die Vorhersage mehrerer Klassen pro Eingabe. Zusätzlich wird garantiert, dass die wahre Klasse mit benutzerdefinierter Wahrscheinlichkeit in dieser Menge enthalten ist. Jedoch wird Conformal Prediction nach dem Training angewandt, sodass sich das Modell während des Trainings nicht auf diese Methodik einstellen kann. Durch unser *Conformal Training* beheben wir dieses Problem, indem wir Conformal Prediction in den Lernprozess integrieren. Dies verbessert nicht nur die entsprechenden Unsicherheitsvorhersagen, sondern erlaubt es uns auch anwendungsspezifische Fehlerfunktionen während des Trainings zu optimieren, ohne die entsprechende Garantie zu verlieren.

Neben unserer Arbeit zur Robustheit und Unsicherheit neuronaler Netze befassen wir uns auch mit der Rekonstruktion von 3D-Formen von unvollständigen Punktwolken. Im Kontext von autonomen Fahrzeugen oder Robotern ist es entscheidend, eine vollständige dreidimensionale Repräsentation der Umgebung zu erhalten. In der Praxis wird dies oft durch LiDAR- oder Tiefensensoren ermöglicht. Allerdings ist es schwierig, die für neuronale Netze nötigen Grundwahrheiten, d.h., vollständigen Formen von relevanten Objekten der Umgebung, zu erhalten. Daher entwickeln wir eine Methode, die nur mit Wissen über die Objektkategorie, also mit schwacher Überwachung, die Rekonstruktion solcher Objekte lernen kann.

Zusammenfassend trägt diese Arbeit zu unserem Verständnis von Robustheit gegenüber Änderungen in den Eingaben und Gewichten bei. Außerdem schlagen wir Methoden zur Erhöhung der Robustheit und verbesserter Quantifizierung von Unsicherheit vor, die den sicheren Einsatz von neuronalen Netzen in sicherheitskritischen Anwendungen gewährleisten sollen. Im konkreten Fall von autonomen Fahrzeugen entwickeln wir zusätzlich eine Methode für die 3D-Rekonstruktion von Punktwolken mit schwacher Überwachung.

Acknowledgements

This thesis concludes a four and a half year endeavor, during which I had the pleasure to meet and work with so many amazing people.

First, I thank my main supervisor, Bernt, for giving me the freedom to work on research questions that I am passionate about and providing invaluable guidance along the way. This includes not only lessons on writing papers or rebuttals, giving talks or reviewing, but also judging relevance, novelty and significance in an incredibly fast-paced research area. In this process, I particularly valued his practical honesty in saying how things are without being discouraging. This helped to establish a high level of mutual trust, giving me a lot of independence in my everyday work. For these reasons, Bernt is an incredible mentor to have.

Second, I thank my second supervisor, Matthias. I honestly believe that our journey in robust machine learning would have been much shorter and significantly less fruitful without him. I particularly value Matthias' dedication to conduct fair and honest research and his detail-orientation in doing so. For example, I remember many heated debates regarding experimental setup and evaluation that always contributed to the quality of our research. In combination with Bernt's higher-level view, this taught me to think at several levels of abstraction when conducting and communicating research.

Third, I am grateful for the mentorship and supervision of Taylan, Dj and Arnaud during my internship at DeepMind. Taylan, in particular, set me up for a successful internship. He provided a unique view on my internship, combining academic curiosity with strategic planning and social engagement. Besides, I experienced Taylan as an incredibly kind and understanding host. Dj was never shy of long discussions that helped to decide which ideas to let go and what next steps to focus on. Combined with Arnaud's readiness to share his incredible expertise in statistics and machine learning, it was an extremely insightful internship. Thanks to their endorsement, especially Taylan's continued support, I am looking forward to the coming years at DeepMind.

Fourth, I thank my co-author Nandhini for taking the time to explain me basic hardware concepts, despite the time difference. Thanks to her, I realized that every discipline has its own vocabulary. I believe that this awareness contributed to my ability to effectively communicate research ideas to diverse audiences and perform interdisciplinary research.

Fifth, I want to express my gratitude towards my master thesis advisor and co-author Andreas for setting me up for a successful PhD.

Thanks also go to Pawan and Mario for readily agreeing to review this thesis. Furthermore, I am grateful for the support by Qualcomm. In this context, I particularly thank Babak for his valuable feedback as well as the whole team at Qualcomm AI Research in Amsterdam.

At the MPII in Saarbrücken, especially D2, I thank all my colleagues for providing a curious, but relaxed research atmosphere. I deeply enjoyed all discussions, lunch or coffee breaks, retreats and Friday seminars. Recently, I realized the uncomplicated way of everyday work at MPII to be a blessing in that any type of unnecessary friction is avoided. I particularly want to thank Moritz, Max, Tribhu, Rakshith, Eldar, Bharat, Thiemo, Sukrut, Mo, 2× Julian, Evgeny, Jan-Hendrik, 2× Anna, Joon, Jan, Vedika, Aymen, Yongqin, Apratim, Fan, Yong, Yaoyao, Singh, Nils, Andrea, Verica, Qianru, Zeynep, Gerard, Björn and Siyu as well as all other current and former colleagues. Special thanks go to Connie for keeping the group running as well as Eldar, Julian and the IST team for helping me as IT administrator.

Outside MPII, I specifically want to thank Kathrin, Marius and Maksym for interesting discussions, reading groups and lunch breaks.

At DeepMind, special thanks go to Tobias. Despite taking separate routes after our studies, I appreciate that you brought me to DeepMind. My internship would have been less successful without you taking the time to get me started with DeepMind's infrastructure and culture. I am looking forward to reviving old traditions in London. I also thank Sven, Robert and Sumedh for many insightful code reviews, Abhijit, Jim and Alan for support in applying my work in health, Stephanie for helping with bureaucracy and Liz, Sharon as well as Sahand for much-needed coffee breaks. I also loved all the insightful discussions with Pawan, Johannes, Mateusz, Alhussein and many others. Finally, thanks to Pushmeet for the opportunity to come back full-time.

I also thank my colleagues from the machine learning group at the University of Tübingen for a fun visit in early 2020 as well as interesting reading groups and discussions. I particularly want to mention Francesco, 2× Maximilian, Alex, Julian, Valentyn, Pedro and Christian.

At the MPI-IS in Tübingen, I am particularly thankful for my colleagues from the autonomous vision group. I particularly want to thank Despoina for her support and encouragement beyond my stay in Tübingen.

For valuable feedback on drafts of this thesis, I thank my colleagues Moritz, Jan, Mo, Kathrin, Valentyn, Max and Anna.

Diese Arbeit ist meinem Großvater Hilmar gewidmet. Er hat mich zu Beginn meiner Promotion aufgenommen und dazu beigetragen, dass ich mich während meiner Zeit in Saarbrücken weniger allein und gestresst gefühlt habe. Unabhängig von Abgabefristen, Erfolgen und Misserfolgen war ich bei ihm immer willkommen, um der Arbeit am Institut zu entfliehen. Dafür bin ich extrem dankbar.

I also dedicate this thesis to my parents Doris and Reiner and my brother Cedric. My parents always made sure to pull me back to earth when I was overwhelmed with deadlines and papers, feeling the need to work harder and harder. Thanks to them, I am living an extremely privileged life with plenty of opportunities and my parents always point that out at exactly the right times. Lastly, Cedric always managed to take my mind off things, nostalgically playing video games or discussing his studies and career for a change. I am extremely lucky to have my brother as my best friend.

Finally, I also dedicate this thesis to Vanessa. I am incredibly blessed for having you in my life and enjoying your support, especially throughout my PhD. I am unlikely to ever find the right words to adequately describe my gratitude. Therefore, with much anticipation, I am looking forward to all future adventures with you.

David Stutz, March 24, 2022

Contents

Introduction 1

Contents

Deep learning is increasingly used in *high-stakes* applications. For example, in autonomous driving, other traffic participants are not only detected, but their size and position are estimated to safely navigate through complex situations [JGBG20]. These components often leverage recent progress in deep learning for 3D detection and reconstruction from cameras or LiDAR sensors [GLU12]. In medical imaging or diagnosis [GvGS16, GPC+16, LKB+17, KGC+18], deep learning is used to provide or help with diagnosis and select appropriate treatments, ranging from simple treatments for minor conditions to potentially invasive ones for severe diseases such as cancer. In these cases, wrong decisions can have tremendous impact on human lives. This motivates a recent shift in evaluation, from looking purely on generalization, e.g., *accuracy* on the test set in image classification, to additionally understanding performance on potentially malicious or corrupted inputs. For example, how does the model perform on noisy inputs or when facing malicious actors that intend to mislead it? Moreover, preserving performance after deployment on specialized hardware with constraints on energy consumption, space and cost becomes increasingly important. For example, is accuracy preserved after quantization and deployment on hardware accelerators, even when facing hardware faults? These questions lead to evaluate the model's *robustness* in various settings in addition to plain accuracy. Furthermore, this motivates *estimating uncertainty* alongside the predictions themselves. For example, does the model assign high confidence to wrong predictions or previously unseen, potentially illogical inputs? Autonomous cars must not malfunction due to, e.g., unexpected stickers or graffiti on street signs and need to estimate the future trajectory of traffic participants with appropriate uncertainty [BFS18]. Similarly, over-confidence is already a serious problem in medical diagnosis [BG08] and robustness regarding population shift or imaging noise is necessary. Overall, studying robustness and uncertainty in unforeseen and adversarial settings before deployment is of vital importance for users and practitioners to develop trust into deep learning models.

This thesis focuses on robustness of deep neural networks against changes in their inputs as well as their parameters and providing appropriate estimates of uncertainty. First, we focus on so-called *adversarial examples* [BCM+13, SZS+14], imperceptibly perturbed inputs that cause misclassification. However, we also consider distribution shifts, including random *corruptions* (e.g., noise, blur, etc.) [HD19] as well as *out-of-distribution examples* that are entirely irrelevant to the problem at hand. Robustness against these perturbations, usually causing devastating accuracy drops, is a crucial prerequisite for *trustworthy* deep learning [Var19, HKR+20]. Second, we consider the impact of bit errors in the network's weights. This is highly relevant when deploying *quantized* networks on special-purpose hardware, so-called *accelerators* [SCYE17], for

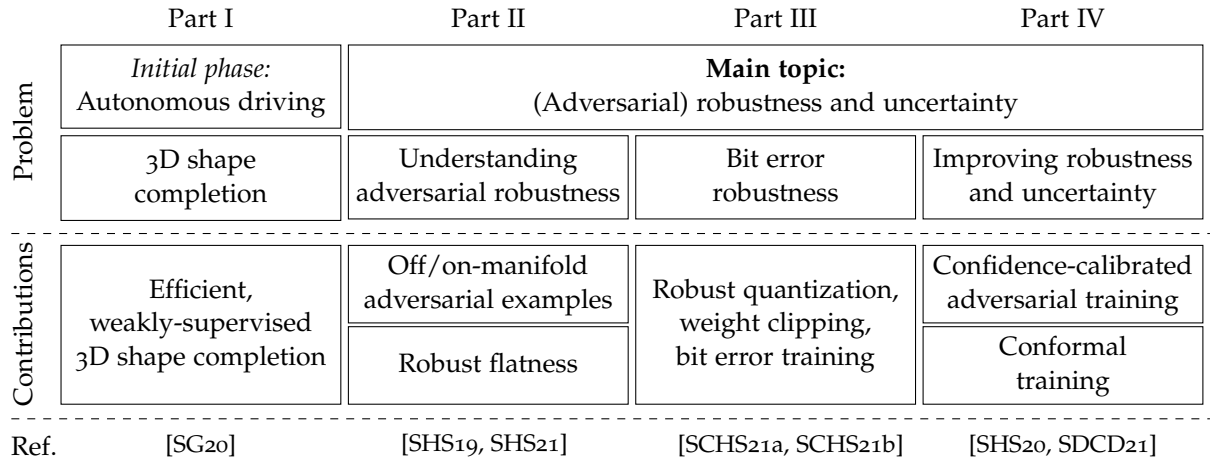


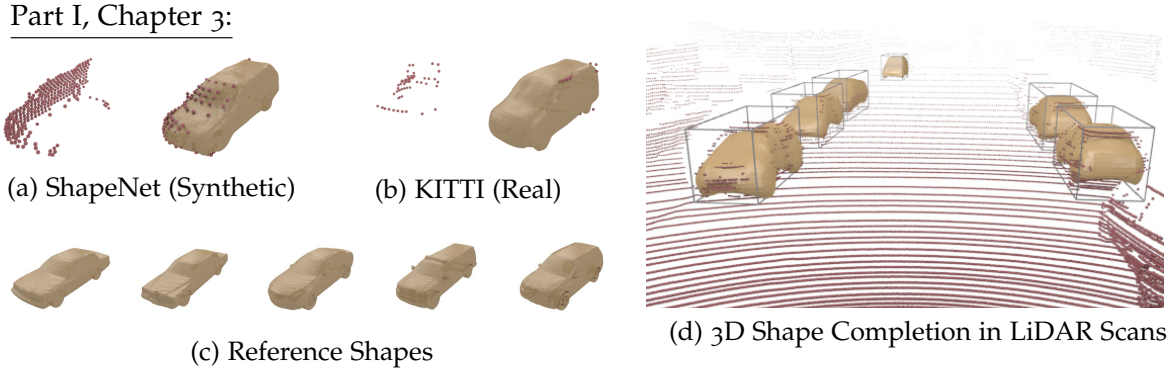
Figure 1.1: **Problems Tackled in this Thesis:** After discussing deep learning for weakly-supervised 3D shape completion in Part I, we mainly focus on robustness and uncertainty in deep learning. Part II intends to understand adversarial examples and adversarial training, while Part III focuses on robustness w.r.t. bit errors in the network's weights. Finally, Part IV proposes methods to improve uncertainty estimation alongside adversarial robustness.

low-cost inference. For example, low-energy operation of digital accelerators causes high rates of bit errors in their memories, directly affecting the stored network weights. Moreover, the memory is vulnerable to hardware- or software-based attacks. Here, robustness is necessary for wide-spread adoption and safe deployment of deep learning accelerators. Finally, we investigate how to provide meaningful uncertainty estimates alongside predictions. For example, deep neural networks tend to assign high confidence to wrong decisions, e.g., on perturbed or out-of-distribution examples. Besides, most deep learning techniques do *not* provide any guarantees beyond empirical test accuracy. However, alongside robustness, well-calibrated uncertainty estimates with relevant guarantees are essential for safe deployment. Altogether, we aim to **quantify, understand and improve robustness as well as uncertainty estimation** in these settings.

Besides the main focus on robustness, in the initial phase of this thesis, we also addressed the problem of 3D reconstruction in autonomous driving. In order to obtain a complete understanding and representation of the surrounding environment for critical components such as path planning or mapping and localization [JGBG20], autonomous vehicles are commonly equipped with 360 degrees LiDAR sensors [GLU12]. However, detecting and completing the 3D shapes of relevant traffic participants such as cars is extremely difficult due to sparse and noisy point cloud observations. Unfortunately, deep learning techniques struggle with difficult-to-obtain ground truth shapes in such settings. We aim to address this problem using a *weakly-supervised* approach to 3D shape completion.

A high-level overview of the problems addressed in this thesis is provided in Figure 1.1. Specifically, in Part I, we propose a weakly-supervised, learning-based approach to tackle 3D shape completion from sparse point clouds. Next, in Part II, we turn towards robustness against adversarial examples, intending to understand why adversarial examples exist and how both adversarially robust and accurate models can be trained effectively. In Part III, we switch to studying bit errors in quantized networks and show how robustness contributes to energy-efficient and secure accelerators. Finally, in Part IV, we investigate how proper uncertainty estimates allow rejecting a wide-range of adversarial, corrupted or out-of-distribution examples and how to obtain guarantees alongside these uncertainty estimates.

Part I, Chapter 3:



(a) ShapeNet (Synthetic) (b) KITTI (Real) (c) Reference Shapes (d) 3D Shape Completion in LiDAR Scans

Figure 1.2: **Weakly-Supervised 3D Shape Completion Problem:** We propose a deep learning approach to complete shapes from noisy point clouds without access to the corresponding ground truth shapes, e.g., for autonomous driving (d). This allows us to learn shape completion not only on synthetic (a) but also on real data (b), where ground truth shapes are unavailable, only by requiring a set of held-out reference shapes of known object class (c). In our *amortized maximum likelihood* approach, we first learn a shape model on the reference shapes before *learning to* align this shape model with the actual observations for efficient inference.

1.1 3D Shape Completion under Weak Supervision

Corresponding to an early phase of this thesis, we first consider a special-case of 3D reconstruction, namely *3D shape completion* from point clouds. This is a fundamental problem in 3D computer vision [FH13] that recently received considerable attention due to promising applications in robotics or autonomous driving [JGBG20]. As depicted in Figure 1.2 (a,b), 3D shape completion describes the problem of predicting a complete shape given an incomplete and noisy point cloud observation. In autonomous driving or robotics, for example, LiDAR sensors and depth cameras are used to provide point clouds of the surrounding environment, c.f. Figure 1.2 (d). Obtaining a complete 3D representation of the environment from this point cloud is crucial for safety and many sub-tasks such as mapping or path planning. This also entails shape completion of particularly relevant objects such as cars, trucks or pedestrians.

Classical approaches to this problem are often data-driven, splitting the problem into shape retrieval and shape-to-observation alignment [SKAG15]. While later data-driven approaches avoid the shape retrieval part by learning a shape model from synthetic reference shapes [NXS12, DPRR13, HSP14, ESL16], see Figure 1.2 (c), alignment remains a complex optimization problem that makes inference very slow. This is addressed in recent work, e.g., [FMAJB16, DQN17], which leverages deep learning to learn shape completion end-to-end, assuming full access to the incomplete point clouds and the corresponding completed shapes. Unfortunately, obtaining annotations with complete shapes is extremely time-consuming and error-prone. Instead, weaker notions of labels such as bounding boxes are often readily available. Thus, we develop a learning-based approach that only requires knowledge of the object categories for training and remains efficient at test time.

Contributions: In Chapter 3 of this thesis, we propose an *amortized maximum likelihood* approach towards *efficient, weakly-supervised* 3D shape completion. Based on a learned shape prior, alignment is formulated as a maximum likelihood problem, which we learn to solve for efficient inference. To this end, we introduce a loss purely operating on the (sparse) observation, not requiring the corresponding ground truth shapes. This allows us to learn 3D shape completion in challenging real-world situations, e.g., on LiDAR or Kinect scans [GLU12, YRM+19], while offering fast inference.

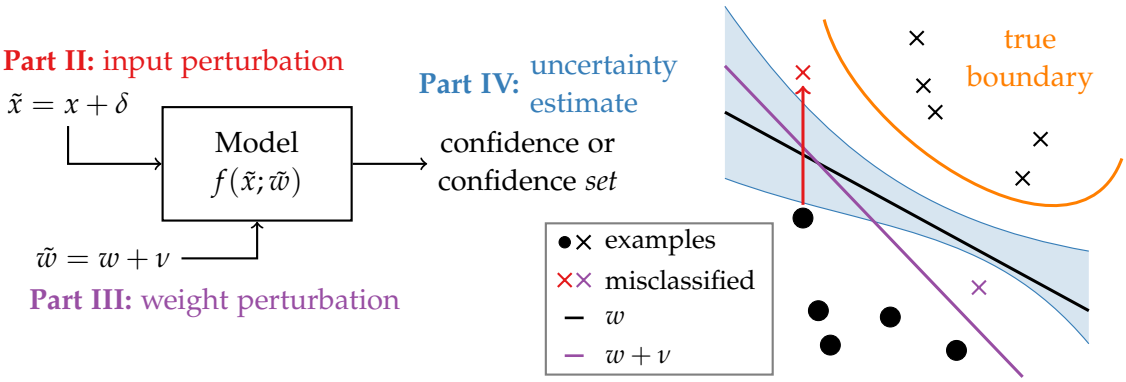


Figure 1.3: **Robustness and Uncertainty Problems: Left:** We consider both perturbations in the inputs (Part II) and the weights (Part III). These can be adversarial, i.e., maliciously crafted, or random. We also study uncertainty estimation for (adversarially robust) models (Part IV), either using the predicted confidence as uncertainty estimate or allowing the model to predict confidence *sets* instead. **Right:** Illustration of input perturbations, weight perturbations, and uncertainty using a 2D and two-class classification problem (● vs. ×), see text for details.

1.2 Robust and Uncertainty-Aware Deep Learning

Turning to the main topic of this thesis, our work on robustness and uncertainty estimation for deep learning is summarized in Figure 1.3 (left). First, we consider a deep neural network f, taking an input x and weights w to make a prediction $f(x; w)$, e.g., classification scores in an image recognition problem. While such models obtain high accuracy on held-out test sets, [BCM+13, SZS+14] demonstrate that slightly perturbed inputs $x + \delta$ can alter the prediction significantly, often changing the predicted class. In Figure 1.3 (right), this is illustrated as moving the example $x + \delta$ (red) beyond the model's decision boundary (black), but not crossing the true one (orange). Next, we consider robustness in terms of the weights: While the examples remain unchanged, the decision boundary changes depending on a weight perturbation $w + v$ (magenta). In both cases, perturbations can be random or adversarially chosen to fool the model. Finally, we consider uncertainty estimation given the model's prediction. In practice, this involves estimating a per-example *confidence* indicating the certainty of the model in its prediction, e.g., when facing input perturbations. More intuitively, however, this can be thought of as an uncertainty in the model's decision boundary (blue).

1.2.1 Understanding Adversarial Examples and Training

Adversarial examples [SZS+14, GSS15], maliciously crafted input perturbations, can be found for the majority of problems in computer vision and beyond. However, especially in the context of *deep* neural networks, their existence is poorly understood. Early hypotheses [SZS+14, GSS15] have been superseded by the manifold assumption: On simplistic toy examples, adversarial examples are hypothesized to leave the underlying data manifold [TG16, GMF+18]. As this would make training on adversarial examples, so-called *adversarial training* [MMS+18], more difficult, this could also explain the observed trade-off between robustness and accuracy [TSE+19]. The manifold assumption is also relevant regarding the recent observation that adversarial training is prone to severe overfitting [RWK20]. This is problematic because robustness does not generalize well to a held-out test set despite robustness on the training set

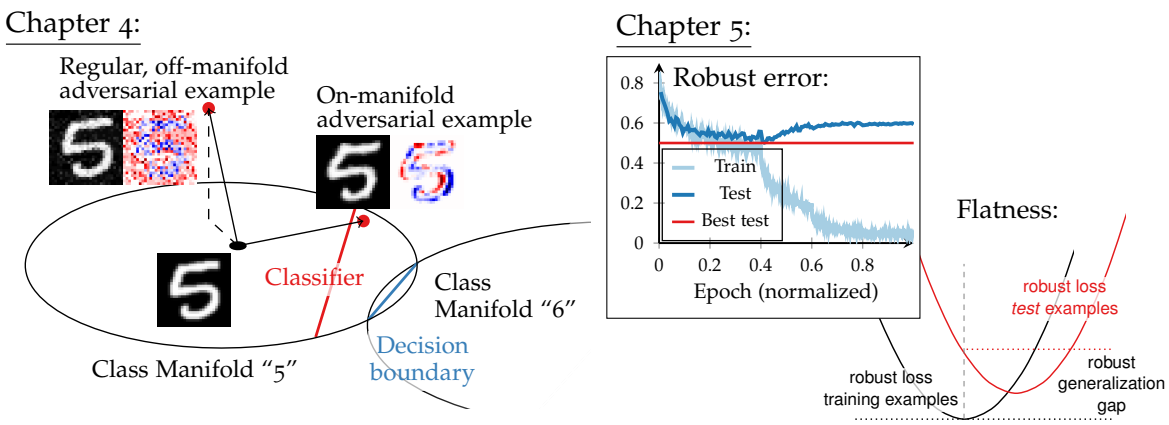


Figure 1.4: **Understanding Adversarial Examples and Training: Left:** We view adversarial examples in the context of the underlying data manifold and show that "regular" ones leave this manifold. However, adversarial examples can also be constrained to the manifold, corresponding to more meaningful perturbations. **Right:** Adversarial training improves robustness but suffers from severe robust overfitting where the test robustness does not continuously decrease throughout training. We study this phenomenon through the lens of flatness in the robust loss landscape w.r.t. weight perturbations.

continuously decreasing throughout training.

We confirm this hypothesis by studying adversarial examples in the context of the underlying data manifold on both synthetic and real datasets. As illustrated in Figure 1.4 (left), we also confirm that adversarial examples can be constrained to the manifold, as previously done on a simplistic toy dataset in [GMF+18]. Regarding the robustness-accuracy trade-off of adversarial training, [SZC+18, TSE+19] suggest that this trade-off is inherent. These results are, however, not unquestioned [RGB16, GMF+18]. By distinguishing between off- and on-manifold adversarial examples, we intend to reconcile these different perspectives. Similarly, we study *robust* generalization and the negative impact of *robust overfitting*, demonstrated in Figure 1.4 (right), from the viewpoint of the loss surface. Flat minima, where the loss does not change significantly with small weight perturbations, are known to be beneficial for generalization [NBMS17, KMN+17, JNM+20]. Intuitively, flatter minima reduce the generalization gap between training and test examples, as illustrated in Figure 1.4 (right). Recent work [WXW20b] argues that encouraging flatness in the *robust loss* surface also improves adversarial robustness. However, flatness is only assessed visually. With our work, we intend to improve our understanding of robust overfitting and how it can be avoided.

Contributions: In Chapter 4, we demonstrate that regular adversarial examples indeed leave the underlying manifold, often in an orthogonal direction. However, we can also constrain adversarial examples to the manifold, resulting in more meaningful perturbations (c.f. Figure 1.4, left). Training on such on-manifold adversarial examples is shown to improve generalization. While our results indicate that there is *no* inherent robustness-accuracy trade-off, regular adversarial training is found to increase sample complexity. This also contributes to severe robust overfitting, as discussed in Chapter 5. In order to better understand this phenomenon, we propose an average- and worst-case measure of flatness in the *robust* loss landscape w.r.t. weight changes. We show that robust overfitting is caused by converging to sharp minima. More broadly, we show that there is a clear relationship between flatness and superior robust generalization.

Chapter 6: Accelerator from [CSC+19]: Profiled (random) bit errors for 2 chips:

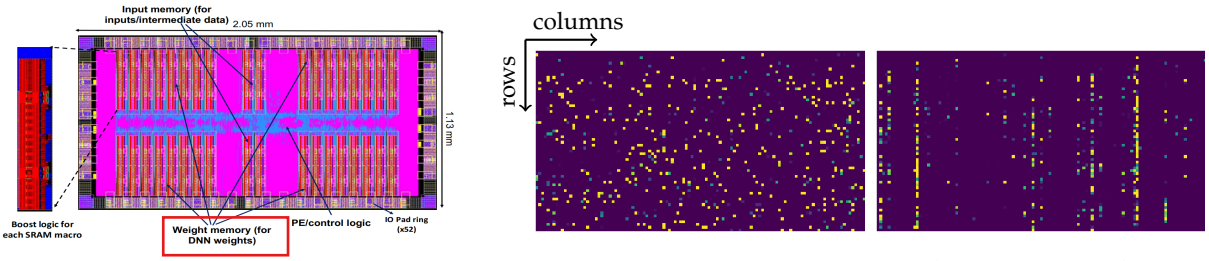


Figure 1.5: **Random Bit Error Robustness for Accelerators: Left:** Accelerators, e.g., the one from [CSC+19] shown here, usually feature a large portion of memory arrays for the model's weights. **Right:** Reducing energy consumption by reducing voltage, however, makes the memory unreliable, leading to bit errors. This is shown exemplarily for two memory arrays from different chips, where yellow indicates a high-probability of bit flips. Thus, improved robustness against such bit errors translates to improved energy-efficiency.

1.2.2 Improving Robustness Against Weight Perturbations

Special-purpose hardware for inference, so-called *accelerators* [CES16, nvd], reduce cost, space and energy-consumption compared to mainstream hardware. An example [CSC+19] is shown in Figure 1.5 (left). The largest component is the memory to store the model's weights. Thus, reading, writing and moving weights constitutes a dominant part of the overall energy-consumption. Recent work [KHM+18a, KOY+19] tried to reduce energy-consumption by reducing the memory voltage. Unfortunately, this leads to unreliable memory operations causing bit errors in the weights, see Figure 1.5 (right). These bit errors are handled with co-design approaches, i.e., a combination of hardware mitigation and neural network architecture or training strategies. In a parallel line of work, researchers also study the robustness of deep neural networks to maliciously injected bit errors [RHF19a, HRL+20], e.g., using hardware- or software-based memory attacks [KDK+14, MOG+20]. Overall, robustness against random and adversarial bit errors is crucial for energy-efficient and secure accelerators.

Unfortunately, co-design methods require extensive hardware knowledge, including the infrastructure to experimentally determine the induced bit errors of a specific chip. This also results in poor generalization across different chips or even voltage-levels, corresponding to different bit error rates. Pure hardware mitigation strategies [SWS+16, RWA+16, MUDV17, CSC+19, CH21a], however, add space and energy overhead while classical error correction is not applicable for high bit errors rates. To address these problems, we follow a purely software-based approach by training more robust deep neural networks. Considering the threat of adversarial bit errors, existing attacks [RHF19a, RHL+20] are inefficient and inflexible. Moreover, previous defenses [HRL+20] have been broken [RHL+20]. We show that a new and more efficient attack also allows improving adversarial bit error robustness during training.

Contributions: In Chapter 6, we follow a software-based approach for improving random bit error robustness through a *robust quantization* scheme, *weight clipping* during training as regularization and *random bit error training*. Our approach generalizes across chips and voltages, as validated on profiled bit errors of several real chips for which we are able to quantify the possible energy savings. Furthermore, we consider random bit errors in inputs and activations which are, at least temporarily, also stored on the unreliable memory. We also benchmark these methods against our newly proposed *adversarial bit error attack* that is shown to be more efficient and effective than related work [RHF19a, RHL+20]. Finally, our *adversarial bit error training* is an effective strategy to improve robustness against these attacks.

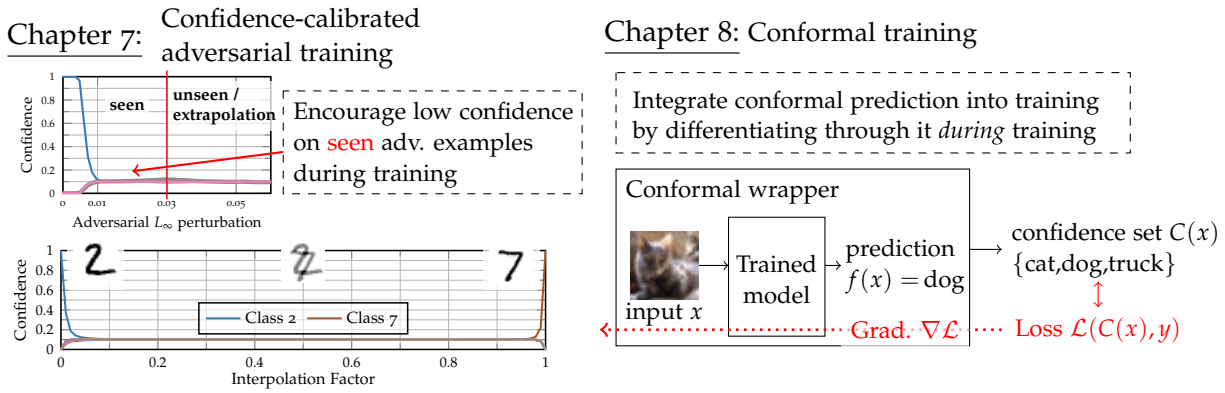


Figure 1.6: **Improving Adversarial Robustness and Uncertainty Estimation: Left:** Our *confidence-calibrated adversarial training* enforces low confidence on adversarial examples during training. This behavior is extrapolated to previously unseen adversarial or out-of-distribution examples, allowing to easily reject them without significant drop in accuracy. **Right:** Conformal prediction allows the model to predict confidence sets $C(x)$ of classes. Uncertainty is intuitively quantified by the number of predicted classes $|C(x)|$, so-called *inefficiency*. Our *conformal training* allows training conformal predictors jointly with the model instead of being applied after training. At the same time, it enables optimizing a wide-range of special-purpose losses $\mathcal{L}$ defined directly on the confidence sets, e.g., to influence their composition.

1.2.3 Improving Adversarial Robustness and Uncertainty Estimation

Uncertainty is usually estimated per-example, by providing a confidence in the prediction or a confidence set of possible classes. The former is a simple measure used extensively for adversarial example and *out-of-distribution detection* [HG17a, MLW+18, LLLS18b]. This improves the robustness-accuracy trade-off and achieves robustness against various types of adversarial examples, which is difficult for adversarial training [TB19, MWK20]. Unfortunately, many detectors have been broken repeatedly [CW17a, YBTV21, BHP+21]. Thus, [HMD19, HAB19] incorporate this idea into training by explicitly enforcing low confidence on out-of-distribution examples. These works show that deep neural networks are able to easily extrapolate this behavior beyond the examples seen during training. We intend to reconcile the shortcomings of detection schemes and adversarial training by encouraging low confidence on adversarial examples during training, as illustrated in Figure 1.6 (left).

Unfortunately, many robust deep neural networks do not provide any guarantee beyond empirical performance. In the context of uncertainty estimation, *conformal prediction* [GVV98, VGS05] allows the model to predict confidence sets containing multiple classes instead of making a single class prediction. These confidence sets are constructed such that the true class is guaranteed to be included with user-specified probability. Conformal prediction is widely applicable as a post-training calibration step [RSC20], e.g., to large-scale image recognition problems [ABJM21]. However, conformal predictors are usually hand-designed for different objectives and guarantees [BAL+21] and there is an inherent discrepancy between training and calibration: Deep neural networks trained with cross-entropy loss might not be optimal for the used conformal predictor. As shown in Figure 1.6 (right), we address both limitations by training the model *through* the conformal predictor, without losing its guarantee after training, but allowing arbitrary losses on confidence sets to be optimized.

Contributions: In Chapter 7, we introduce *confidence-calibrated adversarial training* which biases the model towards low-confidence predictions on adversarial examples seen during training, see Figure 1.6 (left). In contrast to standard adversarial training, the model can extrapolate

this behavior to previously unseen adversarial attacks. By rejecting low-confidence examples, robustness also generalizes to corrupted examples and out-of-distribution examples and the robustness-accuracy trade-off improves significantly. For evaluation, we propose a per-example worst-case evaluation across many adaptive adversarial attacks, as also adopted by standard benchmarks [CH20c, YBTV21].

Chapter 8 introduces *conformal training*, a method for end-to-end training of deep neural networks and conformal predictors. By differentiating through the conformal predictor, we can optimize arbitrary objectives defined directly on the confidence sets, c.f. Figure 1.6 (right). After training, we re-calibrate to obtain the original statistical guarantee. This approach is shown to reduce inefficiency, i.e., the average size of confidence sets representing the model's uncertainty. We can further improve on various application-specific metrics that standard conformal predictors cannot easily optimize.

1.3 Outline

This thesis is divided into nine chapters, organized in four parts:

Chapter 2, Related Work: We review previous work directly related to the topics discussed in this thesis. These include, e.g., secure deep learning in general and adversarial robustness as well as uncertainty estimation in particular. Moreover, we discuss related work on robustness for accelerators and quantization and recent work on conformal prediction.

Part I, Deep Learning for 3D Shape Completion

Chapter 3, Learning 3D Shape Completion under Weak Supervision: We present an efficient, weakly-supervised and learning-based approach to 3D shape completion. Specifically, we use deep neural networks to learn a powerful shape prior, which is subsequently used for completion in an *amortized maximum likelihood* approach. Our method can be trained on real-world datasets as no supervision besides object category is required.

This chapter corresponds to the IJCV publication [SG20] of the same title which represents a significant extension of previous work [SG18b] based on the master thesis [Stu17], see Section 3.1 for further details.

Part II, Understanding Adversarial Examples and Training

Chapter 4, Disentangling Adversarial Robustness and Generalization: We study adversarial examples w.r.t. the underlying data manifold, showing that they leave the manifold. We also conclude that adversarial robustness is not inherently in conflict with accuracy, but adversarial training exhibits higher sample complexity. Moreover, *on-manifold* adversarial examples can be used during training to improve accuracy.

This work corresponds to the CVPR 2019 publication [SHS19] with the same title. It was also presented at the Workshop on Uncertainty and Robustness in Deep Learning (UDL) 2019 and the Heidelberg Laureate Forum (HLF) 2019.

Chapter 5, Relating Adversarially Robust Generalization to Flat Minima: In this chapter, we intend to understand how different variants of adversarial training improve robustness and avoid robust overfitting. To this end, we relate robust generalization to flat minima in the robust loss surface w.r.t. weight changes.

This chapter corresponds to the ICCV 2021 paper [SHS21] with the same title. Short versions were presented at the Workshop on Uncertainty and Robustness in Deep Learning (UDL) 2021, the Workshop on Adversarial Learning Methods for Machine Learning and Data Mining (AdvML) 2021, and the Workshop on Adversarial Machine Learning in Real-World Computer Vision Systems and Online Challenges (AML-CV) 2021. Invited talks were given at the Max Planck Institute for Mathematics in the Sciences and the University of California Los Angeles as well as at the University of Cagliari.

Part III, Improving Weight Robustness

Chapter 6, Random and Adversarial Bit Error Robustness: This chapter shows how bit error robustness of *quantized* deep neural networks can improve energy-efficiency and security of hardware accelerators. We use *robust quantization*, *weight clipping* and *random or adversarial bit error training* to improve robustness. To this end, we also propose an efficient and effective adversarial bit error attack.

The chapter corresponds to the MLSys 2021 paper *"Bit Error Robustness for Energy-Efficient DNN Accelerators"* [SCHS21a], subsumed in our work on *"Random and Adversarial Bit Error Robustness: Energy-Efficient and Secure DNN Accelerators"* [SCHS21b] which is currently under review. This work was conducted in collaboration with IBM Research and presented at the IBM Research Workshop on the Future of Computing Architectures (FOCA) 2020, the Lorentz Center Workshop on Robust Artificial Intelligence (RobustAI), the Workshop on Adversarial Learning Methods for Machine Learning and Data Mining (AdvML) 2021, and the Workshop on Adversarial Machine Learning in Real-World Computer Vision Systems and Online Challenges (AML-CV) 2021 where it was recognized as distinguished paper. It was also part of the Qualcomm Innovation Fellowship Europe 2019 and an invited talk was given at TU Dortmund.

Part IV, Improving Adversarial Robustness and Uncertainty Estimation

Chapter 7, Confidence-Calibrated Adversarial Training: We present *confidence-calibrated adversarial training* which biases the model towards low-confidence predictions on adversarial examples. By rejecting examples with low confidence, adversarial robustness generalizes to unseen threat models, corrupted examples and out-of-distribution examples. This also improves accuracy compared to standard adversarial training.

The content of this chapter is based on our ICML 2020 publication [SHS20] with the title *"Confidence-Calibrated Adversarial Training: Generalizing to Unseen Attacks"*. This work was presented at the Workshop on Uncertainty and Robustness in Deep Learning (UDL) 2020, the Bosch Center for Artificial Intelligence, the Qian Xuesen Laboratory of Space Technology, and as part of the Qualcomm Innovation Fellowship Europe 2019.

Chapter 8, Learning Optimal Conformal Classifiers: We present *conformal training*, a way of training model and conformal predictor end-to-end to minimize uncertainty and optimize arbitrary application-specific losses directly defined on confidence sets. Re-calibrating after training preserves the conformal predictor's statistical guarantee.

This chapter is based on our ICLR 2021 paper [SDCD21] with the same title. This work was conducted while at DeepMind and presented at UC Berkeley.

Chapter 9, Conclusion: We conclude this thesis by summarizing our key findings and discussing promising directions of future work.

1.4 Publications

The content of this thesis has previously appeared in the following publications, ordered as outlined above:

[SG20] David Stutz and Andreas Geiger. Learning 3D shape completion under weak supervision. *International Journal of Computer Vision (IJCV)*, 128(5):1162–1181, 2020.

[SHS19] David Stutz, Matthias Hein, and Bernt Schiele. Disentangling adversarial robustness and generalization. *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[SHS21] David Stutz, Matthias Hein, and Bernt Schiele. Relating adversarially robust generalization to flat minima. In *Proc. of the IEEE International Conference on Computer Vision (ICCV)*, 2021.

[SCHS21a] David Stutz, Nandhini Chandramoorthy, Matthias Hein, and Bernt Schiele. Bit error robustness for energy-efficient DNN accelerators. In *Proc. of Machine Learning and Systems (MLSys)*, 2021.

[SCHS21b] David Stutz, Nandhini Chandramoorthy, Matthias Hein, and Bernt Schiele. Random and adversarial bit error robustness: Energy-efficient and secure DNN accelerators. *arXiv.org*, abs/2104.08323, 2021 (under review).

[SHS20] David Stutz, Matthias Hein, and Bernt Schiele. Confidence-calibrated adversarial training: Generalizing to unseen attacks. In *Proc. of the International Conference on Machine Learning (ICML)*, 2020.

[SDCD21] David Stutz, Krishnamurthy Dvijotham, Ali Taylan Cemgil, and Arnaud Doucet. Learning optimal conformal classifiers. In *Proc. of the International Conference on Learning Representations (ICLR)*, 2022.

Further contributions were made to the following works not discussed in this thesis:

[KSA+21] Iryna Korshunova, David Stutz, Alexander A. Alemi, Olivia Wiles, and Sven Gowal. A closer look at the adversarial robustness of information bottleneck models. In *Proc. of the International Conference on Machine Learning (ICML) Workshops*, 2021.

[RSS20] Sukrut Rao, David Stutz, and Bernt Schiele. Adversarial training against location-optimized adversarial patches. In *Proc. of the European Conference on Computer Vision (ECCV) Workshops*, 2020.

[GSS22] Yong Guo, David Stutz, and Bernt Schiele. Improving corruption and adversarial robustness by enhancing weak subnets. *arXiv.org*, abs/2201.12765, 2022 (under review).

Related Work 2

Contents

In this chapter, we review literature on the interdisciplinary topics of deep learning in computer vision as well as security and hardware accelerators. Deep learning has enabled tremendous progress in many difficult tasks. For example, in Part I of this thesis, we specifically discuss the use of deep learning for *3D shape completion*, i.e., the reconstruction of partly observed shapes from sparse point clouds. However, the great success of deep learning also raises concerns regarding *robustness* and *uncertainty* in deep learning, which we focus on in the remaining parts of this thesis, i.e., Part II, III and IV. Both topics are important, as a lack of trust by practitioners and users can seriously hinder wide-spread adoption, especially in high-stakes applications.

In Section 2.1, we discuss related work on 3D shape completion, also touching on research regarding useful 3D representations and shape modeling. Subsequently, in Section 2.2, we provide a high-level overview of robustness and uncertainty in the broader context of security and trustworthiness in deep learning, where both topics play a crucial role. Based on this overview, Section 2.3 specifically discusses relevant work on adversarial attacks and defenses, also considering adversarial training and detection methods. Then, Section 2.4 switches to weight robustness, discussing approaches to energy-efficient and secure deep learning accelerators, i.e., specialized hardware for inference. Here, we particularly discuss related work tackling robustness against random and adversarial bit errors in quantized weights. Finally, in Section 2.5, we discuss two research directions in uncertainty estimation, namely out-of-distribution detection and conformal methods.

2.1 3D Shape Completion

3D vision problems such as 3D reconstruction have a long tradition of "classical" methods that leverage extensive geometry and domain knowledge [SCD+06]. Recently, however, deep learning started to replace many of these methods due to the increasing availability of large amounts of data [HLB21]. In the following, we consider a special case of 3D reconstruction:

shape completion from partially observed point clouds. Such point clouds are commonly available in autonomous driving or robotics settings due to the increased use of laser- or camera-based depth sensors such as LiDAR and Kinect. We briefly review both optimization and learning-based approaches to shape completion in Section 2.1.1. Subsequently, we discuss relevant work on 3D shape priors which are commonly used for completion in Section 2.1.3 and touch on effective shape representations for learning in Section 2.1.2.

2.1.1 Shape Completion

Following [SKAG15], classical shape completion approaches can roughly be categorized into optimization-based methods and data-driven methods. Representative works on optimization-based shape completion include [CT11, TOZ+11, KH13] and references therein. These methods are, e.g., based on variational or level-set methods for optimization. Often, symmetry is leveraged [TW05, PMW+08, ZSW+10, LA11, KAEP12]. In contrast, data-driven approaches [PMG+05] pose shape completion as retrieval and alignment problem. While [PMG+05] allow shape deformations, [GAGM15], use the iterative closest point algorithm [BM92] for fitting rigid shapes. [SFCH12] additionally considers completing shapes based on individual parts. Subsequent work usually avoids explicit shape retrieval by learning a latent space of shapes [NXS12, BCLS13, DPRR13, HSP14, RGT+15, LDGN15, ESL16, NHT+16]. Alignment is then formulated as an optimization task over the learned, low-dimensional latent space. In both cases, inference is typically slow as a complex optimization problem needs to be solved. With the recent success of deep learning, learning-based approaches are becoming more and more popular [SGF16, REM+16, FMAJB16, SM17, DQN17, FSG17, RUBG17, HLH+17, YWW+17, GFK+18, YRM+19, WLT20]. Strictly speaking, these are data-driven as well. However, shape retrieval and fitting are *both* avoided by directly learning shape completion end-to-end, usually on synthetic data from ShapeNet [CFG+15] or ModelNet [WSK+15]. Unfortunately, these approaches require full supervision, i.e., access to the completed shapes during training.

In Chapter 3, we develop a *learning-based* approach that addresses the shortcomings of both lines of work: While we still learn a 3D shape prior and formulate shape completion as an optimization problem, expensive inference is avoided by *learning to infer* [RM15a, WL16, RHG16]. To this end, we propose a loss that does *not* require full supervision. Thus, we can train on real-world data, e.g., from LiDAR sensors [GLU12] or Kinect cameras [YRM+19], and generalize very well. This is also in contrast to recent few-shot approaches [WH19, MTP+21] that still need *full* supervision, but on few examples.

2.1.2 Shape Models

Shape priors are a core ingredient for many 3D shape completion and reconstruction methods [DPRR13, GG15, KTCM15, RLT+20]. Originally, most approaches used hand-crafted shape models, for example based on anchor points or part annotations [ZSSS13, ZSS14, LMHD14, PSG+15]. Recent work, however, has shown that generative models such as variational auto-encoders (VAEs) [KW14a] or generative adversarial networks (GANs) [GPM+14a] allow efficiently generating, manipulating and reasoning about 3D shapes. Exemplary work includes [WSK+15, SGF16, GFRG16, WZX+16a, SM17, LYF17, NW17, LYF17]. Beyond 3D vision, shape priors are also heavily used in other tasks such as pose estimation [SDYT09, SDYT11, PSR13, AME+14], tracking [LM09, MS14], segmentation [SDYT09, SDYT11, PSR13], object detection [ZSSS13, ZSS14, SX14, PSG+15, ZPA+15] or recognition [LMHD14], to name just a few exam-

ples. Besides the scale and quality of available shapes, we will see that shape representations are in the center of recent research.

In Chapter 3, we utilize VAEs in order to learn shape models on ShapeNet or ModelNet. However, as long as a low-dimensional representation with known, parametric prior is learned, our approach is agnostic to the type of generative model used.

2.1.3 Shape Representations

For both shape completion and shape modeling, the used shape representation is a crucial choice as it determines the complexity during training and the level of details recovered at inference. Originally, occupancy-based voxel representations were common, as in [WSK+15, BLRW16, WZX+16a, HLH+17, DQN17]. However, high resolutions required to predict detailed shapes make training with 3D convolutions expensive. Thus, several representations have been explored: (truncated) signed distance [DQN17, LSJ+17, RUBG17], point clouds [QSMG17, FSG17, QYSG17, ADMG18, YFST18, YLF+18], meshes [LDG18, RBSB18, KJP+18, DN19], or implicit functions [MPJ+19, PFS+19, MON+19, CZ19]. Signed distance functions, for example, allow reconstructing sub-voxel details using marching cubes [LC87], while still predicting a fixed resolution. The computational overhead of occupancy grids and signed distance functions can be reduced using efficient data structures such as octrees or kd-trees [RUG17, TDB17, HTM17, WLG+17, KL17, WLT20]. Recently, however, implicit function approaches [PFS+19, MON+19, CZ19, SHN+19, GCV+19, XWC+19, NG21] have become extremely popular, often combined with differentiable rendering [LZP+20, YKM+20, NMOG20]. The idea is similar to signed distance functions. However, instead of predicting a fixed grid of distances, a distance function is learned that can be evaluated at arbitrary 3D points, see [TFT+20, DL21] for an overview. All of these representations are of utility in 3D reconstruction in general, independent of the specific task and we refer to several surveys [ICNK17, ASS+18, BYW+20, GaWH+21] for details.

In Chapter 3, we use truncated signed distance functions alongside occupancy grids in order to complete shapes with sub-voxel details. This can be seen as an early pre-cursor to nowadays implicit representations [MPJ+19, PFS+19, MON+19, CZ19]. Our approach is, however, agnostic to the used shape representation as long as our weakly-supervised loss is adapted accordingly. In Chapter 3, we do this by generalizing signed distance functions to sparse point cloud observations.

2.2 Robustness and Uncertainty for Secure and Trustworthy Deep Learning

The success of deep learning in numerous vision tasks, not only 3D reconstruction, sparked a large body of research on problems related to robustness and uncertainty alongside many other topics surrounding machine learning security and trustworthiness. In the context of security, for example, deep neural networks are vulnerable to a wide variety of attack vectors [KNL+20, BGBK21], often aimed at compromising performance. In the even broader context of trustworthiness [Var19, KR19, HKR+20], privacy concerns as well as interpretability, fairness or causality are studied in addition to security threats. In the following, we give a high-level overview of the three main topics of this thesis, namely adversarial robustness, weight robustness and uncertainty estimation, and embed them in the larger context of security and trustworthiness.

Adversarial Robustness: Despite adversarial examples already being reported in [DDM+04, LM05], they received considerable attention after their "re-discovery" in [BCM+13, SZS+14]. In the security community, adversarial examples are also known as *evasion attacks*, which are extensively studied, e.g., in the context of network security [HPK01]. Early work on adversarial examples considers very specific problems such as spam filtering [WW04, NBC+08] or malware detection [FSP+06, BCM+13]. While [SZS+14] originally considered "imperceptible" adversarial perturbations, recent work also focuses on related robustness properties such as "natural" corruptions [HD19, HD19, GRM+19] or (adversarial) transformations [ETSM17, XZL+18, AAG19]. Evasion attacks can be constructed for many machine learning methods besides neural networks, including logistic regression [FXMY14], support vector machines [ZKTX12], boosted decision trees [AH19, WZC+20], Gaussian processes [GPSB18] or k-nearest neighbors [SW19, SW20]. In this thesis, however, we are mostly concerned with deep neural networks. Moreover, adversarial examples can be found for many vision tasks and beyond [CW18, ZSA19, HP20], ranging from recognition, over object detection [LSFF17] and semantic segmentation [FKMB17, CANK17] to 3D vision problems [XQL19, LCY+20, KHNY21], to name just a few. Throughout this thesis, we consider the task of image classification. As we tackle adversarial robustness in both Part II and IV, we provide a detailed discussion in Section 2.3.

Weight Robustness: In contrast to the model's inputs, its weights are generally not exposed to the user, leaving little room for malicious manipulation. Nevertheless, various types of backdooring and trojaning attacks [JZJ+18, DS20, RHF20] manipulate the weights in order to integrate "hidden", potentially malicious behavior. Moreover, weight robustness becomes relevant for model deployment: [SSH15, MAA+16, MDI19, ABvB+20] study the impact of quantization errors. More recently, the deployment on special hardware exposes the weights to hardware faults or attacks. For example, [KDK+14, MOG+20] demonstrate bit-level attacks on the memory of digital accelerators. Moreover, memory faults also occur in [GKKR17, GKB+19] when optimizing the power consumption by lowering voltage. Finally, works such as [NSY92, CMMR94, DVK98] also consider robustness to various types of structural network faults, e.g., removed neurons or weights. In Part III, we address both random and adversarial bit errors in the memory and show that more robust models contribute to secure and energy-efficient accelerators. We review relevant research on these topics in Section 2.4. Furthermore, Part II shows that weight robustness can have unexpected benefits in terms of improving (adversarially robust) generalization [JNM+20, WXW20b, FKMN21].

Uncertainty Estimation: Not only adversarial examples, but known or unknown distribution shifts [QCSLS09, WGS+21, YZLL21] in general, demonstrate that recent deep learning methods provide no or very unreliable uncertainty estimates alongside their predictions. In practice, however, proper uncertainty estimates are crucial, for example for detecting out-of-distribution examples [NYC15, HG17a, SLH+21]. Similarly, uncertainty plays a key role in anomaly, novelty, and outlier detection [AY01, HA04, PCCT14, WBH19, RKV+21, PSCvdH21], also see [YZLL21]. This sparked extensive research on improving uncertainty estimates, giving rise to Bayesian approaches [Qaz96, PEM01, WT11, BCKW15, GG16, KHH20a, KHH20b, WRV+20, KHH20c, KHH21], ensemble techniques [Hes96, LPB17, WTB20, WSTJ20, HJF+21], data augmentation methods [HMKS19, TCB+19, MKH19, HMC+20, WJM+20, WJM+21] and generative models [Bis93, NMT+19a, NMT+19b], see [APH+21, GTA+21] for detailed surveys. Recently, conformal methods [GVV98, VGS05, RSC20, ABJM21] also received considerable attention as they promise to provide statistical guarantees on top of intuitive uncertainty estimates. We address uncertainty estimation in the context of adversarial robustness, out-of-distribution detection and conformal prediction in Part IV and review the most relevant related works in Section 2.5.

Security and Privacy: Besides adversarial robustness or weight robustness, various other security and privacy issues have been raised recently. On the security side, data poisening or backdooring attacks intend to integrate malicious behavior into models [RNH+09, BNL11, BNaL12, CLL+17, LMA+18, LZS+18, ZGJ+18, SHN+18, JZJ+18, DS20, GKGL20], sponge attacks intend to increase inference time [SZB+21], or adversarial re-programming can be seen as a resource stealing attack by maliciously re-purposing a model to perform a different task [EGS19]. Regarding privacy, membership inference attacks reveal private training examples [SRS17, SSSS17, RRLM18, CLE+19, SeSM19, LF20, CYZF20, SSZ21, ROF21] or model stealing allows "copying" the functionality of commercial models without consent [TZJ+16, OAFS18, WG18, dSBB+18, RK20, OSF19, OSF20, KTP+20, JCB+20, TMWP21, KPQ21]. Both are also relevant from an intellectual property perspective where backdooring mechanisms have been adapted to "watermark" deep learning models [ABC+18, GP18, RCK18, MPT20, LWB21]. In many cases, there is an ongoing arms race between attacks and defenses [PMSW18].

Trustworthiness: Security and privacy are just individual ingredients in developing trustworthy deep learning models [Var19, KR19, HKR+20]. Alongside reliable uncertainty estimates [SOF+19], additional topics include interpretability [DVK17, GBY+18, Rud19], fairness and ethics [BHN19], or causality [Pea19]. The negative impact of not addressing these problems has already been demonstrated in various applications, e.g., in the form of deep fakes [CC18, AFG+19, YSAF21] or language model toxicity [WGU+21]. However, it is generally agreed that trust requires appropriately addressing all of these topics. Thus, recent work started jointly tackling some of these problems: For example, [GAZ19, NADL19, KSJ19, SKM+20, AMH20] study interpretability in conjunction with adversarial robustness. Similarly, adversarial examples are used in privacy settings [JG18b], differential privacy [Dwo06] can be used for adversarial robustness [LAG+19], and trade-offs between robustness and privacy are considered in [Hay20, HBK22]. Similar work exists at the intersection of fairness and robustness [XLL+21, NDS+21]. Overall, this illustrates the complexity of deploying deep learning applications while taking into account human users, potential misuse and the impact on society.

For this thesis, many of the above-mentioned problems are out-of-scope. However, the problems we address, i.e., adversarial robustness, weight robustness and uncertainty estimation, represent key challenges that need to be solved as part of developing secure and trustworthy deep learning techniques for high-stakes applications. Thus, in the following sections, we specifically discuss literature on these three topics.

2.3 Adversarial Robustness

Adversarial examples pose a severe threat to deployed deep learning systems. In the following, we discuss existing work on developing neural networks that are robust against such adversarial perturbations. To this end, we review several adversarial attacks and defenses on images, with specific focus on adversarial training [GSS15, MMS+18] and detection of adversarial examples. In the context of adversarial training, we also discuss the generally observed trade-off between accuracy and robustness [TSE+19] as well as overfitting issues [RWK20]. For further references, we refer to recent surveys in [AM18, BR18, YHZL19, XML+20, SN20, CAB+20].

2.3.1 Adversarial Attacks

In computer vision, the actual adversarial perturbation is usually constrained, e.g., using some L_p norm, to make sure that the (true) label is preserved. In this thesis, we partic-

ularly focus on these L_p-constrained, often imperceptible, adversarial examples in an image recognition setting. Such L_p-constrained adversarial examples have been reported for numerous tasks in computer vision [TTV16, CZC+17, CANK17, LSFF17, KFS18, XYL+19, XQL19, XQL19, WZYS19, SKM+20, BCL+20, HGXL21]. However, we note that other formulations exist, especially for other modalities such as text, speech, graphs or tabular data [DLT+18, ZAG19, BRA+19, ZSA19, HP20, SYO+18]. We follow common practice and categorize adversarial attacks by the adversary's knowledge: *white-box* attacks assume full access to the model, while *black-box* attacks assume limited access, e.g., only to the model's predictions.

White-Box Attacks: The original adversarial attacks reported for deep neural networks are white-box attacks that utilize the model's gradient in order to compute adversarial perturbations in one [GSS15] or multiple steps [SZS+14]. These attacks usually include a hard constraint forcing the L_p norm of the perturbation to be $\leq \epsilon$ or minimize the L_p norm without explicit constraint [MFF16, CW17b, LZLY17, RGPA21]. Replacing the L_p norm with perceptual metrics [ZIE+18, WBSS04] has also been studied [JMGD19, LSF21, GMVP21, ZLL20b]. Building on the framework proposed in [MMS+18], these gradient-based approaches have been extended in numerous ways, e.g., additionally including a momentum term [DLP+18], random step sizes [CGG+20], automatic step size [CH20c], different objectives [RGB17] or optimizers [UOKvdO18], additional attention [JK21] and many more [GUQ+19, ZCR19, BRK+19]. Most are applicable in a targeted or untargeted fashion, i.e., with or without an explicit target label. While attacks for L_∞ and L_2 are most commonly used, there are also successful attacks for L_1, L_0 or L_p in general [PMJ+16, BRK+19, MMF19, PRBB21]. However, L_0 or L_1 attacks can be more difficult from an optimization perspective [PMJ+16, CH21c]. Overall, implementation details have been shown to be important and many variants adapted to specific defenses have been proposed [ZYJ+19, WRK20, ZXH+20, PL21].

Large parts of this thesis, Chapter 4, 5 and 7 in particular, utilize the gradient-based attack of [MMS+18] and its variants [DLP+18, CH20c] to evaluate adversarial robustness. Chapter 7 additionally extends [MMS+18, DLP+18] using a backtracking scheme and an objective tailored to maximize the confidence of adversarial examples.

Black-Box Attacks: The simplest way to compute attacks without access to the model are *transfer* attacks [PMG16, PMJ+16, PMG+17]: adversarial examples computed for a known, accessible model are frequently also adversarial for other, similar models. This can be made practical for attackers by training a substitute model or directly stealing one [OAFS18, WG18, RK20]. Since, transferability of adversarial examples has been studied extensively [LCLS17, TPG+17, DMP+18, XZZ+19, DPSZ19, DMP+19, XZZ+19, WWX+20]. Alternatively, early black-box attacks are generally based on sampling perturbations and querying the target model [BRB18, KH18, ACFH20, GGY+19, LJL+20, SHT20, CH20a, BDK19, LJC+21]. Gradient-based approaches can also be applied by approximating the gradients through finite differences [CZS+17, IEAL18, IEM19, CLC+19, CJW20, ZCWL20, CG20], sign-based approaches [AO20, CSC+20, CSC+20, CZHW20], or Bayesian optimization [SSWK21], again using queries. Here, query-efficiency is crucial for the attacker. Thus, recent work focuses in particular on minimizing the number of queries [BHLS18, TTC+19, SCET20, CAS+20b, YMH21, MFM21, XLG+21, DCP+21, WSS+21]. Black-box attacks are useful in evaluation to overcome gradient obfuscation or non-differentiability of defenses [ACW18, QZZ+20, CRH20].

In Chapter 4 we study the robustness-accuracy trade-off also for transfer attacks and Chapter 7 adapts several black-box attacks [NK17, IEAL18, KH18, CH19, ACFH20] to our *confidence-calibrated adversarial training* for thorough evaluation.

Other Attacks: Besides these L_p-constrained attacks, many other types of adversarial examples have been studied. For example, adversarial patches or frames [BMR+17, KZG18, ZZRP19,

RSS20] allow unrestricted changes in a small part of the image. These are commonly also printed or physically realizable [BMR+17, JM19, LSK19, XZL+20, LWL+20]. However, more unconstrained adversarial examples are also possible [NYC15], often based on generative models [SSKE18, WLX+20, WK21] to craft visually more meaningful adversarial examples. Such adversarial examples are also similar to so-called on-manifold adversarial examples which are obtained using a generative model trained on clean images [GMF+18]. Similarly, there are adversarial transformations [ETSM17, DMM18, XZL+18, HP18, XZL+18, AAG19] or color/saturation/contrast changes [HP18, SSC20, ZLL20a], which are both often referred to as "semantic" adversarial examples. All of these attacks are example-specific, however, so-called *universal* adversarial examples are also studied [MFFF17, MGB19, DZGZ21].

In Chapter 4, we generalize the notion of on-manifold adversarial examples from the toy dataset in [GMF+18] to real datasets, an idea later extended in [LTL+19, XTCZ21]. In Chapter 7, we also evaluate robustness against various types of attacks besides L_p constrained ones in order to obtain a more holistic evaluation of adversarial robustness.

Benchmarks: Reliable robustness evaluation usually involves an ensemble of white- and black-box attacks. Then, the worst-case across these attacks is used for evaluation [CH20c]. More recently, these attacks are learned, considering different objectives as well as targeted and untargeted attacks [YBTV21]. Benchmarks attempt to keep track of the quick progress[1]. At the same time, toolboxes bundling adversarial attacks have been released [NST+18, DWJ19, EIS+19, RZBB20, GXY+20] and competitions are organized [KGB+18, DFY+21][2].

In Chapter 7, we also use an ensemble of (adaptive) attacks and propose a *per-example* worst-case evaluation as in standard benchmarks such as [CH20c].

Cause for Adversarial Examples: It is largely unclear why adversarial examples actually exist. Originally, [SZS+14] found that adversarial examples are "extremely" rare negatives and [GSS15] attributes them to the linearity in many architectures. Others [TG16, GMF+18, SSRD19] argue against these assumptions. Today, the manifold assumption is a widely accepted theory [SKN+18, IJA+17, PS18, SRBB19]: adversarial examples are assumed to leave the data manifold. Moreover, it was shown that this is facilitated by models relying on non-robust features, e.g., spurious correlations, to boost accuracy [IST+19, YLS+19, WWHX20, SMK21, AL21].

Our work in Chapter 4 contributes to the manifold assumption by experimentally validating it on real-world image datasets. This also lead to the notion of *on- and off-manifold adversarial examples*. These insights have since been refined [XYL21, SMB21, GCJ+21] and used for defenses [LLL+20, LLS+21], improved generalization [GQH+20], or calibration [PBZ+20].

2.3.2 Defenses Against Adversarial Attacks

There is an extremely wide variety of adversarial defenses, including regularization schemes [LHL15, HA17, RD18, JG18a, SGOS+18, HRY19, YLW+18, CTOF20, RNR20, LCC20, YRZ+20a], adapted architectures [NG17, ZNR17, RSCC17, LBG+18, RB19, SRPR20, XSG20], ensemble methods [LCZH17, SHJU17, HWC+17, ZKS+18, ZKX18, SRR20], distillation [PMW+16, WJC18, GFFG20], pre-processing or dimensionality-reduction approaches [BCM17, XEQ18, BRRG18, MST18, PMG+18b, SGHG19, MWYA19, QZZ+20], so-called manifold-based methods [IJA+17, PS18, JSZ+19, SRBB19, NHL20, DLJ+20, CGDK20] or randomness/stochasticity based schemes [XWZ+18, GRCvdM18, WWZ+18, PMG+18a, ZKX18, ZL19, QZZ+20, AF21]. However, a significant portion of these methods have been broken by adaptive attacks [CW16, CW17a,

[1] See https://robustbench.github.io/ or https://ml.cs.tsinghua.edu.cn/adv-bench/
[2] See https://github.com/MadryLab/mnist_challenge.

CW17c, LZLY17, EIA18, MAT+18, AC18, Car19, CH20c], i.e., attacks designed to break a specific defense, stressing the importance of proper robustness evaluation [PDS+21]. Thus, recent work also considers adaptive defenses that typically involve test-time optimization [KSDT21, CLZ21, APT+21, SHM21, HLR21, MCW+21, YHL21], also see [CGB+22]. Besides, there is a significant line of work on certified robustness [YWW+17, GMD+18, MGV18, SGM+18, ZWC+18, WK18, GDS+18, CRK19, LAJ19, CAH19, CH20b, YDH+20, KLFG20] as well as verification approaches [Zako1, KBD+17, BTT+18, WRTK19, BLT+20]. Nevertheless, *adversarial training* [GSS15, MMS+18] has become the de facto standard to train adversarially robust models as the obtained (empirical) robustness typically exceeds the (certified) robustness of such methods. Besides discussing work on adversarial training, we address the accuracy drop generally observed in conjunction with adversarial training [TSE+19] and the problem of *robust overfitting* [RWK20]. Finally, we consider not directly improving adversarial robustness, but detecting adversarial examples instead.

Adversarial Training: Training against adversarial examples generated on-the-fly was first proposed in [GSS15]. However, it gained popularity in [MMS+18] when used with a multistep attack. While [MMS+18] trains *only* with adversarial examples, mixing clean and adversarial examples is also possible [GSS15]. In [ZYJ+19], this is combined with a Kullback-Leibler divergence between clean and adversarial predictions. [HMKS19, KTH20] further boost adversarial with additional self-supervised training, [AUH+19, CRS+19, GRW+21, SMH+22] utilize additional unlabeled/generated examples during training and [HLM19, JCCW20] employ pre-training strategies. More recently, [WXW20b] use weight perturbations during training to avoid robust overfitting. Plenty of additional variants have been proposed, including instance-aware threat models [BGH19, DSLH20], additional regularizers [ZW19, RHF19b, PYD+20, SF20, WCY20, BLZ+20, LML+21], curriculum training [CLS18, YLL+19], Bayesian networks [YZ18, LLWH19], and many more [LVKB19, KW20, WCG+20, CLC+20, ZCS+19, ZXH+20, JSW20, ZZMJ21, LK21, ZZN+21, RMD21, WW22, PLY+22]. Despite challenges, adversarial training with universal adversarial examples [SNX+18, PMPP18] is also researched. Unfortunately, computational complexity increases significantly. This is addressed in several works [SNG+19, WRK20, ZZL+19] by re-using adversarial examples during training or resorting to adversarial training with single-step attacks [NKM18, SNG+19, SB20, WXW20a, PL21], which comes with its own problems [LWJC20, AF20, KLL21, KM21]. Recent studies further focus on tuning hyperparameters [GQU+20, PYD+21], architectures [WCC+21, HWE+21, DMM21], the role of data augmentation [RGC+21b, RGC+21a], the generative capabilities of adversarially robust models [ZMS+21] or their loss landscape [YLW+18, XYP19, PYXW19, LSL+20]. Training schemes similar to adversarial training also play an important role in certified robustness [WK18, MGV18, WSMK18, GDS+18, SLR+19]. In all of these cases, the obtained robustness is limited to the type of adversarial examples used during training, e.g., a specific L_p threat model. Various approaches [TB19, MWK20, MSH21, PTSS21] attempt to address this problem by training with multiple types of adversarial examples. However, again, robustness generalizes poorly beyond those adversarial examples.

In Chapter 7, we present *confidence-calibrated adversarial training* which intends to obtain generalizable robustness against multiple types of adversarial examples *without* seeing them during training. Thereby, it tackles adversarial robustness, out-of-distribution detection and calibration jointly. This is achieved by enforcing low confidence on adversarial examples during training and in stark contrast to all the methods outlined above. These usually intend to classify adversarial examples correctly with high confidence. Since, the idea of integrating rejection through confidence in adversarial training has been studied in several additional works [LF19, KCF20, BBSZ20, PZH+21, JPK+21, SLK21, CRC+22].

Robustness-Accuracy Trade-Off: Adversarial training generally leads to decreased accuracy on clean examples. This trade-off between robustness and accuracy has explicitly been studied in several works: [TSE+19, SZC+18] argue that there exists an inherent trade-off between robustness and generalization. However, the theoretical argument in [TSE+19] is questionable as adversarial examples are allowed to change their actual, true label and the experimental results obtained in [SZC+18] stem from comparing different architectures and training strategies. [SST+18, RXY+19] argue that the trade-off stems from finite data and adversarial robustness has higher sample complexity, while [CMZK20] argues that more data can actually worsen the trade-off. In contrast, [RGB16, YRZ+20b] suggest that both accuracy and robustness should be achievable, i.e., the trade-off is not inherent. [JSH20, DHHR20, RXY+20, CHY22] study the trade-off in simple settings, e.g., linear cases or Gaussian cases and [Nak19] argues that more complex models are necessary. Beyond adversarial training, this trade-off is also observed for most certified robustness approaches [ZCX+20, GRF+20, MKW+21] and remains difficult to overcome, even in recent training methods [BGH19, WCG+20, ZXH+20, ASZ20, RMD21, PLY+22]

Supported by [YRZ+20b, SST+18], our work in Chapter 4 concludes that the trade-off is not inherent and includes a detailed discussion of the theoretical argument of [TSE+19]. However, we also find adversarial training to have higher sample complexity. Moreover, our work suggests that adversarial examples leaving the underlying manifold could be the core reason for this trade-off. Our adversarial training variant in Chapter 7 improves the trade-off significantly by encouraging low-confidence prediction on adversarial examples.

Robust Overfitting: Adversarial training also suffers from severe robust overfitting [RWK20]. While early stopping was shown to be reasonably effective, this motivated work [WXW20b, SSJF21, CZL+21] trying to mitigate robust overfitting altogether. While [SSJF21] studies the use of different activation functions, [WXW20b] proposes adversarial training with *adversarial weight perturbations* explicitly aimed at finding flatter minima in order to reduce overfitting. Instead, [CZL+21] explores self-training and distillation. Robust overfitting is also studied in relation to memorization in adversarial training [DXY+22], the impact of problematic training data [DLS21a], and double descent [DLS21b].

In Chapter 5, we empirically link robust overfitting and good robust generalization to the flatness of the found minima. This connection has been studied extensively in the context of *clean* generalization [NBMS17, KMN+17, CCS+17, DPBB17, STIM18, BGSW18, IPG+18, JNM+20, FKMN21]. Using flatness measures adapted to the *robust loss*, we show that many well-performing methods find flat minima.

Detection of Adversarial Examples: Instead of correctly classifying adversarial examples, as intended in adversarial training, some early works [LL17, BCM17, FCSG17, GWK17, GMP+17, HG17b, MGFB17] try instead to *detect* and *reject* adversarial examples. However, they have been shown to be ineffective against adaptive attacks [CW17a]. Nevertheless, rejecting adversarial examples by, e.g., confidence or other statistics, is still an active area of research [ABB+17, ZH18, SG18a, CBC+18, RSJK18, LLD+18, MLW+18, LLLS18b, RKH19, LZZ+19, AD19a, SWW+20, SKC+20, CSG20, ECW20, TZLD21, LTHL21] despite approaches being broken continuously [YBTV21, BHP+21]. Thus, in recent benchmarks, most detectors are applied on top of adversarially trained models [YBTV21]. Some of these approaches are also used to detect out-of-distribution examples [MLW+18, LLLS18b, LLS18]. We also refer to [AHFD21] for a recent survey on adversarial example detection.

Our *confidence-calibrated adversarial training* of Chapter 7 also functions as a detector. However, it is directly trained in a min-max fashion instead of being applied after adversarial training. We also use an extensive range of adaptive attacks for evaluation, thereby breaking several detector baselines [MLW+18, LLLS18b].

2.4 Robustness for Neural Network Accelerators

Besides robustness w.r.t. input perturbations, many applications require robustness in terms of the model's parameters. For example, quantization often expects models to be robust against quantization errors in their weights [SSH15, MAA+16, MDI19, ABvB+20]. In this thesis, we consider deploying quantized models on special-purpose hardware for inference, so-called *accelerators*. Here, robustness in the weights is relevant independent of the hardware type: For example, analog hardware inherently induces noise in computations, which can be modeled using noisy weights, see [SHES17, HGP19, ZKMW20, ICZ+20, FQ21, ICZ+21] for representative recent work. While analog hardware promises cheap and energy-efficient accelerators, digital hardware is still dominating in the context of deep learning [HGP19]. However, in digital hardware weight robustness is also very relevant: Besides quantization, memory arrays commonly exhibit bit errors, especially when optimizing for energy efficiency [KHM+18a, KOY+19]. Moreover, attacks on the memory are wide-spread and possible using both hardware- and software-based approaches [KDK+14, MOG+20]. In both cases, robustness against bit errors in the model's quantized weights is crucial. We review relevant work on random and adversarial bit errors in Section 2.4.1 as well as discuss related work on quantization and weight robustness in general, in Section 2.4.2 and 2.4.3, respectively.

2.4.1 Bit Error Robustness for Accelerators

Random Bit Error Robustness for Energy-Efficiency: The weights of a model are the most significant part in terms of memory. As a result, data movement during inference plays a significant role in the energy consumption of an accelerator [SCYE17]. This has been shown for both accelerators with off-chip [KOY+19] and on-chip [KHM+18a, CSC+19] memory. Besides optimizing data flow or exploiting access patterns [JHHM21], recent work tries to reduce the energy needs of memory arrays by lowering voltage. However, [GKKR17, GKB+19] demonstrate that this leads to an exponentially increasing rate of bit errors which directly affect the stored weights, leading to severe drops in accuracy [CYG+17, CSC+19, SOY+20]. To prevent accuracy drops at low voltages, [RWA+16] combines fault detection with logic to set the result of faulty data reads to zero. [MUDV17, CSC+19, CH21a] uses supply voltage boosting to ensure error-free, low-voltage operation, while [SWS+16] proposes storing critical bits in specifically robust memory cells. However, such methods incur power and area overhead. Thus, [KHM+18a, KHM+18b] and [KOY+19] propose co-design approaches combining training on profiled bit errors with hardware mitigation strategies and clever weight to memory mapping. However, these approaches are specific to the profiled chips and do not necessarily generalize well to different voltage settings. Besides low-voltage operation for energy efficiency, recent work [TSS17] also demonstrates an attack that maliciously reduces voltage and [HPG+08, HS21] show that aging also causes voltage shifts. Thus, robustness against the induced bit errors is also relevant from a security and durability perspective.

In Chapter 6, we present *random bit error training* with *weight clipping* to obtain robustness that generalizes across chips and voltages without expensive profiling or hardware mitigation strategies. The obtained energy savings from low-voltage operation can directly be combined with lower precision [PKY18]. We also study low-voltage induced bit errors in the model's activations and inputs. Concurrent work [BCC+21] also proposes a max-margin loss for improving bit error robustness in *binary* quantization. Similarly, [YBG+21] introduces another co-design approach to handle the impact of temperature variations on non-volatile memory.

Adversarial Bit Error Robustness for Security: Works such as [KDK+14, MOG+20] demonstrate methods to induce few, but targeted bit flips in memory. The impact of such attacks on the weights has recently been studied in [RHF19a]: The proposed bit flip attack (BFA) is a search-based strategy to find as few bit errors as possible such that accuracy reduces to chance level. However, the binarization approach of [HRL+20], improving robustness against *untargeted* BFA, has been shown to be ineffective against a *targeted* version of BFA [RHL+20, RHF20]. These attacks were also demonstrated on real hardware in [YRF20]. Moreover, the authors of [HRL+20] conclude that training on adversarial bit errors is *not* a promising defense. Instead, [RYL+21] resorts to binary quantization and [LRH+21] employ a checksum-based detector.

In Chapter 6, we first propose a more effective and efficient, gradient-based adversarial bit error attack, outperforming [RHF19a, RHL+20]. Subsequently, we demonstrate that *adversarial bit error training* improves robustness against both untargeted *and* targeted attacks.

2.4.2 Quantization Robustness

Network quantization [Guo18, GKD+21] is motivated by faster inference through fixed-point quantization and arithmetic [LTA16, SBS17, LDX+17], energy savings and reduced memory requirements. To avoid reduced accuracy, quantization is already applied during training [JKC+18, Kri18] or in a fine-tuning stage [GDAT18, BNS19, nvr, ner]. This generally enables significantly lower precisions down to binary weights [RORF16, CBD15]. Some works also consider quantizing activations [RORF16, HCS+17, CWV+18] and/or gradients [SFD+14, ZNZ+16, AGL+17]. We refer to [Guo18, GKD+21] for recent surveys which also discuss the importance of quantization in neural network-hardware co-design. As data movement makes up a significant part of the energy consumption of accelerators [SCYE17, CBM+20], quantization has direct impact on energy-efficiency.

In Chapter 6, we use a simple but efficient quantization scheme during training as well as for activations during inference. However, in contrast to, e.g., [ZNZ+16, JKC+18, LWL+19], we also quantize the batch or group normalization [IS15a, WH18] parameters. While [SSH15, MAA+16, MDI19, ABvB+20] study the robustness to *quantization*, the robustness of various quantization schemes against *injected bit errors* is not considered. This is surprising as our work shows that quantization impacts robustness against bit errors significantly. We further combine this with *weight clipping* during training. However, in contrast to [SSH15, ZST+18, PKY18], which clip outliers for quantization, we use weight clipping as regularization *during* training to improve robustness.

2.4.3 Weight Robustness and Fault Tolerance

Few works explicitly study the impact of (random or adversarial) weight perturbations: [WZL+20] considers robustness w.r.t. L_∞ weight perturbations, while [CSK17] studies Gaussian noise on weights. More recently, [TCC+20, THYC21b, THYC21a] started to formalize weight robustness by providing generalization bounds and some backdooring attacks also manipulate weights explicitly [JZJ+18, DS20, ZWG+19]. Besides these works, there is a more established line of work on *fault tolerance*, i.e., considering structural changes such as removed units. Early work goes back to [APS94, NSY92, CMMR94]. These approaches obtain fault-tolerant neural networks using approaches similar to adversarial training [DVK98, LHS14]. We refer to [TG17] for a comprehensive survey including more recent work. Generally, fault tolerance addresses a range of different errors, including node faults [LHS14, DVK98], hardware soft

errors [AGGC18], timing errors [DFD+15] or transient errors in general [SUC18].

In contrast to our discussion in Chapter 6, however, large rates of non-transient bit errors provoked through low-voltage operation have not been considered. Nevertheless, some of these approaches are related to ours in spirit: [DPA+14] consider inexact computation for energy-efficiency and [CM99, KMS19, HHS20] constrain weights and/or activations to limit the impact of various errors – similar to our weight clipping. However, in Chapter 6, we do not consider structural errors.

2.5 Uncertainty Estimation

Obtaining good estimates of a model's uncertainty plays a key role in almost all applications of machine learning [Gal16, KSB21, HW21], including robustness. In the following, we focus on uncertainty estimation techniques for deep neural networks, focusing particularly on directly training calibrated models. In particular, we consider two directions: First, considering out-of-distribution detection and, second, regarding *conformal prediction* [GVV98, VGS05], which aims to predict efficient (i.e., small) confidence sets with an underlying coverage guarantee independent of the data distribution and model.

2.5.1 Out-of-Distribution Calibration and Detection

Calibration [GPSW17] generally refers to the predicted confidence being close to the likelihood of being correct. Even though measuring calibration is understood to be difficult [NDZ+19, VWA+19, RCSM20, GRA+21, MDR+21], researchers are aware that deep neural networks produce overconfident predictions [YLD11, NO15, SOF+19, NDZ+19]. This becomes especially apparent in the context of out-of-distribution examples where confidence should generally be low in order to reliably detect them [LLS17, HG17a, LLLS18b, LLS18, MLW+18, VJZ+18, RLF+19, CDD19, ANKT19, CLW+20, SO20]. To address this problem [RLF+19] use likelihood ratios for detection, [AD19b] proposes a patch Gaussian data augmentation scheme, [HMKS19] augments training with a self-supervised loss or uses AugMix [HMC+20], [WBR+20] uses contrastive learning, [TCB+19] trains on interpolated examples, [vASTG20, CMS+22] use radial basis function (RBF) networks or [JHS19, XYA20] utilize generative models. These approaches do not explicitly use out-of-distribution examples during training but aim to improve calibration in general to allow better detection at test time. Instead, [LLLS18a, HMD19] explicitly enforce a uniform distribution on out-of-distribution examples supplied during training. Similarly, [HAB19] use out-of-distribution examples during training, but compute them in an adversarial way starting from random noise. These approaches have been extended in various ways [GE18, MPYW20, TMJS20, LV20, CLW+21, PRSW21, TTD+21], e.g., using contrastive learning in [WBR+20] or Bayesian approaches in [KHH20a], and is compared with a standard binary classifier in [BMAH22]. Similar to adversarial robustness, there is also a line of work on certifiable out-of-distribution detection [BMH20, MH20, MBH21].

Our *confidence-calibrated adversarial training* presented in Chapter 7 does *not* see out-of-distribution examples during training or calibration. However, biasing the model to yield low confidence on adversarial examples is similar to [LLLS18a, HMD19, HAB19] and we show that our approach improves detection of distal adversarial examples as computed in [HAB19].

2.5.2 Conformal Prediction

Conformal prediction builds on early work in [GVV98, VGS05] allowing the model to predict confidence intervals for regression [RPC19] or confidence sets for classification [DN10, HPW18, RSC20, ABJM21, CGD21]. Most of these approaches follow a *split* conformal prediction approach [LRW13] where a held-out set of exchangeable calibration examples is used after training to define these confidence sets. However, earlier variants based on cross-validation [Vov13] or jackknife (i.e., leave-one-out) [BCRT21] are also available. The assumption of exchangeability can be relaxed as discussed in [DWR18]. These approaches mostly provide *marginal coverage*, i.e., an unconditional guarantee that the true label is included with a specific probability. [Vov12, BCRT20] suggest that it is generally difficult or impossible to obtain *conditional coverage*, e.g., conditioned on the true label. However, [RSC20, FBR21] work towards empirically better conditional coverage and [SLW19] show that efficient *class-conditional coverage* is possible. [ABJM21] extends the work by [RSC20] to obtain smaller confidence sets at the expense of the obtained empirical conditional coverage and [FSJB22] allows to trade coverage for improved precision, i.e., less false positives. Conformal prediction has also been studied in the context of ensembles [YK21], allowing to perform model selection based on inefficiency while keeping coverage guarantees. The work of [BAL+21] can be seen as an extension in which a guarantee on an arbitrary, user-specified risk can be obtained. There also exist conformal methods for weakly-supervised [CGAD22], few-shot [FSJB21] or structured prediction tasks [ABC+21]. Moreover, [ABZJ21] provide differentiable private confidence sets. Finally, [Shi00, CMLZ16, TBCR19, PR21, BCL+21] propose conformal predictors for settings with distribution or label shift. We refer to [SV08, BHV14, ZFV20, AB21] for a comprehensive introduction to conformal prediction. All of these approaches come into play *post-training*.

In contrast, Chapter 8 proposes *conformal training*, a strategy to train deep neural networks jointly with a conformal predictor. This allows to reduce inefficiency and optimize arbitrary objectives defined directly on the predicted confidence sets, e.g., to influence their composition, which is difficult for standard conformal methods. After training, re-calibration with any arbitrary conformal method preserves the obtained coverage guarantee.

I

Deep Learning for 3D Shape Completion

In the first part of this thesis, we use deep learning to address a fundamental problem in 3D computer vision, a field that has received considerable attention in recent years due to an increased interest in autonomous vehicles and the availability of low-cost 3D sensors.

Specifically, Chapter 3 considers the task of (single-view) 3D shape completion from point clouds with *weak* supervision. Using only 3D bounding box annotations, we are able to complete 3D shapes such as cars from very sparse and noisy point clouds. This is achieved *without* requiring matching pairs of point clouds and completed shapes during training. Thereby, we address two limitations of prior work: the inefficient inference of data-driven approaches that perform shape alignment at test time and the required supervision of deep learning based methods.

3 Learning 3D Shape Completion under Weak Supervision

Contents

In this first chapter, we address a fundamental problem in 3D computer vision: *3D shape completion* from (single-view) sparse and noisy point clouds. Existing approaches are usually data-driven or learning-based: Data-driven approaches rely on a shape model whose parameters are optimized to fit the observations. Learning-based approaches, in contrast, avoid the expensive optimization step by learning to directly predict complete shapes from incomplete observations in a fully-supervised setting. However, full supervision is often not available in practice. In this chapter, we propose a **weakly-supervised learning-based approach to 3D shape completion** which neither requires slow optimization nor direct supervision. While we also learn a shape prior on synthetic data, we amortize, i.e., *learn*, maximum likelihood fitting using deep neural networks resulting in efficient shape completion without sacrificing accuracy. On synthetic benchmarks based on ShapeNet [CFG+15] and ModelNet [WSK+15] as well as on real data from KITTI [GLU12] and Kinect [YRM+19], we demonstrate that the proposed amortized maximum likelihood approach is able to compete with the fully supervised baseline of [DQN17] and outperforms the data-driven approach of [ESL16], while requiring less supervision and being significantly faster.

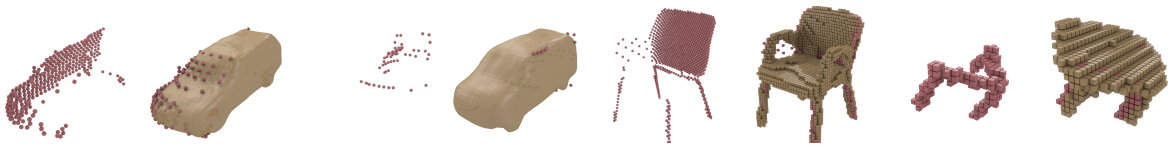
This chapter is based on [SG20]: As first author, David Stutz conducted all the experiments and was the main writer. This work also represents a significant extension of previous work [SG18b] published at CVPR 2018 and based on the master thesis [Stu17] which received the MINT Award IT 2018. See Section 3.1 for details.

The **code** for this work can be found on GitHub[1]

3.1 Introduction

3D shape perception is a long-standing and fundamental problem both in human and computer vision [Piz07, Piz10, FH13] with many applications to robotics. A large body of work focuses

[1] https://github.com/davidstutz/ijcv2018-improved-shape-completion



(a) ShapeNet (Synthetic) (b) KITTI (Real) (c) ModelNet (Synthetic) (d) Kinect (Real)

Figure 3.1: **3D Shape Completion:** Results for cars on ShapeNet [CFG+15] and KITTI [GLU12] and for chairs and tables on ModelNet [WSK+15] and Kinect [YRM+19]. Learning shape completion on real-world data is challenging due to sparse and noisy observations and missing ground truth. Occupancy grids (right) or meshes from signed distance functions (SDFs, left) at various resolutions in beige and point cloud observations in red.

on *3D reconstruction*, e.g., reconstructing objects or scenes from one or multiple views, which is an inherently ill-posed inverse problem where many configurations of shape, color, texture and lighting may result in the very same image. While the primary goal of human vision is to understand how the human visual system accomplishes such tasks, research in computer vision is focused on the task of devising 3D reconstruction systems. Generally, work by [Piz10] suggests that the constraints and priors used for 3D perception are innate and not learned. Similarly, in computer vision, cues and priors are commonly built into 3D reconstruction pipelines through explicit assumptions. Recently, however – leveraging the success of deep learning – researchers started to *learn* shape models from large collections of data, as for example ShapeNet [CFG+15]. These generative models have been used to learn how to generate, manipulate and reason about 3D shapes [GFRG16, BLRW16, SGF16, WZX+16a, WSK+15].

Here, we focus on the specific problem of inferring and completing 3D shapes based on sparse and noisy 3D point observations as illustrated in Figure 3.1. This problem occurs when only a single view of an individual object is provided or large parts of the object are occluded as common, e.g., in robotic applications. For example, autonomous vehicles are commonly equipped with LiDAR scanners providing a 360 degree point cloud of the surrounding environment in real-time. This point cloud is inherently incomplete: back and bottom of objects are typically occluded and – depending on material properties – the observations are sparse and noisy, see Figure 3.1 for an illustration. Similarly, indoor robots are generally equipped with low-cost, real-time RGB-D sensors providing noisy point clouds of the observed scene. In order to make informed decisions (e.g., for path planning and navigation), it is of utmost importance to efficiently establish a representation of the environment which is as complete as possible.

Existing approaches to 3D shape completion can be categorized into *data-driven* and learning-based methods. The former usually rely on learned shape priors and formulate shape completion as an optimization problem over the corresponding (lower-dimensional) latent space [NXS12, BCLS13, DPRR13, HSP14, LDGN15, RGT+15, ESL16, NHT+16]. These approaches have demonstrated good performance on real data, e.g., on KITTI [GLU12], but are often slow in practice.

Learning-based approaches, in contrast, assume a fully supervised setting in order to directly learn shape completion on synthetic data [SGF16, REM+16, RUBG17, SM17, DQN17, FSG17, WHY+17, VDR+17, HLH+17, YRM+19]. They offer advantages in terms of efficiency as prediction can be performed in a single forward pass, however, require full supervision during training. Unfortunately, even multiple, aggregated observations (e.g., from multiple views) will not be fully complete due to occlusion, sparse sampling of views and noise.

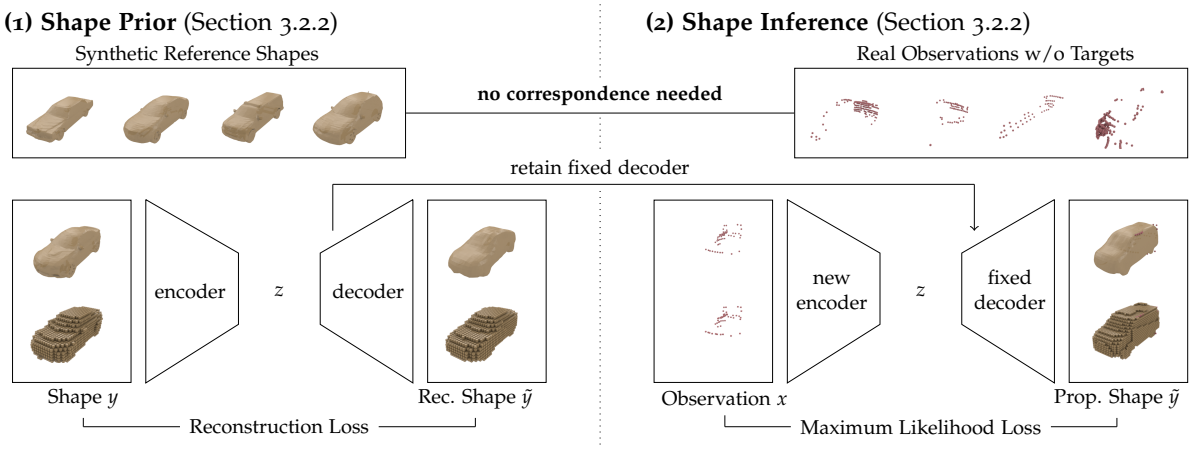


Figure 3.2: **Amortized Maximum Likelihood (AML) for 3D Shape Completion on KITTI:** (1) We train a denoising variational auto-encoder (DVAE) [KW14a, IAMB17] as shape prior on ShapeNet using occupancy grids and signed distance functions (SDFs) to represent shapes. (2) The fixed generative model, i.e., decoder, then allows us to learn shape completion using an unsupervised maximum likelihood (ML) loss by training a new recognition model, i.e., encoder. The retained generative model constraints the space of possible shapes while the ML loss aligns the predicted shape with the observations.

Contributions: In this chapter, we propose an **amortized maximum likelihood approach for 3D shape completion** (c.f. Figure 3.2) *avoiding the slow optimization* problem of data-driven approaches and the *required supervision* of learning-based approaches. Specifically, we first learn a shape prior on synthetic shapes using a (denoising) variational auto-encoder [KW14a, IAMB17]. Subsequently, 3D shape completion can be formulated as a maximum likelihood problem. However, instead of maximizing the likelihood independently for distinct observations, we follow the idea of amortized inference [GG14] and *learn* to predict the maximum likelihood solutions directly. Towards this goal, we train a new encoder which embeds the observations in the same latent space using an unsupervised maximum likelihood loss. This allows us to learn 3D shape completion in challenging real-world situations, e.g., on KITTI, and obtain sub-voxel accurate results using signed distance functions at resolutions up to 64^3 voxels. For experimental evaluation, we introduce two novel, synthetic shape completion benchmarks based on ShapeNet and ModelNet [WSK+15]. We compare our approach to the data-driven approach of [ESL16], a baseline inspired by [GAGM15] and the fully-supervised learning-based approach of [DQN17]. We additionally present experiments on real data from KITTI and Kinect [YRM+19]. Experiments show that our approach outperforms data-driven techniques and rivals learning-based techniques while significantly reducing inference time and using only a fraction of supervision.

This work is a significant extension of previous work published at CVPR 2018 [SG18b], based on the master thesis [Stu17]. Compared to [Stu17, SG18b], we improved the proposed shape completion method, the constructed datasets and present more extensive experiments. Specifically, we extended our amortized maximum likelihood approach to enforce more variety and increase visual quality significantly. On ShapeNet and ModelNet, we use volumetric fusion to obtain more detailed, watertight meshes and manually selected – per object-category – 220 high-quality models to synthesize challenging observations. We additionally increased the spatial resolution, consider two additional baselines [GAGM15, DQN17] and present results on an additional real-world dataset [YRM+19].

(a) Reference Shapes $\mathcal{Y}$ (b) Observation x_n (c) GT y_n^*

Figure 3.3: **Weakly-Supervised Shape Completion:** Given reference shapes $\mathcal{Y}$ and incomplete observations $\mathcal{X}$, we want to learn a mapping $x_n \mapsto \tilde{y}(x_n)$ such that $\tilde{y}(x_n)$ matches the *unknown* ground truth shape y_n^* as close as possible. The observations x_n are split into free space (i.e., $x_{n,i} = 0$, right) and point observations (i.e., $x_{n,i} = 1$, left). Shapes are shown in beige and observations in red.

3.2 3D Shape Completion with Amortized Maximum Likelihood

In the following, we introduce the mathematical formulation of the weakly-supervised 3D shape completion problem. Subsequently, we briefly discuss denoising variational auto-encoders (DVAEs) [KW14a, IAMB17] which we use to learn a strong shape prior that embeds a set of reference shapes in a low-dimensional latent space. Then, we formally derive our proposed amortized maximum likelihood (AML) approach. Here, we use maximum likelihood to learn an embedding of the observations within the same latent space – thereby allowing to perform shape completion. The overall approach is also illustrated in Figure 3.2.

3.2.1 Problem Formulation

In a supervised setting, the task of 3D shape completion can be described as follows: Given a set of incomplete observations $\mathcal{X} = \{x_n\}_{n=1}^N \subseteq \mathbb{R}^R$ and corresponding ground truth shapes $\mathcal{Y}^* = \{y_n^*\}_{n=1}^N \subseteq \mathbb{R}^R$, learn a mapping $x_n \mapsto y_n^*$ that is able to generalize to previously unseen observations and possibly across object categories. We assume $\mathbb{R}^R$ to be a suitable representation of observations and shapes. In practice, we resort to occupancy grids and signed distance functions (SDFs) defined on regular grids, i.e., $x_n, y_n^* \in \mathbb{R}^{H \times W \times D} \simeq \mathbb{R}^R$. Specifically, occupancy grids indicate occupied space, i.e., voxel $y_{n,i}^* = 1$ if and only if the voxel lies on or inside the shape's surface. To represent shapes with sub-voxel accuracy, SDFs hold the distance of each voxel's center to the surface; for voxels inside the shape's surface, we use negative sign. Finally, for the (incomplete) observations, we write $x_n \in \{0, 1, \perp\}^R$ to make missing information explicit; in particular, $x_{n,i} = \perp$ corresponds to unobserved voxels, while $x_{n,i} = 1$ and $x_{n,i} = 0$ correspond to occupied and unoccupied voxels, respectively.

On real data, e.g., KITTI [GLU12], supervised learning is often not possible as obtaining ground truth annotations is labor-intensive [MG15, XKSG16]. Therefore, we target a weakly-supervised variant of the problem instead: Given observations $\mathcal{X}$ and reference shapes $\mathcal{Y} = \{y_m\}_{m=1}^M \subseteq \mathbb{R}^R$ both of the same, known object category, learn a mapping $x_n \mapsto \tilde{y}(x_n)$ such that the predicted shape $\tilde{y}(x_n)$ matches the unknown ground truth shape y_n^* as close as possible – or, in practice, the sparse observation x_n while being plausible considering the set of reference shapes, c.f. Figure 3.3. Here, supervision is provided in the form of the known object category. Alternatively, the reference shapes $\mathcal{Y}$ can also include multiple object categories resulting in an even weaker notion of supervision as the correspondence between observations and object categories is unknown. Except for the object categories, however, the set of reference shapes $\mathcal{Y}$, and its size M, is completely independent of the set of observations $\mathcal{X}$, and its size N, as also highlighted in Figure 3.2. On real data, e.g., KITTI, we additionally assume the object

locations to be given in the form of 3D bounding boxes in order to extract the corresponding observations $\mathcal{X}$. In practice, the reference shapes $\mathcal{Y}$ are derived from watertight, triangular meshes, e.g., from ShapeNet [CFG+15] or ModelNet [WSK+15].

3.2.2 Shape Prior

We approach the weakly-supervised shape completion problem by first learning a shape prior using a denoising variational auto-encoder (DVAE). Later, this prior constrains shape inference (see Section 3.2.3) to predict reasonable shapes. In the following, we briefly discuss the standard variational auto-encoder (VAE), as introduced in [KW14a], as well as its denoising extension described in [IAMB17].

Variational Auto-Encoder (VAE): We propose to use the provided reference shapes $\mathcal{Y}$ to learn a generative model of possible 3D shapes over a low-dimensional latent space $\mathcal{Z} = \mathbb{R}^Q$, i.e., $Q \ll R$. In the framework of VAEs, the joint distribution $p(y, z)$ of shapes y and latent codes z decomposes into $p(y|z)p(z)$ with $p(z)$ being a unit Gaussian $\mathcal{N}(z; 0, I_Q)$ and $I_Q \in \mathbb{R}^{R\times R}$ being the identity matrix. This decomposition allows sampling $z \sim p(z)$ and $y \sim p(y|z)$ to generate random shapes. For training, however, we additionally need to approximate the posterior $p(z|y)$. To this end, the so-called recognition model $q(z|y) \approx p(z|y)$ takes the form

$$q(z|y) = \mathcal{N}(z; \mu(y), \mathrm{diag}(\sigma^2(y))) \tag{3.1}$$

where $\mu(y), \sigma^2(y) \in \mathbb{R}^Q$ are predicted using the encoder neural network. The generative model $p(y|z)$ decomposes over voxels y_i and the corresponding probabilities $p(y_i|z)$ are represented using Bernoulli distributions for occupancy grids or Gaussian distributions for SDFs:

$$p(y_i|z) = \mathrm{Ber}(y_i; \theta_i(z)) \quad \text{or} \quad p(y_i|z) = \mathcal{N}(y_i; \mu_i(z), \sigma^2). \tag{3.2}$$

In both cases, the parameters, i.e., $\theta_i(z)$ or $\mu_i(z)$, are predicted using the decoder neural network. For SDFs, we explicitly set σ^2 to be constant (see Section 3.3.3). Then, σ^2 merely scales the corresponding loss, thereby implicitly defining the importance of accurate SDFs relative to occupancy grids as described below.

In the framework of variational inference, the parameters of the encoder and the decoder neural networks are found by maximizing the likelihood $p(y)$. In practice, the likelihood is usually intractable and the evidence lower bound is maximized instead, see [KW14a, BKM16]. This results in the following loss to be minimized:

$$\mathcal{L}_{\mathrm{VAE}}(w) = -\mathbb{E}_{q(z|y)}[\log p(y|z)] + \mathrm{KL}(q(z|y)|p(z)). \tag{3.3}$$

Here, w are the weights of the encoder and decoder hidden in the recognition model $q(z|y)$ and the generative model $p(y|z)$, respectively. The Kullback-Leibler divergence KL can be computed analytically as described in the appendix of [KW14a]. The negative log-likelihood $-\log p(y|z)$ corresponds to a binary cross-entropy error for occupancy grids and a scaled sum-of-squared error for SDFs. The loss $\mathcal{L}_{\mathrm{VAE}}$ is minimized using stochastic gradient descent (SGD) by approximating the expectation using samples:

$$-\mathbb{E}_{q(z|y)}[\log p(y|z)] \approx -\sum_{l=1}^{L} \log p(y|z^{(l)}) \tag{3.4}$$

The required samples $z^{(l)} \sim q(z|y)$ are computed using the so-called reparameterization trick,

$$z^{(l)} = \mu(y) + \epsilon^{(l)}\sigma(y) \quad \text{with} \quad \epsilon^{(l)} \sim \mathcal{N}(\epsilon; 0, I_Q), \tag{3.5}$$

in order to make $\mathcal{L}_{\text{VAE}}$, specifically the sampling process, differentiable. In practice, we found $L = 1$ samples to be sufficient – which conforms with results by [KW14a]. At test time, the sampling process $z \sim q(z|y)$ is replaced by the predicted mean $\mu(y)$. Overall, the standard VAE allows us to embed the reference shapes in a low-dimensional latent space. In practice, however, the learned prior might still include unreasonable shapes.

Denoising VAE (DVAE): In order to avoid inappropriate shapes to be included in our shape prior, we consider a denoising variant of the VAE allowing to obtain a tighter bound on the likelihood $p(y)$. More specifically, a corruption process $y' \sim p(y'|y)$ is considered and the corresponding evidence lower bound results in the following loss:

$$\mathcal{L}_{\text{DVAE}}(w) = -\mathbb{E}_{q(z|y')}[\log p(y|z)] + \text{KL}(q(z|y')|p(z)). \tag{3.6}$$

Note that the reconstruction error $-\log p(y|z)$ is still computed with respect to the uncorrupted shape y while z, in contrast to Equation (3.3), is sampled conditioned on the corrupted shape y'. In practice, the corruption process $p(y'|y)$ is modeled using Bernoulli noise for occupancy grids and Gaussian noise for SDFs. In experiments, we found DVAEs to learn more robust latent spaces – meaning the prior is less likely to contain unreasonable shapes. In the following, we always use DVAEs as shape priors.

3.2.3 Shape Inference

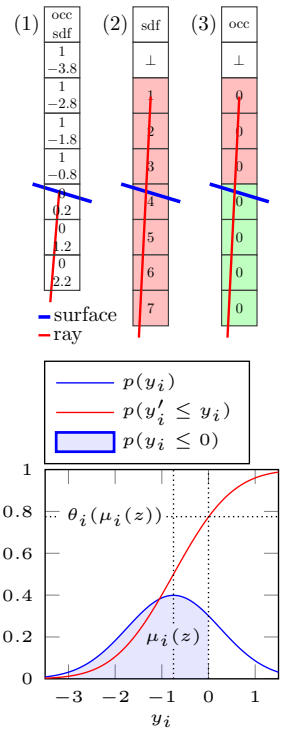
After learning the shape prior, defining the joint distribution $p(y, z)$ of shapes y and latent codes z as product of generative model $p(y|z)$ and prior $p(z)$, shape completion can be formulated as a maximum likelihood (ML) problem for $p(y, z)$ over the lower-dimensional latent space $\mathcal{Z} = \mathbb{R}^Q$. The corresponding negative log-likelihood $-\log p(y, z)$ to be minimized can be written as

$$\mathcal{L}_{\text{ML}}(z) = -\sum_{x_i \neq \perp} \log p(y_i = x_i|z) - \log p(z). \tag{3.7}$$

As the prior $p(z)$ is Gaussian, the negative log-probability $-\log p(z)$ is proportional to $\|z\|_2^2$ and constrains the problem to likely, i.e., reasonable, shapes with respect to the shape prior. As before, the generative model $p(y|z)$ decomposes over voxels. Here, we can only consider actually observed voxels $x_i \neq \perp$. We assume that the learned shape prior can complete the remaining, unobserved voxels $x_i = \perp$. Instead of solving Equation (3.7) for each observation $x \in \mathcal{X}$ independently, however, we follow the idea of amortized inference [GG14] and train a new encoder $z(x; w)$ to *learn* ML. To this end, we keep the generative model $p(y|z)$ fixed and train only the weights w of the new encoder $z(x; w)$ using the ML objective as loss:

$$\mathcal{L}_{\text{dAML}}(w) = -\sum_{x_i \neq \perp} \log p(y_i = x_i|z(x; w)) - \lambda \log p(z(x; w)). \tag{3.8}$$

Here, λ controls the importance of the shape prior. The exact form of the probabilities $p(y_i = x_i|z)$ depends on the used shape representation. For occupancy grids, this term results in a cross-entropy error as both the predicted voxels y_i and the observations x_i are, for $x_i \neq \perp$, binary. For SDFs, however, the term is not well-defined as $p(y_i|z)$ is modeled with a continuous Gaussian distribution, while the observations x_i are binary. As solution, we could compute (signed) distance values along the rays corresponding to observed points (e.g., following [SKC13]) in order to obtain continuous observations $x_i \in \mathbb{R}$ for $x_i \neq \perp$. However, as illustrated in Figure 3.4, noisy observations cause the distance values along the whole ray to be invalid.



(a) **Top: Problem with SDF Observations:** Illustration of a ray (red line) correctly hitting a surface (blue line) causing the (signed) distance values and occupancy values computed for voxels along the ray to be correct (c.f. (1)). A noisy ray, however, causes all voxels along the ray to be assigned incorrect distance values (marked red) w.r.t. to the true surface (blue line) because the ray ends far behind the actual surface (c.f. (2)). When using occupancy only, in contrast, only the voxels behind the surface are assigned invalid occupancy states (marked red); the remaining voxels are labeled correctly (marked green, c.f. (3)).

(b) **Bottom: Proposed Gaussian-to-Bernoulli Transformation:** For $p(y_i) := p(y_i|z) = \mathcal{N}(y_i; \mu_i(z), \sigma^2)$ (blue), we illustrate the transformation discussed in Section 3.2.3 allowing to use the binary observations x_i (for $x_i \neq \perp$) to supervise the SDF predictions. This is achieved by transforming the predicted Gaussian distribution to a Bernoulli distribution with occupancy probability $\theta_i(\mu_i(z)) = p(y_i \leq 0)$ (blue area).

Figure 3.4: **Handling Noisy Data:** Illustration of the problems with SDF observations and noisy data as obtained on KITTI (a) as well as our proposed Gaussian-to-Bernoulli transformation allowing to use binary, i.e., occupancy, observations to supervise SDF predictions (b).

This can partly be avoided when relying only on occupancy to represent the observations. In this case, free space (c.f. Figure 3.3) observations are partly correct even though observed points may lie within the corresponding shapes.

For making SDFs tractable (i.e., to predict sub-voxel accurate, visually smooth and appealing shapes, see Section 3.3.5) while using binary observations, we propose to define $p(y_i = x_i|z)$ through a simple transformation. In particular, as $p(y_i|z)$ is modeled using a Gaussian distribution $\mathcal{N}(y_i; \mu_i(z), \sigma^2)$ where $\mu_i(z)$ is predicted using the fixed decoder (σ^2 is constant), and x_i is binary (for $x_i \neq \perp$), we introduce a mapping $\theta_i(\mu_i(z))$ transforming the predicted mean SDF value to an occupancy probability $\theta_i(\mu_i(z))$:

$$p(y_i = x_i|z) = \text{Ber}(y_i = x_i; \theta_i(\mu_i(z))) \tag{3.9}$$

As, by construction (see Section 3.2.1), occupied voxels have negative sign or value zero in the SDF, we can derive the occupancy probability $\theta_i(\mu_i(z))$ as the probability of a non-positive distance:

$$\theta_i(\mu_i(z)) = \mathcal{N}(y_i \leq 0; \mu_i(z), \sigma^2) = \frac{1}{2}\left(1 + \text{erf}\left(\frac{-\mu_i(z)}{\sigma\sqrt{\pi}}\right)\right). \tag{3.10}$$

Here, erf is the error function which, in practice, can be approximated following [Abr74]. Equation (3.10) is illustrated in Figure 3.4 where the occupancy probability $\theta_i(\mu_i(z))$ is computed as the area under the Gaussian bell curve for $y_i \leq 0$. This per-voxel transformation can easily be implemented as non-linear layer and its derivative w.r.t. $\mu_i(z)$ is, by construction, a Gaussian. Note that the transformation is correct, not approximate, based on our model assumptions and the definitions in Section 3.2.1. Overall, this transformation allows us to easily minimize Equation (3.8) for both occupancy grids and SDFs using binary observations. The obtained encoder embeds the observations in the latent shape space to perform shape completion.

3.2.4 Practical Considerations

Encouraging Variety: So far, our AML formulation assumes a deterministic encoder $z(x, w)$ which predicts, given the observation x, a single code z corresponding to a completed shape. A closer look at Equation (3.8), however, reveals an unwanted problem: the data term scales with the number of observations, i.e., $|\{x_i \neq \bot\}|$, while the regularization term stays constant – with less observations, the regularizer gains in importance leading to limited variety in the predicted shapes because $z(x; w)$ tends towards zero.

In order to encourage variety, we draw inspiration from the VAE shape prior. Specifically, we use a probabilistic recognition model

$$q(z|x) = \mathcal{N}(z; \mu(x), \mathrm{diag}(\sigma^2(x))) \tag{3.11}$$

(c.f. see Equation (3.1)) and replace the negative log-likelihood $-\log p(z)$ with the corresponding Kullback-Leibler divergence $\mathrm{KL}(q(z|x)|p(z))$ with $p(z) = \mathcal{N}(z; 0, I_Q)$. Intuitively, this makes sure that the encoder's predictions "cover" the prior distribution – thereby enforcing variety. Mathematically, the resulting loss

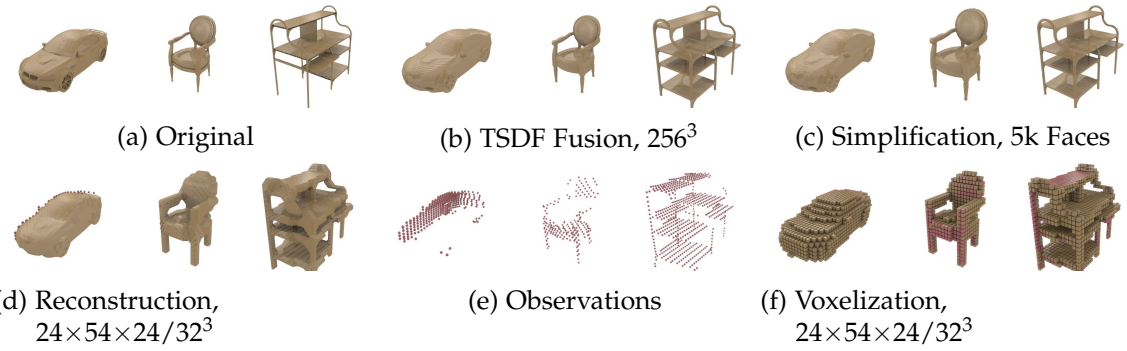
$$\mathcal{L}_{\mathrm{AML}}(w) = -\mathbb{E}_{q(z|x)}\left[\sum_{x_i \neq \bot} \log p(y_i = x_i|z)\right] + \lambda \mathrm{KL}(q(z|x)p(z)) \tag{3.12}$$

can be interpreted as the result of maximizing the evidence lower bound of a model with observation process $p(x|y)$ (analogously to the corruption process $p(y'|y)$ for DVAEs in [IAMB17] and Section 3.2.2). The expectation is approximated using samples following the reparameterization trick in Equation (3.5) and, during testing, the sampling process $z \sim q(z|x)$ is replaced by the mean prediction $\mu(x)$. In practice, we find that Equation (3.12) improves visual quality of the completed shapes. We compare this AML model to its deterministic variant dAML in Section 3.3.5.

Handling Noise: Another problem of our AML formulation concerns noise. On KITTI, for example, specular or transparent surfaces cause invalid observations – laser rays traversing through these surfaces cause observations to lie within shapes or not get reflected. However, our AML framework assumes deterministic, i.e., trustworthy, observations – as can be seen in the reconstruction error in Equation (3.12). Therefore, we introduce per-voxel weights κ_i computed using the reference shapes $\mathcal{Y} = \{y_m\}_{m=1}^M$:

$$\kappa_i = 1 - \left(\frac{1}{M}\sum_{m=1}^{M} y_{m,i}\right) \in [0, 1] \tag{3.13}$$

where $y_{m,i} = 1$ if and only if the corresponding voxel is occupied. Applied to observations $x_i = 0$, these are trusted less if they are unlikely under the shape prior. Note that for point observations, i.e., $x_i = 1$, this is not necessary as we explicitly consider "filled" shapes (see Section 3.3.1). This can also be interpreted as imposing an additional mean shape prior on the predicted shapes with respect to the observed free space. In addition, we use a corruption process $p(x'|x)$ consisting of Bernoulli and Gaussian noise during training (analogously to the DVAE shape prior).



(a) Original (b) TSDF Fusion, 256^3 (c) Simplification, 5k Faces

(d) Reconstruction, $24\times54\times24/32^3$ (e) Observations (f) Voxelization, $24\times54\times24/32^3$

Figure 3.5: **ShapeNet and ModelNet Data Generation Pipeline:** On ShapeNet and ModelNet we illustrate: **(a)** samples from the original datasets; **(b)** fused watertight meshes from TSDF fusion at 256^3 voxels resolution using [RUBG17]; **(c)** simplified meshes ($5k$ faces); **(d)** marching cubes [LC87] reconstructions from the SDFs computed from (c) (resolutions $24\times54\times24$ and 32^3 voxels; note that steps (b) and (c) are necessary to derive exact SDFs); **(e)** observations obtained by projection into a single view; and **(f)** voxelized observations and shapes. Shapes (meshes and occupancy grids) in beige and observations in red.

3.3 Experiments

We present experiments on multiple synthetic and real datasets, see Section 3.3.1. Before presenting quantitative and qualitative experimental results in Section 3.3.5, including a discussion of failure cases and runtime, we also detail used evaluation metrics in Section 3.3.2, our training procedure and the used architecture in Section 3.3.3 and introduce both data-driven and learning-based baselines in Section 3.3.4

3.3.1 Data

We briefly introduce our synthetic shape completion benchmarks, derived from ShapeNet [CFG+15] and ModelNet [WSK+15] (c.f. Figure 3.5), and our data preparation for KITTI [GLU12] and Kinect [YRM+19] (c.f. Figure 3.6). Table 3.1 summarizes key statistics of these benchmarks including the level of supervision computed as the fraction of observed voxels, i.e. $|\{x_{n,i}\neq\perp\}|/HWD$, averaged over observations x_n.

ShapeNet: We utilize the truncated SDF (TSDF) fusion approach of [RUBG17] to obtain watertight versions of the provided car shapes allowing to reliably and efficiently compute occupancy grids and SDFs. Specifically, we use 100 depth maps of 640×640 pixels resolution, distributed uniformly on the sphere around the shape, and perform TSDF fusion at a resolution of 256^3 voxels. Detailed watertight meshes, without inner structures, can then be extracted using marching cubes [LC87] and simplified to 5k faces using MeshLab's quadratic simplification algorithm [CCC+08], see Figure 3.5a to c. Finally, we manually selected 220 shapes from this collection, removing exotic cars, unwanted configurations, or shapes with large holes (e.g., missing floors or open windows).

The shapes are split into $|\mathcal{Y}| = 100$ reference shapes, $|\mathcal{Y}^*| = 100$ shapes for training the inference model, and 20 test shapes. We randomly perturb rotation and scaling to obtain 5 variants of each shape, voxelize them using triangle-voxel intersections and subsequently "fill" the obtained volumes using a connected components algorithm [JOP+01]. For computing SDFs

(a) KITTI, Point Clouds (b) Kinect, Occupancy Grids

Figure 3.6: **Extracted KITTI and Kinect Data:** For KITTI, we show observed points in red and the accumulated, partial ground truth in green. Note that for the first example ground truth is not available due to missing past/future observations. For Kinect, we show observations in red and ElasticFusion [WLS+15] ground truth in beige. Note that the objects are rotated and not aligned as in ModelNet (c.f. Figure 3.5).

we use SDFGen[2]. We use three different resolutions: $H \times W \times D = 24 \times 54 \times 24$, $32 \times 72 \times 32$ and $48 \times 108 \times 48$ voxels. Examples are shown in Figure 3.5d to f.

Finally, we use the OpenGL renderer of [GG15] to obtain 10 depth maps per shape. The incomplete observations $\mathcal{X}$ are obtained by re-projecting them into 3D and marking voxels with at least one point as occupied and voxels between occupied voxels and the camera center as free space. We obtain more dense point clouds at 48×64 pixels resolution and sparser point clouds using depth maps of 24×32 pixels resolution. For the latter, more challenging case we also add exponentially distributed noise (with rate parameter 70) to the depth values, or randomly (with probability 0.075) set them to the maximum depth to simulate the deficiencies of point clouds captured with real sensors, e.g., on KITTI. These two variants are denoted **SN-clean** and **SN-noisy**. The obtained observations are illustrated in Figure 3.5e.

KITTI: We extract observations from KITTI's Velodyne point clouds using the provided ground truth 3D bounding boxes to avoid the inaccuracies of 3D object detectors (train/test split by [CKZ+18]). As the 3D bounding boxes in KITTI fit very tightly, we first padded them by factor 0.25 on all sides. Afterwards, the observed points are voxelized into voxel grids of size $H \times W \times D = 24 \times 54 \times 24$, $32 \times 72 \times 32$ and $48 \times 108 \times 48$ voxels. To avoid taking points from the street, nearby walls, vegetation or other objects into account, we only consider those points lying within the original (i.e., not padded) bounding box. Finally, free space is computed using ray tracing as described above. We filter all observations to ensure that each observation contains a minimum of 50 observations. For the bounding boxes in the test set, we additionally generated partial ground truth by accumulating the 3D point clouds of 10 future and 10 past frames around each observation. Examples are shown in Figure 3.6.

ModelNet: We use ModelNet10, comprising 10 popular object categories (bathtub, bed, chair, desk, dresser, monitor, night stand, table, toilet) and select, for each category, the first 200 and 20 shapes from the provided training and test sets. Then, we follow the pipeline outlined in Figure 3.5, as on ShapeNet, using 10 random variants per shape. Due to thin structures, however, SDF computation does not work well (especially for low resolution, e.g., 32^3 voxels). Therefore, we approximate the SDFs using a 3D distance transform on the occupancy grids. Our experiments are conducted at a resolution of $H \times W \times D = 32^3$, 48^3 and 64^3 voxels. Given the increased difficulty, we use a resolution of 64^2, 96^2 and 128^2 pixels for the observation generating depth maps. In our experiments, we consider bathtubs, chairs, desks and tables individually, as well as all 10 categories together (resulting in 100k views overall). For Kinect, we additionally used a dataset of rotated chairs and tables aligned with Kinect's ground plane.

Kinect: Yang et al. [YRM+19] provide Kinect scans of various chairs and tables. They provide both single-view observations and ground truth from ElasticFusion [WLS+15] as occupancy

[2]https://github.com/christopherbatty/SDFGen.

	Synthetic		Real	
	SN-clean/ -noisy	ModelNet	KITTI	Kinect
Training/Test Sets #Shapes for Shape Prior, #Views for Shape Inference				
#Shapes	500/100	1000/200	–	–
#Views	5000/1000	10000/2000	8442/9194	30/10
Observed Voxels in % ($< 5\%$) & Resolutions Low = $24\times54\times24/32^3$; Medium = $32\times72\times32/48^3$; High = $48\times108\times48/64^3$				
Low	7.66/3.86	9.71	6.79	**0.87**
Medium	6.1/**2.13**	8.74	5.24	–
High	**2.78/0.93**	8.28	**3.44**	–

Table 3.1: **Dataset Statistics:** We report the number of (rotated and scaled) meshes, used as reference shapes, and the resulting number of observations (i.e., views, 10 per shape). We also report the average fraction of observed voxels, i.e., $|\{x_i \neq \perp\}|/HWD$. For ModelNet, we exemplarily report statistics for chairs, and for Kinect, we report statistics for tables.

grids. However, the ground truth is not fully accurate, and only 40 views are provided per object category. Still, the objects have been segmented to remove clutter and are appropriate for experiments in conjunction with ModelNet10. Unfortunately, Yang et al. do not provide SDFs; again, we use 3D distance transforms as approximation. Additionally, the observations do not indicate free space, and we were required to guess an appropriate ground plane. For our experiments, we use 30 views for training and 10 views for testing, see Figure 3.6 for examples.

3.3.2 Evaluation

For occupancy grids, we use Hamming distance (Ham) and intersection-over-union (IoU) between the (thresholded) predictions and the ground truth. Note that lower Ham is better, while lower IoU is worse. For SDFs, we consider a mesh-to-mesh distance on ShapeNet and a mesh-to-point distance on KITTI. We follow [JDV+14] and consider accuracy (Acc) and completeness (Comp). To measure Acc, we uniformly sample roughly 10k points on the reconstructed mesh and average their distance to the target mesh. Analogously, Comp is the distance from the target mesh (or the ground truth points on KITTI) to the reconstructed mesh. Note that for both Acc and Comp, lower is better. On ShapeNet and ModelNet, we report both Acc and Comp in voxels, i.e., in multiples of the voxel edge length (i.e., in [vx], as we do not know the absolute scale of the models); on KITTI, we report Comp in meters (i.e., in [m]).

3.3.3 Architectures and Training

As depicted in Figure 3.7, our network architectures are kept simple and shallow. Considering a resolution of $24\times54\times24$ voxels on ShapeNet and KITTI, the encoder comprises three stages, each consisting of two convolutional layers (followed by ReLU activations and batch normalization [IS15a]) and max pooling. The decoder mirrors the encoder, replacing max pooling by nearest neighbor upsampling. We consistently use 3^3 convolutional kernels. We use a latent space of size $Q = 10$ and predict occupancy using Sigmoid activations.

We found that the shape representation has a significant impact on training. Specifically, learning both occupancy grids and SDFs works better compared to training on SDFs only. Additionally, following prior art in single image depth prediction [EPF14, EF15, LRB+16], we consider log-transformed, truncated SDFs (logTSDFs) for training: given a signed distance y_i, we compute $\text{sign}(y_i)\log(1+\min(5,|y_i|))$ as the corresponding log-transformed, truncated signed distance. TSDFs are commonly used in the literature [CL96, NIH+11, ESL16, RUBG17,

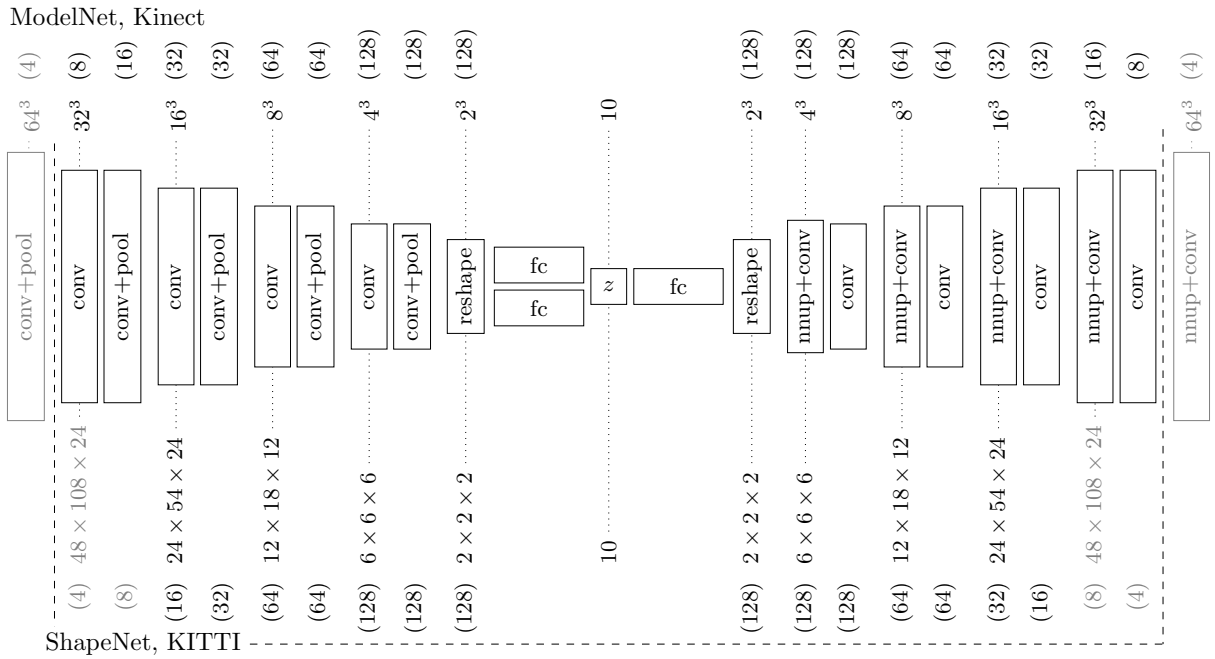
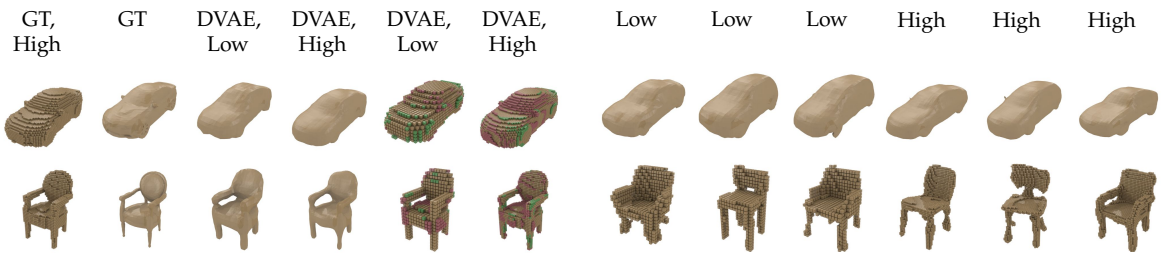


Figure 3.7: **Network Architectures:** We use different resolutions for ShapeNet and KITTI as well as ModelNet and Kinect (bottom and top, respectively). In both cases, architectures for higher resolutions employ one additional stage in the en- and decoder (in gray). Each convolutional layer is followed by ReLU activations and batch normalization [IS15a]. The window sizes for max pooling and nearest-neighbor upsampling can be derived from the context and the number of channels are given in parentheses.

DQN17] and the logarithmic transformation additionally increases the relative importance of values around the surfaces (i.e., around the zero crossing).

For training, we combine occupancy grids and logTSDFs in separate feature channels and randomly translate both by up to 3 voxels per axis. Additionally, we use Bernoulli noise (probability 0.1) and Gaussian noise (variance 0.05). We use Adam [KB15a], a batch size of 16 and the initialization scheme of [GB10a]. The shape prior is trained for 3000 to 4000 epochs with an initial learning rate of 10^{-4} which is decayed by 0.925 every 215 iterations until a minimum of 10^{-16} has been reached. In addition, weight decay (10^{-4}) is applied. For shape inference, training takes 30 to 50 epochs, and an initial learning rate of 10^{-4} is decayed by 0.9 every 215 iterations. For our learning-based baselines (see Section 3.3.4) we require between 300 and 400 epochs using the same training procedure as for the shape prior. On the Kinect dataset, where only 30 training examples are available, we used 5000 epochs. We use $\log \sigma^2 = -2$ as an empirically found trade-off between accuracy of the reconstructed SDFs and ease of training – significantly lower $\log \sigma^2$ may lead to difficulties during training, including divergence. On ShapeNet, ModelNet and Kinect, the weight λ of the Kullback-Leibler divergence KL (for both DVAE and (d)AML) was empirically determined to be $\lambda = 2, 2.5, 3$ for low, medium and high resolution, respectively. On KITTI, we use $\lambda = 1$ for all resolutions. In practice, λ controls the trade-off between diversity (low λ) and quality (high λ) of the completed shapes. In addition, we reduce the weight in free space areas to one fourth on SN-noisy and KITTI to balance between occupied and free space. We implemented our networks in Torch [CKF11].



(a) Reconstructions (b) Random Samples

Figure 3.8: **DVAE Shape Prior:** Reconstructions and random samples on ShapeNet and ModelNet at multiple resolutions (c.f. Table 3.1). False negative and false positive voxels are shown in green and red. Our DVAE shape prior provides high-quality reconstructions and meaningful random samples across resolutions.

3.3.4 Baselines

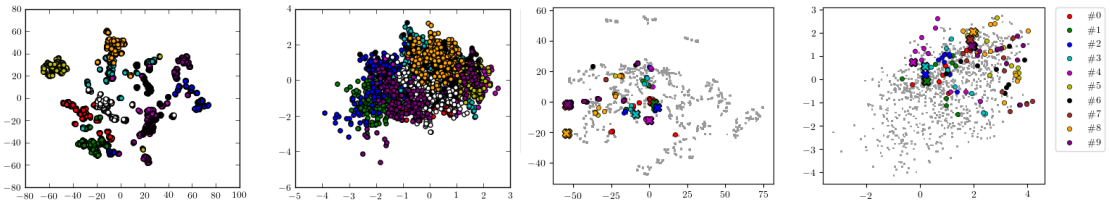
Data-Driven Approaches: We consider [ESL16] and [GAGM15] as data-driven baselines. Additionally, we consider regular maximum likelihood (ML). [ESL16] – referred to as Eng16 – use a principal component analysis shape prior trained on a manually selected set of car models[3]. Shape completion is posed as optimization problem considering both shape and pose. The pre-trained shape prior provided by [ESL16] assumes a ground plane which is, according to KITTI's LiDAR data, fixed at $1m$ height. Thus, we don't need to optimize pose on KITTI as we use the ground truth bounding boxes. On ShapeNet, in contrast, we need to optimize both pose and shape to deal with the random rotations in SN-clean and SN-noisy.

Inspired by [GAGM15], we also consider a shape retrieval and fitting baseline. Specifically, we perform iterative closest point (ICP) [BM92] fitting on all training shapes and subsequently select the best-fitting one. To this end, we uniformly sample 1Mio points on the training shapes, and perform point-to-point ICP[4] for 100 iterations using $\begin{bmatrix} R & t \end{bmatrix} = \begin{bmatrix} I_3 & 0 \end{bmatrix}$ as initialization. On the training set, we verified that this approach is always able to retrieve the perfect shape.

Finally, we consider a simple ML baseline iteratively minimizing Equation (3.7) using stochastic gradient descent (SGD). This baseline is similar to Eng16, however, like ours it is bound to the voxel grid. Per example, we allow a maximum of 5000 iterations, starting with latent code $z = 0$, learning rate 0.05 and momentum 0.5 (decayed every 50 iterations at rate 0.85 and 1.0 until 10^{-5} and 0.9 have been reached).

Learning-Based Approaches: Learning-based approaches usually employ an encoder-decoder architecture to directly learn a mapping from observations x_n to ground truth shapes y_n^* in a fully supervised setting [WHY$^+$17, VDR$^+$17, YWW$^+$17, DQN17, YRM$^+$19]. While existing architectures differ slightly, they usually rely on a U-net architecture [RFB15, ÇAL$^+$16]. Here, we use the approach of [DQN17][5] – referred to as Dai17 – as a representative baseline for this class of approaches. To run Dai17 on ModelNet, we added one convolutional stage in the en- and decoder for larger resolutions. On ShapeNet and KITTI, we needed to adapt the convolutional strides to fit the corresponding resolutions. In addition, we consider a custom learning-based baseline which uses the architecture of our DVAE shape prior, c.f. Figure 3.7. In contrast to [DQN17], this baseline is also limited by the low-dimensional ($Q = 10$) bottleneck as it does not use skip connections.

[3] https://github.com/VisualComputingInstitute/ShapePriors_GCPR16
[4] http://www.cvlibs.net/software/libicp/.
[5] https://github.com/angeladai/cnncomplete



(a) **DVAE t-SNE** (b) **DVAE Projection** (c) **AML t-SNE** (d) **AML Projection**

Figure 3.9: **Learned Latent Spaces:** In (a) and (b), we show a t-SNE [vdMH08] visualization and a two-dimensional projection of the DVAE latent space on ModelNet10. The DVAE is able to separate the ten object categories. In (c) and (d), we show a t-SNE visualization and a projection of the latent space corresponding to our learned AML model on SN-clean. We randomly picked 10 ground truth shapes, "x", and the corresponding observations (10 per shape), points (gray pixels indicate remaining shapes/observations). AML is able to associate observations with the corresponding ground truth shapes under weak supervision.

3.3.5 Experimental Evaluation

Quantitative results are summarized in Table 3.2 (ShapeNet and KITTI) and 3.3 (ModelNet). Qualitative results for the shape prior are shown in Figure 3.8 and 3.9, while shape completion results are shown in Figure 3.11 for ShapeNet and ModelNet and 3.14 for KITTI and Kinect.

Latent Space Dimensionality: Regarding our DVAE shape prior, we found the dimensionality Q to be of crucial importance as it defines the trade-off between reconstruction accuracy and random sample quality (i.e., the quality of the generative model). A higher-dimensional latent space usually results in higher-quality reconstructions but also imposes the difficulty of randomly generating meaningful shapes. Across all datasets, we found $Q = 10$ to be suitable – which is significantly smaller compared to related work: 35 in [LYF17], 6912 in [SGF16], 200 for [WZX+16a, SM17] or 64 in [GFRG16]. Still, we are able to obtain visually appealing results. Finally, in Figure 3.8 we show qualitative results, illustrating good reconstruction performance and reasonable random samples across resolutions.

Figure 3.9 shows a t-SNE [vdMH08] visualization as well as a projection of the $Q = 10$ dimensional latent space, color coding the 10 object categories of ModelNet10. The DVAE clusters the object categories within the support region of the unit Gaussian. In the t-SNE visualization, we additionally see ambiguities arising in ModelNet10, e.g., night stands and dressers often look indistinguishable while monitors are very dissimilar to all other categories. Overall, these findings support our decision to use a DVAE with $Q = 10$ as shape prior.

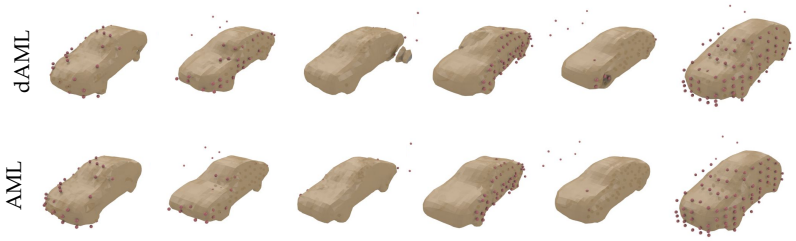


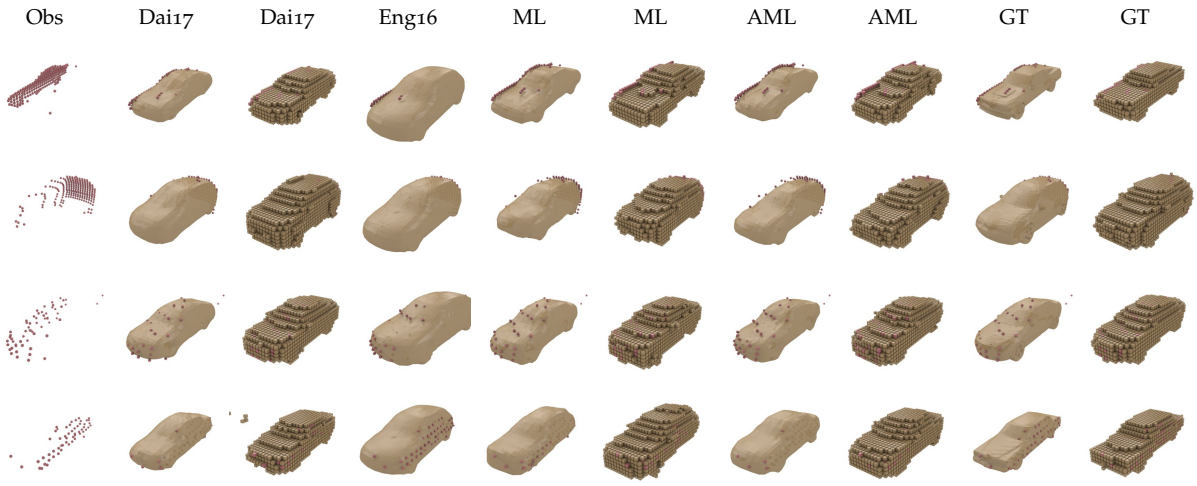
Figure 3.10: **Comparison of AML and dAML:** Our deterministic variant, dAML, suffers from inferior results, including artifacts and less details. Predicted shapes in beige and observations in red at low resolution (24×54×24 voxels).

Supervision in %	Method	SN-clean Ham↓	SN-clean IoU↑	SN-clean Acc [vx] ↓	SN-clean Comp [vx] ↓	SN-noisy Ham↓	SN-noisy IoU↑	SN-noisy Acc [vx] ↓	SN-noisy Comp [vx] ↓	KITTI Comp [m] ↓
Low Resolution: $24 \times 54 \times 24$ voxels; * independent of resolution										
(shape prior)	DVAE	0.019	0.885	0.283	0.527	(same shape prior as on SN-clean)				
100	[DQN17] (Dai17)	**0.021**	**0.872**	**0.321**	**0.564**	**0.027**	**0.836**	**0.391**	**0.633**	0.128
	Sup	0.026	0.841	0.409	0.607	0.028	0.833	0.407	0.637	**0.091**
< 7.7	Naïve	0.067	0.596	0.999	1.335	0.064	0.609	0.941	1.29	–
	Mean	0.052	0.697	0.79	0.938	0.052	0.696	0.79	0.938	–
	ML	0.04	0.756	0.637	0.8	0.041	0.755	0.625	0.829	(too slow)
	*[GAGM15] (ICP)	(mesh only)		0.534	**0.503**	(mesh only)		7.551	6.372	(too slow)
	*[ESL16] (Eng16)	(mesh only)		1.235	1.237	(mesh only)		1.974	1.312	0.13
	dAML	**0.034**	**0.784**	**0.532**	0.741	**0.036**	**0.772**	**0.557**	**0.76**	(see AML)
	AML	**0.034**	0.779	0.549	0.753	**0.036**	0.771	0.57	0.761	**0.12**
Low Resolution: $24 \times 54 \times 24$ voxels; Multiple, $k > 1$ Fused Views										
100	[DQN17] (Dai17), $k = 5$	**0.012**	**0.924**	**0.214**	**0.436**	**0.018**	**0.887**	**0.278**	**0.491**	n/a
	Sup, $k = 5$	0.022	0.866	0.336	0.566	0.024	0.86	0.331	0.573	
< 16	AML, $k = 2$	0.032	0.794	0.489	0.695	0.034	0.79	0.52	0.725	
< 24	AML, $k = 3$	**0.031**	**0.809**	**0.471**	**0.667**	**0.031**	**0.81**	**0.493**	**0.67**	n/a
< 40	AML, $k = 5$	**0.031**	0.804	0.502	0.686	0.035	0.799	0.523	0.7	
Medium Resolution: $32 \times 72 \times 32$ voxels										
(shape prior)	DVAE	0.019	0.877	0.24	0.47	(same shape prior as on SN-clean)				
100	[DQN17] (Dai17)	**0.02**	**0.869**	**0.399**	**0.674**	**0.026**	**0.83**	**0.51**	**0.767**	**0.074**
	Sup	0.027	0.834	0.498	0.789	0.029	0.815	0.571	0.843	0.09
≤ 6.1	AML	**0.031**	**0.788**	**0.415**	**0.584**	**0.036**	**0.766**	**0.721**	**0.953**	**0.083**
High Resolution: $48 \times 108 \times 48$ voxels										
(shape prior)	DVAE	0.018	0.87	0.272	0.434	(same shape prior as on SN-clean)				
100	Dai17	**0.017**	**0.88**	**0.517**	**0.827**	0.054	0.664	1.559	2.067	**0.066**
	Sup	0.023	0.843	0.677	1.032	**0.052**	**0.674**	**1.52**	**1.981**	0.091
< 3.5	AML	**0.028**	**0.796**	**0.433**	**0.579**	**0.045**	**0.659**	**1.4**	**1.957**	**0.078**

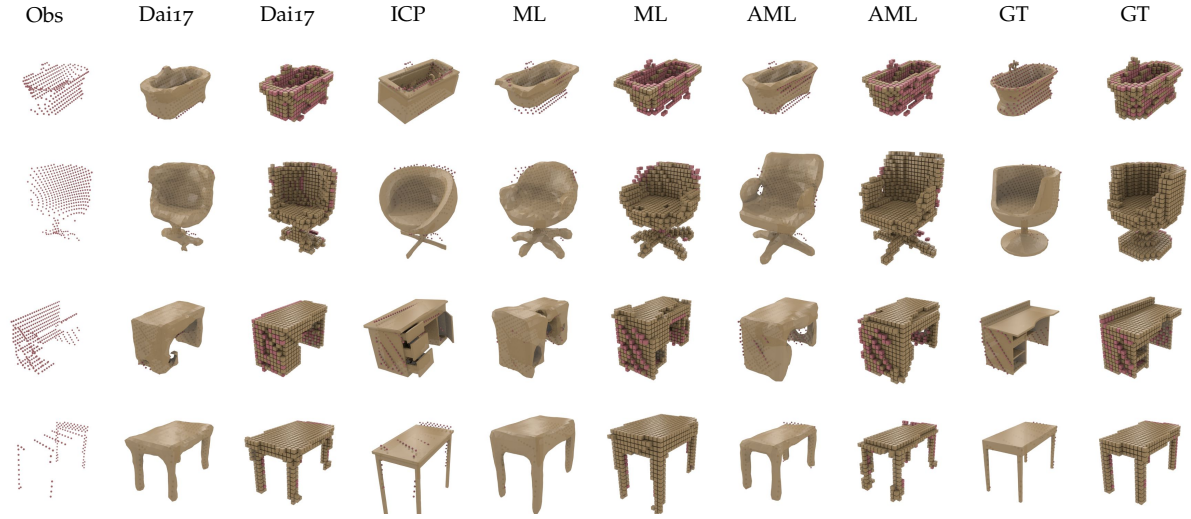
Table 3.2: **Quantitative Results on ShapeNet and KITTI:** We consider Hamming distance (Ham) and intersection over union (IoU) for occupancy grids as well as accuracy (Acc) and completeness (Comp) for meshes on SN-clean, SN-noisy and KITTI. For Ham, Acc and Comp, lower is better; for IoU, higher is better. The unit of Acc and Comp is voxels (voxel length at $24 \times 54 \times 48$ voxels) or meters. Note that the DVAE shape prior (in gray) is only reported as reference (i.e., bound on (d)AML). We indicate the level of supervision in percentage, relative to the corresponding resolution (see Table 3.1) and mark the best results under full supervision in **red** and under weak supervision in **green**. See text for discussion.

Ablation Study: In Table 3.2, we show quantitative results of our model on SN-clean and SN-noisy. First, we report the reconstruction quality of the DVAE shape prior as reference. Then, we consider the DVAE shape prior (Naïve), and its mean prediction (Mean) as simple baselines. The poor performance of both illustrates the difficulty of the benchmark. For AML, we also consider its deterministic variant, dAML (see Section 3.2). Quantitatively, there is essentially no difference; however, Figure 3.10 demonstrates that AML is able to predict more detailed shapes. We also found that using both occupancy and SDFs is necessary to obtain good performance – as is using both point observations and free space.

Considering Figure 3.9, we additionally demonstrate that the embedding learned by AML, i.e., the embedding of incomplete observations within the latent shape space, is able to associate observations with corresponding shapes even under weak supervision. In particular, we show a t-SNE visualization and a projection of the latent space for AML trained on SN-clean. We color-code 10 randomly chosen ground truth shapes, resulting in 100 observations (10 views per shape). AML is usually able to embed observations near the corresponding ground truth shapes, without explicit supervision (e.g., for violet, pink, blue or teal, the observations – points



(a) SN-clean (Top) and SN-noisy (Bottom), Low Resolution ($24\times54\times24$)



(b) ModelNet Bathtubs, Chairs, Desks and Tables, Low Resolution (32^3)

Figure 3.11: **Qualitative Results on ShapeNet and ModelNet:** Results for AML, Dai17, Eng16, ICP and ML on SN-clean, SN-noisy and ModelNet's bathtubs, chairs, desks and tables. AML outperforms data-driven approaches (ML, Eng16, ICP) and rivals Dai17 while requiring significantly less supervision. Occupancy grids and meshes in beige, observations in red.

– are close to the corresponding ground truth shapes – "x"). Additionally, AML also matches the unit Gaussian prior distribution reasonably well.

Comparison to Baselines on Synthetic Data: For ShapeNet, Table 3.2 demonstrates that AML outperforms data-driven approaches such as Eng16, ICP and ML and is able to compete with fully-supervised approaches, Dai17 and Sup, while using only 8% or less supervision. We also note that AML outperforms ML, illustrating that amortized inference is beneficial. Furthermore, Dai17 outperforms Sup, illustrating the advantage of propagating low-level information (through skip connections) without bottleneck. Most importantly, the performance gap between AML and Dai17 is rather small considering the difference in supervision (more than 92%) and on SN-noisy, the drop in performance for Dai17 and Sup is larger than for AML suggesting that AML handles noise and sparsity more robustly. Figure 3.11 shows that these conclusions also apply visually where AML performs en par with Dai17.

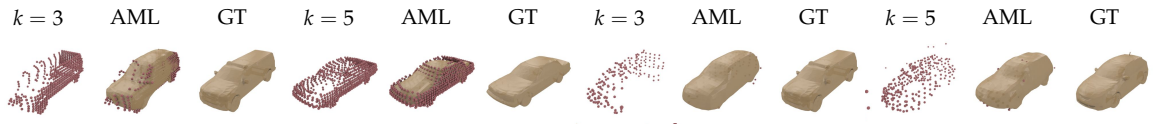
Supervision in %	Method	bathtub		chair				desk		table		ModelNet10	
		Ham↓	IoU↑	Ham↓	IoU↑	Acc [vx]↓	Comp [vx]↓	Ham↓	IoU↑	Ham↓	IoU↑	Ham↓	IoU↑
Low Resolution: 32^3 voxels; * independent of resolution													
(shape prior)	DVAE	0.015	0.699	0.025	0.517	0.884	0.72	0.028	0.555	011	0.608	0.023	0.714
100	[DQN17] (Dai17)	**0.022**	0.59	**0.019**	**0.61**	**0.663**	**0.671**	**0.027**	**0.568**	**0.011**	**0.648**	**0.03**	**0.646**
	Sup	0.023	**0.618**	0.03	0.478	0.873	0.813	0.036	0.458	0.017	0.497	0.038	0.589
< 10	* [GAGM15] (ICP)	(mesh only)		(mesh only)		1.483	0.89	(mesh only)		(mesh only)		(mesh only)	
	ML	0.028	**0.503**	**0.033**	**0.414**	1.489	1.065	0.048	0.323	0.029	0.318	(too slow)	
	AML	**0.026**	**0.503**	**0.033**	0.373	**1.088**	**0.785**	**0.041**	**0.389**	**0.018**	**0.423**	**0.04**	**0.509**
Medium Resolution: 48^3 voxels													
(shape prior)	DVAE	0.014	0.671	0.021	0.491	0.748	0.697	0.025	0.525	0.01	0.548		
100	[DQN17] (Dai17)	**0.018**	**0.609**	**0.016**	**0.576**	**0.513**	**0.508**	**0.023**	**0.532**	**0.008**	**0.65**		
< 9	AML	**0.024**	**0.459**	**0.029**	**0.347**	**1.025**	**0.805**	**0.034**	**0.361**	**0.015**	**0.384**		
High Resolution: 64^3 voxels													
(shape prior)	DVAE	0.014	0.644	0.02	0.474	0.702	0.705	0.024	0.506	0.009	0.548		
100	[DQN17] (Dai17)	**0.018**	**0.54**	**0.016**	**0.548**	**0.47**	**0.53**	**0.021**	**0.525**	**0.007**	**0.673**		
< 9	AML	**0.023**	**0.46**	**0.026**	**0.333**	**0.893**	**0.852**	**0.042**	**0.31**	**0.012**	**0.407**		

Table 3.3: **Quantitative Results on ModelNet:** Results for bathtubs, chairs, desks, tables and all ten categories combined (ModelNet10). As the ground truth SDFs are merely approximations (c.f. Section 3.3.1), we concentrate on Hamming distance (Ham; lower is better) and intersection-over-union (IoU; higher is better). Only for chairs, we report accuracy Acc and completeness Comp in voxels (voxel length at 32^3 voxels). We also indicate the level of supervision (see Table 3.1). Again, we report the DVAE shape prior as reference and color the best weakly-supervised approach using **green** and the best fully-supervised approach in **red**.

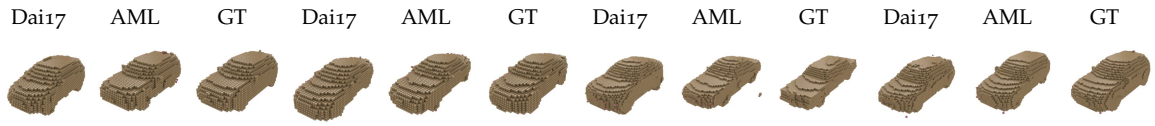
For ModelNet, in Table 3.3, we mostly focus on occupancy grids (as the derived SDFs are approximate, c.f. Section 3.3.1) and show that chairs, desks or tables are more difficult. However, AML is still able to predict high-quality shapes, outperforming data-driven approaches. Additionally, in comparison to ShapeNet, the gap between AML and fully-supervised approaches (Dai17 and Sup) is surprisingly small – not reflecting the difference in supervision. This means that even under full supervision, these object categories are difficult to complete. In terms of accuracy (Acc) and completeness (Comp), e.g., for chairs, AML outperforms ICP and ML. Dai17 and Sup, on the other hand, outperform AML. Still, considering Figure 3.11, AML predicts visually appealing meshes although the reference shape SDFs on ModelNet are merely approximate. Qualitatively, AML also outperforms its data-driven rivals and only Dai17 predicts shapes slightly closer to the ground truth.

Multiple Views and Higher Resolutions: In Table 3.2, we consider multiple, $k \in \{2, 3, 5\}$, randomly fused observations (from the 10 views per shape). Generally, additional observations are beneficial (also c.f. Figure 3.12). However, fully-supervised approaches such as Dai17 benefit more significantly from more views than AML. Intuitively, especially on SN-noisy, $k = 5$ noisy observations seem to impose contradictory constraints that cannot be resolved under weak supervision. We also show that higher resolution allows both AML and Dai17 to predict more detailed shapes, see Figure 3.12. For AML this is significant as, e.g., on SN-noisy, the level of supervision reduces to less than 1%. Also note that AML is able to handle the slightly asymmetric desks in Figure 3.12 due to the strong shape prior which itself includes symmetric and less symmetric shapes.

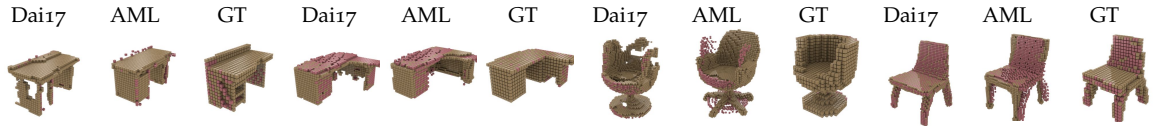
Multiple Object Categories: We also investigate the category-agnostic case, considering all ten ModelNet10 object categories. Here, we train a single DVAE shape prior (and a single model for Dai17 and Sup) across all ten object categories. In Table 3.3, the gap between AML and fully-supervised approaches further shrinks. This means that even fully-supervised methods



(a) SN-clean and -noisy, k Views, Low Resolution ($24\times54\times24$)



(b) SN-clean and -noisy, Medium ($32\times72\times32$) and High ($48\times108\times48$) Resolution



(c) ModelNet desks and chairs, Medium (48^3) and High (64^3) Resolution

Figure 3.12: **Multi-View and Higher-Resolution Results on ShapeNet and ModelNet:** While AML is designed for especially sparse observations, it also performs well in a multi-view setting. Additionally, higher resolutions allow predicting more detailed shapes. Shapes, occupancy grids or meshes, in beige and observations in red.

have difficulties distinguishing object categories based on sparse observations. Figure 3.13 shows that AML is able to not only predict reasonable shapes, but also identify the correct object category. In contrast to Dai17, which predicts slightly more detailed shapes, this is significant as AML does not have access to the object category during training.

Comparison on Real Data: On KITTI, considering Figure 3.14, we illustrate that AML consistently predicts detailed shapes regardless of the noise and sparsity in the inputs. Our qualitative results suggest that AML is able to predict more detailed shapes compared to Dai17 and Eng16. Additionally, Eng16 is distracted by sparse and noisy observations. Quantitatively, instead, Dai17 and Sup outperform AML. However, this is mainly due to two factors: first, the ground truth collected on KITTI does rarely cover the full car, and second, we put significant effort into faithfully modeling KITTI's noise statistics in SN-noisy, allowing Dai17 and Sup to generalize very well. The latter effort, especially, can be avoided by using our weakly-supervised approach.

On Kinect, also considering Figure 3.14, only 30 observations are available for training. It can be seen that AML predicts reasonable shapes for tables. We find it interesting that AML is able to generalize from only 30 training examples. In this sense, AML functions similar to ML, in that the objective is trained to overfit to few samples. This, however, cannot work in

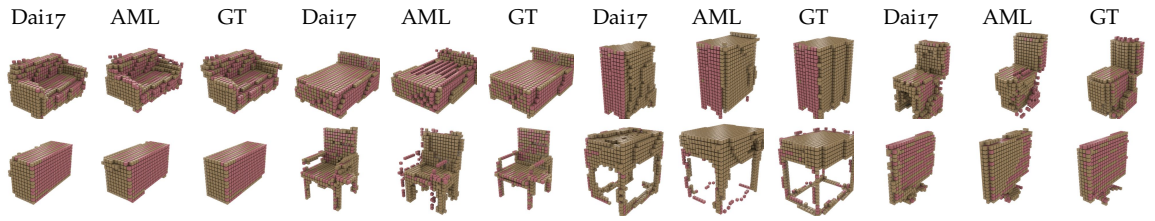
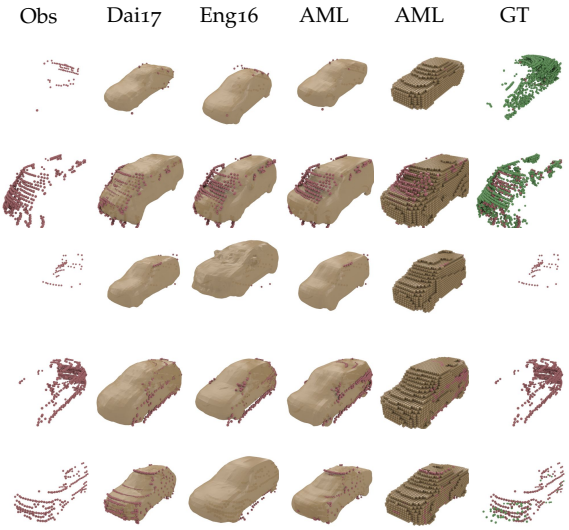
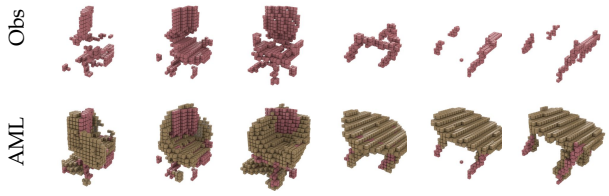


Figure 3.13: **Category-Agnostic Results on ModelNet10:** AML recovers detailed shapes of the correct object category even without category supervision (as provided to Dai17). Shapes (occupancy grids and meshes) in beige and observations in red at low resolution (32^3 voxels).



(a) KITTI, Medium Resolution (32×72×32)



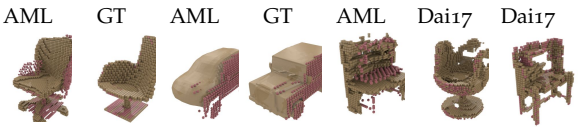
(b) Kinect, Low Resolution (32^3)

Figure 3.14: **Qualitative Results on KITTI and Kinect:** On KITTI, AML visually outperforms both Dai17 and Eng16 while being faster and requiring less supervision. On Kinect, AML demonstrates that it is able to generalize from as few as 30 training samples. Predicted shapes (occupancy grids or meshes) in beige and observations in red; additionally, partial ground truth in green.

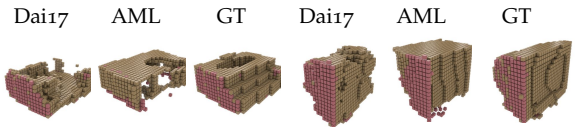
all cases, as demonstrated by the chairs where AML tries to predict a suitable chair, but does not fit the observations as well. Another problem witnessed on Kinect, is that the shape prior training samples need to be aligned to the observations (with respect to the viewing angles). For the chairs, we were not able to guess the viewing trajectory correctly (c.f. [YRM+19]).

Failure Cases: AML and Dai17 often face similar problems, as illustrated in Figure 3.15, suggesting that these problems are inherent to the used shape representations or the learning approach independent of the level of supervision. For example, both AML and Dai17 have problems with fine, thin structures that are hard to reconstruct properly at any resolution. Furthermore, identifying the correct object category on ModelNet10 from sparse observations is difficult for both AML and Sup. Finally, AML additionally has difficulties with exotic objects that are not well represented in the latent shape space as, e.g., designed chairs.

Runtime: At low resolution, AML as well as the fully-supervised approaches Dai17 and Sup, are particular fast, requiring up to 2*ms* on a NVIDIA™ GeForce® GTX TITAN using Torch [CKF11]. Data-driven approaches (e.g., Eng16, ICP and ML), on the other hand, take considerably longer. Eng16, for instance requires 168*ms* on average for completing the shape of a sparse LIDAR observation from KITTI using an Intel® Xeon® E5-2690 @2.6Ghz and the multithreaded Ceres solver [AMO12]. ICP and ML take longest, requiring up to 38*s* and 75*s* (not taking into account the point sampling process for the shapes), respectively. Except for Eng16 and ICP, all approaches scale with the used resolution and the employed architecture.



(a) Exotic Shapes and Fine Structures



(b) Difficulties with Multiple Categories

Figure 3.15: **Failures Cases: Left:** We show that AML has difficulties with exotic shapes, not represented in the latent space; and both AML and Dai17 have difficulties with fine details. **Right:** Results demonstrating that it is difficult to infer the correct object category from sparse observations, even under full supervision as required by Dai17. Shapes (occupancy grids and meshes) in beige and observations in red from various resolutions.

3.4 Conclusion

In this chapter, we presented a novel, weakly-supervised learning-based approach to 3D shape completion from sparse and noisy point cloud observations. We used a (denoising) variational auto-encoder [KW14a, IAMB17] to learn a latent space of shapes for one or multiple object categories using synthetic data from ShapeNet [CFG+15] or ModelNet [WSK+15]. Based on the learned generative model, i.e., decoder, we formulated 3D shape completion as a maximum likelihood problem. In a second step, we then fixed the learned generative model and trained a new recognition model, i.e. encoder, to amortize, i.e. *learn*, the maximum likelihood problem. Thus, our **Amortized Maximum Likelihood (AML)** approach to 3D shape completion can be trained in a weakly-supervised fashion. Compared to related data-driven approaches, e.g., [NXS12, BCLS13, DPRR13, HSP14, RGT+15, LDGN15, NHT+16, ESL16, ESL17], our approach offers fast inference at test time; in contrast to other learning-based approaches, e.g., [SGF16, REM+16, FSG17, RUBG17, SM17, DQN17, WHY+17, VDR+17, HLH+17, YRM+19], we do not require full supervision during training. Both characteristics render our approach useful for robotic scenarios where full supervision is often not available such as in autonomous driving, e.g., on KITTI [GLU12], or indoor robotics, e.g., on Kinect [YRM+19].

On two newly created synthetic shape completion benchmarks, derived from ShapeNet's cars and ModelNet10, as well as on real data from KITTI and, we demonstrated that AML outperforms related data-driven approaches [ESL16, GAGM15] while being significantly faster. We further showed that AML is able to compete with fully-supervised approaches [DQN17], both quantitatively and qualitatively, while using only $3-10\%$ supervision or less. In contrast to [NXS12, BCLS13, DPRR13, HSP14, RGT+15, LDGN15, ESL16, ESL17], we additionally showed that AML is able to generalize across object categories without category supervision during training. On Kinect, we also demonstrated that our AML approach is able to generalize from very few training examples. In contrast to [WSK+15, GFRG16, SGF16, FMAJB16, DQN17, LYF17, HLH+17, FSG17], we considered resolutions up to $48\times108\times48$ and 64^3 voxels as well as significantly sparser observations. Overall, our experiments demonstrate two key advantages of the proposed approach: significantly reduced runtime and increased performance compared to data-driven approaches and no need for ground truth compared to learning-based approaches showing that amortizing inference is highly effective.

II

Understanding Adversarial Examples and Training

In this second part, we intend to understand the phenomenon of *adversarial examples*. These are imperceptibly perturbed images causing misclassification. Despite good performance on test sets, adversarial examples are able to fool almost any deep neural network. Thus, it is crucial to understand why these adversarial examples exist in order to improve robustness against such attacks. For example, a popular approach to obtain adversarially robust models is *adversarial training*, i.e., injecting adversarial examples into the training procedure. However, it is often observed that adversarial training leads to reduced accuracy and easily overfits. For wide-spread use of adversarial training both problems have to be understood and addressed.

In Chapter 4, we find that adversarial examples tend to leave the underlying data manifold, explaining why normally trained models are easily fooled. Furthermore, by considering *on-manifold adversarial examples* that are explicitly constrained to the data manifold, we show that there is no *inherent* trade-off between robustness and accuracy. Instead, the robustness-accuracy trade-off of adversarial training is caused by higher sample complexity.

Subsequently, in Chapter 5, we intend to understand why some adversarially trained models provide higher robustness than others. To this end, we particularly consider the phenomenon of robust overfitting where robustness does *not* reduce continuously throughout training. We show that better robustness correlates strongly with finding flatter minima in the robust loss landscape w.r.t. to perturbations in the model's weights. Based on new *robust flatness* measures, we find that favoring flatness during training avoids robust overfitting and improves overall robustness.

4 Disentangling Adversarial Robustness and Generalization

Contents

Deep deep learning enabled significant progress on almost all computer vision problems as well as in many other domains such as natural language processing, speech recognition or robotics. However, the ever-increasing size of deep neural networks and their black-box nature raises many open questions including their failure modes. In the following, we switch focus to one particular kind of failure mode: *adversarial examples*, imperceptibly perturbed images causing misclassification, as first reported for deep neural networks in [SZS+14]. Obtaining models that are robust against adversarial examples *and* generalize well is an open problem. A recent hypothesis [SZC+18, TSE+19] even states that *both* robust *and* accurate models are impossible, i.e., adversarial robustness and generalization are conflicting goals. In an effort to **clarify the relationship between robustness and generalization**, we assume an underlying, low-dimensional data manifold and show that:

1. regular adversarial examples leave the manifold;
2. on-manifold adversarial examples, explicitly constrained to the manifold, exist;
3. on-manifold adversarial examples are generalization errors, and on-manifold adversarial training boosts generalization;
4. regular robustness and generalization are not necessarily contradicting goals.

These assumptions imply that *both* robust *and* accurate models are possible. However, different deep neural networks (architectures, training strategies etc.) can exhibit different robustness and generalization characteristics. To confirm our claims, we present experiments on synthetic data as well as on MNIST [CATvS17], Fashion-MNIST [XRV17] and CelebA [LLWT15].

This chapter is based on [SHS19]: As first author, David Stutz ran all experiments and was the main writer of the paper. However, the theoretical argument for off-manifold adversarial examples in Section 4.2.2 was contributed by Matthias Hein. This work was also presented at the Workshop on Uncertainty and Robustness in Deep Learning (UDL) held in conjunction with ICML 2019 and the Heidelberg Laureate Forum (HLF) 2019.

The **code** for this chapter can be found on GitHub[1].

[1] https://github.com/davidstutz/cvpr2019-adversarial-robustness

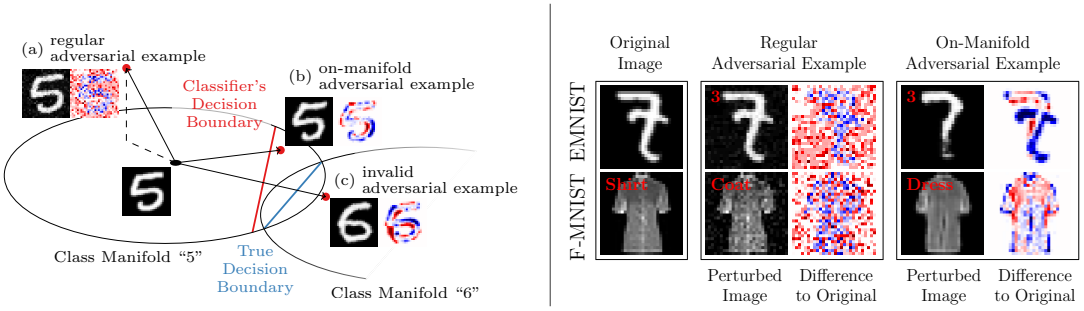


Figure 4.1: **On- and Off-Manifold Adversarial Examples:** We consider adversarial examples, shown with their (normalized) difference to the original image, in the context of the underlying manifold, e.g., class manifolds "5" and "6" on MNIST [CATvS17]. This allows us to study their relation to generalization. Regular adversarial examples are not constrained to the manifold, c.f. (a), and often result in (seemingly) random noise patterns. In fact, we show that they leave the manifold. However, adversarial examples on the manifold can be found as well, c.f. (b), resulting in meaningful manipulations of the image content. However, care needs to be taken that the actual, true label w.r.t. the manifold does not change, c.f. (c).

4.1 Introduction

Adversarial robustness describes the ability to defend against adversarial examples [SZS+14], imperceptibly perturbed images causing misclassification. These adversarial attacks pose severe security threats, as demonstrated against Clarifai.com [LCLS17, BHLS17] or Google Cloud Vision [IEAL18]. Despite these serious risks, many proposed defenses have been ineffective. Only adversarial training, i.e., training on adversarial examples [GSS15, MMS+18], has been shown to work well in practice [ACW18, AC18] – at the cost of computational overhead and reduced accuracy. Overall, the problem of adversarial robustness is left open and poorly understood.

The phenomenon of adversarial examples itself, i.e., their mere existence, has also received considerable attention. Early explanations attributing adversarial examples to "rare pockets" of the classification surface [SZS+14] or linearities in deep networks [GSS15] have been superseded by the manifold assumption [TG16, GMF+18]: adversarial examples are assumed to leave the underlying, low-dimensional but usually unknown data manifold. However, only [SKN+18] provide experimental evidence supporting this assumption. Yet, on a simplistic toy dataset, Gilmer et al. [GMF+18] also found adversarial examples on the manifold, as also tried on real datasets [BCZ+18, SSKE18, ZDS18], rendering the manifold assumption questionable. Still, the manifold assumption fostered research on novel defenses [IJA+17, PS18, SRBB19].

Beyond the existence of adversarial examples, their relation to generalization is an important open problem. Recently, it has been argued [SZC+18, TSE+19] that there exists an inherent trade-off, i.e., robust and accurate models seem impossible. While Tsipras et al. [TSE+19] provide a theoretical argument on a toy dataset, Su et al. [SZC+18] evaluate the robustness of different models on ImageNet [RDS+15]. However, these findings have to be questioned given the results in [RGB16, GMF+18] showing the opposite, i.e., better generalization helps robustness.

In order to address this controversy, and in contrast to [RGB16, SZC+18, TSE+19], we consider adversarial robustness in the context of the underlying manifold. In particular, to break the hypothesis down, we explicitly ask whether adversarial examples leave, or stay on, the manifold. On MNIST [LBBH98, CATvS17], for example, considering the class manifolds

for "5" and "6", as illustrated in Figure 4.1, adversarial examples are not guaranteed to lie on the manifold, c.f. Figure 4.1 (a). Adversarial examples can, however, also be constrained to the manifold, c.f. Figure 4.1 (b). In this case, it is important to ensure that the adversarial examples do not actually change their label, i.e., are more likely to be a "6" than a "5", as in Figure 4.1 (c). For clarity, we refer to unconstrained adversarial examples, as illustrated in Figure 4.1 (a), as *regular adversarial examples*; in contrast to adversarial examples constrained to the manifold, so-called *on-manifold adversarial examples*.

Contributions: Based on this distinction between regular robustness against unconstrained adversarial examples and on-manifold robustness against adversarial examples constrained to the manifold, we show:

1. regular adversarial examples leave the manifold;
2. adversarial examples constrained to the manifold, i.e., on-manifold adversarial examples, exist and can be computed using an approximation of the manifold;
3. on-manifold robustness is essentially generalization;
4. and regular robustness and generalization are not necessarily contradicting goals, i.e., for any arbitrary but fixed model, better generalization through additional training data does not worsen robustness.

We conclude that **both robust and accurate models are possible** and can, e.g., be obtained through adversarial training on larger training sets. Additionally, we propose on-manifold adversarial training to boost generalization in settings where the manifold is known, can be approximated, or invariances of the data are known. We present experimental results on a novel MNIST-like, synthetic dataset with known manifold, as well as on MNIST [CATvS17], Fashion-MNIST [XRV17] and CelebA [LLWT15].

4.2 Adversarial Robustness and Generalization

To clarify the relationship between adversarial robustness and generalization, we explicitly distinguish between regular and on-manifold adversarial examples, as illustrated in Figure 4.1. Then, the hypothesis [SZC+18, TSE+19] that robustness and generalization are contradicting goals is challenged in four arguments: regular unconstrained adversarial examples leave the manifold; adversarial examples constrained to the manifold exist; robustness against on-manifold adversarial examples is essentially generalization; and robustness against regular adversarial examples is not influenced by generalization when controlled through the amount of training data. Altogether, our results imply that adversarial robustness and generalization are not opposing objectives and both robust and accurate models are possible but require higher sample complexity.

4.2.1 Experimental Setup

Datasets: We use MNIST [CATvS17], F(ashion)-MNIST [XRV17] and CelebA [LLWT15] for our experiments (240k/40k, 60k/10k and 182k/20k training/test images); CelebA has been re-sized to 56×48 and we classify "Male" vs. "Female".

Our synthetic dataset, FONTS, consists of letters "A" to "J" from different fonts that are randomly transformed using a spatial transformer network [JSZK15] such that the generation process is completely differentiable. Specifically, we consider 1000 Google Fonts obtained from

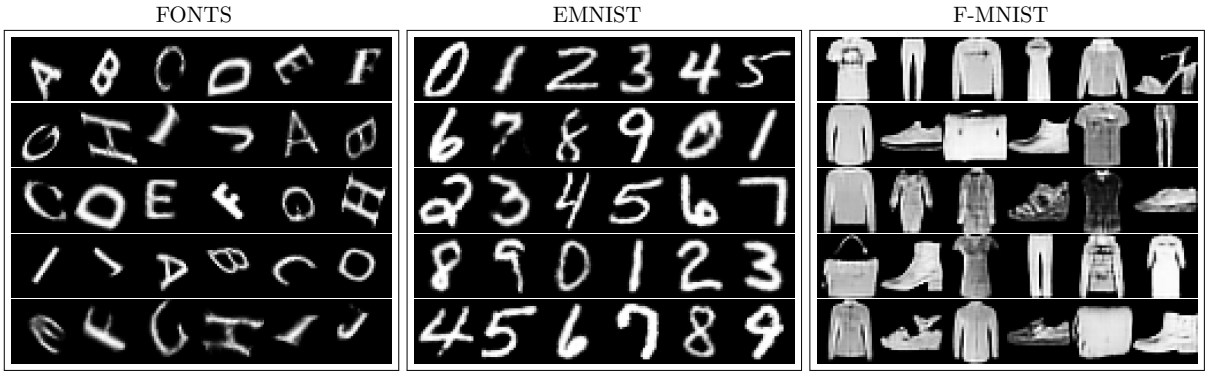


Figure 4.2: **Random Sampled from VAE-GANs:** For FONTS (left), MNIST (middle) and Fashion (right), we show random samples from the learned, class-specific VAE-GANs used to craft on-manifold adversarial examples. Our VAE-GANs generate realistic looking samples. However, we also include problematic samples illustrating the discrepancy between true and approximated data distribution.

the corresponding GitHub repository[2], manually curated to exclude, e.g., fonts consisting of symbols. The obtained letters are transformed using translation, shear, scaling and rotation, uniformly sampled from $[-0.2, 0.2]$, $[-0.5, 0.5]$, $[0.75, 1.15]$, and $[-\pi/2, \pi/2]$, respectively. With 112 transformations per letter, we obtain 1.12Mio images of size 28×28, split into 960k training images and 160k test images (of which we only use 40k for simplicity). To make the generation process differentiable w.r.t. the transformation parameters, the spatial transformer network is applied using the transformation matrix

$$\begin{bmatrix} \cos(r)s - \sin(r)s\lambda_1 & -\sin(r)s + \cos(r)s\lambda_1 & t_1 \\ \cos(r)s\lambda_2 + \sin(r)s & -\sin(r)s\lambda_2 + \cos(r)s & t_2 \end{bmatrix}, \tag{4.1}$$

with translation $[t_1, t_2]$, shear $[\lambda_1, \lambda_2]$, scale s and rotation r. Overall, this results in FONTS offering full control over the manifold, i.e., the transformation parameters, font and class, with differentiable generative model, i.e., decoder.

Models: We consider classifiers with three (four on CelebA) convolutional layers (4×4 kernels; stride 2; 16, 32, 64 channels), each followed by ReLU activations and batch normalization [IS15b], and two fully connected layers. The models are trained using ADAM [KB15b], with learning rate 0.01 (decayed by 0.95 per epoch), weight decay 10^{-4} and batch size 100, for 20 epochs. Most importantly, to control their generalization performance, we use N training images, with N between 250 and 40k. For each N, we train 5 models with random weight initialization [GB10b] and report averages.

Approximating Manifolds: We learn class-specific VAE-GANs to approximate the underlying manifold. In contrast to [LSLW16], however, we use a reconstruction loss on the image, not on the discriminator's features, and in contrast to [RLWFM17], we use the standard Kullback-Leibler divergence to regularize the latent space. Specifically, the model consists of an encoder enc, approximating the posterior $q(z|x) \approx p(z|x)$ of latent code z given image x, a (deterministic) decoder dec, and a discriminator dis. During training, the sum of the following

[2]https://github.com/google/fonts

losses is minimized:

$$\mathcal{L}_{\text{enc}} = \mathbb{E}_{q(z|x)}\left[\lambda\|x - \text{dec}(z)\|_1\right] + \text{KL}(q(z|x)|p(z)) \tag{4.2}$$

$$\mathcal{L}_{\text{dec}} = \mathbb{E}_{q(z|x)}\left[\lambda\|x - \text{dec}(z)\|_1 - \log(\text{dis}(\text{dec}(z)))\right] \tag{4.3}$$

$$\mathcal{L}_{\text{dis}} = -\mathbb{E}_{p(x)}\left[\log(\text{dis}(x))\right] - \mathbb{E}_{q(z|x)}\left[\log(1 - \text{dis}(\text{dec}(z)))\right] \tag{4.4}$$

using a standard Gaussian prior $p(z)$. Here, $q(z|x)$ is modeled by predicting the mean $\mu(x)$ and variance $\sigma^2(x)$ such that $q(z|x) = \mathcal{N}(z; \mu(x), \text{diag}(\sigma^2(x)))$ and the weighting parameter λ controls the importance of the L_1 reconstruction loss relative to the Kullback-Leibler divergence KL and the adversarial loss for decoder and discriminator. As in [KW14b], we use the reparameterization trick with one sample to approximate the expectations in Equation (4.2), (4.3) and (4.4), and the Kullback-Leibler divergence $\text{KL}(q(z|x)|p(z))$ is computed analytically.

The encoder, decoder and discriminator consist of three (four for CelebA) (de-) convolutional layers (4×4 kernels; stride 2; 64, 128, 256 channels), followed by ReLU activations and batch normalization [IS15b]. The encoder uses two fully connected layers to predict mean and variance and the discriminator uses two fully connected layers to predict logits. We tuned λ to dataset- and class-specific values: on FONTS, $\lambda = 3$ worked well for all classes, on MNIST, $\lambda = 2.5$ except for classes "0" ($\lambda = 2.75$), "1" ($\lambda = 5.6$) and "8" ($\lambda = 2.25$), on Fashion, $\lambda = 2.75$ worked well for all classes, on CelebA $\lambda = 3$ worked well for both classes. Finally, we use a learning rate 0.005 (decayed by 0.9 every epoch), weight decay 10^{-4} and batch size 100 for 10, 30, 60 and 30 epochs on FONTS, MNIST, Fashion and CelebA, respectively. Random samples of the class-specific VAE-GANs are shown in Figure 4.2. Especially on MNIST and FONTS, our VAE-GANs generate realistic looking samples with sharp edges. However, we also show several problematic random samples, illustrating the discrepancy between the true data distribution and the approximation – as particularly highlighted on FONTS.

Attack: Given an image-label pair (x, y) from an unknown data distribution p and a classifier f, an adversarial example is a perturbed image $\tilde{x} = x + \delta$ which is misclassified by the model, i.e., $f(\tilde{x}) \neq y$. While our results can be confirmed using other attacks and norms, for clarity, we concentrate on the L_∞ white-box attack by Madry et al. [MMS+18] that directly maximizes the training loss,

$$\max_\delta \mathcal{L}(f(x+\delta), y) \quad \text{s.t.} \quad \|\delta\|_\infty \leq \epsilon, \tilde{x}_i \in [0,1], \tag{4.5}$$

using projected gradient descent. Here, $\mathcal{L}$ is the cross-entropy loss and $\tilde{x} = x + \delta$. The ϵ-constraint is meant to ensure perceptual similarity. We run 40 iterations of ADAM [KB15b] with learning rate 0.005 and consider 5 restarts, (distance and direction) uniformly sampled in the ϵ-ball for $\epsilon = 0.3$. Optimization is stopped as soon as the predicted label changes, i.e., $f(\tilde{x}) \neq y$. We attack 1000 test images.

Adversarial Training: An established defense is adversarial training, i.e., training on adversarial examples crafted during training. [MMS+18] considers the min-max problem

$$\min_w \mathbb{E}\left[\max_{\|\delta\|_\infty \leq \epsilon, x_{n,i}+\delta_i \in [0,1]} \mathcal{L}(f(x_n+\delta; w), y_n)\right] \tag{4.6}$$

where w are the classifier's weights and x_n the training images. We considered different variants [SZS+14, GSS15, MMS+18] and decided to follow common practice and train on 50% clean images and 50% adversarial examples [SZS+14]. For $\epsilon = 0.3$, the attack (for the inner optimization problem) is run for full 40 iterations, i.e., is not stopped at the first adversarial example found. Robustness of the obtained deep neural network is measured by computing the attack **success rate**, i.e., the fraction of successful attacks on correctly classified test images, as, e.g., in [CW17b], for a fixed ϵ; lower success rate indicates higher robustness.

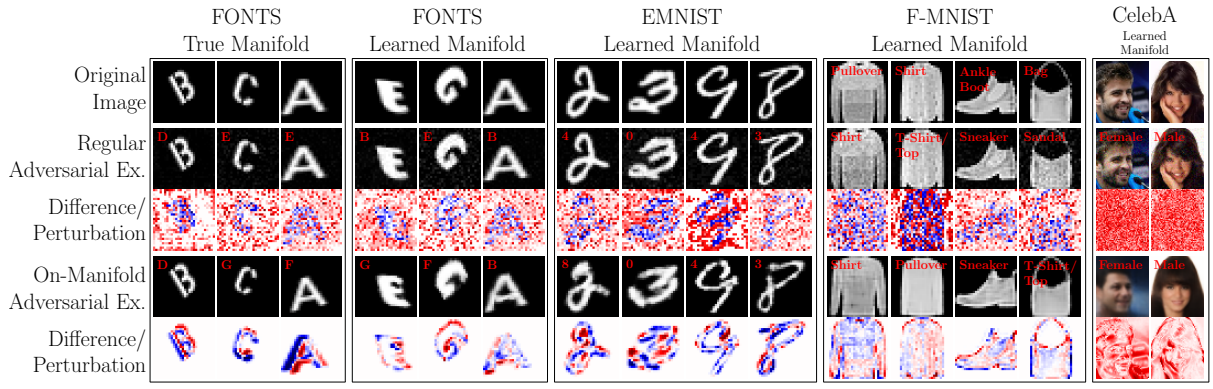


Figure 4.3: **Regular and On-Manifold Adversarial Examples** on our synthetic dataset, FONTS, consisting of randomly transformed characters "A" to "J", MNIST [CATvS17], Fashion [XRV17] and CelebA [LLWT15]. On FONTS, the manifold is known by construction. In the other cases, the class manifolds have been approximated using VAE-GANs [LSLW16, RLWFM17]. The difference (normalized; or their magnitude on CelebA) to the original test image reveals the (seemingly) random noise patterns of regular adversarial examples in contrast to reasonable concept changes of on-manifold adversarial examples.

4.2.2 Adversarial Examples Leave the Manifold

The idea of adversarial examples leaving the manifold is intuitive on MNIST where particular background pixels are known to be constant, see Figure 4.3. If an adversarial example $\tilde{x}$ manipulates these pixels, it has zero probability under the data distribution and its distance to the manifold, i.e., the distance to its projection $\pi(\tilde{x})$ onto the manifold, should be non-zero. On FONTS, with known generative process in the form of a decoder dec mapping latent variables z to images x, the projection is obtained iteratively: $\pi(\tilde{x}) = \text{dec}(\tilde{z})$ with $\tilde{z} = \arg\min_z \| \text{dec}(z) - \tilde{x}) \|_2$ and z constrained to valid transformations (font and class, known from the test image x, stay constant). To this end, we use 100 iterations of ADAM with learning rate 0.09, decayed every 10 iterations by factor 0.95. Additional iterations did not improve the results. On MNIST, as illustrated in Figure 4.4, the manifold is approximated using 50 nearest neighbors and the projection $\pi(\tilde{x})$ onto the sub-space spanned by the x-centered nearest neighbors is computed through least squares. Specifically, we consider the vector $\delta = \tilde{x} - x$, i.e., we assume that the "adversarial direction" originates at the mean $\bar{x} = 1/50 \sum_{i=1}^{50} x_i$. Then,

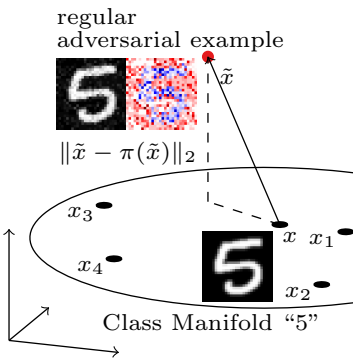


Figure 4.4: **Off-Manifold Distance Computation.** The distance of a regular adversarial example $\tilde{x}$ to the manifold, is computed as the distance to its orthogonal projection $\pi(\tilde{x})$: $\|\tilde{x} - \pi(\tilde{x})\|_2$. Here, the manifold is approximated linearly using least squares based on 50 nearest neighbors x_i of x. Then, large distances indicate that the adversarial example left the manifold, which we find to hold for both synthetic as well as real datasets.

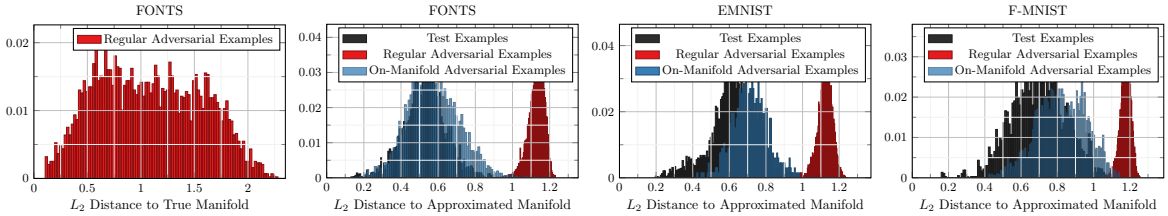


Figure 4.5: **Adversarial Examples Leave the Data Manifold:** Distance of adversarial examples to the true, on FONTS (left), or approximated, on FONTS, MNIST and Fashion (right), manifold. We show normalized histograms of the L_2 distance of adversarial examples to their projections onto the manifold. Regular adversarial examples exhibit a significant distance to the manifold, clearly distinguishable from on-manifold adversarial examples and test images. We also note that, depending on the VAE-GAN approximation, on-manifold adversarial examples are hardly distinguishable from test images.

we solve

$$\beta^* = \arg\min_{\beta} \|X\beta - \delta\|_2^2 \tag{4.7}$$

where the columns X_i are the vectors $x_i - \bar{x}$. The projection $\pi(\tilde{x})$ is obtained as $\pi(\tilde{x}) = X\beta^*$. The same approach can be applied to projecting the test image x. Note that it is crucial to consider the adversarial direction δ itself, instead of the adversarial example $\tilde{x}$ because $\|\delta\|_2$ is small by construction, i.e., the projections of $\tilde{x}$ and x are very close. On both FONTS and MNIST, the distance $\|\tilde{x} - \pi(\tilde{x})\|_2$ is considered to assess whether the adversarial example $\tilde{x}$ actually left the manifold.

On FONTS, Figure 4.5 (left) shows that regular adversarial examples clearly exhibit non-zero distance to the manifold. In fact, the projections of these adversarial examples to the manifold are almost always the original test images. As a result, the distance to the manifold is essentially the norm of the corresponding perturbation: $\|\tilde{x} - \pi(\tilde{x})\|_2 \approx \|\tilde{x} - x\|_2 = \|\delta\|_2$. This suggests that the adversarial examples leave the manifold in an almost orthogonal direction. On MNIST and Fashion, in Figure 4.5 (right), these results can be confirmed in spite of the crude local approximation of the manifold. Again, regular adversarial examples seem to leave the manifold almost orthogonally, i.e., their distance to the manifold coincides with the norm of the corresponding perturbations. These results show that regular adversarial examples are essentially off-manifold adversarial examples. This finding is intuitive as for well-trained classifiers, leaving the manifold should be the "easiest" way to fool it.

Intuition and Theoretical Argument: Having empirically shown that regular adversarial examples tend to leave the manifold, often in a nearly orthogonal direction, we also discuss a theoretical argument supporting this observation. The main assumption is that the training loss is constant on the manifold (normally close to zero) due to training and proper generalization, i.e., low training and test loss. Thus, the loss gradient is approximately orthogonal to the manifold as this is the direction to increase the loss most efficiently.

More formally, let $f(x)$ denote the classifier which – for simplicity – takes inputs $x \in \mathbb{R}^d$ and predicts outputs $y \in \mathbb{R}^K$ for K classes. We assume both the classifier and the used loss, e.g., cross-entropy loss, to be differentiable. We further expect the data to lie on a manifold $\mathcal{M}$ and the loss to be constant on $\mathcal{M} \cap B(x, \epsilon)$ where $B(x, \epsilon) = \{x' \in \mathbb{R}^d : \|x' - x\| \leq \epsilon\}$ denotes the ϵ-ball around x. Let

$$g(x) = \mathbb{E}\left[\mathcal{L}(f(x), y)|x\right] \tag{4.8}$$

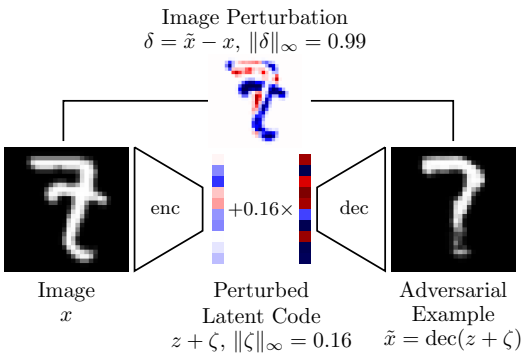


Figure 4.6: **On-Manifold Adversarial Examples:** We constrain adversarial examples to the manifold using learned, class-specific VAE-GANs [LSLW16, RLWFM17]. The perturbation ζ is obtained via Equation (4.11) and added to the latent code $z = \mathrm{enc}(x)$ yielding the adversarial example $\tilde{x} = \mathrm{dec}(z + \zeta)$ with difference $\delta = \tilde{x} - x$ in image space. On-manifold adversarial examples tend to correspond to visually more meaningful changes.

be the conditional expectation of the loss $\mathcal{L}$. Then, by the mean value theorem, there exists $\theta(x') \in [0, 1]$ for each $x' \in \mathcal{M} \cap B(x, \epsilon)$ such that

$$0 = g(x') - g(x) = \left\langle \nabla g(\theta(x')x + (1 - \theta(x'))x'), x' - x \right\rangle \tag{4.9}$$

As this holds for all $\epsilon > 0$ and as $\epsilon \to 0$, every vector $x' - x$ becomes a tangent of $\mathcal{M}$ at x and

$$\lim_{\epsilon \to 0} \nabla g(\theta(x')x + (1 - \theta(x'))x') = \nabla g(x), \tag{4.10}$$

it holds that $\nabla g(x)$ is orthogonal to the tangent space of $\mathcal{M}$ at x. As $\nabla g(x)$ is the gradient of the expected loss, it implies that adversarial examples leave the manifold $\mathcal{M}$ in order to fool the classifier $f(x)$.

4.2.3 On-Manifold Adversarial Examples

Given that regular adversarial examples leave the manifold, we intend to explicitly compute on-manifold adversarial examples. To this end, we assume our data distribution $p(x, y)$ to be conditional on the latent variables z, i.e., $p(x, y|z)$, corresponding to the underlying, low-dimensional manifold. On this manifold, however, there is no notion of "perceptual similarity" in order to ensure label invariance, i.e., distinguish valid on-manifold adversarial examples, Figure 4.1 (b), from invalid ones that change the actual, true label, Figure 4.1 (c):

Definition 1 (On-Manifold Adversarial Example)**.** Given the data distribution p, an on-manifold adversarial example for x with label y is a perturbed version $\tilde{x}$ such that $f(\tilde{x}) \neq y$ but $p(y|\tilde{x}) > p(y'|\tilde{x}) \forall y' \neq y$.

Note that the posteriors $p(y|\tilde{x})$ correspond to the true, unknown data distribution. Any on-manifold adversarial example $\tilde{x}$ that violates Definition 1 changed its actual, true label.

In practice, we assume access to an encoder and decoder modeling the (class-conditional) distributions $p(z|x, y)$ and $p(x|z, y)$ – in our case, achieved using VAE-GANs [LSLW16, RLWFM17]. Then, given the encoder enc and decoder dec and as illustrated in Figure 4.6, we obtain the latent code $z = \mathrm{enc}(x)$ and compute the perturbation ζ by maximizing:

$$\max_\zeta \mathcal{L}(f(\mathrm{dec}(z + \zeta)), y) \quad \text{s.t.} \quad \|\zeta\|_\infty \leq \eta. \tag{4.11}$$

The image-constraint, i.e., $\mathrm{dec}(z + \zeta) \in [0, 1]$, is enforced by the decoder and the η-constraint can, again, be enforced by projection. We can additionally enforce a constraint on $z + \zeta$, e.g., corresponding to a prior on z. Label invariance, as in Definition 1, is ensured by considering only class-specific encoders and decoders, i.e., the data distribution is approximated per class.

We use $\eta = 0.3$ and the same optimization procedure as for Equation (4.5). On approximated manifolds, the perturbation $z + \zeta$ is additionally constrained to $[-2, 2]^{10}$, corresponding to a truncated normal prior from the class-specific VAE-GANs. We attack 2500 test images.

On-manifold adversarial examples obtained through Equation (4.11) are similar to those crafted in [GMF+18, ACW18, ZDS18, SRBB19]. However, in contrast to [GMF+18, ACW18, SRBB19], we directly compute the perturbation ζ on the manifold instead of computing the perturbation δ in the image space and subsequently projecting $x + \delta$ to the manifold. Also note that enforcing any similarity constraint through a norm on the manifold is significantly more meaningful compared to using a norm on the image space, as becomes apparent when comparing the obtained on-manifold adversarial examples in Figure 4.3 to their regular counterparts. Compared to [ZDS18], we find on-manifold adversarial examples using a gradient-based approach instead of randomly sampling the latent space.

Figure 4.3 shows on-manifold adversarial examples for all datasets, which we found significantly harder to obtain compared to their regular counterparts. On FONTS, using the true, known class manifolds, on-manifold adversarial examples clearly correspond to transformations of the original test image – reflecting the true latent space. For the learned class manifolds, the perturbations are less pronounced, often manipulating boldness or details of the characters. Due to the approximate nature of the learned VAE-GANs, these adversarial examples are strictly speaking not always part of the true manifold – as can be seen for the irregular "A" (Figure 4.3, 6th column). On MNIST and Fashion, on-manifold adversarial examples represent meaningful manipulations, such as removing the tail of a hand-drawn "8" (Figure 4.3, 10th column) or removing the collar of a pullover (Figure 4.3, 11th column), in contrast to the random noise patterns of regular adversarial examples. However, these usually incur a smaller change in the images space which also explains why regular, unconstrained adversarial examples almost always leave the manifold. Still, on-manifold adversarial examples are perceptually close to the original images. On CelebA, the quality of on-manifold adversarial examples is clearly limited by the approximation quality of our VAE-GANs. Finally, Figure 4.5 (right) shows that on-manifold adversarial examples are closer to the manifold than regular adversarial examples – in spite of the crude approximation of the manifold on MNIST.

4.2.4 On-Manifold Robustness is Essentially Generalization

We argue that on-manifold robustness is not different from generalization: as on-manifold adversarial examples have non-zero probability under the data distribution, they are merely generalization errors. This is shown in Figure 4.7 (top left) where test error and on-manifold success rate on FONTS are shown. As expected, better generalization, i.e., using more training images N, also reduces on-manifold success rate. In order to make this relationship explicit, Figure 4.7 (bottom) plots on-manifold success rate against test error. Then, especially for FONTS and MNIST, the relationship of on-manifold robustness and generalization becomes apparent. On Fashion, the relationship is less pronounced because on-manifold adversarial examples, computed using our VAE-GANs, are not close enough to real generalization errors. However, even on Fashion, the experiments show a clear relationship between on-manifold robustness and generalization.

On-Manifold Adversarial Training Boosts Generalization: Given that generalization positively influences on-manifold robustness, we propose to adapt adversarial training to the on-manifold case in order to boost generalization:

$$\min_w \mathbb{E}\left[\max_{\|\zeta\|_\infty \leq \eta} \mathcal{L}(f(\text{dec}(z_n + \zeta); w), y_n)\right]. \tag{4.12}$$

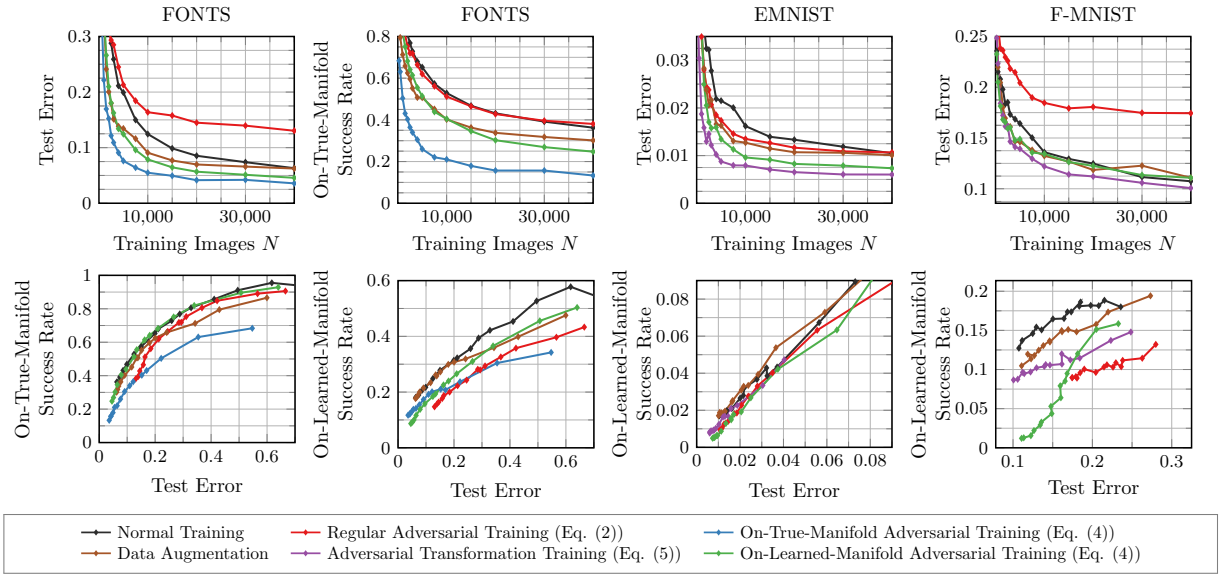


Figure 4.7: **On-Manifold Robustness and Generalization:** On-manifold robustness is strongly related to generalization, as shown on FONTS, MNIST and Fashion considering on-manifold success rate and test error. Top: Test error and on-manifold success rate shown in relation to the number of training images. As test error reduces, so does on-manifold success rate. Bottom: On-manifold success rate plotted against test error reveals the strong relationship between on-manifold robustness and generalization.

with $z_n = \text{dec}(x_n)$ being the latent codes corresponding to training images x_n. Then, on-manifold adversarial training corresponds to robust optimization w.r.t. the true, or approximated, data distribution. For example, with the perfect decoder on FONTS, the inner optimization problem finds "hard" images irrespective of their likelihood under the data distribution. For approximate dec, the benefit of on-manifold adversarial training depends on how well the true data distribution is matched, i.e., how realistic the obtained on-manifold adversarial examples are. In our case, this depends on the quality of the learned VAE-GANs.

Instead of approximating the manifold using generative models, we can exploit known invariances of the data. Then, adversarial training can be applied to these invariances, assuming that they are part of the true manifold. In practice, this can, for example, be accomplished using adversarial deformations [ETSM17, XZL+18, AAG19], i.e., adversarially crafted transformations of the image. For example, as on FONTS, we consider 6-degrees-of-freedom transformations corresponding to translation, shear, scaling and rotation:

$$\min_w \mathbb{E}\left[\max_{\|t\|_\infty \leq \eta, t \in \mathbf{m}R^6} \mathcal{L}(f(T(x_n;t);w), y_n)\right]. \tag{4.13}$$

where $T(x;t)$ denotes the transformation of image x with parameters t and the η-constraint ensures similarity and label invariance. Again, the transformations can be applied using spatial transformer networks [JSZK15] such that T is differentiable and t can be constrained to a reasonable space of transformations. We note that a similar approach has been used by Fawzi et al. [FSTF16] to boost generalization on, e.g., MNIST [LBBH98]. However, the approach was considered as an adversarial variant of data augmentation and not motivated through the lens of on-manifold robustness. We refer to Equation (4.13) as adversarial transformation training and note that, on FONTS, this approach is equivalent to on-manifold adversarial training as the transformations coincide with the actual, true manifold by construction. We also include a data augmentation baseline, where the transformations t are applied randomly.

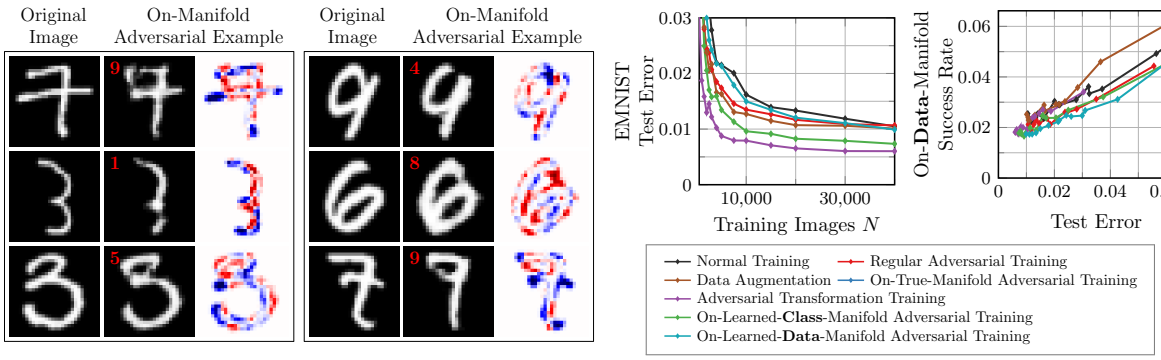


Figure 4.8: **Left: Class-Agnostic On-Manifold Adversarial Examples.** On-manifold adversarial examples crafted using class-agnostic VAE-GANs on MNIST. We show examples illustrating the problematic of unclear class boundaries within the learned manifold. On-manifold adversarial examples are not guaranteed to be label invariant, i.e., they may change the actual, true label according to the approximate data distribution. **Right: Relationship to Generalization.** Test error and on-data-manifold success rate on FONTS and MNIST. Using class-agnostic VAE-GANs, without clear class boundaries, on-manifold adversarial training loses its effectiveness – the on-manifold adversarial examples cross the true class boundaries too often. The strong relationship between on-manifold robustness and generalization can still be confirmed.

We demonstrate the effectiveness of on-manifold adversarial training in Figure 4.7. On FONTS, with access to the true manifold, on-manifold adversarial training is able to boost generalization significantly, especially for low N, i.e., few training images. Our VAE-GAN approximation on FONTS seems to be good enough to preserve the benefit of on-manifold adversarial training. On MNIST and Fashion, the benefit reduces with the difficulty of approximating the manifold; this is the "cost" of imperfect approximation. While the benefit is still significant on MNIST, it diminishes on Fashion. However, both on MNIST and Fashion, identifying invariances and utilizing adversarial transformation training recovers the boost in generalization, especially in contrast to the random data augmentation baseline. Overall, on-manifold adversarial training is a promising tool for improving generalization and we expect its benefit to increase with better generative models.

From Class Manifolds to Data Manifold: So far, we considered approximating the manifold using class-*specific* VAE-GANs. Instead, we can also train class-agnostic VAE-GANs where the marginals $p(x)$ are approximated instead of the class-conditionals $p(x|y)$. This means that images from different classes are embedded in the same latent space. Then, however, ensuring label invariance, as required by Definition 1 becomes more difficult. We attempt to ensure label invariance through a particularly small L_∞-constraint on the perturbation, specifically $\|\zeta\|_\infty \leq \eta$ with $\eta = 0.1$. Still, as can be seen in Figure 4.8 (left), on-manifold adversarial examples might cross class boundaries, i.e., they change their actual label rendering them invalid according to our definition.

In Figure 4.8 (right), we clearly distinguish between on-*class*-manifold and on-*data*-manifold adversarial training, corresponding to the used class-specific or -agnostic VAE-GANs. Robustness, however, is measured w.r.t. on-data-manifold adversarial examples. As can be seen, the positive effect of on-manifold adversarial training diminishes when using on-data-manifold adversarial examples during training. Both, on FONTS and MNIST, generalization slightly decreases in comparison to normal training because adversarial examples are not useful for learning the task if label invariance cannot be ensured. When evaluating robustness against

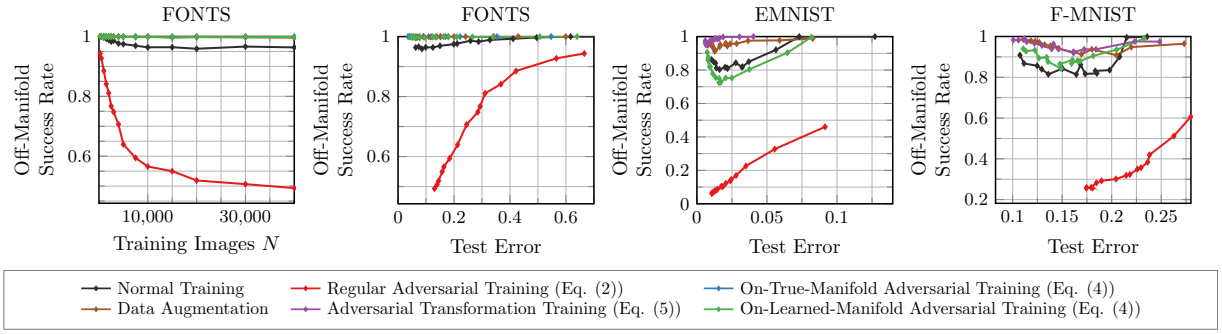


Figure 4.9: **Adversarial Robustness and Generalization:** Regular robustness is not related to generalization, as demonstrated on FONTS, MNIST and Fashion considering test error and (regular) success rate. On FONTS (left), success rate is not influenced by test error, except for adversarial training. Plotting success rate against test error highlights the independence of robustness and generalization; however, different training strategies exhibit different robustness-generalization characteristics.

on-data-manifold adversarial examples, however, the relation of on-data-manifold robustness to generalization can clearly be seen. Overall, this shows that this relationship also extends to more general, less strict definitions of on-manifold adversarial examples.

4.2.5 Regular Robustness is Independent of Generalization

We argue that generalization, as measured *on* the manifold w.r.t. the data distribution, is mostly independent of robustness against regular, possibly off-manifold, adversarial examples when varying the amount of training data. Specifically, in Figure 4.9 (left) for FONTS, it can be observed that – except for adversarial training – the success rate is invariant to the test error. This can best be seen when plotting the success rate against test error for different numbers of training examples, c.f. Figure 4.9 (middle left): only for adversarial training there exists a clear relationship. For the remaining training schemes success rate is barely influenced by the test error. In particular, better generalization does not worsen robustness. Similar behavior can be observed on MNIST and Fashion, see Figure 4.9 (right). Here, it can also be seen that different training strategies exhibit different characteristics w.r.t. robustness and generalization. Overall, regular robustness and generalization are not necessarily contradicting goals.

As mentioned in Section 4.1, these findings are in contrast to related work [SZC+18, TSE+19] claiming that an inherent trade-off between robustness and generalization exists. For example, Tsipras et al. [TSE+19] use a synthetic toy dataset to theoretically show that no model can be both robust and accurate (on this dataset). However, they allow the adversary to produce perturbations that change the actual, true label w.r.t. the data distribution. That is, the considered adversarial examples are not adversarial examples according to Definition 1, as discussed in detail below. Thus, it is unclear whether the suggested trade-off actually exists for real datasets. Our experiments, at least, seem to indicate the contrary. Similarly, Su et al. [SZC+18] experimentally show a trade-off between adversarial robustness and generalization by studying different models on ImageNet [RDS+15]. However, Su et al. compare the robustness and generalization characteristics of different models (i.e., different architectures, training strategies etc.), while we found that the generalization performance does not influence robustness for any *arbitrary, but fixed* model.

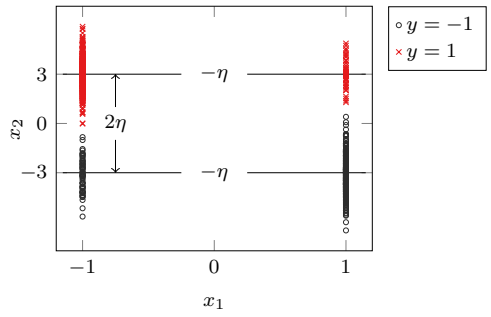


Figure 4.10: **Illustration of the Toy Dataset of [TSE+18]:** For labels $y = 1$ and $y = -1$, the two-dimensional observations $x \in \{-1, 1\} \times \mathbb{R}$ are plotted. The first dimension x_1 mirrors the label with probability 0.9; the second dimension x_2 is drawn from a Gaussian $\mathcal{N}(y3, I)$, i.e., η from the text is 3. As illustrated on the left, perturbing an observation x with label $y = 1$ but $x_1 = -1$ by $2\eta = 6$ results in an adversarial example $\tilde{x}$ indistinguishable from observations with label $y = -1$.

Discussion of [TSE+19]: Tsipras et al. argue that there exists an inherent trade-off between regular robustness and generalization based on a simple toy example. We follow the notation of [TSE+19], but restrict ourselves to the two-dimensional case. Note that [TSE+19] also considers the general d-dimensional case. The conclusions, however, remain the same. Specifically, for labels $y = 1$ and $y = -1$ with $p(y = 1) = p(y = -1) = 0.5$, the observations $x \in \{-1, 1\} \times \mathbb{R}$ are drawn as follows:

$$p(x_1|y) = \begin{cases} p & \text{if } x_1 = y \\ 1 - p & \text{if } x_1 = -y \end{cases} \qquad p(x_2|y) = \mathcal{N}(x_2; y\eta, 1), \tag{4.14}$$

where η defines the degree of overlapping between the two classes and $p \geq 0.5$. Figure 4.10 illustrates this dataset for $p = 0.9$ and $\eta = 3$. For a L_∞-bounded adversary with $\epsilon \geq 2\eta$, Tsipras et al. show that no model can be both accurate and robust. Specifically, for x with $y = 1$ but $x_1 = -1$ and $x_2 = \eta$, we consider replacing x_2 with $\tilde{x}_2 = x_2 - 2\eta = -\eta$, as considered in [TSE+19]. However, this adversary does not produce proper adversarial examples according to Definition 1. Note that, in this case and for $p < 1$, $p(x) > 0$ everywhere, meaning that our definition of on-manifold adversarial examples applies. Indeed,

$$\begin{aligned} p(y = 1|x = \tilde{x}) &= p(y = 1|x_1 = -1) \cdot p(y = 1|x_2 = -\eta) = (1 - p) \cdot \mathcal{N}(x_2 = -\eta; \eta, 1) \\ &\not> p \cdot \mathcal{N}(x_2 = -\eta; -\eta, 1) \\ &= p(y = -1|x_1 = -1) \cdot p(y = -1|x_2 = -\eta) = p(y = -1|x = \tilde{x}) \end{aligned} \tag{4.15}$$

which contradicts our definition. Thus, the suggested trade-off is questionable. However, we note that this argument explicitly depends on our definition of proper and invalid adversarial examples. Other definitions of adversarial examples or adversarial robustness, e.g., in the context of the adversarial loss defined in [TSE+19], may lead to different conclusions.

4.2.6 Discussion

Our results imply that robustness and generalization are not necessarily conflicting goals, as believed in related work [SZC+18, TSE+19]. This means, in practice, for any arbitrary but fixed model, better generalization will not worsen regular robustness. Different models (architectures, training strategies etc.) might, however, exhibit different robustness and generalization characteristics, as also shown in [RGB16, SZC+18]. For adversarial training, on regular adversarial examples, the commonly observed trade-off between robustness and generalization is explained by the tendency of adversarial examples to leave the manifold. As a result, the network has to learn (seemingly) random, but adversarial, noise patterns *in addition* to the actual task at hand, rendering the learning problem harder. On simple datasets, such as MNIST,

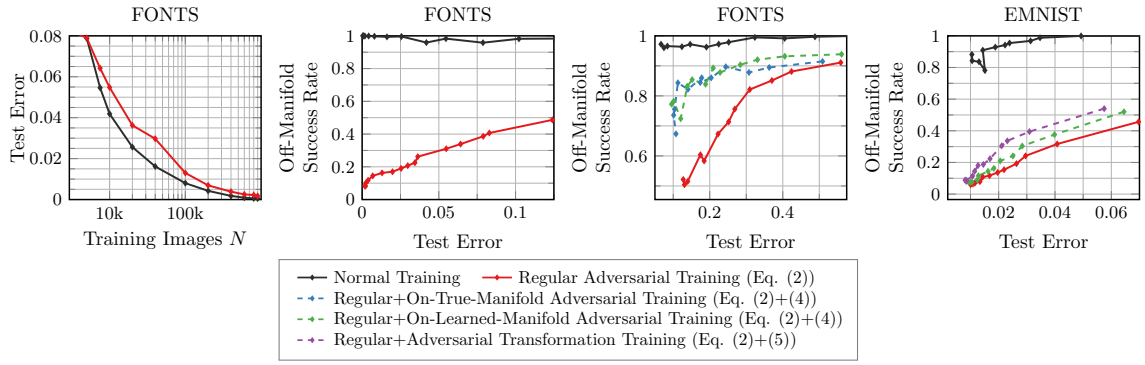


Figure 4.11: **Sample Complexity of Adversarial Training:** Adversarial training on regular adversarial examples, potentially leaving the manifold, renders the learning problem more difficult. Top: With roughly 1.5 to 2 times the training data, adversarial training can still reach the same accuracy as normal training; results for ResNet-13 [HZRS16a]. Bottom: Additionally, the trade-off can be controlled by combining regular and on-manifold adversarial training; results averaged over 3 models.

these adversarial directions might avoid overfitting. On harder tasks, e.g., FONTS or Fashion, the discrepancy in test error between normal and adversarial training increases. Our results also support the hypothesis that regular adversarial training has higher sample complexity [SST+18, KH18]. In fact, on FONTS, adversarial training can reach the same accuracy as normal training with roughly twice the amount of training data, as demonstrated in Figure 4.11 (left). Furthermore, as illustrated in Figure 4.11 (right), the trade-off between regular robustness and generalization can be controlled by combining regular and on-manifold adversarial training, i.e. boost generalization while reducing robustness.

The presented results can also be confirmed on more complex datasets, such as CelebA, and using different threat models, i.e., attacks. On CelebA, where VAE-GANs have difficulties approximating the manifold, Figure 4.12 (left) shows that on-manifold robustness still improves with generalization although most on-manifold adversarial examples are not very realistic, see Figure 4.3. Similarly, regular robustness is not influenced by generalization. Here, we also show that the average distance of the perturbation, i.e., average $\|\delta\|_\infty$, when used to

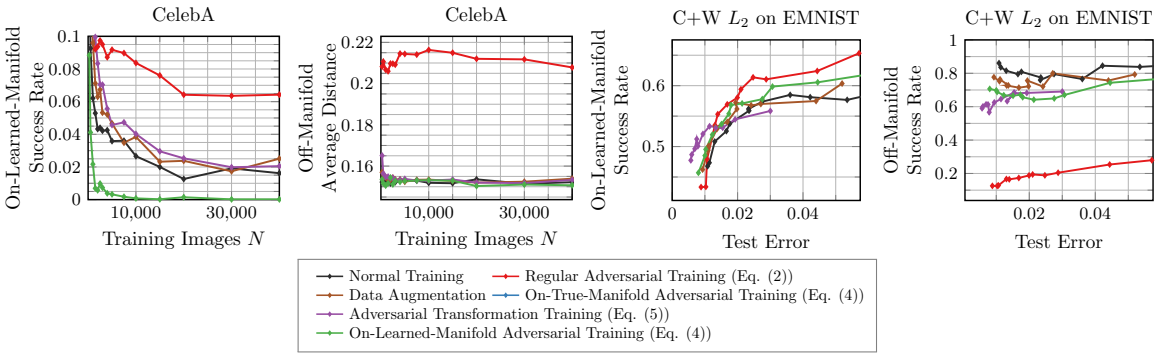


Figure 4.12: **Results on CelebA and using Carlini and Wagner Adversarial Examples:** On CelebA, as the class manifolds are significantly harder to approximate, the benefit of on-manifold adversarial training diminishes. For [CW17b], we used 120 iterations. Our hypotheses are confirmed, although [CW17b] does not use the training loss as attack objective and the L_2 norm changes the similarity-constraint for regular and on-manifold adversarial examples.

assess robustness leads to the same conclusions. Similarly, as shown in Figure 4.12 (right), our findings are confirmed using Carlini and Wagner's attack [CW17b] with L_2-norm – to show that the results generalize across norms. However, overall, we observed lower success rates using [CW17b] and the L_2 norm. Our results can also be reproduced using transfer attacks (i.e., black-box attacks, which are generally assumed to be subsumed by white-box attacks [ACW18]). However, results are generally less pronounced due to significantly lower success rates of transfer attacks. Finally, these observations can be confirmed using different architectures such as multi-layer perceptrons, ResNets [HZRS16a] and VGG [SZ15]. These results can be found in Appendix A.

4.3 Conclusion

In this chapter, we intended to disentangle the relationship between adversarial robustness and generalization by initially adopting the hypothesis that robustness and generalization are contradictory [SZC+18, TSE+19]. By considering adversarial examples in the context of the low-dimensional, underlying data manifold, we formulated and experimentally confirmed four assumptions. First, we showed that regular adversarial examples indeed leave the manifold, as widely assumed in related work [TG16, IJA+17, GMF+18, PS18, SRBB19]. Second, we demonstrated that adversarial examples can also be found on the manifold, so-called on-manifold adversarial examples, even if the manifold has to be approximated, e.g., using VAE-GANs [LSLW16, RLWFM17]. Third, we established that robustness against on-manifold adversarial examples is clearly related to generalization. Our proposed on-manifold adversarial training exploits this relationship to boost generalization using an approximate manifold, or known invariances. Fourth, we provided evidence that robustness against regular, unconstrained adversarial examples and generalization are not necessarily contradicting goals: for any arbitrary but fixed model, better generalization, e.g., through more training data, does not reduce robustness.

5 Relating Adversarially Robust Generalization to Flat Minima

Contents

Adversarial training (AT) has become the de facto standard to obtain models robust against adversarial examples. Besides the robustness-accuracy trade-off discussed in the previous chapter, AT also exhibits severe *robust overfitting*: cross-entropy loss on adversarial examples, so-called robust loss, decreases continuously on training examples, while eventually increasing on test examples. In practice, this leads to poor generalization of robustness to new examples. To address this phenomenon, in this chapter, we study the relationship between robust generalization and flatness of the robust loss landscape in weight space, i.e., whether robust loss changes significantly when perturbing weights. As flat minima have been argued to be beneficial for generalization, we propose average- and worst-case metrics to measure flatness in the robust loss landscape and show a **correlation between good robust generalization and flatness**. For example, throughout training, flatness reduces significantly during overfitting such that early stopping effectively finds flatter minima in the robust loss landscape. Similarly, AT variants achieving higher adversarial robustness also correspond to flatter minima. This holds for many popular choices, e.g., AT-AWP [WXW20b], TRADES [ZYJ+19], AT with self-supervision [HMKS19] or additional unlabeled examples [CRS+19], as well as simple regularization techniques, e.g., AutoAugment [CZM+19], weight decay or label noise. For fair comparison, our flatness measures are specifically designed to be scale-invariant, and we conduct extensive experiments to validate our findings.

This chapter is based on [SHS21]: As first author, David Stutz conducted all experiments and was the main writer of the paper. A short version was presented at the Workshop on Uncertainty and Robustness in Deep Learning (UDL) held in conjunction with ICML 2021, the Workshop on Adversarial Learning Methods for Machine Learning and Data Mining (AdvML) during KDD 2021, and the Workshop on Adversarial Machine Learning in Real-World Computer Vision Systems and Online Challenges (AML-CV) at CVPR 2021. Invited talks were given at the machine learning seminar organized by the Max Planck Institute for Mathematics in the Sciences and the University of California Los Angeles as well as at the machine learning security seminar series organized by the University of Cagliari.

Code: The source for this chapter is available on GitHub[1].

[1] https://github.com/davidstutz/iccv2021-robust-flatness

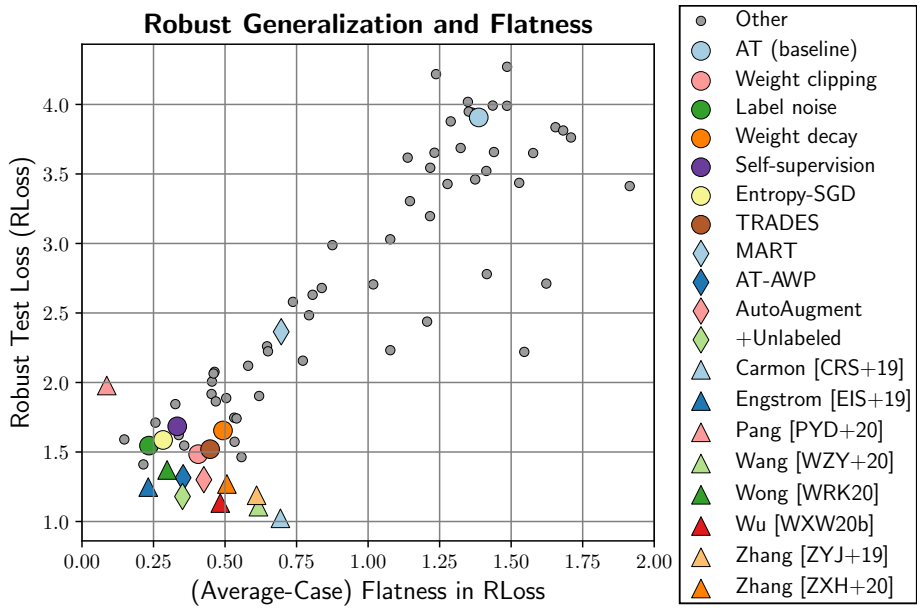


Figure 5.1: **Robust Generalization and Flatness:** Robust loss (RLoss, lower is more robust, y-axis), i.e., cross-entropy loss on PGD adversarial examples [MMS+18], against our *average-case flatness* measure of RLoss in weight space (lower is "flatter", x-axis). Popular AT variants improving adversarial robustness on CIFAR10, e.g., TRADES [ZYJ+19], AT-AWP [WXW20b] or AT with self-supervision [HMKS19]/unlabeled examples [CRS+19], also correspond to flatter minima. Vice-versa, regularization explicitly improving flatness, e.g., Entropy-SGD [CCS+17], weight decay or weight clipping [SCHS21a], also improve robustness. Across all models, there is a **clear relationship between good robust generalization and flatness in RLoss.** ●,♦ Our models, w/o early stopping. ▲ RobustBench [CAS+20a] models *w/* early stopping.

5.1 Introduction

In order to obtain robustness against adversarial examples [SZS+14], *adversarial training (AT)* [MMS+18] augments training with adversarial examples that are generated on-the-fly. While many different variants have been proposed, AT is known to require more training data [KH18, SST+18], generally leading to generalization problems [FZT19]. In fact, *robust overfitting* [RWK20] has been identified as the main problem in AT: adversarial robustness on test examples eventually starts to decrease, while robustness on training examples continues to increase. This is typically observed as increasing *robust loss (RLoss)* or *robust test error (RErr)*, i.e., (cross-entropy) loss and test error on adversarial examples. As a result, the *robust generalization gap*, i.e., the difference between test and training robustness, tends to be very large. In [RWK20], early stopping is used as a simple and effective strategy to avoid robust overfitting. However, despite recent work tackling robust overfitting [WXW20b, SSJF21, HLOB21], it remains an open and poorly understood problem.

In *"clean"* generalization (i.e., on natural examples), overfitting is well-studied and commonly tied to flatness of the loss landscape in weight space, both visually [LXTG18] and empirically [NBMS17, KMN+17, JNM+20]. In general, the optimal weights on test examples do not coincide with the minimum found on training examples. Flatness ensures that the loss does *not* increase significantly in a neighborhood around the found minimum. Therefore, flatness leads to good generalization because the loss on test examples does not increase significantly (i.e., small generalization gap). [LXTG18] showed that *visually* flatter minima cor-

respond to better generalization. [NBMS17] and [KMN+17] formalize this idea by measuring the change in loss within a local neighborhood around the minimum considering random [NBMS17] or "adversarial" weight perturbations [KMN+17]. These measures are shown to be effective in predicting generalization in a recent large-scale empirical study [JNM+20] and explicitly encouraging flatness during training has been shown to be successful in practice [CCS+17, IPG+18, CS19, LSPJ20, ZZM21].

Recently, [WXW20b] applied the idea of flat minima to AT: through *adversarial weight perturbations*, AT is regularized to find flatter minima of the *robust* loss landscape. This reduces the impact of robust overfitting and improves robust generalization, but does not *avoid* robust overfitting. As a result, early stopping is still necessary. Furthermore, flatness is only assessed *visually* and it remains unclear whether flatness does actually improve in these adversarial weight directions. Similarly, [GQU+20] shows that weight averaging [IPG+18] can improve robust generalization, indicating that flatness might be beneficial in general. This raises the question whether other "tricks" [GQU+20, PYD+21], e.g., different activation functions [SSJF21] or label smoothing [SVI+16], or approaches such as AT with self-supervision [HMKS19]/unlabeled examples [CRS+19] are successful *because of* finding flatter minima.

Contributions: This chapter studies **whether flatness of the robust loss (RLoss) in weight space improves robust generalization**. To this end, we propose both average- and worst-case flatness measures for the *robust* case, thereby addressing challenges such as scale-invariance [DPBB17], estimation of RLoss on top or jointly with weight perturbations, and the discrepancy between RLoss and RErr. We show that **robust generalization generally improves alongside flatness** and vice-versa: Figure 5.1 plots RLoss (lower is more robust, y-axis) against our average-case flatness in RLoss (lower is flatter, x-axis), showing a clear relationship. In contrast to [WXW20b], not providing empirical flatness measures, our results show that this relationship is stronger for average-case flatness. This trend covers a wide range of AT variants on CIFAR10, e.g., AT-AWP [WXW20b], TRADES [ZYJ+19], MART [WZY+20], AT with self-supervision [HMKS19] or additional unlabeled examples [CRS+19, AUH+19], as well as various regularization schemes, including AutoAugment [CZM+19], label smoothing [SVI+16] and noise or weight clipping [SCHS21a]. Furthermore, we consider hyperparameters, e.g., learning rate schedule, weight decay, batch size, or different activation functions [HG16, EUD18, Mis20], and methods explicitly improving flatness, e.g., Entropy-SGD [CCS+17] or weight averaging [IPG+18].

5.2 Robust Generalization and Flat Minima

We study robust generalization and overfitting in the context of flatness of the *robust* loss landscape in weight space, i.e., w.r.t. changes in the weights. While flat minima have consistently been linked to standard generalization [HS97, NBMS17, KMN+17, LXTG18], this relationship remains unclear for adversarial robustness. We start by briefly introducing the robust overfitting phenomenon (Section 5.2.1) and discussing problems in judging flatness visually [LXTG18] (Section 5.2.2). Then, we are inspired by [KMN+17, NBMS17] and introduce average- and worst-case flatness measures based on the change in robust loss along random or adversarial weight directions in a local neighborhood (Section 5.2.3), c.f. Figure 5.3. We also discuss the connection of flatness to the Hessian eigenspectrum [YGL+18] and the importance of scale-invariance as in [DPBB17].

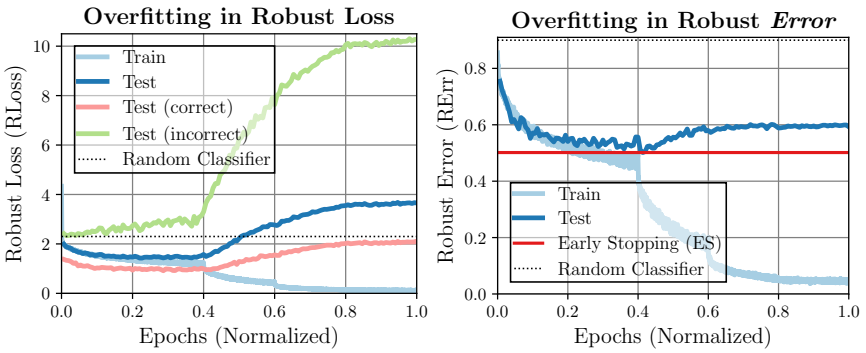


Figure 5.2: **Robust Overfitting:** Robust cross-entropy loss (RLoss) and robust error (RErr) over epochs (normalized by 150 epochs) for AT, using a ResNet-18 on CIFAR10 (c.f. Section 5.3), to illustrate *robust* overfitting. **Left:** Training RLoss (light blue) reduces continuously throughout training, while test RLoss (dark blue) eventually increases again. We also highlight that robust overfitting is *not* limited to incorrectly classified examples (green), but also affects correctly classified ones (rose). **Right:** Similar behavior, but less pronounced, can be observed considering RErr. We also show RErr obtained through early stopping (red).

5.2.1 Adversarial Training and Robust Overfitting

Adversarial Training (AT): Let f be a (deep) neural network taking input $x \in [0,1]^D$ and weights $w \in \mathbb{R}^W$ and predicting a label $f(x;w)$. Given a true label y, an adversarial example is a perturbation $\tilde{x} = x + \delta$ such that $f(\tilde{x};w) \neq y$. The perturbation δ is intended to be nearly invisible which is, in practice, enforced using a L_p constraint: $\|\delta\|_p \leq \epsilon$. To obtain robustness against these perturbations, adversarial training injects adversarial examples during training:

$$\min_w \mathbb{E}_{x,y} \left[\max_{\|\delta\|_p \leq \epsilon} \mathcal{L}(f(x+\delta;w),y) \right] \tag{5.1}$$

where $\mathcal{L}$ denotes the cross-entropy loss. The outer minimization problem can be solved using regular stochastic gradient descent (SGD) on mini-batches. To compute adversarial examples, the inner maximization problem is tackled using projected gradient descent (PGD) [MMS+18]. Here, we focus on $p = \infty$ as this constrains the maximum change per feature/pixel, e.g., $\epsilon = 8/255$ on CIFAR10. For evaluation (at test time), we consider both robust loss (RLoss) $\max_{\|\delta\|_\infty \leq \epsilon} \mathcal{L}(f(x+\delta;w),y)$, approximated using PGD, and robust test error (RErr), computed using AutoAttack [CH20c]. Note that AutoAttack stops when adversarial examples are found and does *not* maximize cross-entropy loss, rendering it unfit to estimate RLoss.

Robust Overfitting: Following [RWK20], Figure 5.2 illustrates the problem of *robust* overfitting, plotting RLoss (left) and RErr (right) over epochs, which we normalize by the total number of epochs for clarity. Shortly after the first learning rate drop (at epoch 60, i.e., 40% of training), test RLoss and RErr start to increase significantly, while robustness on training examples continues to improve. Robust overfitting was shown to be independent of the learning rate schedule [RWK20] and, as we show (Section 5.3.1), occurs across various different activation functions as well as many popular AT variants. In contrast to [RWK20], mostly focusing on RErr, Figure 5.2 shows that RLoss overfits more severely, indicating a "disconnectedness" between RLoss and RErr that we consider in detail later. For now, RLoss and RErr do clearly not move "in parallel" and RLoss, reaching values around 4, is higher than for a random classifier (which is possible considering *adversarial* examples). This is primarily due to an extremely high RLoss on incorrectly classified test examples (which are "trivial" adversarial

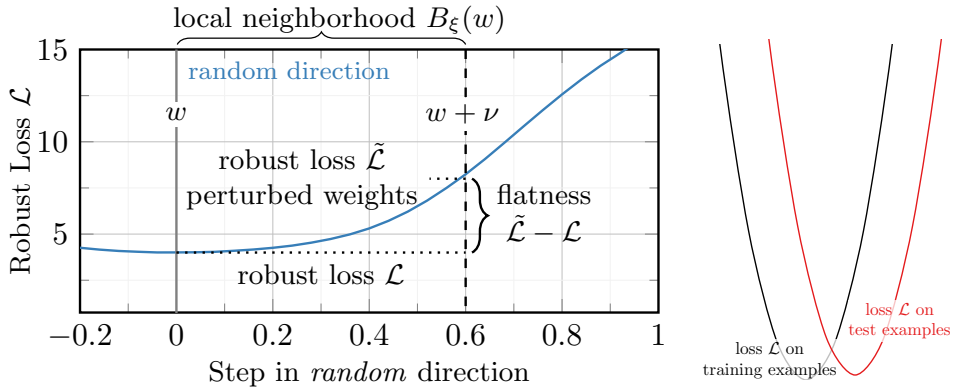


Figure 5.3: **Measuring Flatness. Left:** Illustration of measuring flatness in a random (i.e., average-case, blue) direction by computing the difference between RLoss $\tilde{\mathcal{L}}$ *after* perturbing weights (i.e., $w + v$) and the "reference" RLoss $\mathcal{L}$ given a local neighborhood $B_\xi(w)$ around the found weights w, see Section 5.2.3. In practice, we average across/take the worst of several random/adversarial directions. **Right:** Large changes in RLoss around the "sharp" minimum causes poor generalization from training (black) to test examples (red).

examples). We emphasize, however, that robust overfitting also occurs on correctly classified test examples.

5.2.2 Intuition and Visualizing Flatness

For judging robust flatness, we consider how RLoss changes w.r.t. random or adversarial perturbations in the weights w. Generally, we expect flatter minima to generalize better as the loss does not change significantly within a small neighborhood around the minimum, i.e., the found weights. Then, even if the loss landscape on test examples does not coincide with the loss landscape on training examples, loss remains small, ensuring good generalization. The contrary case, i.e., that sharp minima generalize poorly is illustrated in Figure 5.3 (right). Before considering to *measure* flatness, we discuss the easiest way to "judge" flatness: visual inspection of the RLoss landscape along random or adversarial directions in weight space.

In [LXTG18], loss landscape is visualized along *normalized* random directions. Normalization is important to handle different scales, i.e., weight distributions, and allow comparison across models. We follow [WXW20b] and perform *per-layer* normalization: Letting $v \in \mathbb{R}^W$ be a direction in weight space, it is normalized as

$$\hat{v}^{(l)} = \frac{v^{(l)}}{\|v^{(l)}\|_2} \|w^{(l)}\|_2 \quad \text{for layer } l. \tag{5.2}$$

This is in contrast to [LXTG18] where a filter-wise normalization is used. However, we found this to make no difference in practice. Moreover, we also consider biases, treating them as individual layer, but we exclude batch normalization parameters. Then, the loss landscape is visualized in 51 discrete (evenly spaced) steps along this direction, i.e., $w + s\hat{v}$ for $s \in [-1, 1]$. Adversarial examples are computed "on-the-fly", i.e., for each $w + s\hat{v}$ individually, to avoid underestimating RLoss as in [YLW+18, PYXW19]. The result is indeed scale-invariant: Figure 5.4 (top) shows that the loss landscapes for scaled versions (factors 0.5 or 2) of our AT baseline coincide with the original landscape. Here, we additionally scale the weights by factors 0.5 and 0.025 for random/Hessian eigenvalue and adversarial directions, respectively. This essentially "zooms in" and is particularly important when visualizing along adversarial

Model	RErr ↓
AT (baseline) ●	62.8
Scaled ×0.5 ●	62.8
Scaled ×2 ●	62.8
MiSH ●	59.8
Batch size 8 ●	58.2
Adam ●	57.5

Model	RErr ↓
Label smoothing ●	61.2
MART ●	61
Entropy-SGD ●	58.6
Self-supervision ●	57.1
TRADES ●	56.7
AT-AWP ●	54.3

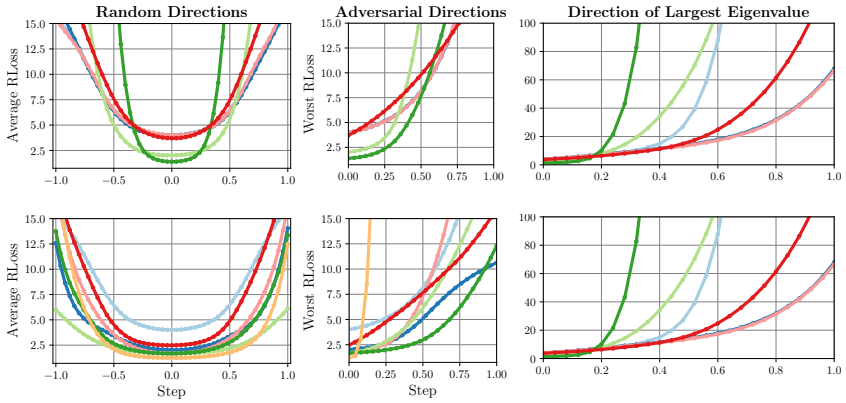


Figure 5.4: **Visualizing Flatness:** RLoss landscape across 10 random/adversarial directions or in the direction of the largest Hessian eigenvalue estimated on a batch of 128 test examples. **Top:** Our AT baseline (ResNet-18) and scaled variants (×2 and ×0.5). Training with smaller batch size or Adam [KB15b] improves adversarial robustness (lower RErr vs. AutoAttack [CH20c]) but does *not* result in *visually* flatter minima. **Bottom:** AT-AWP [WXW20b] or Entropy-SGD [CCS+17] improve robustness *and* visual flatness in random directions. In adversarial directions, however, AT-AWP looks very sharp. Overall, visual inspection does not provide a clear, objective picture of flatness.

directions. However, Figure 5.4 also illustrates that judging flatness visually is difficult: Considering random weight directions, AT with Adam [KB15b] or small batch size improves adversarial robustness, but the found minima look less flat (top). For other approaches, e.g., TRADES [ZYJ+19] or AT-AWP [WXW20b], results look indeed flatter while also improving robustness (bottom). In adversarial directions, in contrast, AT-AWP looks particularly sharp. Furthermore, not only flatness but also the vertical "height" of the loss landscape matters, and it is impossible to tell "how much" flatness is necessary.

5.2.3 Average- and Worst-Case Flatness Measures

In order to objectively measure and compare flatness, we draw inspiration from [NBMS17, KMN+17] and propose average- and worst-case flatness measures adapted to the robust loss. We emphasize that measuring flatness in RLoss is non-trivial and flatness in (clean) Loss *cannot* be expected to correlate with robustness. For example, we need to ensure scale-invariance [DPBB17] and estimate RLoss *on top* of random or adversarial weight perturbations:

Average-Case / Random Flatness: Considering random weight perturbations $v \in B_\xi(w)$ within the ξ-neighborhood of w, average-case flatness is computed as

$$\mathbb{E}_v[\max_{\|\delta\|_\infty \leq \epsilon} \mathcal{L}(f(x+\delta; w+v), y)] - \max_{\|\delta\|_\infty \leq \epsilon} \mathcal{L}(f(x+\delta; w), y) \tag{5.3}$$

averaged over test examples x, y, as illustrated in Figure 5.3. We define $B_\xi(w)$ using *relative* L_2-balls per layer (c.f. Equation (5.2)):

$$B_\xi(w) = \{w + v : \|v^{(l)}\|_2 \leq \xi \|w^{(l)}\|_2 \forall \text{ layers } l\}. \tag{5.4}$$

This ensures scale-invariance w.r.t. the weights as $B_\xi(w)$ scales with the weights on a *per-layer* basis. Note that the second term in Equation (5.3), i.e., the "reference" robust loss, is important to make the measure independent of the absolute loss (i.e., corresponding to the vertical shift

in Figure 5.4, left). Sampling in $B_\xi(w)$ is accomplished by sampling individually per layer. That is, for each layer l, we compute $\xi' := \xi \cdot \|w^{(l)}\|_2$ given the original weights w. Then, a random vector $v^{(l)}$ with $\|v^{(l)}\|_2 \leq \xi'$ is sampled. This is done for each layer, handling weights and biases as separate layers, but ignoring batch normalization [IS15b] parameters. In practice, ξ can be as large as 0.5. We refer to Equation (5.3) as **average-case flatness in RLoss**.

Worst-Case / Adversarial Flatness: [WXW20b] explicitly optimizes flatness in *adversarial weight* directions and shows that average-case flatness is not sufficient to improve adversarial robustness. As it is unclear whether [WXW20b] actually improves worst-case flatness, we define

$$\max_{v \in B_\xi(w)} \left[\max_{\|\delta\|_\infty \leq \epsilon} \mathcal{L}(f(x+\delta; w+v), y) \right] - \max_{\|\delta\|_\infty \leq \epsilon} \mathcal{L}(f(x+\delta; w), y) \tag{5.5}$$

as **worst-case flatness in RLoss**. Here, the expectation over v in Equation (5.3) is replaced by a maximum over $v \in B_\xi(w)$, considering smaller ξ. We use the same definition of $B_\xi(w)$ as above (aligned with [WXW20b]). Regarding *standard* performance, this worst-case notion of flatness has been shown to be a reliable predictor of generalization [JNM+20, KMN+17]. For computing Equation (5.5) in practice, we jointly optimize over v and δ (for each batch individually) using PGD. This means, after random initialization of δ_b, $\forall b = 1, \ldots, B$, and $v \in B_\xi(w)$, each iteration computes and applies updates

$$\Delta_v = \nabla_v \sum_{b=1}^{B} \mathcal{L}(f(x_b + \delta_b; w + v), y_b) \quad \Delta_{\delta_b} = \nabla_{\delta_b} \sum_{b=1}^{B} \mathcal{L}(f(x_b + \delta_b; w + v), y_b) \tag{5.6}$$

before projecting δ_b and v onto the constraints $\|\delta_b\|_\infty \leq \epsilon$ and $\|v^{(l)}\|_2 \leq \xi \|w^{(l)}|_2$. The latter projection is applied in a per-layer basis. As illustrated in Figure 5.4, RLoss increases quickly along adversarial directions, even for very small values of ξ, e.g., $\xi = 0.005$.

Flatness of Clean Loss Landscape: We can also consider both Equation (5.3) and Equation (5.5) on the *clean* (cross-entropy) loss ("Loss"), i.e.,

$$\mathcal{L}(f(x, w+v), y) \quad \text{instead of} \quad \max_{\|\delta\|_\infty \leq \epsilon} \mathcal{L}(f(x+\delta, w+v), y). \tag{5.7}$$

We note that RLoss is an upper bound of (clean) Loss. Thus, flatness in RLoss and Loss are connected. However, we found that measuring flatness in RLoss is essential in order to judge *robust* generalization, as detailed later in Section 5.3.

5.2.4 Discussion

In the context of flatness, there has also been some discussion concerning the meaning of Hessian eigenvalues [LXTG18, YGL+18] as well as concerns regarding the scale-invariance of flatness measures [DPBB17]. First, regarding the Hessian eigenspectrum, [YGL+18] shows that large Hessian eigenvalues indicate poor adversarial robustness. However, Hessian eigenvalues are generally *not* scale-invariant (which is acknowledged in [YGL+18]): Our AT baseline has a maximum eigenvalue of 1990 which reduces to 505 when *up-scaling* the model and increases to 7936 when *down-scaling*, without affecting robustness (c.f. $\times 0.5$ and $\times 2$ in Figure 5.4). We also found that the largest eigenvalue is *not* correlated with adversarial robustness. Similarly, we did *not* find a strong correlation between the fraction $|\lambda_{\min}|/|\lambda_{\max}|$ of smallest and largest eigenvalue and adversarial robustness, as proposed in [LXTG18]. The results are summarized

Model	RErr ↓	$\lambda_{\max}$	$\frac{\lvert\lambda_{\min}\rvert}{\lvert\lambda_{\max}\rvert}$
AT (baseline)	62.8	1990	0.088
Scaled $\times 0.5$	62.8	7936	0.088
Scaled $\times 2$	62.8	505	0.088
Batch size 8	58.2	3132	0.027
Adam	57.5	540	0.047
Label smoothing	61.2	2484	0.085
Self-supervision	57.1	389	0.041
Entropy-SGD	58.6	5773	0.054
TRADES	56.7	947	0.089
MART	61	1285	0.087
AT-AWP	54.3	1200	0.241

Table 5.1: **Hessian Eigenvalues and Convexity:** We report RErr against AutoAttack [CH20c], the maximum Hessian eigenvalue $\lambda_{\max}$ and the convexity measure of [LXTG18] computed as $|\lambda_{\min}|/|\lambda_{\max}|$. This fraction is supposed to quantify the degree of non-convexity around the found minimum. As can be seen, neither $\lambda_{\max}$ nor convexity correlate well with adversarial robustness. Regarding $\lambda_{\max}$ this is due to the Hessian eigenspectrum not being scale-invariant, as shown for scaled versions ($\times 0.5$ and $\times 2$) of our AT baseline.

in Table 5.1. Also due to these shortcomings, we decided not to attempt and compute the Hessian based on adversarial examples for further experiments.

Second, following a similar train of thought, [DPBB17] criticizes the flatness measures of [NBMS17, KMN+17] as not being scale-invariant. That is, through clever scaling of weights, without changing predictions, arbitrary flatness values can be "produced". However, the analysis in [DPBB17] does not take into account the relative neighborhood as defined in [KMN+17], which renders the measure explicitly scale-invariant. This also applies to our definition of $B_{\xi}(w)$ in Equation (5.4) and is shown in Figure 5.4 where normalization is performed relative (per-layer) to the weights. In detail, scaling the used ResNet-18 was accomplished by scaling only the convolutional layers followed by batch normalization – batch normalization essentially "cancels" out the scaling after re-calibrating the statistics. As $B_{\xi}(w)$ is defined *per-layer*, *relative* to w, the neighborhood increases alongside the weights. This also applies to the example of [DPBB17], scaling up the first layer of a two-layer ReLU network (without batch normalization) by α and scaling down the second layer by $1/\alpha$.

5.3 Experiments

We start with a closer look at RLoss in robust overfitting (Section 5.3.1, Figure 5.5). Then, we show a strong correlation between good robust generalization and flatness (Section 5.3.2). For example, robust overfitting causes sharper minima (Figure 5.6). More importantly, more robust models generally find flatter minima and, vice-versa, methods encouraging flatness improve adversarial robustness (Figure 5.7 and 5.9). In fact, flatness improves robust generalization by lowering the robust generalization gap and avoiding robust overfitting (Figure 5.10).

Setup: On CIFAR10 [Kri09], our *AT baseline* uses ResNet-18 [HZRS16a] with batch normalization [IS15b] and ReLU activations, trained for 150 epochs with batch size 128, learning rate 0.05, reduced by factor 0.1 at 60, 90 and 120 epochs, using weight decay 0.005 and momentum 0.9 with standard SGD. We whiten input examples by subtracting the (per-channel) mean and dividing by standard deviation and subsequently apply random flips and cropping as data augmentation. During training, we use 7 iterations PGD, with learning rate 0.007, signed gradient and $\epsilon = 8/255$ for L_∞ adversarial examples. PGD-7 is also used for early stopping (every 5th epoch) on the last 500 test examples. We do *not* use early stopping by default. For evaluation on the first 1000 test examples, we run PGD with 20 iterations, 10 random restarts to estimate RLoss and AutoAttack [CH20c] to estimate RErr (c.f. Section 5.2.1). Note that AutoAttack does *not* maximize cross-entropy loss as it stops when adversarial examples are

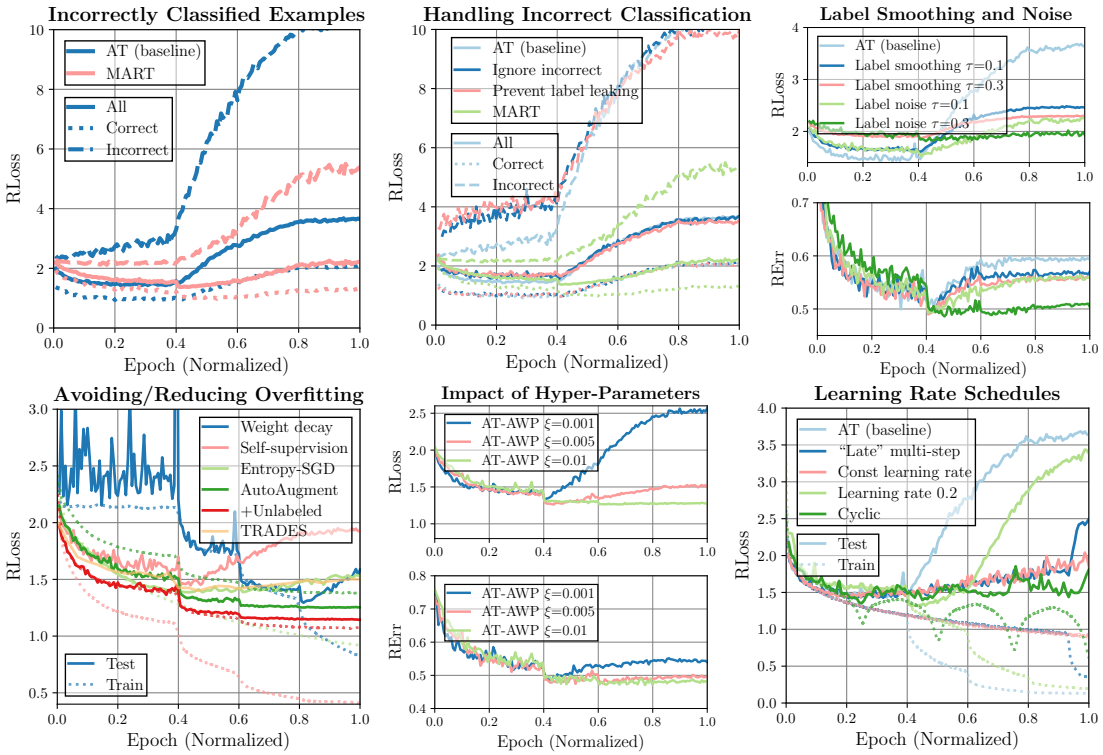


Figure 5.5: **Understanding Robust Overfitting:** Training curves over epochs, see Section 5.2.4 for discussion. **First row, first column:** RLoss for correct/incorrect test examples, for AT and MART. **First row, second column:** We consider ignoring incorrectly classified training examples in the RLoss computation during training (dark blue) and preventing label leaking by computing adversarial examples against the *predicted* labels (rose). However, these "tricks" do not reduce robust overfitting. In contrast, MART [WZY+20] (green) dampens overfitting through an additional robust KL-loss weighted by confidence. **First row, third column:** Both label smoothing and label noise reduce robust overfitting w.r.t. RLoss. However, this does not translate to a similar reduction of RErr. **Second row:** RLoss for various AT variants and different learning rate schedules. While AT-AWP can avoid robust overfitting altogether, others methods like weight decay merely reduce its impact. Both depend on hyperparameters.

found. Thus, it is not suitable to estimate RLoss. We do *not* use momentum [DLP+18] or backtracking [SHS20] for PGD.

For *average-case flatness of RLoss*, we take the average of 10 random weight perturbations with ξ=0.5. For *worst-case flatness*, we maximize RLoss jointly over adversarial examples and adversarial weights with ξ=0.00075, taking the worst of 10 restarts. We use a learning rate 0.001, after normalizing the update Δ_v in Equation (5.6) per-layer as in Equation (5.2). In both cases, 20-step PGD as described above is used to compute adversarial examples (choosing the worst-case one per test example) and we use a batch size of 128 on the first 1000 test examples.

Methods: Besides our AT baseline, we consider AT-AWP [WZX+16b], TRADES [ZYJ+19], MART [WZY+20], AT with self-supervision [HMKS19] or additional unlabeled examples [CRS+19, AUH+19], weight averaging [IPG+18] and AT with "early-stopped" PGD [ZXH+20]. We investigate different hyperparameters and "tricks" recently studied in [GQU+20, PYD+21]: learning rate schedules, batch size, weight decay, label smoothing [SVI+16] as well as SiLU, Mish, and GeLU [HG16, EUD18, Mis20] activation functions. Furthermore, we consider Entropy-SGD [CCS+17], label noise, weight clipping [SCHS21a] and AutoAugment [CZM+19].

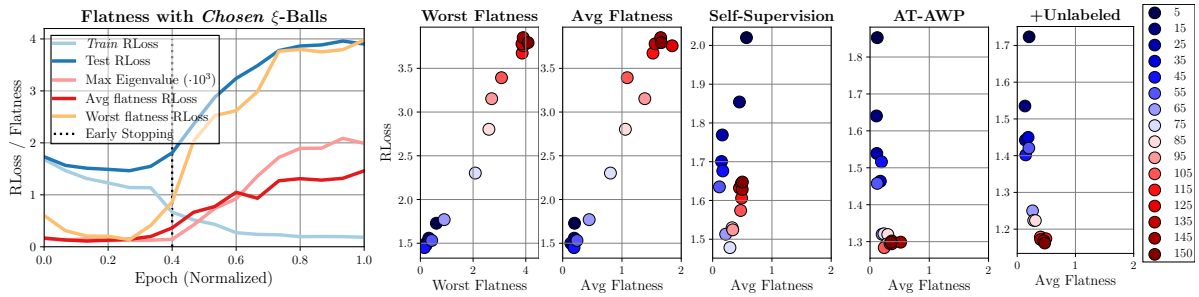


Figure 5.6: **Flatness Throughout Training. Left:** We plot train and test RLoss, maximum Hessian eigenvalue $\lambda_{\max}$, average-/worst-case flatness of RLoss. Flatness and $\lambda_{\max}$ increase quickly during robust overfitting. **Right:** Test RLoss (y-axis) plotted against flatness in RLoss (x-axis) during training (early epochs in dark blue, late epochs in dark red), showing a clear correlation, for both average- and worst-case flatness. AT with self-supervision reduces the impact of robust overfitting (RLoss increases less) and simultaneously favors flatter minima. This behavior is pronounced for AT-AWP and AT with additional unlabeled examples.

We emphasize that weight averaging, Entropy-SGD and weight clipping are known to improve flatness of the (clean) loss. *If not stated otherwise, these methods are applied on top or as replacement of our AT baseline.* We report results using the best hyperparameters per method. Finally, we also use pre-trained models from RobustBench [CAS+20a], which were obtained using early stopping. Appendix B.4 discusses each of the evaluated methods individually in more detail.

5.3.1 Understanding Robust Overfitting

In contrast to related work [RWK20], we take a closer look at RLoss during robust overfitting because RErr is "blind" to many improvements in RLoss, especially on incorrectly classified examples. Figure 5.5 shows training curves for various methods, i.e., RLoss/RErr over (normalized) epochs. For example, explicitly handling incorrectly classified examples during training, using MART, helps but does not *prevent* overfitting: RLoss for MART reduces compared to AT (top row, middle). Unfortunately, this improvement does *not* translate to significantly better RErr. This discrepancy between RLoss and RErr can be reproduced for other methods, as well: label smoothing and label noise enforce, in expectation, the same target distribution. Thus, both reduce RLoss during overfitting (top row, right, rose and dark green). Label smoothing, however, does not improve RErr as significantly as label noise, i.e., does not *prevent* misclassification. This illustrates an important aspect: against adversarial examples, "merely" improving RLoss

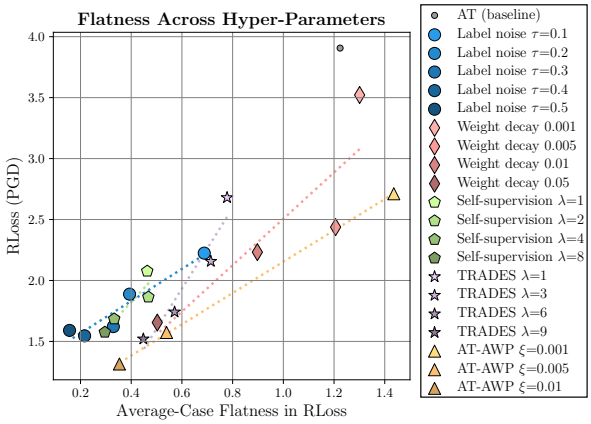


Figure 5.7: **Flatness Across Hyperparameters:** RLoss (y-axis) vs. average-case flatness (x-axis) for selected methods and hyperparameters. For example, we consider different strengths of weight decay (rose) or sizes ξ of adversarial weight perturbations for AT-AWP (orange). For clarity, we plot (dotted) lines representing the trend per method. Clearly, improved adversarial robustness, i.e., low RLoss, is related to improved flatness.

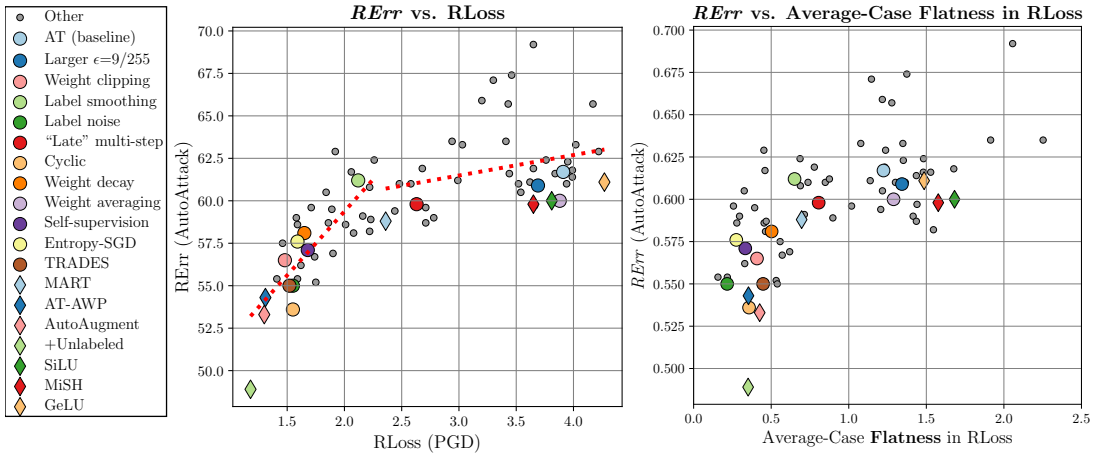


Figure 5.8: **Relationship between RLoss and RErr: Left:** RErr plotted against RLoss, showing that improved RLoss does not directly translate to reduced RErr for large RLoss. In these cases, reducing RLoss mainly means reducing the confidence of adversarial examples, which is necessary to improve adversarial robustness. **Right:** For average-case flatness in RLoss, subject to the non-trivial interplay between RErr and RLoss (c.f. left), we can also see a relationship between flatness and RErr.

does not translate to improved RErr if RLoss is high to begin with, i.e., "above" $-\log(1/K)\approx 2.3$ for $K=10$ classes. However, this is usually the case during robust overfitting. RErr, on the other hand, does not take into account the confidence of wrong predictions, i.e., it is "blind" for these improvements in RLoss. Label noise, in contrast, also improves RErr, which might be due to the additional randomness.

Similar to established methods, many "simple" regularization schemes prove surprisingly effective in tackling robust overfitting. For example, strong weight decay delays robust overfitting and AutoAugment prevents overfitting entirely, c.f. Figure 5.5 (bottom row). This indicates that popular AT variants, e.g., TRADES, AT with self-supervision or unlabeled examples, improve adversarial robustness by avoiding robust overfitting through regularization. This is achieved by preventing convergence on training examples (dotted). In regularization, however, hyperparameters play a key role: even AT-AWP does not prevent robust overfitting if regularization is "too weak" (blue). This is particularly prominent in terms of RLoss. Finally, learning rate schedules play an important role in how and *when* robust overfitting occurs. However, as in [RWK20], all schedules are subject to robust overfitting.

5.3.2 Robust Generalization and Flatness

As robust overfitting is primarily avoided through strong regularization, we hypothesize that this is because strong regularization finds flatter minima in the RLoss landscape. These flat minima help to improve robust generalization.

Flatness in RLoss "Explains" Overfitting: Using our average- and worst-case flatness measures in RLoss, we find that flatness reduces significantly during robust overfitting. Namely, flatness "explains" the increased RLoss caused by overfitting very well. Figure 5.6 (left) plots RLoss, alongside average- and worst-case flatness and the maximum Hessian eigenvalue throughout training of our AT baseline. Clearly, flatness increases alongside (test) RLoss as soon as robust overfitting occurs. Note that the best epoch is 60, meaning 0.4 (black dotted). For further illustration, Figure 5.6 (middle) explicitly plots RLoss (y-axis) against flatness in RLoss

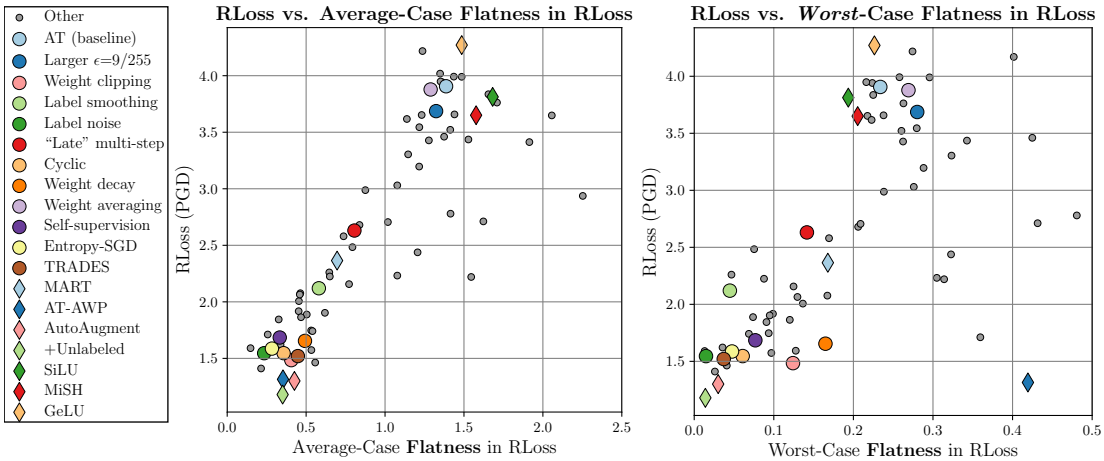


Figure 5.9: **Robustness in RLoss and Flatness: Left:** RLoss (y-axis) plotted against our *average-case* flatness in RLoss. We highlight selected models, as in Table 5.2 and reveal a striking correlation between adversarial robustness and flatness. Popular AT variants improving robustness (e.g., TRADES, MART, etc.) also correspond to flatter minima. Vice versa, methods improving flatness (e.g., Entropy-SGD, weight decay, etc.) improve robustness obtained through AT. Subject to the non-trivial interplay between RErr and RLoss (c.f. left), this relationship is also visible using RErr to quantify robustness. **Right:** RLoss (y-axis) plotted against *worst-case* flatness (x-axis) shows a less clear relationship. Still, improved flatness remains a necessity for better robust generalization.

(x-axis) across epochs (dark blue to dark red): RLoss and flatness clearly worsen "alongside" each other during overfitting, for both average- and worst-case flatness. Methods such as AT with self-supervision, AT-AWP or AT with unlabeled examples avoid both robust overfitting *and* sharp minima (right). This relationship generalizes to different hyperparameter choices of these methods: Figure 5.7 plots RLoss (y-axis) vs. average-case flatness (x-axis) across different hyperparameters. Again, e.g., for TRADES or AT-AWP, hyperparameters with lower RLoss also correspond to flatter minima. In fact, Figure 5.7 indicates that the connection between robustness and flatness also generalizes *across* different methods (and individual models).

Improved Robustness Through Flatness: Indeed, across all trained models, we found a strong correlation between robust generalization and flatness, using RLoss as measure for robust generalization. We mainly consider RLoss to assess robust generalization as improvements in RLoss above $\sim$2.3 have, on average, only small impact on RErr. Pushing RLoss below 2.3, in contrast, directly translates to better RErr. This is illustrated in Figure 5.8 (left) which plots RErr vs. RLoss for all evaluated models. To avoid this "kink" in the dotted red lines around RLoss$\approx$2.3, Figure 5.9 (left) plots *RLoss* (y-axis) against *average-case* flatness in RLoss (x-axis), highlighting selected models. This reveals a *clear correlation between robustness and flatness*: More robust methods, e.g., AT with unlabeled examples or AT-AWP, correspond to flatter minima. Methods improving flatness, e.g., Entropy-SGD, weight decay or weight clipping, improve adversarial robustness. This also translates to RErr subject to the described bend at RLoss$\approx$2.3, c.f. Figure 5.8 (right). While many robust methods still obtain better flatness, activation functions such as SiLU, MiSH or GeLU also seem to improve flatness, without clear advantage in terms of robustness. Overall, with Pearson/Spearman correlation coefficients of 0.85/0.87 (p-values $<10^{-21}$), we revealed a strong relationship between robustness and flatness. We emphasize that, with a Pearson correlation between RLoss and average-case flatness in *clean* Loss of only 0.27, it is essential to measure flatness in RLoss.

Model	Robustness ↓		Flatness ↓		Early Stop.
(sorted asc. by test RErr) (split at 70%/30% percentiles)	RErr (test)	RErr (train)	Avg (RLoss)	Worst (RLoss)	RErr ↓ (early stop)
+Unlabeled	48.9	43.2 (-5.7)	0.32	1.20	48.9 (-0.0)
Cyclic	53.6	35.4 (-18.2)	0.35	1.50	53.6 (-0.0)
AutoAugment	54.0	47.9 (-6.1)	0.49	0.69	53.5 (-0.5)
AT-AWP	54.3	43.1 (-11.2)	0.35	2.68	53.6 (-0.7)
Label noise	56.2	30.0 (-26.2)	0.33	0.93	55.5 (-0.7)
Weight clipping	56.5	39.0 (-17.5)	0.41	4.57	56.5 (-0.0)
TRADES	56.7	15.8 (-40.9)	0.57	2.25	53.4 (-3.3)
Self-supervision	57.1	45.0 (-12.1)	0.33	2.63	56.8 (-0.3)
Weight decay	58.1	32.8 (-25.3)	0.50	3.93	54.8 (-3.3)
Entropy-SGD	58.6	46.1 (-12.5)	0.28	1.80	56.9 (-1.7)
MiSH	59.8	5.3 (-54.5)	1.56	3.54	53.7 (-6.1)
"Late" multistep	59.8	18.4 (-41.4)	0.80	2.96	57.8 (-2.0)
SiLU	60.0	5.6 (-54.4)	1.71	4.20	53.7 (-6.3)
Weight averaging	60.0	10.0 (-50.0)	1.28	5.98	53.0 (-7.0)
Larger ϵ=9/255	60.9	11.1 (-49.8)	1.33	5.84	53.8 (-7.1)
MART	61.0	20.8 (-40.2)	0.73	3.17	54.7 (-6.3)
GeLU	61.1	3.2 (-57.9)	1.55	4.12	56.7 (-4.4)
Label smoothing	61.2	8.0 (-53.2)	0.65	2.72	54.0 (-7.2)
AT (baseline)	62.8	10.7 (-52.1)	1.21	6.48	54.6 (-8.2)
Robustness	**Averages (across models)**				
Good (RErr<57%≈30% percentile)	54.3	36.3 (-18.0)	0.40	2.00	53.6 (-0.7)
Average (57%≥RErr < 60%)	58.7	29.5 (-29.2)	0.69	2.9	56.0 (-2.7)
Poor (RErr≥60%≈70% percentile)	61.0	9.9 (-51.1)	1.21	4.67	54.4 (-6.6)

Table 5.2: **Robustness and Flatness, Quantitative Results:** Test and train RErr (first, second column, early stopping in fifth column) as well as average-/worst-case flatness in RLoss (third, fourth column) for selected methods, c.f. Figure 5.9. We split methods into good, average, and poor robustness using the 30% and 70% percentiles. Most methods improve adversarial robustness alongside both average- and worst-case flatness.

Figure 5.9 (right) shows that this relationship is less clear when considering *worst-case* flatness in RLoss (Pearson coefficient 0.54). This is in contrast to [WXW20b] suggesting that worst-case flatness, in particular, is important to improve robustness of AT. However, worst-case flatness is more sensitive to ξ and thus less comparable across methods. Note that worst-case robustness is still a good indicator for overfitting, c.f. Figure 5.6. All results are summarized in tabular form in Table 5.2: Grouping methods by good, average or poor robustness, we find that methods need at least "some" flatness, average- *or* worst-case, to be successful.

Decomposing Robust Generalization: So far, we used (absolute) RLoss on test examples as proxy of robust generalization. This is based on the assumption that deep models are generally able to obtain nearly zero *train* RLoss. However, this is not the case for many methods in Table 5.2 (second column). Thus, we also consider the robust generalization *gap* and the RLoss difference between last and best (early stopped) epoch. Specifically, Figure 5.10 (left) explicitly plots the RLoss generalization gap (test−train RLoss, y-axis) against average-case flatness in RLoss (x-axis). Robust methods generally reduce this gap by *both* reducing test RLoss *and* avoiding convergence in train RLoss. Furthermore, Figure 5.10 (right) considers the difference between last and best epoch, essentially quantifying the extent of robust overfitting. Again, methods with small difference, i.e., little robust overfitting, generally correspond to flatter minima. This is also confirmed in Figure 5.11 showing that early stopping essentially finds flatter minima along the training trajectory, thereby improving adversarial robustness. Finally,

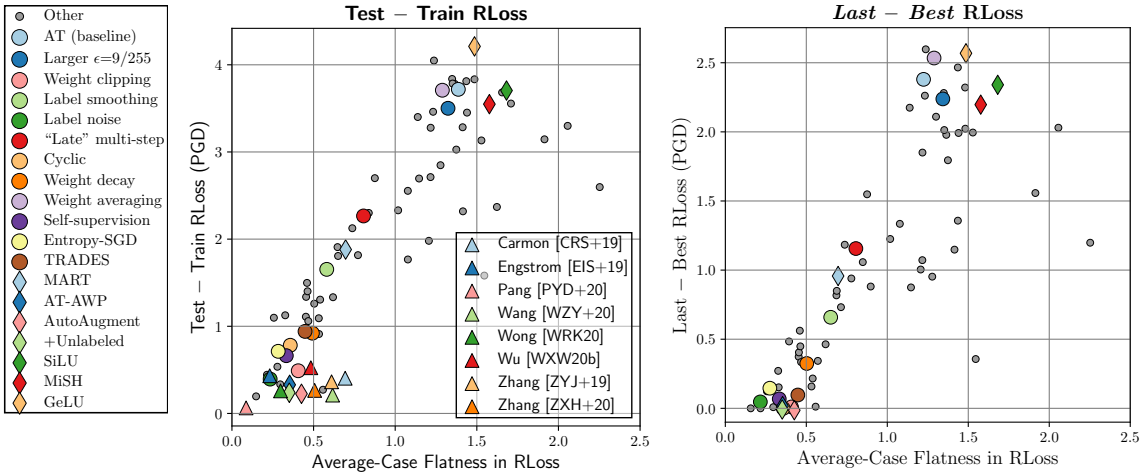


Figure 5.10: **Flatness and Robust Generalization.** Robust generalization (RLoss) decomposed into the test-train difference and the last-best (epoch) improvement (y-axis), both plotted against average-case flatness in RLoss (x-axis). In both cases, flatness seems to play an important role, i.e., flatness clearly reduces both the robust generalization gap *and* robust overfitting.

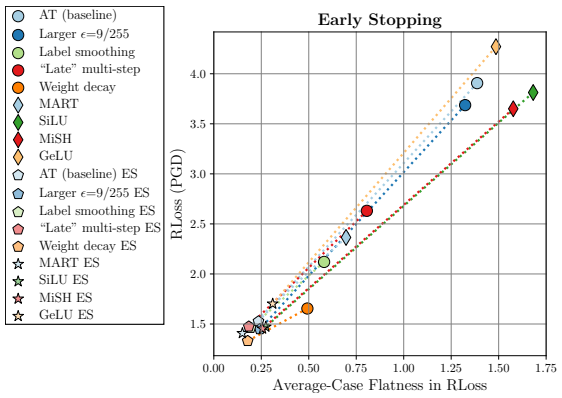


Figure 5.11: **Flatness and Early Stopping.** RLoss vs. average-case flatness in RLoss for selected models with and without early stopping ("ES"). Early stopping consistently leads to improved adversarial robustness *and* better flatness.

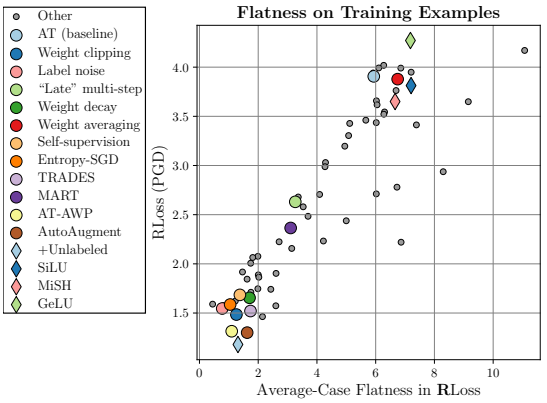


Figure 5.12: **Flatness on Training Examples.** RLoss plotted vs. average-case flatness measured *on training examples*. Even on training examples, flatness is a good indicator for robust generalization. This suggests that robust flatness evaluated during training could already be used for model selection.

Figure 5.12 highlights that our conclusions also generalize to computing robust flatness on training examples. Altogether, flatness improves robust generalization by reducing both the robust generalization gap *and* robust overfitting.

More Results: Figure 5.1 and 5.10 show that the pre-trained models from RobustBench [CAS+20a] confirm our observations so far. While detailed analysis is not possible as only early stopped models are provided, they are consistently more robust *and* correspond to flatter minima compared to our models. This is despite using different architectures (commonly Wide ResNets [ZK16]).

5.4 Conclusion

In this paper, we studied the relationship between adversarial robustness, specifically considering robust overfitting [RWK20], and flatness of the robust loss (RLoss) landscape w.r.t. perturbations in the weight space. We introduced both average- and worst-case measures for flatness in RLoss that are scale-invariant and allow comparison across models. Considering adversarial training (AT) and several popular variants, including TRADES [ZYJ+19], AT-AWP [WXW20b] or AT with additional unlabeled examples [CRS+19], we show a **clear relationship between adversarial robustness and flatness** in RLoss. More robust methods predominantly find flatter minima. Vice versa, approaches known to improve flatness, e.g., Entropy-SGD [CCS+17] or weight clipping [SCHS21a] can help AT become more robust, as well. Moreover, even simple regularization methods such as AutoAugment [CZM+19], weight decay or label noise, are effective in increasing robustness by improving flatness. These observations also generalize to pre-trained models from RobustBench [CAS+20a].

III

Improving Weight Robustness

While the previous part focused on understanding robustness against adversarial examples, i.e., perturbations in the *input* space, this part considers robustness in *weight* space instead. There are numerous applications where robustness against weight perturbations is relevant, including, e.g., network quantization or security against backdoor or trojan attacks. Moreover, robustness in weight space is closely related to the flatness measures from Chapter 5. In the following, however, we particularly focus on the importance of weight robustness for deploying deep neural networks on deep learning *accelerators*, specialized chips for inference.

In particular, Chapter 6 considers the robustness of models against random or adversarial bit errors in their quantized weights. As we will show, robustness against random bit errors is highly relevant in the context of reducing energy consumption of accelerators. This is because robustness allows enduring bit errors induced through lower operating voltage, directly improving energy efficiency without reducing performance significantly. Additionally, defending adversarial bit errors also improves security against malicious hardware- of software-based attacks on the accelerator's memory.

6 Random and Adversarial Bit Error Robustness: Energy-Efficient and Secure Neural Network Accelerators

Contents

This chapter addresses robustness of deep neural networks against perturbations in their weights, specifically considering bit errors in quantized weights. In the context of specialized hardware for inference, so-called *accelerators*, we show that bit error robustness contributes to more energy-efficient and secure deployment.

These deep neural network accelerators received considerable attention in recent years due to the potential to save energy compared to mainstream hardware. Low-voltage operation of accelerators allows further reducing energy consumption significantly, however, causes bit-level failures in the memory storing the quantized weights. Furthermore, accelerators have been shown to be vulnerable to adversarial attacks on voltage controllers or individual bits. We show that a combination of **robust fixed-point quantization**, **weight clipping**, as well as **random bit error training (RandBET)** or **adversarial bit error training (AdvBET)** improves robustness against random or adversarial bit errors in quantized DNN weights significantly. This leads not only to high energy savings from low-voltage operation *as well as* low-precision quantization, but also improves security of accelerators. Our approach generalizes across operating voltages and accelerators, as demonstrated on bit errors from profiled SRAM arrays, and achieves robustness against both targeted and untargeted bit-level attacks. Without losing more than 0.8%/2% in test accuracy, we can reduce energy consumption on CIFAR10 by 20%/30% for 8/4-bit quantization using RandBET. Allowing up to 320 adversarial bit errors, AdvBET reduces test error from above 90% (chance level) to 26.22% on CIFAR10.

This chapter is based on [SCHS21a] and its extension [SCHS21b], which is currently

under review. As first author, David Stutz ran all experiments and was the main writer of both papers. Proposition 1 in Section 6.4.2 and its proof were contributed by Matthias Hein and the profiled bit errors of Figure 6.3 and Table 6.6 were provided by Nandhini Chandramoorthy. This work was conducted in collaboration with IBM Research and presented at the Workshop on the Future of Computing Architectures (FOCA) 2020 organized by IBM Research in 2020, the Workshop on Robust Artificial Intelligence (RobustAI) organized by the Lorentz Center, at the Workshop on Adversarial Learning Methods for Machine Learning and Data Mining (AdvML) organized together with KDD 2021, as part of the Qualcomm Innovation Fellowship Europe 2019, as well as the Workshop on Adversarial Machine Learning in Real-World Computer Vision Systems and Online Challenges (AML-CV) at CVPR 2021 where it was recognized as distinguished paper. Finally, an invited talk was given at TU Dortmund.

The **code** for this chapter can be found on GitHub[1].

6.1 Introduction

Energy-efficiency is an important goal to lower carbon-dioxide emissions of deep neural network driven applications and is a critical prerequisite to enable applications in edge computing. Deep neural network accelerators, i.e., specialized hardware for inference, are used to reduce and limit energy consumption alongside cost and space compared to mainstream hardware, e.g., GPUs. These accelerators, e.g., [CDS+14, DFC+15, CES16, RWA+16, SPS+18, CSC+19, nvd], generally feature large on-chip SRAM used as scratchpads, e.g., to store weights and intermediate computations. The overall dynamic energy of the accelerator SRAM is proportional to the number of SRAM accesses (e.g., to read weights) times the energy of a single access. Thus, data access/movement constitutes a dominant component of accelerator's energy consumption [SCYE17]. While both optimized data flow, e.g., re-using weights, and low-precision quantization [LTA16] reduce the number of memory accesses, recent accelerators [RWA+16, KHM+18a, CSC+19] further try to lower the energy per SRAM access. This is achieved by lowering the memory supply voltage (dynamic power per access varies quadratically with voltage). However, aggressive SRAM supply voltage scaling causes bit-level failures on account of process variation [GCP+09, GKKR17] with direct impact on the stored model weights. The rate p of these errors increases exponentially with lowered voltage, causing devastating drops in accuracy. Thus, such accelerators are also vulnerable to maliciously reducing voltage [TSS17] or adversarially inducing individual bit errors [KDK+14, MOG+20]. In the following, we aim to enable very low-voltage operation of accelerators by developing deep neural networks robust to *random bit errors* in their (quantized) weights. This also improves security against manipulation of voltage settings [TSS17]. Furthermore, we address robustness against a limited number of *adversarial bit errors*, similar to [RHF19a, RHL+20, HRL+20]. In general, robustness to bit errors is a desirable goal to maintain safe operation and should become a standard performance metric in low-power and quantized neural network design.

Figure 6.1 shows the average bit error rates of SRAM arrays as supply voltage is scaled below $V_{\min}$, i.e., the measured lowest voltage at which there are no bit errors. Voltage (x-axis) and energy (red, right y-axis) are normalized w.r.t. $V_{\min}$ and the energy per access at $V_{\min}$, respectively. Deep neural networks robust to a bit error rate (blue, left y-axis) of, e.g., $p = 1\%$ allow reducing SRAM energy by roughly 30%. To improve robustness to the induced *random bit errors*, we first consider the impact of fixed-point quantization on robustness. While prior work [SSH15, MAA+16, MDI19] studies robustness *to* quantization, the impact of random

[1] https://github.com/davidstutz/mlsys2021-bit-error-robustness

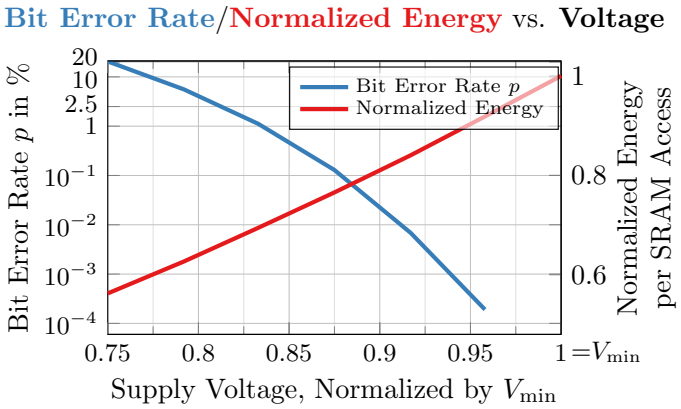


Figure 6.1: **Energy and Low-Voltage Operation:** Average bit error rate p (blue, left y-axis) from 32 14nm SRAM arrays of size 512×64 from [CSC+19] fabricated using 14nm FinFET technology and energy (red, right y-axis) vs. voltage (x-axis). Voltage is normalized by $V_{\min}$, the minimal measured voltage for error-free operation, and the energy per SRAM access at $V_{\min}$. Bit error rate at a given supply voltage is measured as the count of read or write bit cell failures averaged over the total number of bit cells in the SRAM. A bit cell failure refers to reading 1 on writing 0 or reading 0 on writing 1, c.f. [GKKR17]. Energy per write and read access of a 4KB (512 × 64 bit) SRAM is obtained from Cadence Spectre simulations (constant clock frequency at all supply voltages). SRAM accesses have significant impact on the accelerator's energy [CES16]. Reducing voltage leads to exponentially increasing bit error rates.

bit errors *in* quantized weights has not been considered so far. We find that the choice of quantization scheme has tremendous impact on robustness, even though accuracy is not affected. In particular, we identify a particularly **robust quantization scheme**, RQuant in Figure 6.2 (red). Additionally, independent of the quantization scheme, we propose aggressive **weight clipping** during training. This acts as an explicit regularizer leading to spread out weight distributions, improving robustness significantly, Clipping in Figure 6.2 (blue). This is in contrast to, e.g., [ZST+18, SSH15] ignoring weight outliers to reduce quantization range, with sole focus on improving accuracy.

Conventional error mitigation strategies or circuit techniques are not applicable to mitigate larger rates of bit errors or incur a significant energy/space overhead. For example, common error correcting codes (ECCs such as SECDED), cannot correct *multiple* bit errors per word (containing multiple weights). However, for $p = 1\%$, the probability of two or more random bit errors in a 64-bit word is 13.5%. Furthermore, an adversary may intentionally target multiple bits per word. Considering low-voltage induced random bit errors, error detection via redundancy [RWA+16] or supply voltage boosting [CSC+19] allow error-free operation at the cost of additional energy or space. Therefore, [KHM+18a] and [KOY+19] propose a co-design approach of training deep neural networks on *profiled* bit errors (i.e., post-silicon characterization) from SRAM or DRAM, respectively. These approaches work as long as the spatial bit error patterns can be assumed fixed for a *fixed* accelerator *and* voltage. However, the random nature of variation-induced bit errors requires profiling to be carried out for each voltage, memory array and individual chip in order to obtain the corresponding bit error patterns. This makes training on profiled bit error patterns an expensive process. We demonstrate that the obtained models do *not* generalize across voltages or to unseen bit error patterns, e.g., from other memory arrays, and propose **random bit error training (RandBET)**, in combination with weight clipping and robust quantization, to obtain robustness against completely random bit error patterns, see Figure 6.2 (violet). Thereby, it generalizes across

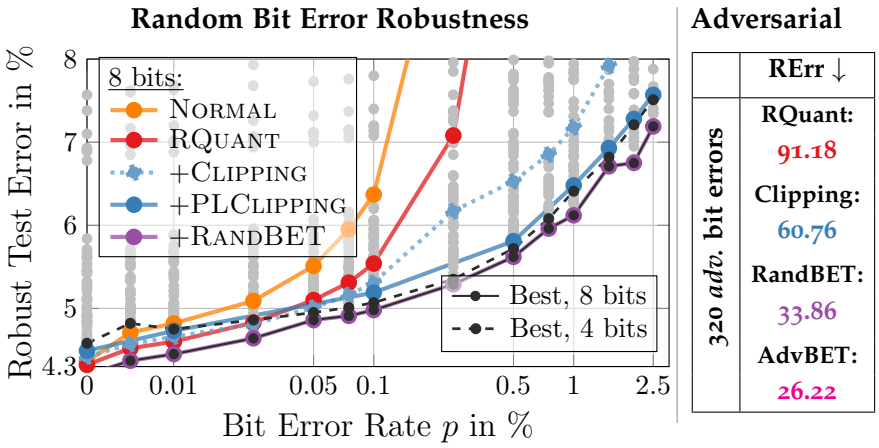


Figure 6.2: **Robustness to Random and Adversarial Bit Errors: Left:** Robust test error RErr after injecting *random* bit errors (lower is better ↓, y-axis) plotted against bit error rate p (x-axis). For 8 bit, robust quantization (RQUANT, red), additionally weight clipping (CLIPPING, dotted blue) or *per-layer* weight clipping (PLCLIPPING, solid blue) and finally adding random bit error training (RANDBET, violet) improve robustness significantly. Robustness to higher bit error rates allows more energy efficient operation, c.f. Figure 6.1. The Pareto optimal frontier is shown for 8 bit (black solid) and 4 bit (dashed) quantization. **Right:** RErr against up to 320 *adversarial* bit errors, showing that CLIPPING combined with RANDBET or ADVBET also allow secure operation.

chips *and* voltages, without any profiling, hardware-specific data mapping or other circuit-level mitigation strategies. Finally, we also consider bit errors in activations and inputs, as both are temporally stored on the chip's memory and thus subject to bit errors.

Besides low-voltage induced random bit errors, [KDK+14, MOG+20] demonstrate the possibility of *adversarially* flipping specific bits. The bit flip attack (BFA) of [RHF19a], an untargeted search-based attack on (quantized) weights, demonstrates that such attacks can easily degrade model accuracy with few bit flips. [HRL+20] proposes a binarization strategy to "defend" against BFA. However, the approach was shown to be ineffective shortly after considering a targeted version of BFA [RHL+20], leaving the problem unaddressed. We propose a novel attack based on projected gradient descent, inspired by recent work on adversarial examples [MMS+18]. We demonstrate that our attack is both more effective and more efficient. Moreover, in contrast to BFA, our adversarial bit attack enables **adversarial bit error training** (**AdvBET**). As shown in Figure 6.2 (right), ADVBET (magenta) improves robustness against adversarial bit errors considerably, outperforming CLIPPING (blue) and RANDBET (violet) which, surprisingly, provide very strong baselines. As a result, we are able to obtain robustness to both random and adversarial bit errors for energy-efficient *and* secure accelerators.

Contributions: We combine our **robust fixed-point quantization** with **weight clipping** and **random bit error training (RandBET)** or **adversarial bit error training (AdvBET)** in order to obtain high robustness against low-voltage induced, *random bit errors* or maliciously crafted, *adversarial bit errors*. We consider fixed-point quantization schemes in terms of robustness *and* accuracy, instead of *solely* focusing on accuracy as related work. Furthermore, we show that aggressive (per-layer) weight clipping, as regularization during training, is an effective strategy to improve robustness through redundancy. In contrast to [KHM+18a, KOY+19], the robustness obtained through RANDBET generalizes across chips *and* voltages, as evaluated on profiled SRAM bit error patterns from [CSC+19]. Moreover, we also consider bit errors in activations and inputs. In contrast to [RHF19a, RHL+20], our (untargeted or targeted)

adversarial bit error attack is based on gradient descent, improving effectiveness and efficiency, and our AdvBET improves robustness against targeted and untargeted attacks, outperforming the recently broken binarization approach of [HRL+20]. Finally, we discuss the involved trade-offs regarding robustness (against random or adversarial bit errors). Figure 6.2 (left) highlights key results for RandBET on CIFAR10: with 8/4 bit quantization and an increase in test error of less than 0.8%/2%, roughly 20%/30% energy savings are possible – on top of energy savings from using low-precision quantization. Similarly, AdvBET, c.f. Figure 6.2 (right), obtains 26.22% (robust) test error against up to 320 adversarial bit errors in the weights.

6.2 Bit Errors in Quantized Weights

In the following, we introduce the bit error models considered in this paper: random bit errors (Section 6.2.1), induced through low-voltage operation of accelerator memory, and adversarial bit errors (Section 6.2.2), maliciously crafted and injected by an adversary to degrade accuracy.

Notation: Let $f(x; w)$ be a deep neural network taking an example $x \in [0,1]^D$, e.g., an image, and weights $w \in \mathbb{R}^W$ as input. The model is trained by minimizing the cross-entropy loss $\mathcal{L}$ on a training set $\{(x_n, y_n)\}_{n=1}^N$ consisting of examples x_n and corresponding labels $y_n \in \{1, \dots, K\}$, K denoting the number of classes. We assume a weight $w_i \in [q_{\min}, q_{\max}]$, i.e., within the *quantization range*, to be quantized using a function Q. As we will detail in Section 6.3.1, Q maps floating-point values to m-bit (signed or unsigned) integers. With $v_i = Q(w_i)$, we denote the integer corresponding to the quantized value of w_i, i.e., v_i is the bit representation of w_i after quantization represented as integer. Finally, $d_H(v, v')$ denotes the bit-level Hamming distance between the integers v and v'.

6.2.1 Low-Voltage Induced Random Bit Errors

We assume the quantized weights to be stored on multiple memory banks, e.g., SRAM in the case of on-chip scratchpads or DRAM for off-chip memory. As shown in [GKKR17, KHM+18a, CSC+19], the probability of memory bit cell failures increases exponentially as operating voltage is scaled below $V_{\min}$, i.e., the minimal voltage required for reliable operation, see Figure 6.1. This is done intentionally to reduce energy consumption [KHM+18a, CSC+19, KOY+19] or adversarially by an attacker [TSS17]. Process variation during fabrication causes a variation in the vulnerability of individual bit cells. As shown in Figure 6.3 (left), for a specific memory array, bit cell failures are typically approximately random and independent of each other [GKKR17] even so chips showing patterns with stronger dependencies are possible, c.f. Figure 6.3 (right). Nevertheless, there is generally an "inherited" distribution of bit cell failures across voltages: as described in [GKB+19], if a bit error occurred at a given voltage, it is likely to occur at lower voltages, as made explicit in Figure 6.3. However, across different SRAM arrays in a chip or different chips, the patterns or spatial distributions of bit errors is usually different and can be assumed random [CSC+19]. Throughout the paper, we use the following bit error model:

Random Bit Error Model: *The probability of a bit error is p (in %) for all weight values and bits. For a fixed memory array, bit errors are persistent across supply voltages, i.e., bit errors at probability $p' \leq p$ also occur at probability p. A bit error flips the currently stored bit. Random bit error injection is denoted $BErr_p$.*

This error model realistically captures the nature of low-voltage induced bit errors, from both SRAM and DRAM as confirmed in [KHM+18a, CSC+19, KOY+19]. In fact, our model can be seen as a very conservative model. This is because low-voltage induced bit errors are

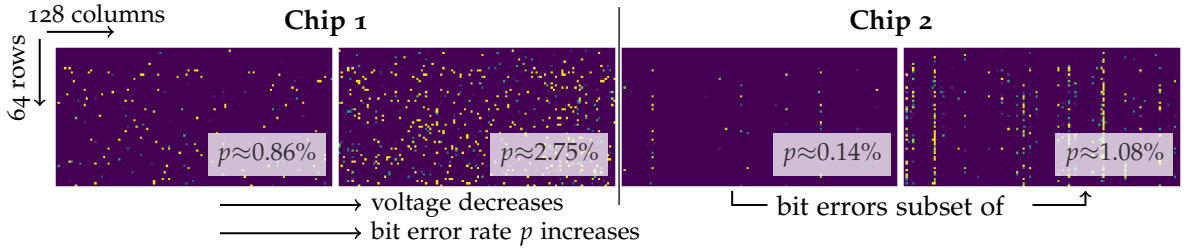


Figure 6.3: **Exemplary SRAM Bit Error Patterns:** Measured bit errors from two chips with on-chip SRAM (left and right), showing bit flip probability for a segment of size 64×128 bits: yellow indicates a bit flip probability of one, violet indicates zero probability. We show measurements corresponding to two supply voltages. With lower voltage, bit error rate increases. Also, the bit errors for higher voltage (= lower bit error rate) are a subset of those for lower voltage (= higher rate), c.f. Section 6.2. Our error model randomly distributes bit errors across space. However, as example, we also show SRAM chip 2 which has a different spatial distribution with bit errors distributed along columns. We aim to obtain robustness across different memory arrays, voltages *and* allowing arbitrary weight to memory mappings.

usually modeled as a two-step process: variations in the fabrication process cause bit cells to be more or less vulnerable. Such vulnerable bit cells are (spatially) distributed approximately uniformly (probability p_{flt}), as described above [GKKR17]. Then, faulty bit cells may cause bit errors when reading. However, these errors are usually transient, e.g., each read has a specific probability p_{err} of causing a bit error. In contrast, we assume the latter *not* to be transient errors, i.e., $p_{\text{err}} = 100\%$. Thereby we follow measurements in [KHM+18a, GKB+19] indicating that p_{err} is often close to 100%. This means that we assume faulty bit cells to *always* provoke bit errors. Note that our bit error model, with $p = p_{\text{flt}} \cdot p_{\text{err}} = p_{\text{flt}}$ (because of p_{err}=100%) remains valid even if relaxing this assumption.

Moreover, our approach in Section 6.3 is model-agnostic: the error model can be refined if extensive memory characterization results are available for individual chips. For example, faulty bit cells with 1-to-0 or 0-to-1 flips might not be equally likely. Similarly, as in [KOY+19], bit errors might be biased towards alignment along rows or columns of the memory array. The latter case is illustrated in Figure 6.3 (right). However, estimating these specifics requires testing infrastructure and detailed characterization of individual chips. More importantly, it introduces the risk of overfitting to few specific memories/chips. Furthermore, we demonstrate that the robustness obtained using our uniform error model generalizes to bit error distributions with strong spatial biases as in Figure 6.3 (right).

We assume the quantized weights to be mapped linearly to the memory. This is the most direct approach and, in contrast to [KOY+19], does not require knowledge of the exact spatial distribution of bit errors. This also means that we do not map particularly vulnerable weights to more reliable memory cells, and therefore no changes to the hardware or the application are required. Thus, in practice, for W weights and m bits per weight value, we sample uniformly $u \sim U(0,1)^{W \times m}$. Then, the j-th bit in the quantized weight $v_i = Q(w_i)$ is flipped iff $u_{ij} \leq p$. Our model assumes that the flipped bits at lower probability $p' \leq p$ are a subset of the flipped bits at probability p and that bit flips to 1 and 0 are equally likely. The noise pattern of random bit errors is illustrated in Figure 6.4: for example, a bit flip in the most-significant bit (MSB) of the signed integer v_i results in a change of half of the quantized range (also c.f. Section 6.3.1).

In the case of on-chip SRAM, inputs and activations will also be subject to low-voltage induced bit errors. This is because the SRAM memory banks are used as scratchpads to temporally store intermediate computations such as inputs and activations. As described in

detail in Section 6.4.7, inputs are subject to random bit errors *once* before being fed into the deep neural network. Activations, i.e., the result of intermediate layers, are subject to random bit errors multiple times throughout a forward pass. This is modeled by (independently) injecting random bit errors in the activations after each "block" consisting of convolution, normalization and ReLU layers. This assumes that activations are temporally stored on the SRAM scratchpads after each such block. In practice, the data flow of a accelerator is manually tailored to the neural network architecture *and* chip design, which is also why energy estimation for accelerators is very difficult [YCES17, WES19]. Furthermore, normalization schemes (group [WH18] or batch normalization [IS15b]) and ReLU activations can be "folded into" the preceding convolutional layer [JKC+18, LWL+19]. Thus, considering the activations to go through temporal storage on the SRAM after each block is a realistic approximation of the actual data flow.

6.2.2 Adversarial Bit Errors

Following recent attacks on memory [KDK+14, BHJ+18, RHF19a, MOG+20] and complementing our work on random bit errors, we also consider adversarial bit errors. We constrain the number of induced bit errors by ϵ, similar to the L_p-constrained adversarial inputs [SZS+14]. Furthermore, we consider only one bit flip per weight value to simplify the projection onto the discrete constraint set. Then, given knowledge of memory layout and addressing schemes, an adversary can use, e.g., RowHammer [KDK+14], in order to flip as many of the adversarially selected bits as possible. Note that, in practice, not all of these bits will be vulnerable to an end-to-end RowHammer attack on memory, which we do not focus on. However, from a robustness viewpoint, it makes sense to consider a slightly stronger threat model than actually realistic. Overall, our white-box threat model is defined as follows:

Adversarial Bit Error Model: *An adversary can flip up to ϵ bits, at most one bit per (quantized) weight, to reduce accuracy and has full access to the deep neural network, its weights and gradients.*

Note that we do *not* consider adversarial bit errors in inputs (i.e., adversarial examples) or activations. Following the projected gradient ascent approach of [MMS+18] and letting d_H be the (bit-level) Hamming distance, we intend to maximize cross-entropy loss $\mathcal{L}$ on a mini-batch $\{(x_b, y_b)\}_{b=1}^B$ of examples as *untargeted* attack:

$$\max_{\tilde{v}} \sum_{b=1}^{B} \mathcal{L}(f(x_b; Q^{-1}(\tilde{v})), y_b) \quad \text{s.t.} \quad d_H(\tilde{v}, v) \leq \epsilon, \quad d_H(\tilde{v}_i, v_i) \leq 1 \tag{6.1}$$

Note that y_b are the ground truth labels. We also consider a targeted version similar to [RHL+20], where we minimize the cross-entropy loss between predictions and an arbitrary but fixed target label: $\min_{\tilde{v}} \sum_{b=1}^{B} \mathcal{L}(f(x_b; Q^{-1}(\tilde{v})), y_t)$ where y_t is the same target label across all examples x_b. As made explicit in Equation (6.1), we work on bit-level, i.e., optimize over the two's complement signed integer representation $\tilde{v}_i \in \{-2^{m-1}-1, \dots, 2^{m-1}-1\}$ corresponding to the underlying bits of the perturbed weights $\tilde{w} = Q(\tilde{v})$. We will adversarially inject bit errors based on the gradient of Equation (6.1) and perform a projection onto the Hamming constraints $d_H(\tilde{v}, v) \leq \epsilon$ and $d_H(\tilde{v}_i, v_i) \leq 1$ with respect to the quantized, clean weights $v = Q(w)$. This means that we maximize Equation (6.1) through projected gradient ascent where the forward and backward pass are performed in floating point:

$$\tilde{w}^{(t+1)} = \tilde{w}^{(t)} + \gamma \Delta^{(t)} \quad \text{with } \Delta^{(t)} = \sum_{b=1}^{B} \nabla_w \mathcal{L}(f(x_b; \tilde{w}_q^{(t)}), y_b) \text{ and } \tilde{w}_q^{(t)} = Q^{-1}(Q(\tilde{w}^{(t)})) \tag{6.2}$$

followed by the projection of $\tilde{v}^{(t+1)} = Q(\tilde{w}^{(t+1)})$ onto the (bit-level) Hamming constraints of Equation (6.1). Here, γ is the step size. The updates are performed in floating point, while

Algorithm 1 Adversarial Bit Errors: We maximize cross-entropy loss using projected gradient ascent while ensuring that at most ϵ bits are flipped. Quantized weights in red; dequantized weights in blue; and floating-point operations in magenta. For coherence with Algorithm 2, AdvBitErrors takes the quantized weights $v = Q(w)$ as input.

AdvBitErrors(v, ϵ)
# perturb quantized weights by flipping at most ϵ bits:
initialize $\tilde{v}^{(0)}$ subject to $d_H(\tilde{v}^{(0)}, v) \leq \epsilon$, $d_H(\tilde{v}_i^{(0)}, v_i) \leq 1$
$\tilde{w}_q^{(0)} = Q^{-1}(\tilde{v}^{(0)})$ # dequantize perturbed weights
$\tilde{w}^{(0)} = \tilde{w}_q^{(0)}$ # floating-point weights to acc. updates
for $t = 0, \ldots, T-1$ **do**
  # fixed batch $\{(x_b, y_b)\}_{b=1}^B$
  # forward + backward pass w/ dequantized weights:
  $\Delta^{(t)} = \nabla_w \sum_{b=1}^B \mathcal{L}(f(x_b; \tilde{w}_q^{(t)}), y_b)$
  $\tilde{w}^{(t+1)} = \tilde{w}^{(t)} + \gamma \Delta^{(t)}$ # update w/o quantization:
  $\tilde{v}^{(t+1)} = Q(\tilde{w}^{(t+1)})$ # quantization for projection
  project onto $d_H(\tilde{v}^{(t+1)}, v) \leq \epsilon$, $d_H(\tilde{v}_i^{(t+1)}, v_i) \leq 1$
  $\tilde{w}_q^{(t+1)} = Q^{-1}(\tilde{v}^{(t+1)})$ # dequantization
end for
return $\tilde{w}_q^{(T)}$ # dequantized weights after projection

the forward pass is performed using the dequantized weights $\tilde{w}_q^{(t)}$. The perturbed weights $\tilde{w}^{(0)} = Q^{-1}(\tilde{v}^{(0)})$ are initialized by uniformly picking $k \in [0, \epsilon]$ bits to be flipped in $v = Q(w)$ in order to obtain $\tilde{v}^{(0)}$. Our adversarial bit attack is summarized in pseudocode in Algorithm 1.

The Hamming-projection is similar to the L_0 projection used for adversarial inputs, e.g., in [CH19]. Dropping the superscript t for brevity, in each iteration, we solve the following projection problem:

$$\min_{\tilde{v}'} \|Q^{-1}(\tilde{v}) - Q^{-1}(\tilde{v}')\|_2^2 \quad \text{s.t.} \quad d_H(v_i, \tilde{v}_i') \leq 1, \quad d_H(v, \tilde{v}') \leq \epsilon \tag{6.3}$$

where $\tilde{v} = Q(\tilde{w})$ are the quantized, perturbed weights after Equation (6.2) and $v = Q(w)$ are the quantized, clean weights. We optimize over $\tilde{v}'$ which will be the perturbed weights after the projection, i.e., as close as possible to $\tilde{v}$ while fulfilling the constraints above. This can be solved in two steps as the objective and the constraint set are separable: The first step involves keeping only the top-ϵ changed values, i.e., the top-ϵ weights with the largest difference

$$\left| Q^{-1}(Q(w_i)) - Q^{-1}(\tilde{v}_i) \right| = |w_{q,i} - \tilde{w}_{q,i}| \tag{6.4}$$

where w are the original, clean weights and w_q the corresponding dequantized weights. All other perturbed weights $\tilde{w}_{q,i}$ are reset to the original, clean weights $w_{q,i}$. For the selected weights, only the most significant changed bit is kept. In practice, considering $\tilde{v}_i$ and v_i from Equation (6.3) corresponding to one of the top-ϵ changes, if $d_H(\tilde{v}_i, v_i) > 1$, only the highest changed bit is kept. In practice, this can be implemented (and parallelized) easily on the m-bit integers $\tilde{v}_i$ and v_i while computing the (bit-level) Hamming distance d_H.

Implementation Details: The optimization problem in Equation (6.1) is challenging due to the non-convex constraint set that we project onto after each iteration. Therefore, we use several random restarts, each initialized by randomly selecting $k \in [0, \epsilon]$ bits to be flipped in v to obtain $\tilde{v}^{(0)}$. We note that initialization by randomly flipping bits is important as, without

initialization, i.e., $\tilde{w}^{(0)} := w$, the loss $\mathcal{L}$ will be close to zero. We also found that initializing with $k = \epsilon$ leads to difficulties in the first few iterations, which is why we sample $k \in [0, \epsilon]$ uniformly. Additionally, we normalize the gradient $\Delta^{(t)}$ in Equation (6.2):

$$\hat{\Delta}^{(t,l)} = \frac{\Delta^{(t,l)}}{\|\Delta^{(t,l)}\|_\infty} \tag{6.5}$$

for each layer l individually (considering biases as separate layer), before applying the update, i.e., $\tilde{w}^{(t+1)} = \tilde{w}^{(t)} + \gamma\hat{\Delta}^{(t)}$. Instead of considering $\tilde{w}^{(T)}$, i.e., the perturbed weights after exactly T iterations, we use

$$\tilde{w}^{(t^*)} \quad \text{with} \quad t^* = \arg\max_t \sum_{b=1}^{B} \mathcal{L}(f(x_b; \tilde{w}_q^{(t)}), y_b) \tag{6.6}$$

instead. Finally, we also use momentum. Nevertheless, despite these optimization tricks, the attack remains very sensitive to hyperparameters, especially regarding the step-size. Thus, running multiple random restarts is crucial.

6.3 Robustness Against Bit Errors

We address robustness against random and/or adversarial bit errors in three steps: First, we analyze the impact of fixed-point quantization schemes on bit error robustness. This has been neglected both in prior work on low-voltage accelerators [KHM+18a, KOY+19] and in work on quantization robustness [SSH15, MAA+16, MDI19]. This yields our **robust quantization** (Section 6.3.1). On top, we propose aggressive **weight clipping** as regularization during training (Section 6.3.2). Weight clipping enforces a more uniformly distributed, i.e., redundant, weight distribution, improving robustness. We show that this is due to minimizing the cross-entropy loss, enforcing large logit differences. Finally, in addition to robust quantization and weight clipping, we perform **random bit error training (RandBET)** (Section 6.3.3) or **adversarial bit error training (AdvBET)** (Section 6.3.4). For RandBET, in contrast to the fixed bit error patterns in [KHM+18a, KOY+19], we train on completely *random* bit errors and thus generalize across chips and voltages. Regarding AdvBET, we train on *adversarial* bit errors, computed as outlined in Section 6.2.2. Generalization of bit error robustness is measured using *robust test error (RErr)*, the test error after injecting bit errors (lower is more robust).

6.3.1 Robust Fixed-Point Quantization

We consider quantization-aware training [JKC+18, Kri18] using a generic, deterministic fixed-point quantization scheme commonly used in accelerators [CSC+19]. However, we focus on the impact of quantization schemes on robustness against random bit errors, mostly neglected so far [SSH15, MAA+16, MDI19]. We find that quantization affects robustness significantly, even if accuracy is largely unaffected.

Fixed-Point Quantization: Quantization determines how weights are represented in memory, e.g., on SRAM. In a *fixed-point quantization* scheme, m bits allow representing 2^m distinct values. A weight $w_i \in [-q_{\max}, q_{\max}]$ is represented by a *signed* m-bit integer $v_i = Q(w_i)$ corresponding to the underlying bits. Here, $[-q_{\max}, q_{\max}]$ is the *symmetric* quantization range and signed integers use two's complement representation. Then, $Q : [-q_{\max}, q_{\max}] \mapsto \{-2^{m-1} -$

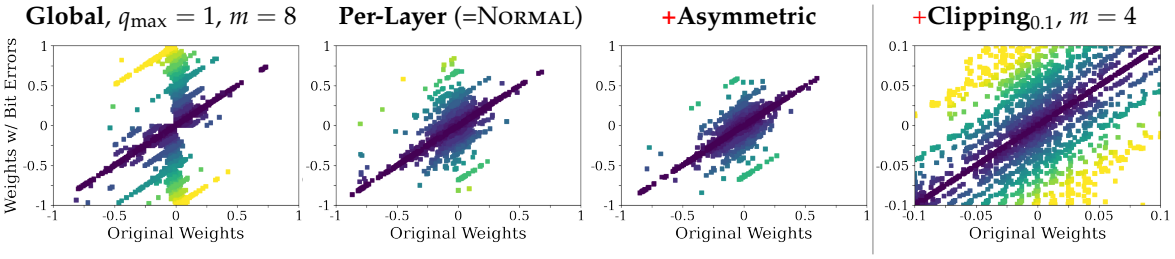


Figure 6.4: **Quantization and Random Bit Errors:** Original weights (x-axis) plotted against perturbed weights with bit errors (y-axis), for different fixed-point quantization schemes with $m = 8$ bit (left) and $p = 2.5\%$. We also show the $m = 4$ bit case with CLIPPING at $w_{max} = 0.1$, c.f. Section 6.3.2. Color indicates absolute error: from zero (violet) to the maximal possible error (yellow) of 1 (left) and 0.1 (right). Asymmetric per-layer quantization reduces the impact of bit errors compared to the symmetric per-layer/global quantization. Clipping reduces absolute error, but the errors *relative* to w_{max} increase.

$1, \dots, 2^{m-1} - 1\}$ is defined as

$$Q(w_i) = \left\lfloor \frac{w_i}{\Delta} \right\rfloor, \quad Q^{-1}(v_i) = \Delta v_i, \quad \Delta = \frac{q_{max}}{2^{m-1} - 1}. \tag{6.7}$$

This quantization is symmetric around zero and zero is represented exactly. By default, we only quantize weights, not activations or gradients. However, in contrast to related work [ZNZ+16, McD18, JKC+18, LWL+19], we quantize *all* layers, including biases and batch normalization parameters [IS15b] (commonly "folded" into preceding convolutional layers). Flipping the most significant bit (MSB, i.e., sign bit) leads to an absolute error of half the quantization range, i.e., q_{max} (yellow in Figure 6.4). Flipping the least significant bit (LSB) incurs an error of Δ. Thus, the impact of bit errors "scales with" q_{max}.

Global and Per-Layer Quantization: q_{max} can be chosen to accommodate all weights, i.e., $q_{max} = \max_i |w_i|$. This is called *global* quantization. However, it has become standard to apply quantization *per-layer* allowing to adapt q_{max} to each layer. As in PyTorch [PGC+17], we consider weights and biases of each layer separately. By reducing the quantization range for each layer individually, the errors incurred by bit flips are automatically minimized, c.f. Figure 6.4. The **per-layer, symmetric quantization is our default reference**, referred to as NORMAL. However, it turns out that it is further beneficial to consider arbitrary quantization ranges $[q_{min}, q_{max}]$ (allowing $q_{min} > 0$). In practice, we first map $[q_{min}, q_{max}]$ to $[-1, 1]$ using the transformation N and then quantize $[-1, 1]$ using Equation (6.7):

$$N(w_i) = \left(\frac{w_i - q_{min}}{q_{max} - q_{min}} \right) \cdot 2 - 1. \tag{6.8}$$

Overall, per-layer asymmetric quantization has the finest granularity, i.e., lowest Δ and approximation error. Nevertheless, it is not the most robust quantization.

Robust Quantization: Equation (6.7) does *not* provide optimal robustness against bit errors. First, the floor operation $\lfloor w_i/\Delta \rfloor$ is commonly implemented as float-to-integer conversion. Using proper rounding $\lceil w_i/\Delta \rfloor$ instead has negligible impact on accuracy, even though quantization error improves slightly. In stark contrast, bit error robustness is improved considerably. During training, deep neural networks can compensate the differences in approximation errors, even for small precision $m < 8$. However, at test time, rounding decreases the impact of bit errors considerably. Second, Equation (6.7) uses signed integers for symmetric quantization. For

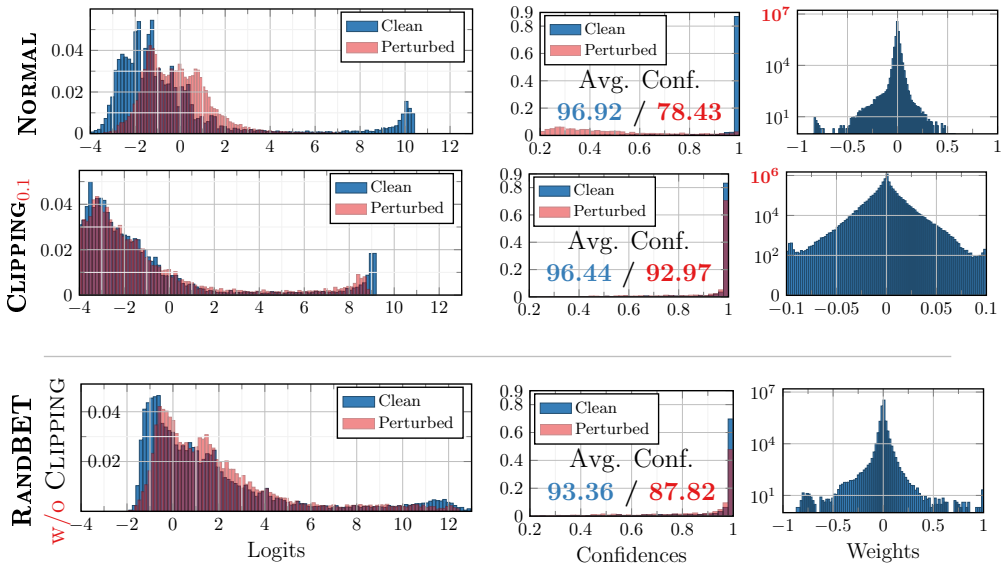


Figure 6.5: **Effect of Weight Clipping:** On CIFAR10, weight clipping constraints the weights (right), thereby implicitly limiting the possible range for logits (left, blue). However, even for $w_{\max}$=0.1 the deep neural network is able to produce high confidences (middle, blue), suggesting that more weights are used to obtain these logits. Furthermore, the impact of random bit errors, $p = 1\%$, on the logits/confidences (red) is reduced significantly. RANDBET (trained with $p = 1\%$, w/o weight clipping), increases the range of weights and is less effective at preserving logit/confidence distribution.

asymmetric quantization, with arbitrary $[q_{\min}, q_{\max}]$, we found quantization into *unsigned* integers to improve robustness, i.e., $Q : [q_{\min}, q_{\max}] \mapsto \{0, \dots, 2^m - 1\}$. This is implemented using an additive term of $2^{m-1} - 1$ in Equation (6.7):

$$Q(w_i) = \left\lceil \frac{w_i}{\Delta} \right\rfloor + (2^{m-1} - 1), \qquad Q^{-1}(v_i) = \Delta(v_i - (2^{m-1} - 1)) \tag{6.9}$$

While accuracy is not affected, the effect of bit errors in the sign bit changes: in symmetric quantization, the sign bit mirrors the sign of the weight value. For asymmetric quantization, an unsigned integer representation is more meaningful. Overall, our **robust fixed-point quantization (RQuant)** uses per-layer, asymmetric quantization into unsigned integers with rounding. These seemingly *small differences* have little to no impact on accuracy but tremendous impact on robustness against bit errors, see Section 6.4.4.

6.3.2 Training with Weight Clipping as Regularization

Simple **weight clipping** refers to constraining the weights to $[-w_{\max}, w_{\max}]$ *during training*, where $w_{\max}$ is a hyperparameter. Generally, $w_{\max}$ is independent of the quantization range(s) which always adapt(s) to the weight range(s) at hand. However, weight clipping limits the maximum possible quantization range (c.f. Section 6.3.1), i.e., $q_{\max} \leq w_{\max}$. It might seem that weight clipping with small $w_{\max}$ automatically improves robustness against bit errors as the absolute errors are reduced. However, the *relative* errors are not influenced by rescaling. As the model's decision is usually invariant to rescaling, reducing the scale of the weights does not impact robustness. In fact, the mean relative error of the weights in Figure 6.4 (right) increased with clipping at $w_{\max}$=0.1. Thus, weight clipping does *not* "trivially" improve robustness by

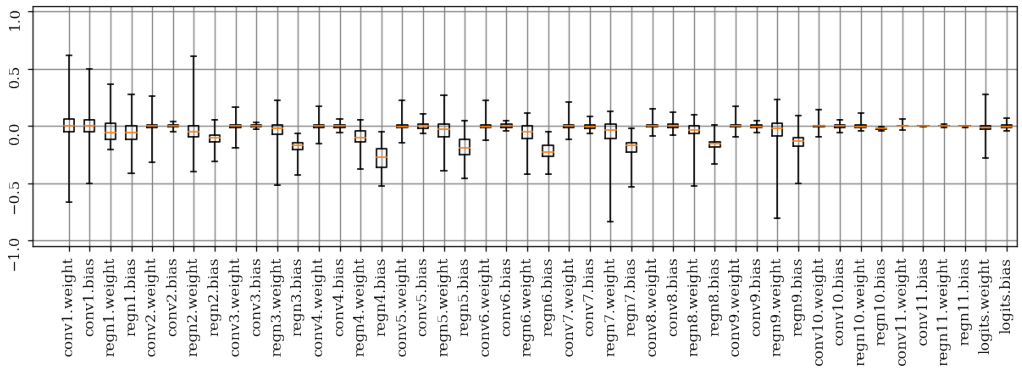


Figure 6.6: **Per-Layer Weight Clipping:** On CIFAR, we show the weight ranges, per layer, corresponding to RQUANT (i.e., without weight clipping). As can be seen, few layers exhibit large ranges. Thus, per-layer clipping computes a layer-specific weight constraint $[-w_{\max,l}, w_{\max,l}]$, derived from a global $w_{\max}$ and based on the weight range relative to the layer with largest range. As not to "over-constraint" layers with small weight range, a minimum range of $[-0.2w_{\max}, 0.2w_{\max}]$ is enforced.

reducing the scale of weights. Nevertheless, we found that weight clipping actually improves robustness considerably on top of our robust quantization.

The interplay of weight clipping and minimizing the cross-entropy loss during training is the key. High confidences can only be achieved by large differences in the logits. Because the weights are limited to $[-w_{\max}, w_{\max}]$, large logits can only be obtained using more weights in each layer to produce larger outputs. This is illustrated in Figure 6.5 (right): using $w_{\max}$=0.1, the weights are (depending on the layer) up to 5 times smaller. Considering deep neural networks, the "effective" scale factor for the logits is significantly larger, scaling exponentially with the number of layers. Thus, using $w_{\max}$=0.1 is a significant constraint on the ability of the deep neural network to produce large logits. As a result, weight clipping produces a much more uniform weight distribution. Figure 6.5 (left and middle) shows that a model constrained at $w_{\max}$=0.1 can produce similar logit and confidence distributions (in blue) as the unclipped one. And random bit errors have a significantly smaller impact on the logits and confidences (in red). Figure 6.5 (right column) also shows the induced redundancy in the weight distribution. Weight clipping leads to more weights being utilized, i.e., less weights are zero (note log-scale, marked in red, on the y-axis). Also, more weights reach large values. We found weight clipping to be an easy-to-use but effective measure to improve weight robustness.

Per-Layer Weight Clipping: We further extend "global" weight clipping by allowing per-layer weight constraints $w_{\max,l}$. This is based on the observation that weights in different layers can have radically different ranges as explicitly shown in Figure 6.6. We found that only few layers exhibit large weight ranges, e.g., the first few convolutional layers ("conv1" "and conv2" in Figure 6.6), few group normalization layers ("regn7" or "regn9") and the final logit layer ("logits"). Thus, these layers are affected significantly when reducing $w_{\max}$ and lead to poor (clean) Err in Table 6.4, e.g., for $w_{\max} = 0.05$. The regularization effect is less pronounced for the remaining layers, reducing the potential impact in terms of robustness. Thus, *per-layer* weight clipping constraints each layer l individually to $[-w_{\max,l}, w_{\max,l}]$. Here, weights and biases are treated individually as biases exhibit significantly different ranges. The per-layer constraints $w_{\max,l}$ are derived from the relative weight ranges without weight clipping. Letting $w_{l,i}$ be the weights of layer l with the largest absolute weight value, we define $\kappa_{l'} = {}^{\max_i |w_{l',i}|}/_{\max_i |w_{l,i}|} \leq 1$ for all other layers l'. Then, for each l', we define $w_{\max,l}$

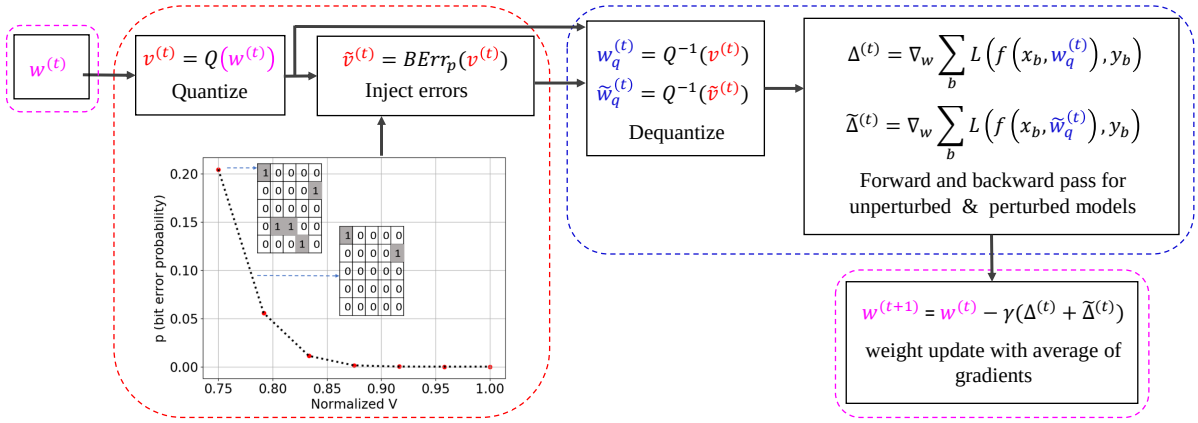


Figure 6.7: **Random Bit Error Training (RandBET):** We illustrate the data-flow for RandBET as in Algorithm 2. Here, BErr_p injects random bit errors in the quantized weights $v^{(t)} = Q(w^{(t)})$, resulting in $\tilde{v}^{(t)}$, while the forward pass is performed on the dequantized perturbed weights $\tilde{w}_q^{(t)} = Q^{-1}(\tilde{v}^{(t)})$, i.e., fixed-point arithmetic is not emulated. The weight update during training is not affected by bit errors and computed in floating point.

as $\max(0.2, \kappa_{l'}) w_{\max}$. We use $\text{Clipping}_{w_{\max}=0.1}$ to refer to *global* weight clipping with, e.g., $w_{\max} = 0.1$, and $\text{PLClipping}_{w_{\max}=0.25}$ to denote *per-layer* weight clipping with, e.g., $w_{\max} = 0.25$.

Weight Clipping with Group/Batch Normalization: While weight clipping during training is easy to implement, we have to make a simple adjustment to group and batch normalization layers [IS15b, WH18]: we reparameterize the scale parameter α of group/batch normalization, which usually defaults to $\alpha = 1$ and may cause problems when clipped, e.g., to $[-0.1, 0.1]$. In particular with aggressive weight clipping, $\alpha \leq w_{\max} \ll 1$, the normalization layers lose their ability to represent the identity function, considered important in [IS15b]. Our reparameterization introduces a learnable, auxiliary parameter α' such that α is $\alpha = 1 + \alpha'$ to solve this problem.

6.3.3 Random Bit Error Training (RandBET)

In *addition to* weight clipping and robust quantization, we inject random bit errors with probability p during training to further improve robustness. This results in the following learning problem, which we optimize as illustrated in Figure 6.7:

$$\min_w \mathbb{E}[\mathcal{L}(f(x; \tilde{w}), y) + \mathcal{L}(f(x; w), y)] \quad \text{s.t. } v = Q(w),\ \tilde{v} = \text{BErr}_p(v),\ \tilde{w} = Q^{-1}(\tilde{v}). \tag{6.10}$$

where (x, y) are labeled examples, $\mathcal{L}$ is the cross-entropy loss and $v = Q(w)$ denotes the (element-wise) quantized weights w which are to be learned. $\text{BErr}_p(v)$ injects random bit errors with rate p in v. Note that we consider both the loss on clean weights and weights with bit errors. This is desirable to avoid an increase in (clean) test error and stabilizes training compared to training only on bit errors in the weights. Note that bit error rate p implies, in expectation, pmW bit errors. Following Algorithm 2, we use stochastic gradient descent to optimize Equation (6.10), by performing the gradient computation using the perturbed weights $\tilde{w} = Q^{-1}(\tilde{v})$ with $\tilde{v} = \text{BErr}_p(v)$, while applying the gradient update on the (floating-point) clean weights w. In spirit, this is similar to data augmentation, however, the perturbation is applied on the weights instead of the inputs. As we found that introducing bit errors right from

Algorithm 2 Random Bit Error Training (RandBET): The forward passes are performed using dequantized weights (blue). Perturbed weights are obtained by injecting bit errors in the quantized weights (in red). The update, averaging gradients from both forward passes, is performed in floating-point (magenta). Also see Figure 6.7.

RandBET(p):
initialize $w^{(0)}$
for $t = 0, \dots, T-1$ **do**
  sample batch $\{(x_b, y_b)\}_{b=1}^{B}$
  $w^{(t)} = \min(w_{\max}, \max(-w_{\max}, w^{(t)}))$ # clipping
  $v^{(t)} = Q(w^{(t)})$ # quantization
  $w_q^{(t)} = Q^{-1}(v^{(t)})$ # dequantization
  # clean forward and backward pass:
  $\Delta^{(t)} = \nabla_w \sum_{b=1}^{B} \mathcal{L}(f(x_b; w_q^{(t)}), y_b)$
  # *perturbed* forward and backward pass:
  $\tilde{w}_q^{(t)} = Q^{-1}(\text{BErr}_p(v^{(t)}))$ (or AdvBitErrors($v^{(t)}, \epsilon$))
  $\tilde{\Delta}^{(t)} = \nabla_w \sum_{b=1}^{B} \mathcal{L}(f(x_b; \tilde{w}_q^{(t)}), y_b)$
  # average gradients and weight update:
  $w^{(t+1)} = w^{(t)} - \gamma(\Delta^{(t)} + \tilde{\Delta}^{(t)})$
end for
return $w_q^{(T)} = Q^{-1}(Q(w^{(T)}))$

the start may prevent the deep neural network from converging, we apply bit errors as soon as the (clean) cross-entropy loss is below 1.75. Interestingly, weight clipping and RandBET have somewhat orthogonal effects, which allows combining them easily in practice: While weight clipping encourages redundancy in weights by constraining them to $[-w_{\max}, w_{\max}]$, RandBET (w/o weight clipping) causes the model to have larger tails in the weight distribution, as shown in Figure 6.5 (bottom). However, considering logits and confidences, especially with random bit errors (in red), RandBET alone performs slightly worse than Clipping$_{0.1}$. Thus, RandBET becomes particularly effective when combined with weight clipping, as we make explicit using the notation RandBET$_{w_{\max}}$ and in Algorithm 2.

6.3.4 Adversarial Bit Error Training (AdvBET)

In order to specifically address adversarial bit errors (c.f. Section 6.2.2), RandBET can be reformulated to train with adversarial bit errors. Essentially, this results in a min-max formulation similar to [MMS+18]:

$$\min_w \mathbb{E}[\max_{\tilde{v}} \mathcal{L}(f(x; Q^{-1}(\tilde{v})), y)] \quad \text{s.t. } d_H(\tilde{v}, v) \leq \epsilon, d_H(\tilde{v}_i, v_i) \leq 1 \tag{6.11}$$

where the inner maximization problem, i.e., the attack is solved following Algorithm 1. In addition to not training on adversarial bit errors for a (clean) cross-entropy above 1.75, we clip gradients to $[-0.05, 0.05]$. This is required as the cross-entropy loss on adversarially perturbed weights $\tilde{w}$ can easily be one or two magnitudes larger than on the clean weights. Unfortunately, training is very sensitive to the hyperparameters of the attack, including the step size, gradient normalization and momentum. This holds both for convergence during training and for the obtained robustness after training.

6.4 Experiments

We present experiments on MNIST [LBBH98] and CIFAR [Kri09], considering *random* bit error robustness first, followed by discussing *adversarial* bit errors. After introducing our experimental setup (Section 6.4.1) and introducing robustness metrics including a simple generalization bound (Section 6.4.2), we first show that batch normalization (BN) [IS15a] reduces robustness significantly. Subsequently, we analyze the impact of fixed-point quantization schemes on robustness (Section 6.4.4) and discuss weight clipping (Clipping, Section 6.4.5), showing that improved robustness originates from increased redundancy in the weight distribution. Then, we focus on random bit error training (RandBET, Section 6.4.6). We show that related work [KHM+18a, KOY+19] does not generalize, while RandBET generalizes across chips and voltages, as demonstrated on profiled bit errors. We further consider random bit errors in inputs and activations (Section 6.4.7). Finally, we discuss our adversarial bit error attack in comparison to BFA [RHL+20] (Section 6.4.8) and show that Clipping as well as RandBET or AdvBET increase robustness against adversarial bit errors significantly. We start by discussing our experimental set

6.4.1 Setup and Baselines

Architecture: We use SimpleNet [HRFS16], providing comparable performance to ResNets [HZRS16a] with only W=5.5Mio weights on CIFAR10. On MNIST, we halve all channel widths and one stage of convolutional layers including a pooling layer is skipped, resulting in roughly 1Mio weights. SimpleNet compares favorably to, e.g., VGG [SZ15]: VGG-16 has 14M weights on CIFAR. Additionally, we found SimpleNet to be easier to train without batch normalization (BN) [IS15b]. On CIFAR100, we use a Wide ResNet (WRN) [ZK16]. In all cases, we replaced BN with (reparameterized) group normalization (GN) [WH18], as models using BN yield consistently worse robustness against bit errors.

Training: We use stochastic gradient descent to minimize cross-entropy loss. We use an initial learning rate of 0.05, multiplied by 0.1 after $^2/_5$, $^3/_5$ and $^4/_5$ of 100/250 epochs on MNIST/CIFAR. Our batch size is 128 and momentum of 0.9 is used together with weight decay of $5 \cdot 10^{-4}$. On CIFAR, we whiten the input images and use AutoAugment[2] [CZM+19] with Cutout [DT17]. Cutout is applied with a window size of 16×16, and independent of AutoAugment, we apply random cropping with up to 4 pixels. Created black spaces are filled using the mean image color (grayish). Initialization follows [HZRS15]. The full training set is used for training, and we do *not* rely on early stopping. For RandBET, we use $\lambda = 1$ and start injecting bit errors when the loss is below 1.75 on MNIST/CIFAR10 or 3.5 on CIFAR100. Normal training with the standard and our robust quantization are denoted Normal and RQuant, respectively. Weight clipping with $w_{\max}$ is referred to as $\text{Clipping}_{w_{\max}}$ and its per-layer variant is denoted $\text{PLClipping}_{w_{\max}}$. Similarly, we refer to RandBET/AdvBET with (global) weight clipping as $\text{RandBET}_{w_{\max}}$/$\text{AdvBET}_{w_{\max}}$ and with per-layer weight clipping as $\text{PLRandBET}_{w_{\max}}$. For RQuant, $m = 8$, we obtain 4.3% Err on CIFAR10 and 18.5% Err on CIFAR100; on MNIST, 0.47% Err for $m = 2$. For AdvBET, we use $\lambda = 0$ and also start injecting adversarial bit errors when the loss is 1.75 or smaller (3.5 on CIFAR100). We use $T = 10$ iterations of our adversarial bit error attack, with learning rate 0.5, no backtracking [SHS20] or momentum [DLP+18], and per-layer L_∞ gradient normalization.

[2] https://github.com/DeepVoltaire/AutoAugment

Bit Flip Attack (BFA) [RHF19a] Baseline: We follow the official PyTorch code[3]. Specifically, we use the provided implementation of BFA to attack our models by integrating our SimpleNet models into the provided attack/evaluation code. This means that we also use the quantization scheme of [RHF19a], not our robust fixed-point quantization scheme. However, our models still used RQUANT (and optionally CLIPPING or RANDBET) during training. We use $m = 8$ bits. We allow 5 bit flips per iteration, for a total of T iterations, totaling $\epsilon = 5 \cdot T$ bit flips. For comparability, we adapted the code to compute adversarial bit flips on the last 100 test examples and evaluate on the first 9000 test examples. We allow 5 restarts for each ϵ.

6.4.2 Robust Test Error and Generalization Bound

For evaluation, we report (clean) test error Err (lower is better, ↓), corresponding to *clean* weights, and **robust test error RErr** (↓) which is the **test error after injecting bit errors into the weights** on 9000 test examples. For random bit errors, we simulate 50 different chips with enough memory arrays to accommodate all weights as described in detail in Section 6.2.1. The pattern, i.e., spatial distribution, of bit errors for each chip is fixed, while across all 50 chips, bit errors are uniformly distributed. These bit errors are pre-determined once for all experiments, making our robustness results entirely comparable across all models and bit error rates p.

For adversarial bit errors, we report *max* (i.e., worst-case) RErr across a total of 80 restarts using the following settings: 5 restarts for the untargeted attack with learning rate 1 with and without momentum 0.9 (10 restarts in total) and 10 restarts for the targeted attack (1 for each potential target label on MNIST and CIFAR10). This is done attacking all layers. Furthermore, we use 5 untargeted restarts with momentum, and 10 targeted restarts for: attacking only the logit layer, only the first convolutional layer, both the logit and the first convolutional layer, or all layers *except* the logit or first convolutional layer. In total, this makes $20 + 4 \cdot 15 = 80$ restarts of our adversarial bit error attack. We use $T = 100$ iterations and perform optimization on 100 held-out test examples.

Bounding Generalization to Random Bit Errors: While we focus on *empirically* evaluating robustness, we briefly derive a simple probabilistic bound that sheds light on which deviations are to be expected from the empirical results reported later on. Let w denote the final weights of a trained neural network f. We test f using n i.i.d. test examples, i.e., $(x_i, y_i)_{i=1}^n$. Further, let $\tilde{w}$ denote the weights where each bit of the (quantized) weights w is flipped with probability p uniformly at random. Then, the *expected* clean and robust errors of f are given by

$$\mathbb{E}[\mathbb{1}_{f(x;w)\neq y}] = \Pr(f(x;w) \neq y), \quad \mathbb{E}[\mathbb{1}_{f(x;\tilde{w})\neq y}] = \Pr(f(x;\tilde{w}) \neq y).$$

Here, the weights of the neural network are themselves random variables. In the following, for simplicity, we use x, y, w, and $\tilde{w}$ to denote the random variables corresponding to test example, test label, weights and weights with random bit errors. With x_j, y_j, w_i and $\tilde{w}_i$ we denote the actual examples, i.e., actual examples (x_j, y_j) as well as clean and perturbed weights w_i, $\tilde{w}_i$. Then, the following proposition derives a simple, probabilistic bound on the deviation of expected robust error from the empirically measured RErr:

Proposition 1. Let $\tilde{w}_i$, $i = 1, \dots, l$ examples of weights with random bit errors. Then it holds

$$\Pr\left(\frac{1}{nl}\sum_{j=1}^{n}\sum_{i=1}^{l}\mathbb{1}_{f(x_j;\tilde{w}_i)\neq y_j} - \Pr(f(x;\tilde{w}) \neq y) \geq \epsilon\right) \leq (n+1)e^{-n\epsilon^2 \frac{l}{(\sqrt{l}+\sqrt{n})^2}}.$$

[3]https://github.com/elliothe/BFA

As alternative formulation, with probability $1-\delta$ it holds

$$\Pr(f(x;\tilde{w}_i) \neq y) \leq \frac{1}{nl}\sum_{j=1}^{n}\sum_{i=1}^{l} \mathbb{1}_{f(x_j;\tilde{w}_i)\neq y_j} + \sqrt{\frac{\log\left(\frac{n+1}{\delta}\right)}{n}}\frac{\sqrt{l}+\sqrt{n}}{\sqrt{l}}.$$

Proof. Let $0 < \alpha < 1$. Using the Hoeffding inequality and union bound, we have:

$$\begin{aligned}&\Pr\Big(\max_{j=1,\dots,n}\frac{1}{l}\sum_{i=1}^{l}\mathbb{1}_{f(x_j;\tilde{w}_i)\neq y_j} - \mathbb{E}_{\tilde{w}}[\mathbb{1}_{f(x_j;\tilde{w})\neq y_j}] > \alpha\epsilon\Big)\\ =&\Pr\Big(\bigcup_{j=1,\dots,n}\{\frac{1}{l}\sum_{i=1}^{l}\mathbb{1}_{f(x_j;\tilde{w}_i)\neq y_j} - \mathbb{E}_{\tilde{w}}[\mathbb{1}_{f(x_j;\tilde{w})\neq y_j}] > \alpha\epsilon\}\Big) \leq\ n\,e^{-l\alpha^2\epsilon^2}.\end{aligned}$$

Then, again by Hoeffding's inequality, it holds:

$$\Pr\Big(\frac{1}{n}\sum_{j=1}^{n}\mathbb{E}_{\tilde{w}}[\mathbb{1}_{f(x_j;\tilde{w})\neq y_j}] - \mathbb{E}_{x,y}[\mathbb{E}_{\tilde{w}}[\mathbb{1}_{f(x;\tilde{w})\neq y}]] > (1-\alpha)\epsilon\Big) \leq\ e^{-n\epsilon^2(1-\alpha)^2}.$$

Thus, using $a+b>\epsilon \Longrightarrow \{a>\alpha\epsilon\}\cup\{b>(1-\alpha)\epsilon\}$ gives us:

$$\begin{aligned}&\Pr\Big(\frac{1}{nl}\sum_{j=1}^{n}\sum_{i=1}^{l}\mathbb{1}_{f(x_j;\tilde{w}_i)\neq y_j} - \Pr(f(x;\tilde{w})\neq y) \geq \epsilon\Big)\\ =&\Pr\Big(\frac{1}{n}\sum_{j=1}^{n}\big(\frac{1}{l}\sum_{i=1}^{l}\mathbb{1}_{f(x_j;\tilde{w}_i)\neq y_j} - \mathbb{E}_{\tilde{w}}[\mathbb{1}_{f(x_j;\tilde{w}_i)\neq y_j}]\big) + \frac{1}{n}\sum_{j=1}^{n}\mathbb{E}_{\tilde{w}}[\mathbb{1}_{f(x_j;\tilde{w}_i)\neq y_j}] - \Pr(f(x;\tilde{w})\neq y) \geq \epsilon\Big)\\ \leq&\Pr\Big(\frac{1}{n}\sum_{j=1}^{n}\big(\frac{1}{l}\sum_{i=1}^{l}\mathbb{1}_{f(x_j;\tilde{w}_i)\neq y_j} - \mathbb{E}_{\tilde{w}}[\mathbb{1}_{f(x_j;\tilde{w})\neq y_j}]\big) > \alpha\epsilon\Big)\\ &+\Pr\Big(\frac{1}{n}\sum_{j=1}^{n}\mathbb{E}_{\tilde{w}}[\mathbb{1}_{f(x_j;\tilde{w})\neq y_j}] - \Pr(f(x;\tilde{w})\neq y) \geq (1-\alpha)\epsilon\Big) \leq n\,e^{-l\alpha^2\epsilon^2} + e^{-n\epsilon^2(1-\alpha)^2}\end{aligned}$$

Having both exponential terms have the same exponent yields $\alpha = \frac{\sqrt{n}}{\sqrt{l}+\sqrt{n}}$ and we get the upper bound of the proposition. □

The samples of bit error injected weights $\{\tilde{w}_i\}_{i=1}^{l}$ can actually be different for every test example (x_j, y_j), even though this is not the case in our evaluation. Thus, the above bound involves a stronger result: for any test example, the empirical test error with random bit errors (i.e., robust test error RErr) and the expected one have to be similar with the same margin. Note also that this bound holds for any fixed bit error distribution as the only requirement is that the bit error patterns we draw are i.i.d., but not necessarily the bit errors on these patterns. Later, we will consider results with $l = 10^6$, i.e., $l \gg n$ with $n = 10^4$ on CIFAR10 such that $l/(\sqrt{l}+\sqrt{n})^2$ tends towards one. With $\delta = 0.01$ the excess term

$$\sqrt{\frac{\log\left(\frac{n+1}{\delta}\right)}{n}}\frac{\sqrt{l}+\sqrt{n}}{\sqrt{l}} \tag{6.12}$$

is equal to 4.09%. Larger test sets provide stronger guarantees, e.g., $n = 10^5$ yields 1.67%.

Normalization		Err	RErr in %	
(CIFAR10)		in %	p=0.1	p=0.5
GN	NORMAL	4.32	5.54	11.28
	CLIPPING$_{0.1}$	4.82	5.58	6.95
BN w/ *Accumulated* Statistics				
BN	NORMAL	3.83	6.36	52.52
	CLIPPING$_{0.1}$	4.46	5.32	8.25
BN w/ Batch Statistics at Test Time				
BN	NORMAL	3.83	6.65	9.63
	CLIPPING$_{0.1}$	4.46	6.57	7.29

Table 6.1: **Batch Normalization is not Robust:** RErr with group normalization (GN) or batch normalization (BN). RErr increases when using BN even though clean Err improves slightly compared GN. However, using batch statistics at test time (i.e., "training mode" in PyTorch) improves RErr significantly indicating that the statistics accumulated throughout training do not account for random bit errors. Thus, we use group normalization as default.

6.4.3 Batch Normalization is not Robust

While BN generally reduces test error, from 4.3% to 3.8% for our SimpleNet on CIFAR10, we found that robustness against random bit errors is reduced significantly. Specifically, Table 6.1 demonstrates that RErr increases significantly when using BN compared to GN indicating that BN is more vulnerable to bit errors in the weights. For example, for p=0.5% and without clipping, RErr increases from 11.28% to staggering 52.52% when replacing GN with BN. We suspect that the running statistics accumulated during training do not account for the random bit errors at test time, even for RANDBET. This is confirmed in Table 6.1 (bottom) showing that RErr reduces significantly when using the batch statistics at test time. Generally, BN improves accuracy, but might not be beneficial in terms of robustness, as also discussed for adversarial examples [GGT+19]. Using GN also motivates our use of SimpleNet instead of, e.g., ResNet-50, which generally performs worse with GN.

6.4.4 Quantization Choice Impacts Robustness

Quantization schemes affect robustness significantly, even when not affecting accuracy. For example, Table 6.2 shows that per-layer quantization reduces RErr significantly for small bit error rates, e.g., $p = 0.05\%$. While asymmetric quantization further reduces the quantization range, RErr increases, especially for large bit error rates, e.g., $p = 0.5\%$ (marked in red). This is despite Figure 6.4 showing a slightly smaller impact of bit errors. This is caused by an asymmetric quantization into *signed* integers in two's complement representation: Bit flips in the most significant bit (MSB, i.e., sign bit) are not meaningful if the quantized range is not symmetric as the sign bit does not reflect the sign of the represented weight value[4]. For larger bit error rates p, this happens more and more frequently, having a larger impact on RErr.

Similarly, replacing integer conversion of w_i/Δ by proper rounding, $\lceil w_i/\Delta \rfloor$, reduces RErr significantly (resulting in our RQUANT). We emphasize that, for $m = 8$ bit, there is *no* significant difference in terms of clean Err. However, using proper rounding reduces the approximation error slightly. Using $p = 2.5\%$ bit error rate, the average absolute error (in the weights) across 10 random bit error patterns reduces by 2%. Nevertheless, it has significantly larger impact on RErr. For $m = 4$, this is more pronounced: rounding reduces the average absolute error by roughly 67%. Surprisingly, this is not at all reflected in the clean Err, which only decreases from 5.81% to 5.29%. It seems that the deep neural network learns to compensate these errors

[4]An *un*signed integer of value 127 is represented as 01111111. Flipping the most significant (left-most) bit gives 11111111 corresponding to 255, i.e., the value increases. For a signed integer in two's complement representation, the same bit flip changes the value to -1, while 0-to-1 flips not affecting the sign bit generally increase the value.

	Quantization Schemes (CIFAR10)	Err in %	RErr in % p=0.05	RErr in % p=0.5
8 bit	Equation (6.7), global	4.63	86.01 ±3.65	90.71 ±0.49
	Equation (6.7), per-layer	4.36	5.51 ±0.19	24.76 ±4.71
	+asymmetric	4.36	6.47 ±0.22	40.78 ±7.56
	+unsigned	4.42	6.97 ±0.28	17.00 ±2.77
	+rounding (=RQUANT)	4.32	**5.10** ±0.13	**11.28** ±1.47
4 bit	w/o rounding*	5.81	90.40 ±0.21	90.36 ±0.2
	w/ rounding*	**5.29**	**5.75** ±0.06	**7.71** ±0.36

Table 6.2: **Quantization Robustness:** RErr for random bit errors at p=0.05% and p=0.5% for normal training with different quantization schemes. Minor differences can have large impact on RErr while clean test error is largely unaffected. For 8 bit the second row corresponds to NORMAL. For 4 bits we show CLIPPING$_{0.1}$ +RQUANT with and without rounding.

Model (CIFAR10)	Err in %	Conf in %	Conf p=1	RErr in % p=0.1	RErr in % p=1
RQUANT	**4.32**	**97.42**	78.43	5.54	32.05
CLIPPING$_{0.15}$	4.42	96.90	88.41	5.31	13.08
CLIPPING$_{0.1}$	4.82	96.66	92.97	5.58	8.93
CLIPPING$_{0.05}$	5.44	95.90	94.73	5.90	7.18
CLIPPING$_{0.025}$	7.10	84.69	83.28	7.40	8.18
CLIPPING$_{0.15}$ +LS	4.67	88.22	47.55	5.83	29.40
CLIPPING$_{0.1}$ +LS	4.82	87.90	78.89	6.10	10.59
CLIPPING$_{0.05}$ +LS	5.30	87.41	85.04	6.43	7.30
PLCLIPPING$_{0.5}$	4.61	97.06	91.31	5.48	10.9
PLCLIPPING$_{0.25}$	4.96	96.90	**95.67**	**5.39**	7.04
PLCLIPPING$_{0.1}$	5.62	94.57	93.98	5.91	**6.65**

Table 6.3: **Weight Clipping Robustness:** Clean Err, RErr, clean confidence and confidence at p=1% bit errors (in %, higher is better, ↑) for CLIPPING, CLIPPING with label smoothing (+LS) and PLCLIPPING. Err increases with extreme weight clipping. LS consistently reduces robustness, indicating that robustness is due to enforcing high confidence during training *and* weight clipping. Per-layer weight constraints are beneficial in terms of both robustness and clean performance.

during training. At test time, however, RErr reflects this difference in terms of robustness.

Overall, we found that robust quantization plays a key role. While both weight clipping (CLIPPING) and random bit error training (RANDBET) can improve robustness further, robust quantization lays the foundation for these improvements to be possible. Thus, we encourage authors to consider robustness in the design of future network quantization schemes. Even simple improvements over our basic fixed-point quantization scheme may have significant impact in terms of robustness. For example, proper handling of outliers [ZST+18, SSH15], learned quantization [ZYYH18], or adaptive/non-uniform quantization [ZMCF18, PYV18, NvBBW19] are promising directions to further improve robustness. Finally, we believe that this also poses new theoretical challenges, i.e., studying (fixed-point) quantization with respect to robustness *and* quantization error.

6.4.5 Weight Clipping Improves Robustness

While the quantization range adapts to the weight range after every update during training, weight clipping explicitly constraints the weights to $[-w_{\max}, w_{\max}]$. Table 6.3 shows the effect of different $w_{\max}$ for CIFAR10 with 8 bit precision. The clean test error is not affected for CLIPPING$_{\mathbf{w_{max}=0.15}}$ but one has already strong robustness improvements for $p = 1\%$ compared to RQUANT (RErr of 13.18% vs 32.05%). Further reducing $w_{\max}$ leads to a slow increase in clean Err and decrease in average clean confidence, while significantly improving RErr to 7.18% for $p = 1\%$ at $w_{\max} = 0.05$. For $w_{\max} = 0.025$ the deep neural network is no longer able to achieve

Model	Err	RErr in %	
(CIFAR10, see text)	in %	p=0.1	p=1
NORMAL	4.32	5.54	32.05
CLIPPING$_{0.1}$	4.82	5.58	8.93
NORMAL $\rightarrow$ CLIPPING$_{0.1}$	4.32	5.55	88.71
CLIPPING$_{0.1}$ $\rightarrow$ NORMAL	4.82	78.47	9.20

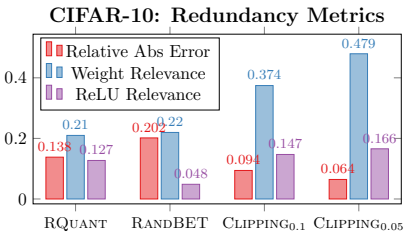


Figure 6.8: **Left: Weight Clipping with Weight Scaling:** For group normalization (GN) without the reparameterization, our deep neural networks are scale-invariant. Scaling RQUANT down to the weight range of CLIPPING$_{0.1}$, however, does not improve robustness. More importantly, scaling CLIPPING$_{0.1}$ up to have the same weight range as RQUANT preserves robustness. Thus, the robustness benefit of CLIPPING is *not* due to reduced quantization range or smaller absolute errors. **Right: Measuring Redundancy in Weight Clipping.** We plot various measures of redundancy, see the text for discussion and details. The *relative* absolute error is computed considering random bit errors with probability $p = 1\%$.

high confidence (marked in red) which leads to stronger loss of clean Err. Interestingly, the gap between clean and perturbed confidences under bit errors for $p = 1\%$ is (almost) monotonically decreasing. These findings generalize to other datasets and precisions. However, for low precision $m \leq 4$ the effects are stronger as RQUANT alone does not yield any robust models and weight clipping is essential for achieving robustness.

As discussed in Section 6.3.2, the robustness originates in the cross-entropy loss enforcing high confidences on the training set and thus large logits while weight clipping works against having large logits. Therefore, the network has to utilize more weights with larger absolute values (compared to $w_{\max}$). In order to test this hypothesis, we limit the confidences that need to be achieved via label smoothing [SVI+16], targeting 0.9 for the true class and $^{0.1}/_{9}$ for the other classes. According to Section 6.3.2, this should lead to less robustness, as the model has to use "less" weights. Indeed, in Table 6.3, RErr at $p = 1\%$ increases from 13.08% for CLIPPING$_{0.15}$ to 29.4% when using label smoothing (marked in blue). Moreover, the difference between average clean and perturbed confidence is significantly larger for neural networks trained with label smoothing.

Figure 6.8 (left) presents another experiment in support of our hypothesis. When using GN, the trained models are scale-invariant in their weights (e.g., as discussed in [DPBB17]). So, we down-scale the weights of RQUANT to have the same maximum absolute weight value as CLIPPING$_{0.1}$, referred to as RQUANT $\rightarrow$ CLIPPING$_{0.1}$. This scaling is applied globally, not per layer and does not affect the GN parameters. Similarly, we up-scale the weights of CLIPPING$_{0.1}$ to the same maximum absolute weight value as RQUANT, denoted CLIPPING$_{0.1}$ $\rightarrow$ RQUANT. However, "just" down-scaling does not induce robustness, as expected. More importantly, up-scaling the weights after training with weight clipping, $w_{\max} = 0.1$, preservers robustness. This emphasizes that robustness does not depend on scale, i.e., weight clipping does *not* reduce the impact of bit errors by reducing scale, but stems from increased weight redundancy.

In order to quantify the obtained redundancy, Figure 6.8 (right) presents three simple measures of redundancy in both weights and activations. The *relative absolute error* is computed with respect to $p = 1\%$ bit error rate and decreases for CLIPPING, meaning that random bit errors have less impact. *Weight relevance* is computed as the mean of absolute weights normalized by the maximum absolute weight. This metric shows that more weights become, considering their absolute value, relevant when using CLIPPING. Finally, We also measure activation redundancy using *ReLU relevance*. Here, we compute the fraction of non-zero activations after the final ReLU activation. Again, CLIPPING increases redundancy significantly. Overall, these measures

Model (CIFAR10)	RErr in %, p in %	
Evaluation on Fixed Pattern	p=1	p=2.5
PATTBET p=2.5	14.14	7.87
PATTBET$_{0.15}$ p=2.5	8.50	7.41
Evaluation on *Random* Patterns	p=1	p=2.5
PATTBET$_{0.15}$ p=2.5	12.09	61.59

Table 6.4: **Fixed Pattern Bit Error Training:** RErr for training on an entirely fixed bit error pattern (PATTBET). *Top:* Evaluation on the same pattern. PATTBET trained on $p = 2.5\%$ does not generalize to $p = 1\%$ even though the bit errors for $p = 1\%$ are a subset of those seen during training for $p = 2.5\%$ (in red). *Bottom:* PATTBET also fails to generalize to random bit errors.

	Model (CIFAR10) p in %	Err in %	RErr in % p=0.5	p=1	p=1.5
8bit	RQUANT	**4.32**	11.28 ±1.47	32.05 ±6	68.65 ±9.23
	CLIPPING$_{0.1}$	4.82	6.95 ±0.24	8.93 ±0.46	12.22 ±1.29
	PLCLIPPING$_{0.25}$	4.96	6.21 ±0.16	7.04 ±0.28	8.14 ±0.49
	RANDBET$_{0.1}$ p=0.1	4.72	6.74 ±0.29	8.53 ±0.58	11.40 ±1.27
	RANDBET$_{0.1}$ p=1	4.90	6.36 ±0.17	7.41 ±0.29	8.65 ±0.37
	PLRANDBET$_{0.25}$ p=0.1	4.49	5.80 ±0.16	6.65 ±0.22	7.59 ±0.34
	PLRANDBET$_{0.25}$ p=1	4.62	**5.62** ±0.13	**6.36** ±0.2	**7.02** ±0.27
4bit	CLIPPING$_{0.1}$	5.29	7.71 ±0.36	10.62 ±1.08	15.79 ±2.54
	PLCLIPPING$_{0.25}$	**4.63**	6.15 ±0.16	7.34 ±0.33	8.70 ±0.62
	RANDBET$_{0.1}$ p=1	5.39	7.04 ±0.21	8.34 ±0.42	9.77 ±0.81
	PLRANDBET$_{0.25}$ p=1	4.83	**5.95** ±0.12	**6.65** ±0.19	**7.48** ±0.32

Table 6.5: **Random Bit Error Training (RandBET).** RErr (with standard deviation) of RANDBET evaluated at various bit error rates p for m=8, 4 bits. For low p, weight clipping provides sufficient robustness, especially considering PLCLIPPING. However for $p \geq 0.5$, RANDBET increases robustness significantly, both based on CLIPPING and PLCLIPPING, especially for m=4bits.

support our hypothesis that improved robustness is caused by increased weight redundancy.

Per-layer weight clipping, i.e., PLCLIPPING, further improves robustness and at the same time lowers test error compared to CLIPPING. For example, in Table 6.3, PLCLIPPING$_{0.2}$ reduces RErr for p=1% to 6.48% compared to 7.18 for CLIPPING$_{0.05}$. Simultaneously, clean Err improves from 5.44% to 4.84. This emphasizes that layers can have radically different weight ranges and thus regularization through weight clipping needs to be layer-specific. Appendix C.4.3 shows that weight clipping also leads to robustness against L_∞ perturbations which generally affect all weights in contrast to random bit errors, and provides more qualitative results about the change of the weight distribution induced by clipping.

6.4.6 RANDBET Yields Generalizable Robustness

In the following, we present experiments on RANDBET, showing that training on fixed, profiled bit errors patterns is not sufficient to generalize across voltages and chips. Thus, training on *random* bit errors in RANDBET is essential, and further improves robustness when applied on top of RQUANT and CLIPPING. Finally, we present results when evaluating RANDBET on real, profiled bit errors corresponding to two different chips. Furthermore, both CLIPPING and RANDBET can also be applied in a post-training quantization setting by replacing random bit errors during RANDBET with L_0 errors in the (non-quantized) weights.

Training on Profiled Errors Does Not Generalize: Co-design approaches such as [KHM+18a, KOY+19] combine training on profiled SRAM or DRAM bit errors with hardware-approaches to limit the errors' impact. However, profiling SRAM or DRAM requires expensive infrastructure, expert knowledge and time. More importantly, training on profiled bit errors does not

Chip (Figure 6.3)	Model (CIFAR10)	RErr in %	
Chip 1		$p\approx0.86$	$p\approx2.75$
	RandBET$_{0.05}$ p=1.5	7.04	9.37
	PLRandBET$_{0.15}$, p=2	6.14	7.58
Chip 2		$p\approx0.14$	$p\approx1.08$
	RandBET$_{0.05}$ p=1.5	6.00	9.00
	PLRandBET$_{0.15}$, p=2	5.34	7.34

Table 6.6: **Generalization to Profiled Bit Errors:** RErr for RandBET and PLRandBET on two different profiled chips. The bit error rates differ across chips due to different voltages. For chip 2, bit errors are strongly aligned along columns and biased towards 0-to-1 flips. Nevertheless, RandBET generalizes surprisingly well.

Model (CIFAR10) (p_{L_0}= L_0 error rate)	Err in %	RErr in %, p = 1%	
		8bit	4bit
PLClipping$_{0.5}$	4.61	11.28 ±1.14	16.93 ±2.77
PLClipping$_{0.2}$	5.08	6.85 ±0.24	7.21 ±0.23
PL$_0$RandBET$_{0.2}$, p_{L_0}=1	5.01	6.58 ±0.17	7.01 ±0.22
PL$_0$RandBET$_{0.2}$, p_{L_0}=4	5.23	**6.57** ±0.13	**6.89** ±0.13
PL$_0$RandBET$_{0.2}$, p_{L_0}=8	5.49	6.73 ±0.16	6.95 ±0.14
PLRandBET$_{0.2}$, p=1	4.92*	6.29 ±0.14	6.60 ±0.18

Table 6.7: **Robustness of Post-Training Quantization:** RErr against p=1% random bit errors for PLClipping and L_0RandBET, i.e., error training with L_0 errors on weights, and post-training quantization. Weight clipping and L_0-based error training allow obtaining robustness without knowing the quantization scheme in advance.

generalize to previously unseen bit error distributions (e.g., other chips or voltages): Table 6.4 (top) shows RErr of PattBET, i.e., pattern-specific bit error training. The main problem is that PattBET does not even generalize to lower bit error rates (i.e., higher voltages) of the same pattern as trained on (marked in red). This is striking as the bit errors form a subset of the bit errors seen during training: training with $p = 2.5\%$ bit errors does not provide robustness for $p = 1\%$, RErr increases 7.9% to 14.1%. It is not surprising, that Table 6.4 (bottom) also demonstrates that PattBET does not generalize to random bit error patterns: RErr increases from 7.4% to 61.6% at $p = 2.5\%$. The same observations can be made when training on real, profiled bit errors corresponding to the chips in Figure 6.3. However, obtaining robustness that generalizes across voltages *and* chips is crucial for low-voltage operation to become practical.

RandBET Improves Robustness: Our RandBET, combined with weight clipping, further improves robustness and additionally generalizes across chips and voltages. Table 6.5 shows results for weight clipping and RandBET with $w_{\max} = 0.1$ and $m = 8, 4$ bits precision. RandBET is particularly effective against large bit error rates, e.g., $p = 1.5\%$, reducing RErr from 12.22% to 8.65% ($m = 8$ bits) with global weight clipping and even further to 7.02% with per-layer clipping, i.e. PLRandBET. The effect is pronounced for 4 bits or even lower precision, where models are generally less robust. The optimal combination of weight clipping and RandBET depends on the bit error rate. However, we note that RandBET consistently improves over Clipping or PLClipping. We also emphasize that RandBET generalizes to lower bit errors than trained on, in stark contrast to the fixed-pattern training PattBET. On other datasets, e.g., MNIST, RandBET allows operating at $p = 15\%$ bit error rate with 0.68% RErr and only $m = 2$ bits. At this point, weight clipping alone yields $\geq$90% RErr.

RandBET Generalizes to Profiled Bit Errors: RandBET also generalizes to bit errors profiled from real chips. Table 6.6 shows results on the two profiled chips of Figure 6.3. Profiling was done at various voltage levels, resulting in different bit error rates for each chip. To simulate various weights to memory mappings, we apply various offsets before linearly mapping weights to the profiled SRAM arrays. Table 6.6 reports average RErr, showing that RandBET generalizes quite well to these profiled bit errors. Regarding chip 1, RandBET performs very

CIFAR10: Stress Test for Guarantees			
Model	Err	RErr in %, $p = 1\%$	
(CIFAR10)	in %	$l = 50$	$l = 1$Mio
RQuant	4.32	32.05 ±6	31.97 ±6.35
Clipping$_{0.05}$	5.44	7.18 ±0.16	7.19 ±0.2
RandBET$_{0.05}$ p=2	5.42	6.71 ±0.11	6.73 ±0.15
PLClipping$_{0.15}$	5.31	6.53 ±0.14	6.52 ±0.14
PLRandBET$_{0.25}$, p=1	4.62	6.36 ±0.2	6.29 ±0.2
PLRandBET$_{0.15}$, p=2	4.99	**6.12** ±0.13	**6.12** ±0.14

Table 6.8: **Results for Probabilistic Guarantees:** Average RErr and standard deviation for $l = 1$Mio random bit error patterns. In comparison with the results for $l = 50$ in Table 6.5, there are no significant changes in RErr.

well, even for large $p \approx 2.75$, as the bit error distribution of chip 1 largely matches our error model. In contrast, with chip 2 we picked a more difficult bit error distribution which is strongly aligned along columns, potentially hitting many MSBs simultaneously. Thus, RErr is similar for chip 2 even for a lower bit error rate $p \approx 1.08$ (marked in red) but energy savings are still possible without degrading prediction performance significantly.

RandBET and Post-Training Quantization: So far, we applied quantization during training, i.e., we performed quantization-aware training [JKC+18, Kri18]. However, both (global and per-layer) weight clipping as well as (bit) error training can be applied in a post-training quantization setting. To this end, for RandBET, bit errors are simulated through L_0 noise on weights. Specifically, with probability p_{L_0} each weight w_i is changed to a (uniformly) random value $\tilde{w}_i \in [-w_{\max}, w_{\max}]$. The same error model applies for per-layer weight clipping. Note that training with L_0 errors with probability p_{L_0} simulates bit error training with $p = m \cdot p_{L_0}$, referred to as L_0RandBET. We apply our robust fixed-point quantization with $m = 8$ bits *after* training to evaluate robustness to random bit errors. In Table 6.7, we demonstrate that both Clipping and L_0RandBET also provide robustness in a post-training quantization context. This allows training robust models without knowing the exact quantization and precision used for deployment in advance.

Guarantees from Proposition 1: Based on the bound derived in Section 6.4.2, we conduct experiments with $l = 1$Mio random bit error patterns, such that $l \gg n$ where $n = 10$k is the number of test examples on CIFAR10. Considering Proposition 1, this would guarantee a deviation in RErr of at most 4.09% with probability at least 99%. As shown in Table 6.8, the obtained RErr with 1Mio random bit error patterns does not deviate significantly from the results with only 50 patterns. Only the standard deviation of RErr increases slightly.

Summary: Our experiments are summarized in Figure 6.9. We consider Normal (orange) quantization vs. our robust quantization RQuant (red) as well as various Clipping and RandBET models with different $w_{\max}$ and p during training (indicated in • gray), highlighting the best model for each bit error rate in blue and violet, respectively. On all datasets RQuant outperforms Normal. On CIFAR10 (left), RErr increases significantly for RQuant starting at $p \approx 0.25\%$ bit error rate. While Clipping generally reduces RErr, only RandBET can keep RErr around 6% or lower for a bit error rate of $p \approx 0.5\%$. CIFAR100 is generally more difficult, while significantly higher bit error rates are possible on MNIST. On CIFAR10, RErr increases slightly for $m = 4$. However, RErr for $m = 3, 2$ increases more significantly as clean Err increases by $1 - 2\%$. Nevertheless, RErr only increases slightly for larger bit error rates p. In all cases, RErr increases monotonically, ensuring safe operation at higher voltages. The best trade-off depends on the application: higher energy savings require a larger "sacrifice" in terms of RErr.

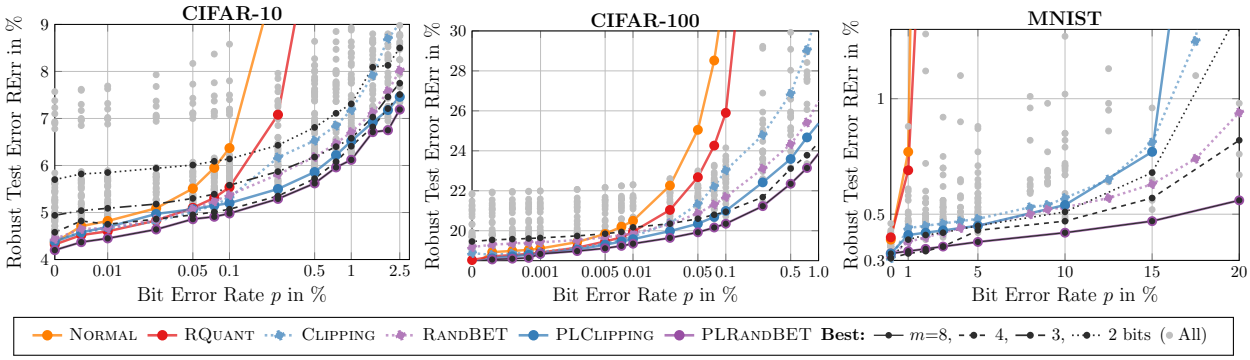


Figure 6.9: **Bit Error Robustness on CIFAR10, CIFAR100 and MNIST:** Average RErr plotted against bit error rate p, both in %. We considered various models (in • gray), corresponding to different $w_{\max}$ and p during training. We explicitly plot the best model for each bit error rate: for NORMAL (orange), RQUANT (red), CLIPPING (blue) and RANDBET (violet). Note that these might correspond to different $w_{\max}$ and p (also across datasets). Across all approaches, we plot the per-error-rate best model in black: for $m = 8, 4, 3, 2$ bits, depending on dataset. For 8 bit and low bit error rates, CLIPPING is often sufficient. However, for 4 bit or higher bit error rates, RANDBET is crucial to keep RErr low.

6.4.7 Robustness to Bit Errors in Inputs and Activations

While RANDBET successfully improves robustness against low-voltage induced bit errors in the *weights*, both inputs and activations might also be subject to random bit errors when (temporarily) stored on the SRAM scratchpad. Thus, we also consider injecting bit errors in inputs and activations, making first steps towards a "fully" robust deep neural network. First, we take a closer look at the impact of bit errors in inputs and activations. Then, we adapt RANDBET to improve robustness. For clarity, in text and Table 6.9, we use p_w, p_i and p_a to denote the bit error rate in weights, inputs and activations, respectively. We further color-code bit errors in inputs as orange and activations as violet.

Bit Error Model in Inputs and Activations: Following our description in Section 6.2.1, we inject bit errors in both inputs and activations. Inputs are quantized using $m = 8$ bit with $[q_{\min}, q_{\max}] = [0, 1]$ using asymmetric quantization into unsigned integers. Note that this does not introduce errors as images are typically provided in 8 bit quantization per channel (i.e., 256 distinct values per channel). Activations are also quantized using $m = 8$ bit using our robust fixed-point quantization scheme. Note that we do not employ any advanced activation quantization schemes such as activation clipping [CWV+18]. Bit errors are injected *once* into inputs before being fed into the model and *once* into the activations after *each* block consisting of convolutional layer, normalization layer (i.e., GN) and ReLU activation. In our SimpleNet, there are 13 such blocks on CIFAR10. This assumes that activations after each such block are temporally stored on SRAM scratchpads. While the actual data flow is highly specific to both chip and deep neural network architecture, this is a realistic assumption. As with bit errors in the weights, we evaluate using 50 random bit error patterns and make sure that for rate $p' \leq p$ the bit errors introduced in inputs/activations are a subset of those for rate p. As we assume activations after different blocks to be stored on separate parts of the memory, the bit errors are uncorrelated across activations.

Input and Activation Bit Error Robustness: Bit errors have severe impact on accuracy not only when occurring in weights but also in inputs and activations. Table 6.9 shows robustness,

Model (CIFAR10)		bit errors in weights		bit errors in inputs			bit errors in act.	
$\mathbf{w}_{\max}$=**0.25**, m=8 bit weight/input/act. quantization bit errors in weights/inp./act., p in %	Err in %	RErr in %		RErr in %		Err in % (act. quant.)	RErr in %	
		p=0.1	p=1	p=0.1	p=0.5		p=0.1	p=0.5
PLClipping	4.96	5.39	7.04	10.80	22.80	5.16	7.38	21.58
PLClipping$_{0.1}$	5.62	5.91	6.65	12.80	26.50	5.84	8.72	27.36
PLRandBET, p_w=0.1	**4.49**	**4.98**	6.65	11.00	22.80	**4.71**	7.25	24.94
PLRandBET, p_w=1	4.62	5.02	**6.36**	11.30	22.40	4.83	**6.92**	19.83
PLRandBET, p_w=1, p_i=0.1	5.50	5.99	7.49	**7.70**	**9.10**	5.71	8.37	25.83
PLRandBET, p_w=1, p_i=0.1, p_a=0.1	9.16	9.60	11.09	11.50	13.80	9.31	10.54	13.51
PLRandBET, p_a=0.5	5.43	5.91	7.96	10.90	21.90	5.68	6.74	**10.16**
PLRandBET, p_w=1, p_a=0.1	7.66	8.27	10.47	13.80	24.70	7.89	9.09	12.17

Table 6.9: **Bit Errors in Inputs and Activations:** Average RErr for PLClipping and PLRandBET against bit errors in weights, inputs and activations. We use PLRandBET to inject bit errors in weights (rate p_w), inputs (rate p_i, orange) and/or activations (rate p_a, violet) during training. Bit errors in inputs and activations are difficult to tolerate. Extreme Clipping (e.g., $w_{\max}$) might worsen robustness against bit errors in inputs/activations. PLRandBET against bit errors in weights, inputs and activations is significantly harder, resulting in higher Err, while improving robustness considerably.

i.e., average RErr, of various models on CIFAR10 against bit errors in weights, inputs (in orange) or activations (in violet). For activation quantization, we additionally report the (clean) Err *after* activation quantization (without bit errors). While being simplistic, our activation quantization has negligible impact on Err. We found bit errors in inputs and activations to be challenging in terms of robustness. Even for small bit error rates, e.g., $p = 0.1\%$, RErr increases significantly, to at least 7.7% and 6.92% RErr for inputs and activations, respectively. While PLRandBET (training on random bit errors *in weights*) helps against bit errors in activations, it has no impact on robustness against bit errors in inputs. Extreme PLClipping, in contrast, e.g., using $w_{\max} = 0.1$ tends to reduce robustness in both cases. These results show that low-voltage operation is complicated when taking inputs and activations into account. While separate SRAM arrays for weights, inputs and activations can be used, allowing varying levels of bit errors, this is potentially undesirable from a design perspective.

RandBET for Inputs and Activations: In order to obtain robustness against random bit errors in inputs and/or activations, we adapt RandBET to allow bit error injection in inputs and/or activations (in addition to weights) during training. Table 6.9 shows that injecting either input bit errors (bit error rate p_i in orange) or activation bit errors (bit error rate p_a in violet) helps robustness, but also makes training significantly more difficult. Indeed, injecting bit errors in weights, inputs *and* activations increases (clean) Err significantly, to 9.16% from 4.62% (for RandBET with bit errors in weights only). We found that this difficulty mainly stems from injecting bit errors in activations during training: While RandBET (activations only) with p_a=0.5% affects (clean) Err only slightly (5.43%), bit errors in weights *and* activations (i.e., p_w=1% and p_a=0.1%) results in an increase to 7.66%. This increase in Err also translates to an increase in RErr against bit errors in weights or activations. As a result, injecting bit errors only in weights and inputs (e.g., p_w=1% and $p_i = 0.1\%$) might be beneficial as it avoids a significant increase in (clean) Err, while still providing some robustness against bit errors in activations. Overall, we made a significant step towards deep neural networks "fully" robust against low-voltage induced random bit errors, but the problem remains difficult.

Model / Dataset	RErr in % for ϵ Bit Errors			
MNIST	ϵ=80	ϵ=160	ϵ=240	ϵ=320
RQUANT	77.88	90.11	89.94	90.04
CLIPPING$_{0.05}$	10.11	11.22	89.18	57.28
ADVBET$_{0.05}$	**0.42**	**10.08**	**10.13**	**20.09**
CIFAR10	ϵ=160	ϵ=320	ϵ=640	ϵ=960
RQUANT BN	89.30	89.26	89.56	89.32
RQUANT *(GN)*	34.42	89.01	90.10	90.02
CLIPPING$_{0.05}$ BN	89.48	89.16	89.19	89.40
CLIPPING$_{0.05}$ *(GN)*	14.84	24.96	49.56	51.88
ADVBET$_{0.05}$ *(GN)*	**13.72**	**15.53**	**25.91**	**44.47**

Model / Dataset	
MNIST	ϵ_{BFA}
RQUANT	98 $\pm$ 7.5
CLIPPING$_{0.05}$	437 $\pm$ 49.7
ADVBET$_{0.05}$	1403 $\pm$ 263.3
CIFAR10	ϵ_{BFA}
RQUANT BN	54 $\pm$ 8.5
RQUANT *GN*	385 $\pm$ 33.9
CLIPPING$_{0.05}$ BN	213 $\pm$ 20.9
CLIPPING$_{0.05}$ *GN*	1725 $\pm$ 60.8
ADVBET$_{0.05}$	2187 $\pm$ 11.2

Table 6.10: **Left: Bit Flip Attack (BFA) [RHF19a].** Worst RErr for RQUANT, CLIPPING and ADVBET against BFA for various allowed budgets ϵ of bit errors. On CIFAR10, we also present results when using batch normalization (BN). Surprisingly, CLIPPING is quite successful in "defending" BFA as long as BN is avoided, even for very large ϵ. **Right: Required Bit Flips for BFA.** We report the average number (and standard deviation) of performed bit flips ϵ_{BFA} to increase RErr to 90% or above. Note that this setting is different from the results left, where ϵ_{BFA} is limited explicitly. ADVBET was trained using $\epsilon = 160$ during training.

6.4.8 Robustness Against Adversarial Bit Errors

In this section, we switch focus and consider *adversarial* bit error robustness. To this end, we consider both the BFA attack from related work [RHF19a] and our own adversarial bit error attack from Section 6.2.2. As "defense", we consider CLIPPING, RANDBET and our adversarial bit error training (ADVBET) which are able to improve robustness considerably – both against BFA and our adversarial bit level attack.

Limitations of BFA: We start by considering BFA, showing that it is not as effective (and efficient) against our models, compared to the results in [RHF19a]. Table 6.10 (left) reports worst RErr on MNIST and CIFAR10 for various models. On MNIST, BFA is effective in attacking our RQUANT model, even for only ϵ=80 bit errors, increasing RErr to 77.88%. However, CLIPPING seems to provide good inherent robustness, at least for low ϵ up to 160. The same holds on CIFAR10. Here, we also show results considering batch normalization (BN, marked in red), as used in [RHF19a]. Similar to RQUANT, when training CLIPPING with BN, the deep neural network is significantly less robust. In fact, BFA is suddenly able to increase RErr beyond 90%, while RErr for CLIPPING with GN is still at 15.53% for ϵ=320. However, we found that BFA does *not* attack the batch normalization parameters (i.e., scale and bias). Instead, as shown in Table 6.3 against random bit errors, we found that BN is generally less robust.

Table 6.10 (right) also reports the average number of bit flips required by BFA in order to increase RErr to 90% or above (i.e., reduce performance to random guessing, as also used in [RHF19a]). Furthermore, we report the standard deviation across these 5 restarts. As shown, BFA requires significantly more bit flips ϵ_{BFA} to break our models than reported in [RHF19a]. We believe that this is mainly due to [RHF19a] relying on BN as discussed above. Furthermore, our CLIPPING, RANDBET or ADVBET models, specifically trained to be robust against random or adversarial bit errors, improve robustness significantly. On MNIST, more than 1k and on CIFAR10 more than 2k bit flips are required.

Finally, Figure 6.10 shows RErr and loss over BFA iterations. While loss increases continuously, RErr tends to increase roughly in steps of 10%. This is because BFA consecutively flips

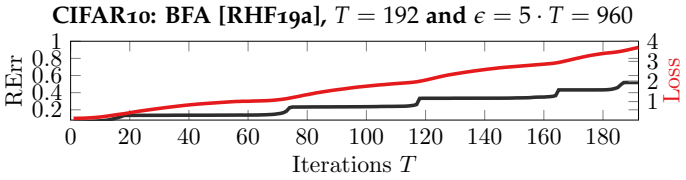


Figure 6.10: **BFA Iterations:** RErr and loss (after bit errors, in red) plotted against BFA iterations. RErr tends to increase (roughly) in steps of $\sim 10\%$ by flipping predictions consecutively to a constant one for each class.

	Err %	Worst RErr in %			
		U (all)	T (all)	log	conv
MNIST		ϵ=160			
RQuant	0.37	0.48	0.51	91.08	85.42
Clipping$_{0.05}$	0.38	2.30	0.74	10.77	85.09
AdvBET$_{0.05}$, ϵ=240	0.29	11.58	0.34	0.36	0.46
CIFAR10		ϵ=320			
RQuant	4.89	8.54	91.18	91.18	89.06
Clipping$_{0.05}$	5.34	24.04	35.20	35.86	60.76
AdvBET$_{0.05}$, ϵ=160	5.54	10.01	20.20	26.22	12.33

Table 6.11: **Adversarial Bit Error Ablation.** Worst RErr, considering different ϵ and settings on MNIST and CIFAR10: attacking only the **log**it layer (i.e., last layer), or only first **conv**olutional layer, using untargeted ("**U**") or targeted ("**T**") attacks. Targeted attacks are usually easier to optimize and more effective. The first convolutional or logit layer are particularly vulnerable.

the labels for each class to a constant class, eventually arriving at 90% RErr which is equivalent to a random or constant classifier. However, BFA needs between 1 and 2 seconds per iteration and the number of bit flips is (indirectly) tied to the number of iterations, which is also the reason why loss increases monotonically in Figure 6.10. This makes BFA unfit to be used for AdvBET. We address some of these limitations using our adversarial bit error attack.

More Effective Adversarial Bit Errors: Using appropriate hyperparameters and considering *both* untargeted and targeted attacks, our adversarial bit error attack is more effective and efficient compared to BFA. Table 6.11 shows (worst) RErr on MNIST and CIFAR10, showing that our adversarial bit errors achieve higher RErr compared to BFA, e.g., for ϵ=320. As also reported in [RHL+20], we found that targeted attacks are generally more effective. This is because the "easiest" way to increase RErr is to force the neural network to predict a constant label, which the targeted attacks explicitly do. Similarly, targeting *only* the logit layer is usually sufficient for high RErr. Interestingly, attacking only the first convolutional layer is quite effective, as well. We also emphasize that we are considering more adversarial bit errors (i.e., larger ϵ) on CIFAR10, even though CIFAR10 is considerably more difficult. This is due to the increased number of weights (roughly 5.5Mio) on CIFAR10. Figure 6.11 also shows that gradient normalization and momentum are essential for the untargeted attack to be successful. This is important as running the targeted attack for each target label during AdvBET is prohibitively expensive. Nevertheless, the attack remains sensitive to, e.g., the learning rate, but γ=1 works reasonably well across models, given enough random restarts to avoid poor optima. Thus, in our evaluation, we run both targeted and untargeted attacks, attacking all weights, only the first convolutional and/or logit layer and consider the worst-case across a total of 80 random restarts. Overall, our attack provides a much more realistic estimate of adversarial bit error robustness. Furthermore, on CIFAR10, our attack requires only between 0.15 and 0.2 seconds per iteration and runtime is independent of ϵ.

Clipping and RandBET Improve Adversarial Bit Error Robustness: As shown for BFA in Table 6.10, we find that Clipping and RandBET are surprisingly robust against adversarial bit errors. Similarly, Table 6.12 reports RErr on MNIST and CIFAR10 against our adversarial bit

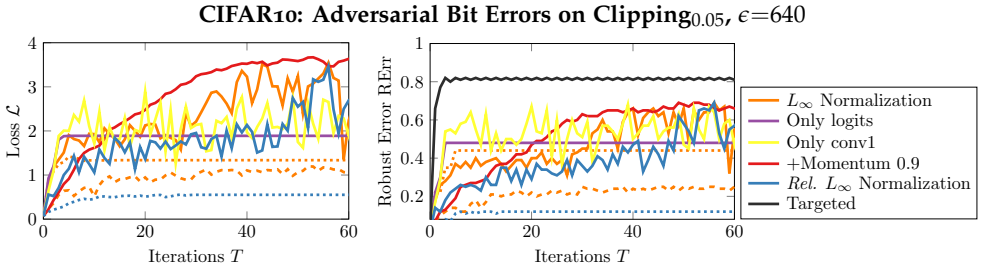


Figure 6.11: **Adversarial Bit Error Iterations:** We plot loss, c.f. Equation (6.1), and robust error RErr against iterations, both *measured on the* 100 *held-out test examples used to find, i.e., train, adversarial bit errors*. Clearly, L_∞ gradient normalization *with* momentum (in red), targeting first convolutional (yellow) or logit layer (violet), is most effective for untargeted attacks. Unfortunately, the attack gets easily stuck in bad optima if learning rate is not optimal. Targeted attacks simplify optimization and are often more effective (black, right).

	Err %	RErr in %		
MNIST		ϵ=160	ϵ=240	ϵ=320
RQuant	0.37	91.08	91.08	91.08
Clipping$_{0.05}$	0.38	88.81	90.11	90.26
RandBET$_{0.05}$, p=20	0.39	69.90	81.16	81.94
AdvBET$_{0.05}$, ϵ=160	**0.31**	21.43	40.30	80.10
AdvBET$_{0.05}$, ϵ=240	**0.31**	28.34	41.66	71.16
CIFAR10		ϵ=320	ϵ=480	ϵ=640
RQuant	**4.89**	91.18	91.18	91.18
Clipping$_{0.05}$	5.34	60.76	79.12	83.93
RandBET$_{0.05}$, p=2	5.42	33.86	**54.24**	80.36
AdvBET$_{0.05}$, ϵ=160	5.54	**26.22**	55.06	**77.43**

Table 6.12: **Adversarial Bit Error Robustness and AdvBET:** Worst RErr against adversarial bit errors for Clipping, RandBET and AdvBET considering multiple restarts, including targeted and untargeted attacks. While AdvBET improves robustness considerably on MNIST, Clipping and RandBET are very strong baselines on CIFAR10. As we consider larger ϵ, this makes it hard for AdvBET to further improve results.

error attack. While Clipping does not perform well on MNIST, RandBET reduces RErr against ϵ=80 from 85.09% to 10.13%. On CIFAR10, in contrast, considering larger ϵ, Clipping alone is quite effective, with 20.48% RErr against ϵ=160. Nevertheless, RandBET further improves over Clipping. This is counter-intuitive considering, e.g., robustness against adversarial examples where training against *random* perturbations does generally not provide *adversarial robustness*. However, RandBET is trained against large bit error rates, e.g., p=2% on CIFAR10, with an expected ϵ=8·W≈880k bit errors, 110k in the most significant bits (MSBs). For adversarial bit error, in contrast, we consider up to ϵ=640 on CIFAR10. In terms of BFA, complementing the results in Table 6.10, we need on average 2253 bit errors to increase RErr above 90% for RandBET on CIFAR10. In contrast, [HRL+20] report 541 required bit flips (ResNet-20, W≈4.3Mio) to "break" their proposed binarized deep neural network, which has been reduced to 35 in [RHL+20] using targeted BFA. Overall, these results show that random and adversarial bit error robustness are aligned well, allowing to *secure* low-voltage operation of accelerators.

AdvBET Improves Adversarial Bit Error Robustness: Using AdvBET, we can further boost robustness against adversarial bit errors, c.f. Table 6.12. On MNIST, in particular, AdvBET is able to reduce RErr from above 80% for RandBET or Clipping, to 40.30% against up to ϵ=240 adversarial bit errors. As Table 6.11 illustrates, targeted attacks are generally considered stronger. Thus, training with a targeted attack, selecting a random target label in each iteration, further boosts robustness to 31.23% RErr. However, these improvements do not easily generalize to CIFAR10. We suspect this is due to two reasons: First, we found that training with too large ϵ is difficult (also c.f. increased Err in Table 6.12), i.e., AdvBET with larger ϵ does not improve robustness because training becomes too hard. This is why we

report results for AdvBET trained on ϵ=160. Second, Clipping alone is significantly more robust on CIFAR10 than on MNIST, resulting in a particularly strong baseline. We suspect that architectural differences have a significant impact on how effective Clipping is against adversarial bit errors. For example, models on CIFAR10 have inherently more weights in the first convolutional layer (relative to W, due to larger input dimensionality,) which Table 6.11 shows to be particularly vulnerable. Overall, AdvBET can be used to further boost robustness against *adversarial* bit errors, beyond Clipping.

6.5 Conclusion

We proposed a combination of **robust quantization**, **weight clipping** and **random bit error training (RandBET)** or **adversarial bit error training (AdvBET)** to train deep neural networks robust against random and adversarial bit errors in their (quantized) weights. This enables secure low-voltage operation of accelerators. Specifically, we considered robustness against random bit errors induced by operating the accelerator memory far below its rated voltage [CSC+19]. We showed that quantization details have tremendous impact on robustness, even though we use a very simple fixed-point quantization scheme without any outlier treatment [ZST+18, SSH15, PKY18]. By encouraging redundancy in the weights, clipping is another simple but effective strategy to improve robustness. In contrast to related work, RandBET does *not* require expert knowledge or profiling infrastructure [KHM+18a, KOY+19] and generalizes across chips, with different bit error patterns, and voltages. As a result, we also avoid expensive circuit techniques [RWA+16, CSC+19]. Furthermore, in contrast to existing research, we discussed low-voltage induced random bit errors in inputs and activations. Finally, we introduced a novel *adversarial* bit error attack that is more effective and efficient compared to existing attacks [RHF19a] and can be utilized for AdvBET. Surprisingly, we found that Clipping and RandBET also improve robustness against adversarial bit errors. However, AdvBET further improves robustness specifically against adversarial bit errors. Altogether, by improving deep neural network robustness against random and adversarial bit errors, we enable both energy-efficient and secure accelerators.

IV

Improving Adversarial Robustness and Uncertainty Estimation

While the previous parts focused on understanding and improving robustness with respect to both input and weight perturbations, this part shifts emphasis to uncertainty estimation. In the context of adversarial robustness, reliable uncertainty quantification helps to generalize robustness to different types of adversarial examples as well as corrupted or out-of-distribution examples. Besides robustness, appropriately estimating uncertainty also allows us to provide statistical guarantees on model performance.

Specifically, in Chapter 7, we frame adversarial training in the larger context of uncertainty estimation. In *confidence-calibrated adversarial training*, instead of requiring high-confidence and correct classification of adversarial examples, we want the model to reduce its confidence, i.e., signal its uncertainty. By rejecting examples with low confidence at test time, we can generalize adversarial robustness to previously unseen types of adversarial examples as well as corrupted or out-of-distribution examples. Besides, this also improves the robustness-accuracy trade-off discussed in Part II.

Then, in Chapter 8, taking a step back from adversarial robustness, we quantify uncertainty not by a (scalar) confidence but by predicting confidence *sets*. Specifically, *conformal prediction* provides a statistical guarantee of the true label being included in the confidence set with user-specified probability. However, conformal prediction is generally applied post-training and not integrated well into today's deep learning methods. Thus, with *conformal training*, we present a novel procedure for training deep neural networks and conformal wrappers end-to-end. This not only reduces uncertainty, but also allows shaping the obtained confidence sets while preserving the original guarantee.

7 Confidence-Calibrated Adversarial Training: Generalizing to Unseen Attacks

Contents

This chapter poses adversarial robustness in the context of uncertainty estimation. Based on Chapter 4, we expect adversarial examples to leave the data manifold and thus encourage the model to reflect this by reducing the predicted confidence. This way, we aim to address a major limitation of adversarial training: While adversarial training is known to yield robust models against a specific threat model, e.g., L_∞ adversarial examples, robustness does typically *not* generalize to previously unseen threat models such as other L_p norms, or larger perturbations. Our **confidence-calibrated adversarial training (CCAT)** tackles this problem by biasing the model towards low confidence predictions on adversarial examples. By allowing to reject examples with low confidence, robustness generalizes beyond the threat model employed during training. CCAT, trained *only* on L_∞ adversarial examples, increases robustness against larger L_∞, L_2, L_1 and L_0 attacks, adversarial frames, distal adversarial examples and corrupted examples and yields better clean accuracy compared to adversarial training. For thorough evaluation we developed novel white- and black-box attacks directly attacking CCAT by maximizing confidence. For each threat model, we use 7 attacks with up to 50 restarts and 5000 iterations and report worst-case robust test error, extended to our confidence-thresholded setting, across *all* attacks.

This chapter is based on [SHS20]: As first author, David Stutz conducted all experiments and was the main writer of the paper. Proposition 2 in Section 7.2.2 and its proof, however, were contributed by Matthias Hein. This work was presented at the Workshop on Uncertainty and Robustness in Deep Learning (UDL) of ICML 2020 and as part of the Qualcomm Innovation Fellowship Europe 2019. This work was also presented during invited talks at the Bosch Center for Artificial Intelligence and the Qian Xuesen Laboratory of Space Technology.

Code: The source for this chapter can be found on GitHub[1].

[1] https://github.com/davidstutz/icml2020-confidence-calibrated-adversarial-training

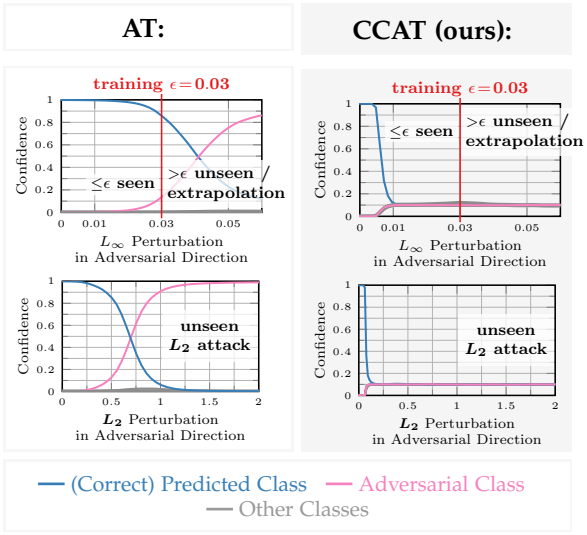


Figure 7.1: **Adversarial Training (AT) versus our CCAT:** We plot the confidence in the direction of an adversarial example. AT enforces high confidence predictions for the correct class on the L_∞-ball of radius ϵ (*"seen"* attack during training, top left). As AT enforces no particular bias beyond the ϵ-ball, adversarial examples can be found right beyond this ball. In contrast CCAT enforces a decaying confidence in the correct class up to uniform confidence within the ϵ-ball (top right). Thus, CCAT biases the model to extrapolate uniform confidence beyond the ϵ-ball. This behavior also extends to *"unseen"* attacks during training, e.g., L_2 attacks (bottom), such that adversarial examples can be rejected via confidence-thresholding.

7.1 Introduction

Deep neural networks were shown to be susceptible to adversarial examples [SZS+14]: adversarially perturbed examples that cause misclassification while being nearly "imperceptible", i.e., close to the original example. Here, "closeness" is commonly enforced by constraining the L_p norm of the perturbation, referred to as threat model. Since then, numerous defenses against adversarial examples have been proposed. However, many were unable to keep up with more advanced attacks [ACW18, AC18]. Moreover, most defenses are tailored to only one specific threat model.

Adversarial training [GSS15, MMS+18], i.e., training on adversarial examples, can be regarded as state-of-the-art. However, following Figure 7.1, adversarial training is known to "overfit" to the threat model *"seen"* during training, e.g., L_∞ adversarial examples. Thus, robustness does not extrapolate to larger L_∞ perturbations, c.f. Figure 7.1 (top left), or generalize to *"unseen"* attacks, c.f. Figure 7.1 (bottom left), e.g., other L_p threat models [SC18, TB19, LCWC19, KSH+19, MWK20]. We hypothesize this to be a result of enforcing high-confidence predictions on adversarial examples. However, high-confidence predictions are difficult to extrapolate beyond the adversarial examples seen during training. Moreover, it is not meaningful to extrapolate high-confidence predictions to arbitrary regions. Finally, adversarial training often hurts accuracy, resulting in a robustness-accuracy trade-off [TSE+19, SHS19, RXY+19, ZYJ+19].

Contributions: We propose **confidence-calibrated adversarial training (CCAT)** which trains the network to predict a convex combination of uniform and (correct) one-hot distribution on adversarial examples that becomes more uniform as the distance to the attacked example increases. This is illustrated in Figure 7.1. Thus, CCAT implicitly biases the network to predict a uniform distribution beyond the threat model *seen* during training, c.f. Figure 7.1 (top right). Robustness is obtained by rejecting low-confidence (adversarial) examples through confidence-thresholding. As a result, having seen *only* L_∞ adversarial examples during training, CCAT improves robustness against previously *unseen* attacks, c.f. Figure 7.1 (bottom right), e.g., L_2, L_1 and L_0 adversarial examples or larger L_∞ perturbations. Furthermore, robustness extends to

adversarial frames [ZZRP19], distal adversarial examples [HAB19], corrupted examples (e.g., noise, blur, transforms etc.) and accuracy of normal training is preserved better than with adversarial training.

For thorough evaluation, following best practices [CAB+19], we adapt several state-of-the-art white- and black-box attacks [NK17, IEAL18, KH18, ACFH20] to CCAT by explicitly maximizing confidence and improving optimization through a backtracking scheme. In total, we consider 7 different attacks for each threat model (i.e., L_p for $p \in \{\infty, 2, 1, 0\}$), allowing up to 50 random restarts and 5000 iterations each. We report worst-case robust test error, extended to our confidence-thresholded setting, across *all* attacks and restarts, on a *per test example* basis. We demonstrate improved robustness against unseen attacks compared to standard adversarial training [MMS+18], TRADES [ZYJ+19], adversarial training using multiple threat models [MWK20] and two detection methods [MLW+18, LLLS18b], while training *only* on L_∞ adversarial examples.

7.2 Confidence Calibration of Adversarial Examples

To start, we briefly review adversarial training on L_∞ adversarial examples [MMS+18], which has become standard to train robust models (Section 7.2.1). However, robustness does not generalize to larger perturbations or unseen attacks. We hypothesize this to be the result of enforcing high-confidence predictions on adversarial examples. CCAT addresses this issue with minimal modifications (Section 7.2.2 and Algorithm 3) by encouraging low-confidence predictions on adversarial examples. During testing, adversarial examples can be rejected by confidence thresholding.

Notation: We consider a classifier $f : \mathbb{R}^d \to \mathbb{R}^K$ with K classes where f_k denotes the confidence for class k. While we use the cross-entropy loss $\mathcal{L}$ for training, our approach also generalizes to other losses. Given $x \in \mathbb{R}^d$ with class $y \in \{1, \dots, K\}$, we let $f(x) := \arg\max_k f_k(x)$ denote the predicted class for notational convenience. For $f(x) = y$, an adversarial example $\tilde{x} = x + \delta$ is defined as a "small" perturbation δ such that $f(\tilde{x}) \neq y$, i.e., the classifier changes its decision. The strength of the change δ is measured by some L_p-norm, $p \in \{0, 1, 2, \infty\}$. Here, $p = \infty$ is a popular choice as it leads to the smallest perturbation per pixel.

7.2.1 Problems of Adversarial Training

Following [MMS+18], adversarial training is given as the following min-max problem:

$$\min_w \mathbb{E}\left[\max_{\|\delta\|_\infty \leq \epsilon} \mathcal{L}(f(x+\delta; w), y)\right] \tag{7.1}$$

with w being the classifier's parameters. During mini-batch training the inner maximization problem,

$$\max_{\|\delta\|_\infty \leq \epsilon} \mathcal{L}(f(x+\delta; w), y), \tag{7.2}$$

is approximately solved. In addition to the L_∞-constraint, a box constraint is enforced for images, i.e., $\tilde{x}_i = (x+\delta)_i \in [0, 1]$. Note that maximizing the cross-entropy loss is equivalent to finding the adversarial example with *minimal* confidence in the true class. For neural networks, this is generally a non-convex optimization problem. In [MMS+18] the problem is tackled using projected gradient descent (PGD), initialized using a random δ with $\|\delta\|_\infty \leq \epsilon$.

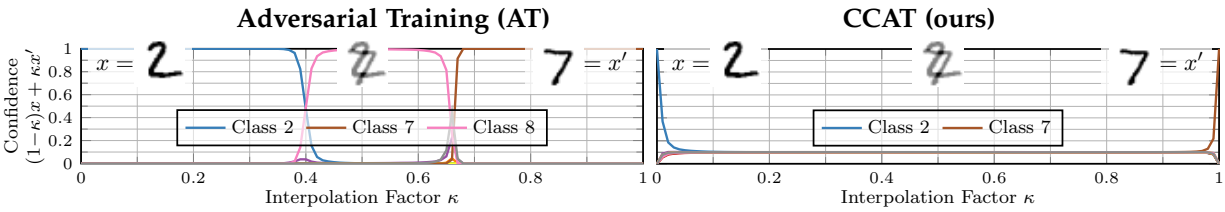


Figure 7.2: **Extrapolation of Uniform Predictions:** We plot the confidence in each class along an interpolation between two test examples x and x', "2" and "7", on MNIST [LBBH98]: $(1-\kappa)x+\kappa x'$ where κ is the interpolation factor. CCAT quickly yields low-confidence, uniform predictions in between both examples, extrapolating the behavior enforced within the ϵ-ball during training. Regular adversarial training, in contrast, consistently produces high-confidence predictions, even on unreasonable inputs.

In contrast to adversarial training as proposed in [MMS+18], which computes adversarial examples for the *full* batch in each iteration, others compute adversarial examples only for *half* the examples of each batch [SZS+14]. Instead of training *only* on adversarial examples, each batch is divided into 50% clean and 50% adversarial examples. Compared to Equation (7.1), 50%/50% adversarial training effectively minimizes

$$\underbrace{\mathbb{E}\Big[\max_{\|\delta\|_\infty \leq \epsilon} \mathcal{L}(f(x+\delta;w),y)\Big]}_{\text{50\% adversarial training}} + \underbrace{\mathbb{E}\big[\mathcal{L}(f(x;w),y)\big]}_{\text{50\% "clean" training}}. \tag{7.3}$$

This improves test accuracy on clean examples compared to 100% adversarial training but typically leads to worse robustness. Intuitively, by balancing both terms in Equation (7.3), the trade-off between accuracy and robustness can already be optimized to some extent [SHS19].

Problems: Trained on L_∞ adversarial examples, the robustness of adversarial training does not generalize to previously unseen adversarial examples, including larger perturbations or other L_p adversarial examples. We hypothesize that this is because adversarial training explicitly enforces high-confidence predictions on L_∞ adversarial examples within the ϵ-ball seen during training ("seen" in Figure 7.1). However, this behavior is difficult to extrapolate to arbitrary regions in a meaningful way. Thus, it is not surprising that adversarial examples can often be found right beyond the ϵ-ball used during training, c.f. Figure 7.1 (top left). This can be described as "overfitting" to the L_∞ adversarial examples used during training. Also, larger ϵ-balls around training examples might include (clean) examples from other classes. Then, Equation (7.2) will focus on these regions and reduce accuracy as considered in our theoretical toy example, see Proposition 2, and related work [JBZB19, JBC+19].

As suggested in Figure 7.1, both problems can be addressed by enforcing low-confidence predictions on adversarial examples in the ϵ-ball. In practice, we found that the low-confidence predictions on adversarial examples within the ϵ-ball are extrapolated beyond the ϵ-ball, i.e., to larger perturbations, unseen attacks or distal adversarial examples. This allows rejecting adversarial examples based on their low confidence. We further enforce this behavior by explicitly encouraging a "steep" transition from high-confidence predictions (on clean examples) to low-confidence predictions (on adversarial examples). As a result, the (low-confidence) prediction is almost flat close to the boundary of the ϵ-ball. Additionally, there is no incentive to deviate from the uniform distribution outside the ϵ-ball. For example, as illustrated in Figure 7.2, the confidence stays low in between examples from different classes and only increases if necessary, i.e., close to the examples.

Algorithm 3 Confidence-Calibrated Adversarial Training (CCAT): The only changes compared to standard adversarial training are the attack and the probability distribution over the classes, which becomes more uniform as distance $\|\delta\|_\infty$ increases. During testing, low-confidence (adversarial) examples are rejected.

while true **do**
  choose random batch $(x_1, y_1), \ldots, (x_B, y_B)$.
  for $b = 1, \ldots, B/2$ **do**
    $\delta_b := \arg\max_{\|\delta\|_\infty \leq \epsilon} \max_{k \neq y_b} f_k(x_b + \delta)$ (Equation (7.4))
    $\tilde{x}_b := x_b + \delta_b$
    $\lambda(\delta_b) := (1 - \min(1, \|\delta_b\|_\infty / \epsilon))^\rho$ (Equation (7.6))
    $\tilde{y}_b := \lambda(\delta_b)\, \text{one\_hot}(y_b) + (1 - \lambda(\delta_b))\frac{1}{K}$ (Equation (7.5))
  end for
  update parameters using Equation (7.3): $\sum_{b=1}^{B/2} \mathcal{L}(f(\tilde{x}_b), \tilde{y}_b) + \sum_{b=B/2}^{B} \mathcal{L}(f(x_b), y_b)$
end while

7.2.2 Confidence-Calibrated Adversarial Training

Confidence-calibrated adversarial training (CCAT) addresses these problems with minimal modifications, as outlined in Algorithm 3. During training, we train the network to predict a convex combination of (correct) one-hot distribution on clean examples and uniform distribution on adversarial examples as target distribution within the cross-entropy loss. During testing, adversarial examples can be rejected by confidence thresholding: adversarial examples receive near-uniform confidence while test examples receive high-confidence. By extrapolating the uniform distribution beyond the ϵ-ball used during training, previously unseen adversarial examples such as larger L_∞ perturbations can be rejected, as well. In the following, we first introduce an alternative objective for generating adversarial examples. Then, we specifically define the target distribution, which becomes more uniform with larger perturbations $\|\delta\|_\infty$.

Given an example x with label y, our adaptive attack during training maximizes the confidence in any other label $k \neq y$. This results in effective attacks against CCAT, as CCAT will reject low-confidence adversarial examples:

$$\max_{\|\delta\|_\infty \leq \epsilon} \max_{k \neq y} f_k(x + \delta; w) \tag{7.4}$$

Note that Equation (7.2), in contrast, minimizes the confidence in the true label y. Similarly, [GQB19] uses targeted attacks in order to maximize confidence, whereas ours is untargeted and thus our objective is the maximal confidence over all other classes.

Then, given an adversarial example from Equation (7.4) during training, CCAT uses the following combination of uniform and one-hot distribution as target for the cross-entropy loss:

$$\tilde{y} = \lambda(\delta)\ \text{one\_hot}(y) + \big(1 - \lambda(\delta)\big)\frac{1}{K} \tag{7.5}$$

with $\lambda(\delta) \in [0, 1]$ and $\text{one\_hot}(y) \in \{0, 1\}^K$ denoting the one-hot vector corresponding to class y. Thus, we enforce a convex combination of the original label distribution and the uniform distribution which is controlled by the parameter $\lambda = \lambda(\delta)$, computed given the perturbation δ. We choose λ to decrease with the distance $\|\delta\|_\infty$ of the adversarial example to the attacked example x with the intention to enforce uniform predictions when $\|\delta\|_\infty = \epsilon$. Then, the network is encouraged to extrapolate this uniform distribution beyond the used ϵ-ball. Even if extrapolation does not work perfectly, the uniform distribution is much more

meaningful for extrapolation to arbitrary regions as well as regions between classes compared to high-confidence predictions as encouraged in standard adversarial training and demonstrated in Figure 7.2. For controlling the trade-off λ between one-hot and uniform distribution, we consider the following "power transition":

$$\lambda(\delta) := \left(1 - \min\left(1, \frac{\|\delta\|_\infty}{\epsilon}\right)\right)^\rho \tag{7.6}$$

This ensures that for $\delta = 0$ we impose the original one-hot label. For growing δ, however, the influence of the original label decays proportional to $\|\delta\|_\infty$. The speed of decay is controlled by the parameter ρ. For $\rho = 10$, Figure 7.1 (top right) shows how the transition is approximated by the network. The power transition ensures that for $\|\delta\|_\infty \geq \epsilon$, i.e., perturbations larger than encountered during training, a uniform distribution is enforced as λ is 0. We train on 50% clean and 50% adversarial examples in each batch, as in Equation (7.3), such that the network has an incentive to predict correct labels.

The convex combination of uniform and one-hot distribution in Equation (7.5) resembles the label smoothing regularizer introduced in [SVI+16]. In concurrent work, label smoothing has also been used as regularizer for adversarial training [CLC+20]. However, in our case, $\lambda = \lambda(\delta)$ from Equation (7.6) is not a fixed hyperparameter as in [SVI+16, CLC+20]. Instead, λ depends on the perturbation δ and reaches zero for $\|\delta\|_\infty = \epsilon$ to encourage low-confidence predictions beyond the ϵ-ball used during training. Thereby, λ explicitly models the transition from one-hot to uniform distribution.

Confidence-Calibrated Adversarial Training Yields Accurate Models: Proposition 2 analyzes 100% adversarial training and its 50%/50% variant as well as our confidence-calibrated variant, CCAT, to show that there exist problems where both 100% and 50%/50% adversarial training are unable to reconcile robustness and accuracy, as recently discussed [TSE+19, SHS19, RXY+19, ZYJ+19]. However, our CCAT is able to obtain *both* robustness and accuracy given that λ in Equation (7.6) is chosen appropriately:

Proposition 2. We consider a classification problem with two points $x = 0$ and $x = \epsilon$ in $\mathbb{R}$ with deterministic labels, i.e., $p(y = 2|x = 0) = 1$ and $p(y = 1|x = \epsilon) = 1$, such that the problem is fully determined by the probability $p_0 = p(x = 0)$. The Bayes error of this classification problem is zero. Let the predicted probability distribution over classes be $\tilde{p}(y|x) = e^{g_y(x)}/e^{g_1(x)} + e^{g_2(x)}$, where $g : \mathbb{R}^d \to \mathbb{R}^2$ is the classifier, and we assume that the function $\lambda : \mathbb{R}_+ \to [0, 1]$ used in CCAT is monotonically decreasing and $\lambda(0) = 1$. Then, the error of the Bayes optimal classifier (with cross-entropy loss) for

- adversarial training on 100% adversarial examples is $\min\{p_0, 1 - p_0\}$.
- adversarial training on 50%/50% adversarial/clean examples per batch is $\min\{p_0, 1 - p_0\}$.
- CCAT on 50% clean and 50% adversarial examples is *zero* if $\lambda(\epsilon) < \min\left\{p_0/1-p_0, 1-p_0/p_0\right\}$.

Proof. First, we stress that we are dealing with three different probability distributions over the labels: the true one $p(y|x)$, the imposed one during training $\hat{p}(y|x)$ and the predicted one $\tilde{p}(y|x)$. We also note that $\hat{p}$ depends on λ as follows:

$$\hat{p}(k) = \lambda p_y(k) + (1 - \lambda)u(k)$$

where $p_y(k)$ is the original one-hot distribution, i.e., $p_y(k) = 1$ iff $k = y$ and $p_y(k) = 0$ otherwise with y being the true label, and $u(k) = 1/K$ is the uniform distribution. We note that this is merely an alternative formulation to the target distribution of Equation (7.5). Also note that λ itself is a function of the norm $\|\delta\|$; here, this dependence is made explicit by writing $\hat{p}(\lambda)(y|x)$. This makes the expressions for the

expected loss of CCAT slightly more complicated. Thus, we first derive the Bayes optimal classifier and its loss for CCAT. To this end, we introduce

$$a = g_1(0) - g_2(0), \quad b = g_1(\epsilon) - g_2(\epsilon) \tag{7.7}$$

and express the logarithm of the predicted probabilities (i.e., the confidences) of classes 1 and 2 in terms of a and b:

$$-\log \tilde{p}(y=2|x=x) = -\log\Big(\frac{e^{g_2(x)}}{e^{g_1(x)}+e^{g_2(x)}}\Big) = \log\big(1+e^{g_1(x)-g_2(x)}\big) = \begin{cases}\log\left(1+e^{a}\right) & \text{if } x=0\\ \log(1+e^{b}) & \text{if } x=\epsilon\end{cases},$$
$$-\log \tilde{p}(y=1|x=x) = -\log\Big(\frac{e^{g_1(x)}}{e^{g_1(x)}+e^{g_2(x)}}\Big) = \log\big(1+e^{g_2(x)-g_1(x)}\big) = \begin{cases}\log(1+e^{-a}) & \text{if } x=0\\ \log\left(1+e^{-b}\right) & \text{if } x=\epsilon\end{cases}.$$

Next, We consider the approach of [MMS+18] with 100% adversarial training. The expected loss can be written as

$$\begin{aligned}
&\mathbb{E}\Big[\max_{\|\delta\|_\infty\leq\epsilon}\mathcal{L}(g(x+\delta),y)\Big] = \mathbb{E}\Big[\mathbb{E}\Big[\max_{\|\delta\|_\infty\leq\epsilon}\mathcal{L}(g(x+\delta),y)|x\Big]\Big]\\
=&p(x=0)p(y=2|x=0)\max\left\{-\log\left(\tilde{p}(y=2|x=0)\right),-\log\left(\tilde{p}(y=2|x=\epsilon)\right)\right\}\\
&+(1-p(x=0))p(y=1|x=\epsilon)\max\left\{-\log\left(\tilde{p}(y=1|x=0)\right),-\log\left(\tilde{p}(y=1|x=\epsilon)\right)\right\}\\
=&p(x=0)\max\left\{-\log\left(\tilde{p}(y=2|x=0)\right),-\log\left(\tilde{p}(y=2|x=\epsilon)\right)\right\}\\
&+(1-p(x=0))\max\left\{-\log\left(\tilde{p}(y=1|x=0)\right),-\log\left(\tilde{p}(y=1|x=\epsilon)\right\}\right.
\end{aligned}$$

In terms of a and b, this yields the expected loss (denoted L instead of $\mathcal{L}$ for clarity)

$$L(a,b) = \max\Big\{\log(1+e^{a}),\log(1+e^{b})\Big\}p_0 + \max\Big\{\log(1+e^{-a}),\log(1+e^{-b})\Big\}(1-p_0).$$

This is minimized if $a=b$ as then both maxima are minimal and yields:

$$L(a) = L(a,a) = \log(1+e^{a})p_0 + \log(1+e^{-a})(1-p_0).$$

The critical point is attained at $a^* = b^* = \log\left(\frac{1-p_0}{p_0}\right)$. Therefore,

$$a^* = b^* = \begin{cases}>0 & \text{if } p_0 < \frac{1}{2}\\ <0 & \text{if } p_0 > \frac{1}{2}\end{cases},$$

and we classify $x=0$ and $x=\epsilon$ correctly if $p_0 > 1/2$ and $p_0 < 1/2$, respectively. As a result, the error of 100% adversarial training is given by $\min\{p_0, 1-p_0\}$ whereas the Bayes optimal error is zero as the problem is deterministic.

Next, we consider 50% adversarial and 50% clean training. Here, the expected loss

$$\mathbb{E}\Big[\max_{\|\delta\|_\infty\leq\epsilon}\mathcal{L}(g(x+\delta),y)\Big] + \mathbb{E}\Big[\mathcal{L}(g(x+\delta),y)\Big],$$

can be written in terms of a and b as

$$\begin{aligned}
L(a,b) =& \max\Big\{\log(1+e^{a}),\log(1+e^{b})\Big\}p_0 + \max\Big\{\log(1+e^{-a}),\log(1+e^{-b})\Big\}(1-p_0)\\
&+\log(1+e^{a})p_0 + \log(1+e^{-b})(1-p_0).
\end{aligned}$$

We make the following case distinction. If $a \geq b$, then the loss reduces to

$$\begin{aligned}
L(a,b) &= \log(1+e^{a})p_0 + \log(1+e^{-b})(1-p_0) + \log(1+e^{a})p_0 + \log(1+e^{-b})(1-p_0)\\
&\geq L(a,a) = 2\log(1+e^{a})p_0 + 2\log(1+e^{-a})(1-p_0).
\end{aligned}$$

Solving for the critical point yields $a^* = b^* = \log\left((1-p_0)/p_0\right)$. Considering $a \leq b$, we obtain the loss

$$L(a,b) = \log(1+e^b)p_0 + \log(1+e^{-a})(1-p_0) + \log(1+e^a)p_0 + \log(1+e^{-b})(1-p_0).$$

Again, solving for the critical point also yields $a^* = b^* = \log\left((1-p_0)/p_0\right)$. As this coincides with the solution found for 100% adversarial training, the error is $\min\{p_0, 1-p_0\}$ and not equal to the Bayes optimal error.

For our confidence-calibrated adversarial training one first has to solve

$$\delta_x^*(j) = \underset{\|\delta\|_\infty \leq \epsilon}{\arg\max} \max_{k \neq j} \tilde{p}(y = k \,|\, x + \delta).$$

With a and b as defined above, and noting that we consider a binary classification problem with examples $x = 0$ as well as $x = \epsilon$ and predicted probability distribution $\tilde{p}$, we get:

$$\delta_0^*(1) = \underset{\|\delta\|_\infty \leq \epsilon}{\arg\max}\, \tilde{p}(y=2|0+\delta) = \begin{cases} 0 & a < b \\ \epsilon & \text{else} \end{cases} \quad \delta_0^*(2) = \underset{\|\delta\|_\infty \leq \epsilon}{\arg\max}\, \tilde{p}(y=1|0+\delta) = \begin{cases} \epsilon & a < b \\ 0 & \text{else} \end{cases},$$

$$\delta_\epsilon^*(1) = \underset{\|\delta\|_\infty \leq \epsilon}{\arg\max}\, \tilde{p}(y=2|\epsilon+\delta) = \begin{cases} -\epsilon & a < b \\ 0 & \text{else} \end{cases} \quad \delta_\epsilon^*(2) = \underset{\|\delta\|_\infty \leq \epsilon}{\arg\max}\, \tilde{p}(y=1|\epsilon+\delta) = \begin{cases} 0 & a < b \\ -\epsilon & \text{else} \end{cases}.$$

Note that the imposed distribution $\hat{p}$ over classes depends on the true label y of x and the perturbation $\delta_x^*(y)$ through $\lambda(\delta_x^*(y))$. This is made explicit using the simplified notation $\hat{p}_y(\lambda(\delta_x^*(y)))(x) := \hat{p}(\lambda(\delta_x^*(y)))(y|x)$. Due to the simple structure of the problem, however, it holds that $\delta_x^*(y)$ is either 0 or ϵ (i.e., $\|\delta_x^*(y)\|_\infty$ is also either 0 or ϵ). In CCAT we use the standard cross-entropy loss for 50% of the batch, while the following loss is used for the other 50% :

$$\mathcal{L}\left(\tilde{p}(x), \hat{p}_y\left(\lambda\left(\delta_x^*(y)\right)\right)(x)\right) = -\sum_{j=1}^{2} \hat{p}_y\left(\lambda\left(\delta_x^*(y)\right)\right)(y = j \,|\, x = x) \log\left(\tilde{p}\left(y = j \,|\, x = x + \delta_x^*(j)\right)\right).$$

Then, the corresponding expected loss is given by

$$\begin{aligned}
&\mathbb{E}\Big[\mathcal{L}\Big(\tilde{p}(x), \hat{p}_y\left(\lambda\left(\delta_x^*(y)\right)\right)(x)\Big)\Big] = \mathbb{E}\Big[\,\mathbb{E}\Big[\mathcal{L}\Big(\tilde{p}(x), \hat{p}_y\left(\lambda\left(\delta_x^*(y)\right)\right)(x)\Big)\,\Big|\, x\Big]\Big] \\
=&p(x=0)\,\mathbb{E}\Big[\mathcal{L}\Big(\tilde{p}(0), \hat{p}_y\left(\lambda\left(\delta_0^*(y)\right)\right)(0)\Big)\,\Big|\, x=0\Big] + p(x=\epsilon)\,\mathbb{E}\Big[\mathcal{L}\Big(\tilde{p}(\epsilon), \hat{p}_Y\left(\lambda\left(\delta_\epsilon^*(y)\right)\right)(\epsilon)\Big)\,\Big|\, x=\epsilon\Big],
\end{aligned}$$

where $p(x=0) = p_0$ and $p(x=\epsilon) = 1 - p(x=0) = 1 - p_0$. With the true conditional probabilities $p(y|x)$ we obtain

$$\begin{aligned}
\mathbb{E}\Big[\mathcal{L}\Big(\tilde{p}(x), \hat{p}_y(\lambda(\delta))(x)\Big)\,\Big|\, x\Big] &= \sum_{s=1}^{2} p(y = s \,|\, x)\, \mathcal{L}\Big(\tilde{p}(x), \hat{p}_s\left(\lambda\left(\delta_x^*(s)\right)\right)(x)\Big) \\
= -\sum_{s=1}^{2} p(y = s \,|\, x) \sum_{j=1}^{2} \hat{p}_s\left(\lambda\left(\delta_x^*(s)\right)\right)(y = j \,|\, x) \log\left(\tilde{p}\left(y = j \,|\, x = x + \delta_x^*(s)\right)\right)
\end{aligned}$$

For the considered problem, it holds $p(y = 2 \,|\, x = 0) = p(y = 1 \,|\, x = \epsilon) = 1$ by assumption. Thus,

$$\mathbb{E}\Big[\mathcal{L}\Big(\tilde{p}(x), \hat{p}_y(\lambda(\delta))(x)\Big)\Big|x=0\Big] = -\sum_{j=1}^{2} \hat{p}_2\left(\lambda\left(\delta_0^*(2)\right)\right)(y = j \,|\, x = 0) \log\left(\tilde{p}\left(y = j \,|\, x = x + \delta_0^*(2)\right)\right)$$

$$\mathbb{E}\Big[\mathcal{L}\Big(\tilde{p}(x), \hat{p}_y(\lambda(\delta))(x)\Big)\,\Big|\, x=\epsilon\Big] = -\sum_{j=1}^{2} \hat{p}_1\left(\lambda\left(\delta_\epsilon^*(1)\right)\right)(y = j \,|\, x = \epsilon) \log\left(\tilde{p}\left(y = j \,|\, x = x + \delta_\epsilon^*(1)\right)\right)$$

As $\|\delta_x^*(y)\|_\infty$ is either 0 or ϵ and $\lambda(0) = 1$, we simplify notation by setting $\lambda := \lambda(\epsilon)$. Moreover, we note that

$$\hat{p}_y(\lambda)(y = j|x = x) = \begin{cases} \lambda + \frac{(1-\lambda)}{K} & \text{if } y = j \\ \frac{(1-\lambda)}{K} & \text{else} \end{cases},$$

where K is the number of classes, i.e., $K = 2$ in our case. With $\lambda + {}^{(1-\lambda)}/_2 = {}^{(1+\lambda)}/_2$, we can write the total loss (i.e., cross-entropy loss with and without modified target distribution) of confidence-calibrated adversarial training in terms of a and b as

$$\begin{aligned}
L(a,b) =& p_0\Big[\log(1+e^a)\mathbb{1}_{a\geq b} + \mathbb{1}_{a<b}\Big(\frac{(1+\lambda)}{2}\log(1+e^b) + \frac{(1-\lambda)}{2}\log(1+e^{-b})\Big)\Big] \\
&+ (1-p_0)\Big[\log(1+e^{-b})\mathbb{1}_{a\geq b} + \mathbb{1}_{a<b}\Big(\frac{(1+\lambda)}{2}\log(1+e^{-a}) + \frac{(1-\lambda)}{2}\log(1+e^a)\Big)\Big] \\
&+ \log(1+e^a)p_0 + \log(1+e^{-b})(1-p_0),
\end{aligned}$$

where we have omitted a global factor ${}^1/_2$ for better readability. Note that the last row corresponds to the "clean" cross-entropy loss while the first two rows represent the adversarial part of the loss. As suggested by the indicator function $\mathbb{1}$, we distinguish two cases $a \geq b$ and $a < b$. First, considering $a \geq b$, it is easy to see that in order to minimize the loss we get $a = b$:

$$\partial_a L = 2\frac{e^a}{1+e^a}p_0 - \frac{e^{-a}}{1+e^{-a}}(1-p_0).$$

This yields $e^a = \frac{1-p_0}{p_0}$ or $a = \log\left(\frac{1-p_0}{p_0}\right)$ and the minimum for $a \geq b$ is attained on the boundary of the domain. Second, for $a \leq b$, we get

$$\begin{aligned}
\partial_a L =& \Big[\frac{(1+\lambda)}{2}\frac{-e^{-a}}{1+e^{-a}} + \frac{(1-\lambda)}{2}\frac{e^a}{1+e^a}\Big](1-p_0) + p_0\frac{e^a}{1+e^a}, \\
\partial_b L =& \Big[\frac{(1+\lambda)}{2}\frac{e^b}{1+e^b} + \frac{(1-\lambda)}{2}\frac{-e^b}{1+e^{-b}}\Big]p_0 + (1-p_0)\frac{-e^{-b}}{1+e^{-b}}.
\end{aligned}$$

This yields the following critical points:

$$a^* = \log\left(\frac{\frac{1+\lambda}{2}(1-p_0)}{p_0 + \frac{1-\lambda}{2}(1-p_0)}\right) \quad \text{and} \quad b^* = \log\left(\frac{\frac{1-\lambda}{2}p_0 + (1-p_0)}{\frac{1+\lambda}{2}p_0}\right).$$

It is straightforward to check that $a^* < b^*$ for all $0 < p_0 < 1$. Indeed, we have

$$\frac{\frac{1+\lambda}{2}(1-p_0)}{p_0 + \frac{1-\lambda}{2}(1-p_0)} = \frac{\frac{1+\lambda}{2}(1-p_0)}{p_0\frac{1+\lambda}{2} + \frac{1-\lambda}{2}} = \frac{1-p_0-\frac{(1-\lambda)}{2}(1-p_0)}{p_0\frac{1+\lambda}{2} + \frac{1-\lambda}{2}} < \frac{\frac{1-\lambda}{2}p_0 + (1-p_0)}{\frac{1+\lambda}{2}p_0}$$

if $0 < p_0 < 1$ and $\lambda < 1$ by assumption. We have $a^* < 0$ with $g_2(0) > g_1(0)$ (i.e., the Bayes optimal decision for $x = 0$) and $b^* > 0$ with $g_1(\epsilon) > g_2(\epsilon)$ (i.e., Bayes optimal decision for $x = \epsilon$) if

$$1 > \frac{1-p_0}{p_0}\lambda \quad \text{and} \quad 1 > \frac{p_0}{1-p_0}\lambda,$$

respectively. Overall, we recover the Bayes classifier if

$$\lambda < \min\Big\{\frac{1-p_0}{p_0}, \frac{p_0}{1-p_0}\Big\}.$$

□

Here, 100% and 50%/50% standard adversarial training are unable to obtain *both* robustness and accuracy: The ϵ-ball used during training contains examples of different classes such that adversarial training enforces high-confidence predictions in contradicting classes. CCAT addresses this problem by encouraging low-confidence predictions on adversarial examples within the ϵ-ball. Thus, CCAT is able to improve accuracy while preserving robustness.

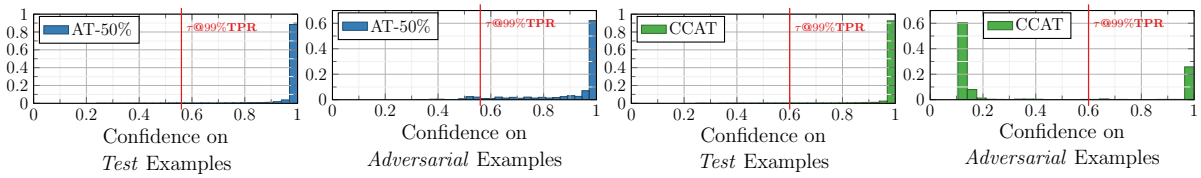


Figure 7.3: **Confidence Histograms:** On SVHN, for AT (50%/50% adversarial training) and CCAT, we show confidence histograms corresponding to *correctly classified* test examples and adversarial examples. We consider the worst-case adversarial examples across all L_∞ attacks for $\epsilon = 0.03$. While the confidence of adversarial examples is reduced slightly for AT, CCAT is able to distinguish the majority of adversarial examples from (clean) test examples by confidence thresholding (in red).

7.3 Detection and Robustness Evaluation with Adaptive Attack

CCAT allows rejecting (adversarial) inputs by confidence-thresholding before classifying them, see Figure 7.3. As we will see, this "reject option", is also beneficial for standard adversarial training (AT). Thus, evaluation also requires two stages: First, we fix the confidence threshold at 99% true positive rate (TPR), where correctly classified clean examples are positives such that at most 1% (correctly classified) clean examples are rejected. Second, on the non-rejected examples, we evaluate accuracy and robustness using *confidence-thresholded* (robust) test error.

7.3.1 Adaptive Attack

As CCAT encourages low confidence on adversarial examples, we use PGD to maximize the confidence of adversarial examples, c.f. Equation (7.4), as effective adaptive attack against CCAT. In order to effectively optimize our objective, we introduce a simple but crucial improvement: after each iteration, the computed update is only applied if the objective is improved; otherwise the learning rate is reduced. Additionally, we use momentum [DLP+18] and run the attack for exactly T iterations, choosing the perturbation corresponding to the best objective across all iterations. In addition to random initialization, we found that $\delta = 0$ is an effective initialization against CCAT. We applied the same principles for [IEAL18], i.e., PGD with approximated gradients, Equation (7.4) as objective, momentum and backtracking. We also use Equation (7.4) as objective for the black-box attacks of [NK17, IEAL18, KH18, ACFH20].

7.3.2 Detection Evaluation

In the first stage, we consider a detection setting: adversarial example are *negatives* and correctly classified clean examples are *positives*. The confidence threshold τ is chosen extremely conservatively by requiring a **99% true positive rate (TPR)**: at most 1% of correctly classified clean examples can be rejected. As a result, the confidence threshold is determined *only* by correctly classified clean examples, independent of adversarial examples. Incorrectly rejecting a significant fraction of correctly classified clean examples is unacceptable. This is also the reason why we do not report the area under the receiver operating characteristic (ROC) curve as related work [LLLS18b, MLW+18] (see Appendix D.3 for a discussion). Instead, we consider the **false positive rate (FPR)** for fixed TPR, see Figure 7.4. Appendix D.4 also contains results for 95% and 98% TPR for comparison.

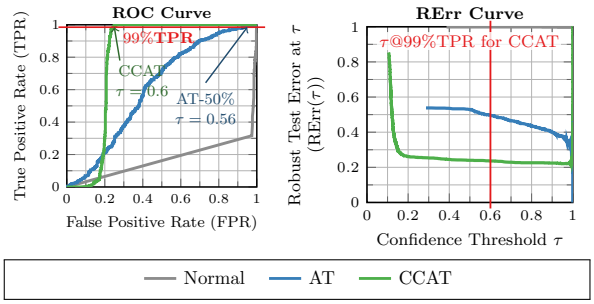


Figure 7.4: **ROC and RErr Curves:** On SVHN, we show ROC curves when distinguishing *correctly classified* test examples from adversarial examples by confidence (left) and (confidence-thresholded) RErr against confidence threshold τ (right) for worst-case adversarial examples across L_∞ attacks with $\epsilon = 0.03$. The confidence threshold τ is chosen exclusively on correctly classified clean examples to obtain 99%TPR. For CCAT, this results in $\tau \approx 0.6$. Note that RErr subsumes both Err and FPR.

7.3.3 Robustness Evaluation

In the second stage, after confidence-thresholding, we consider the widely used robust test error (RErr) [MMS+18]. It quantifies the model's test error in the case where all test examples are allowed to be attacked, i.e., modified within the chosen threat model, e.g., for L_p:

$$\text{"Standard" RErr} = \frac{1}{N}\sum_{n=1}^{N} \max_{\|\delta\|_p \leq \epsilon} \mathbb{1}_{f(x_n+\delta)\neq y_n} \tag{7.8}$$

where $\{(x_n, y_n)\}_{n=1}^N$ are test examples and labels. In practice, RErr is computed empirically using adversarial attacks. Unfortunately, standard RErr does not take into account the option of rejecting (adversarial) examples.

We propose a generalized definition adapted to our confidence-thresholded setting where the model can reject examples. For fixed confidence threshold τ at 99%TPR, the **confidence-thresholded RErr** is defined as

$$\text{RErr}(\tau) = \frac{\sum_{n=1}^{N} \max_{\|\delta\|_p \leq \epsilon, c(x_n+\delta)\geq\tau} \mathbb{1}_{f(x_n+\delta)\neq y_n}}{\sum_{n=1}^{N} \max_{\|\delta\|_p \leq \epsilon} \mathbb{1}_{c(x_n+\delta)\geq\tau}} \tag{7.9}$$

with $c(x) = \max_k f_k(x)$ and $f(x)$ being the model's confidence and predicted class on example x, respectively. Essentially, this is the **test error on test examples that can be modified within the chosen threat model *and* pass confidence thresholding**. For $\tau = 0$ (i.e., all examples pass confidence thresholding) this reduces to the standard RErr, comparable to related work. We stress that our adaptive attack in Equation (7.4) directly maximizes the numerator of Equation (7.9) by maximizing the confidence of classes not equal y. A (clean) **confidence-thresholded test error (Err(τ))** is obtained similarly. In the following, if not stated otherwise, we report *confidence-thresholded* RErr and Err as default and omit the confidence threshold τ for brevity.

Implementation Details: As Equation (7.9) cannot be computed exactly, we instead compute

$$\frac{\sum_{n=1}^{N} \max\{\mathbb{1}_{f(x_n)\neq y_n}\mathbb{1}_{c(x_n)\geq\tau}, \mathbb{1}_{f(\tilde{x}_n)\neq y_n}\mathbb{1}_{c(\tilde{x}_n)\geq\tau}\}}{\sum_{n=1}^{N} \max\{\mathbb{1}_{c(x_n)\geq\tau}, \mathbb{1}_{c(\tilde{x}_n)\geq\tau}\}} \tag{7.10}$$

which is an upper bound assuming that our attack is perfect. Essentially, this counts the test examples x_n that are either classified incorrectly with confidence $c(x_n) \geq \tau$ or that can

be attacked successfully $\tilde{x}_n = x_n + \delta$ with confidence $c(\tilde{x}_n) \geq \tau$. This is normalized by the total number of test examples x_n that have $c(x_n) \geq \tau$ or where the corresponding adversarial example $\tilde{x}_n$ has $c(\tilde{x}_n) \geq \tau$. Again, it can easily be seen that $\tau = 0$ reduces Equation (7.10) to its unthresholded variant, i.e., standard RErr, ensuring full comparability to related work.

We also want to highlight two special cases that are correctly taken into account by Equation (7.10): (a) if a correctly classified test example x_n with $f(x_n) = y_n$ has confidence $c(x_n) < \tau$ and is rejected, but the corresponding adversarial example $\tilde{x}_n$ with $f(\tilde{x}_n) \neq y$ has confidence $c(\tilde{x}_n) \geq \tau$ and is thus *not* rejected, this is counted both in the numerator and denominator; (b) if an incorrectly classified test example x_n with $f(x_n) \neq y$ and confidence $c(x_n) < \tau$ is rejected, but has a corresponding adversarial example $\tilde{x}_n$ with $f(\tilde{x}_n) \neq y$ and confidence $c(x_n) \geq \tau$ such that it is *not* rejected, this is also counted in the numerator as well as denominator. Note that these cases are handled differently in a detection evaluation following related work [MLW+18, LLLS18b]: negatives are adversarial examples corresponding to correctly classified clean examples that are successful.

Connection to FPR: FPR quantifies how well an adversary can perturb (correctly classified) examples while not being rejected. The confidence-thresholded RErr is more conservative as it measures *any* non-rejected error (adversarial or not). For example, case (b) from the above paragraph would not contribute towards the FPR since the original test example is already misclassified. Thus, while RErr implicitly includes FPR as well as Err, it is even more conservative than just considering "FPR + Err". Therefore, we report only RErr and include FPRs for all our experiments in the appendix.

7.3.4 Per-Example Worst-Case Evaluation

Instead of reporting average or per-attack results, we use a per-example *worst-case* evaluation scheme: For each individual test example, all adversarial examples from all attacks (and restarts) are accumulated. Subsequently, *per test example*, only the adversarial example with the highest confidence is considered, resulting in a significantly stronger robustness evaluation compared to related work.

7.4 Experiments

We evaluate CCAT in comparison with AT [MMS+18] and related work [MWK20, ZYJ+19] on MNIST [LBBH98], SVHN [NWC+11] and Cifar10 [Kri09] as well as MNIST-C [MG19] and Cifar10-C [HD19] with corrupted examples (e.g., blur, noise, compression, transforms etc.). We report *confidence-thresholded* test error (Err; ↓ lower is better) and *confidence-thresholded* robust test error (RErr; ↓ lower is better) for a threshold τ corresponding to 99% true positive rate (TPR) and omit τ for brevity. We note that normal and standard adversarial training (AT) are also allowed to reject examples by confidence thresholding. Err is computed on 9000 test examples. RErr is computed on 1000 test examples. The confidence threshold τ depends *only* on correctly classified clean examples and is fixed on the held-out *last* 1000 test examples.

7.4.1 Attacks

We consider several adaptive white- and black-box L_p attacks for $p \in \{\infty, 2, 1, 0\}$ as well as adversarial frames and distal adversarial examples:

Adaptive White-Box Attacks: As white-box attacks, we use PGD to maximize the objectives in Equation (7.2) and (7.4), referred to as PGD-CE and PGD-Conf. We use $T = 1000$ iterations and 10 random restarts with random initialization plus one restart with zero initialization for PGD-Conf, and $T = 200$ with 50 random restarts for PGD-CE. For L_∞, L_2, L_1 and L_0 attacks, we set ϵ **to 0.3, 3, 18, 15 (MNIST) or 0.03, 2, 24, 10 (SVHN/Cifar10)**. Both PGD-CE and PGD-Conf also use momentum and backtracking. These two "tricks" add two additional hyperparameters to the number of iterations T and the learning rate γ, namely the momentum parameter β and the learning rate factor α. After each iteration, the computed update, already including the momentum term, is only applied if this improves the objective. This is checked through an additional forward pass. If not, the learning rate is divided by α, and the update is rejected. It is important to note that the learning rate is updated per test example individually. In practice, for PGD-CE, we use $\gamma = 0.05$, $\beta = 0.9$ and $\alpha = 1.25$; for PGD-Conf, with $T = 1000$ iterations, we use $\gamma = 0.001$, $\beta = 0.9$ and $\alpha = 1.1$.

PGD for L_∞ adversarial examples uses a signed gradient and projection is implemented through clipping to $[-\epsilon, \epsilon]$. For the L_2 norm, the gradient is normalized by dividing by the L_2 norm, for the L_1 norm only the 1% largest values (in absolute terms) of the gradient are kept and normalized by their L_1 norm, and for the L_0 norm, the gradient is normalized by dividing by the L_1 norm. The L_1 projection follows the algorithm of [DSSC08] and for L_0 adversarial examples, only the ϵ largest values are kept. Similarly, initialization for L_2 and L_1 are simple by randomly choosing a direction and then normalizing by their norm. For L_0, we randomly choose pixels with probability $(2/3\epsilon)/(HWD)$ and set them to a uniformly random values $u \in [0, 1]$, where $H \times W \times D$ is the image size. We found that tuning the learning rate for PGD with L_1 and L_0 constraints is more difficult, making multiple restarts necessary.

Adaptive Black-Box Attacks: As black-box attacks, we additionally evaluate random sampling with $T = 5000$ attempts. We also implemented the Query-Limited (QL) black-box attack of [IEAL18] using a population of 50 and variance of 0.1 for estimating the gradient. We use a learning rate of 0.001 (note that the gradient is signed, as in [MMS+18]) and also integrated a momentum term with $\beta = 0.9$ and backtracking with $\alpha = 1.1$ and $T = 1000$ iterations. We use zero and random initialization with 10 random restarts. For the Simple attack we follow the algorithmic description in [NK17] considering only axis-aligned perturbations of size ϵ per pixel. We run the attack for $T = 1000$ iterations and allow 10 random restarts. Following, [KH18], we further use the Geometry attack for $T = 1000$ iterations. Random sampling, QL, Simple and Geometry attacks are run for arbitrary L_p, $p \in \{\infty, 2, 1, 0\}$. For L_∞, we also use the Square attack proposed in [ACFH20] with $T = 5000$ iterations with a probability of change of 0.05. For all attacks, we use Equation (7.4) as objective. Finally, for L_0, we also use Corner Search (CS) [CH19] with the cross-entropy loss as objective, for $T = 200$ iterations. We emphasize that, except for QL, these attacks are not gradient-based and do not approximate the gradient. Furthermore, we note that all attacks except CS are adapted to explicitly attack CCAT by maximizing confidence.

Adversarial Frames and Distal Adversarial Examples: Besides L_p attacks, we also consider adversarial frames [ZZRP19] which allow a 2 (MNIST) or 3 (SVHN/Cifar10) pixel border to be manipulated arbitrarily within $[0, 1]$ to maximize Equation (7.4) using PGD. Furthermore, we compute distal adversarial examples [HAB19] starting with a random image and using PGD to maximize (7.4) within a L_∞-ball of size $\epsilon = 0.3$ (MNIST) or $\epsilon = 0.03$ (SVHN/Cifar10).

Summary: Overall, we evaluate 7 L_p attacks (not counting Random) with up to 50 restarts and 5000 iterations depending on the attack. As outlined in Section 7.3.4, we evaluate using the *per-example* worst-case, i.e., the most effective attack (and restart) per example. All of these attacks are summarized below:

Attack	Objective	T	Restarts
PGD-CE	Equation (7.2), random init.	200	50
PGD-Conf	Equation (7.4), zero + random init.	1000	11
QL†	Equation (7.4), zero + random init.	1000	11
Simple†	Equation (7.4)	1000	10
Square†	Equation (7.4), L_∞, L_2 only	5000	1
CS†	Equation (7.2), L_0 only	200	1
Geometry†	Equation (7.4)	1000	1
Random†	Equation (7.4)	–	5000

† Black-box attacks.

7.4.2 Training and Baselines

We train 50%/50% AT (AT-50%) and CCAT as well as 100% AT (AT-100%) with L_∞ attacks using $T = 40$ iterations for PGD-CE and PGD-Conf (including momentum and backtracking with $\beta = 0.9$, $\alpha = 1.5$), respectively, and $\epsilon = 0.3$ (MNIST) or $\epsilon = 0.03$ (SVHN/Cifar10). We use ResNet-20 [HZRS16a], implemented in PyTorch [PGC+17]. For CCAT, we use $\rho = 10$ and also train on 50% clean/50% adversarial examples (per batch) as for AT-50%. We train using stochastic gradient descent with a batch size of 100 and a total of 100 and 200 epochs on MNIST and SVHN/Cifar10, respectively. For PGD-CE we use a learning rate of 0.05, 0.01 and 0.005 on MNIST, SVHN and Cifar10. For PGD-Conf we use a learning rate of 0.005. For CCAT, we randomly switch between random and zero initialization. For training, we start with a learning rate of 0.1 on MNIST/SVHN and 0.075 on Cifar10. The learning rate is multiplied by 0.95 after each epoch. We do not use weight decay. On SVHN and Cifar10, we use random cropping, random flipping (only Cifar10) and contrast augmentation during training.

As baseline, we use the multi-steepest descent (MSD) adversarial training of [MWK20], using the code and models provided in the official repository[2]. The models correspond to a LeNet-like [LBBH98] architecture on MNIST, and the pre-activation version of ResNet-18 [HZRS16a] on Cifar10. The models were trained with L_∞, L_2 and L_1 adversarial examples and ϵ set to $0.3, 1.5, 12$ and $0.03, 0.5, 12$, respectively. Additionally, we compare to TRADES [ZYJ+19] using the code and pre-trained models from the official repository[3]. The models correspond to a convolutional architecture with four convolutional and three fully-connected layers [CW17b] on MNIST, and a wide ResNet, specifically WRN-10-28 [ZK16], on Cifar10. Both are trained using *only* L_∞ adversarial examples with $\epsilon = 0.3$ and $\epsilon = 0.03$, respectively. Finally, on Cifar10, we also use the pre-trained ResNet-50 from [MMS+18] obtained from the official repository[4]. The model was trained on L_∞ adversarial examples with $\epsilon = 0.03$. We emphasize that for all these baselines the same evaluation protocol as for CCAT applies.

Besides these adversarial training baselines, we evaluate two detection methods: the Mahalanobis detector (MAHA) of [MLW+18] and the local intrinsic dimensionality (LID) detector of [LLLS18b]. We used the code provided by [LLLS18b] from the official repository[5]. For evaluation, we used the provided setup, adding *only* PGD-CE and PGD-Conf with $T = 1000$, $T = 200$ and $T = 40$. For $T = 1000$, we used 5 random restarts, for $T = 200$, we used 25 restarts, and for $T = 40$, we used one restart. These were run for L_∞, L_2, L_1 and L_0. We also evaluated

[2] https://github.com/locuslab/robust_union
[3] https://github.com/yaodongyu/TRADES
[4] https://github.com/MadryLab/robustness
[5] https://github.com/pokaxpoka/deep_Mahalanobis_detector

SVHN: RErr @99%TPR, L_∞, $\epsilon = 0.03$						
	worst case	top-5 attacks/restarts out of 7 attacks with 84 restarts				
AT-50%	56.0	52.1	52.0	51.9	51.6	51.4
CCAT	39.1	23.6	13.7	13.6	12.6	12.5

Table 7.1: **Per-Example Worst-Case Evaluation:** Confidence-thresholded RErr with τ@99%TPR for the per-example worst-case and the top-5 individual attacks/restarts among 7 attacks with 84 restarts in total. Multiple restarts are *necessary* to effectively attack CCAT, demonstrating that it is difficult to "crack".

distal adversarial examples. While the hyperparameters were chosen for our L_∞ PGD-CE attack ($T = 40$, one restart) and kept fixed for other threat models, the logistic regression classifier trained on the computed statistics (e.g., the Mahalanobis statistics) is trained for each threat model individually. This results in a significant advantage over AT and CCAT. For worst-case evaluation, we use the obtained detection score instead of the model's confidence (as done for AT and CCAT). This means, for each test example individually, we consider the adversarial example with the worst detection score.

7.4.3 Ablation Study

In the following, we briefly discuss ablation experiments supporting our choice of evaluation metrics as well as (per-example) worst-case evaluation. We also ablate the proposed backtracking scheme for PGD-Conf. Ablations regarding attack and training hyperparameters are included in Appendix D.3.1.

Evaluation Metrics: Figure 7.4 shows ROC curves, i.e., how well adversarial examples can be rejected by confidence. As marked in red, we are only interested in the FPR for the conservative choice of 99%TPR, yielding the confidence threshold τ. The RErr curves highlight how robustness is influenced by the threshold: AT also benefits from a reject option, however, not as much as CCAT which has been explicitly designed for rejecting adversarial examples.

Worst-Case Evaluation: Table 7.1 illustrates the importance of worst-case evaluation on SVHN, showing that CCAT is significantly "harder" to attack than AT. We show the worst-case RErr over all L_∞ attacks as well as the top-5 individual attacks (each restart treated as separate attack). For AT-50%, a single restart of PGD-Conf with $T = 1000$ iterations is highly successful, with 52.1% RErr close to the overall worst-case of 56%. For CCAT, in contrast, multiple restarts are crucial as the best individual attack, PGD-Conf with $T = 1000$ iterations and zero initialization obtains only 23.6% RErr compared to the overall worst-case of 39.1%.

Backtracking: Figure 7.5 illustrates the advantage of backtracking for PGD-Conf with T=40 iterations on 5 test examples of SVHN. Backtracking results in better objective values and avoids oscillation, i.e., a stronger attack for training and testing. In addition, while T=200 iterations are sufficient against AT, we needed up to T=1000 iterations for CCAT.

7.4.4 Main Results

Our main results are presented in Table 7.2. In this section, we discuss robustness against both seen and unseen adversarial examples and compare to the baselines from Section 7.4.2. We also report the worst-case robustness across all threat models in Table 7.3. We note that results for 95% and 98%TPR, including evaluated FPRs, can be found in Appendix D.4.

Robustness Against seen L_∞ Attacks: Considering Table 7.2 and L_∞ adversarial examples as seen during training, CCAT exhibits comparable robustness to AT. With 7.4%/67.9% RErr on

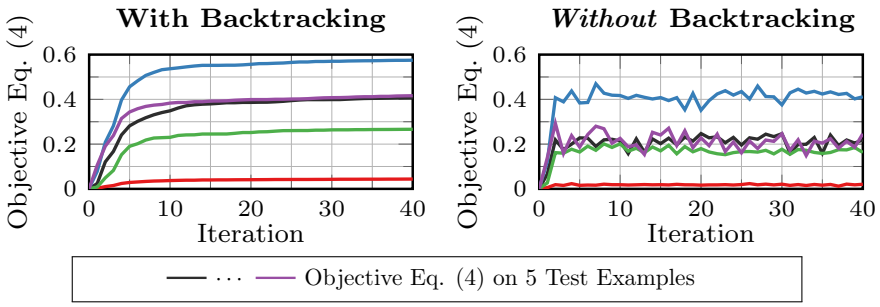


Figure 7.5: **Backtracking:** Our L_∞ PGD-Conf attack using 40 iterations with momentum and our developed backtracking scheme (left) and without both (right) on SVHN. We plot Equation (7.4) over iterations for the first 5 test examples corresponding to different colors. Backtracking avoids oscillation and obtains higher overall objective values within the same number of iterations.

MNIST/Cifar10, CCAT lacks behind AT-50% (1.7%/62.7%) only slightly. On SVHN, in contrast CCAT outperforms AT-50% and AT-100% significantly with 39.1% vs. 56.0% and 48.3%. We note that CCAT and AT-50% are trained on 50% clean / 50% adversarial examples. This is in contrast to AT-100% trained on 100% adversarial examples, which improves robustness slightly, e.g., from 56%/62.7% to 48.3%/59.9% on SVHN/Cifar10.

Robustness Against unseen L_p Attacks: Regarding unseen attacks, AT's robustness deteriorates quickly while CCAT is able to generalize robustness to novel threat models. On SVHN, for example, RErr of AT-50% goes up to 88.4%, 99.4%, 99.5% and 73.6% for larger L_∞, L_2, L_1 and L_0 attacks. In contrast, CCAT's robustness generalizes to these unseen attacks significantly better, with 53.1%, 29%, 31.7% and 3.5%, respectively. The results on MNIST and Cifar10 or for AT-100% tell a similar story. However, AT generalizes better to L_1 and L_0 attacks on MNIST, possibly due to the large L_∞-ball used during training ($\epsilon = 0.3$). Here, training purely on adversarial examples, i.e., AT-100% is beneficial. On Cifar10, CCAT has more difficulties with large L_∞ attacks ($\epsilon = 0.06$) with 92% RErr. As shown in the appendix, AT benefits from considering FPR as clean Err is not taken into account. On Cifar10, for example, 47.6% FPR compared to 62.7% RErr for AT-50%. This is obviously less pronounced for CCAT due to the improved Err compared to AT. Overall, CCAT improves robustness against arbitrary (unseen) L_p attacks, demonstrating that CCAT indeed extrapolates near-uniform predictions beyond the L_∞ ϵ-ball used during training.

Comparison to MSD and TRADES: TRADES is able to outperform CCAT alongside AT (including AT-Madry) on Cifar10 with respect to the L_∞ adversarial examples seen during training: 43.5% RErr compared to 68.4% for CCAT. This might be a result of training on 100% adversarial examples and using more complex models: TRADES uses a WRN-10-28 with roughly 46.1M weighs, in contrast to our ResNet-20 with 4.3M (and ResNet-18 with 11.1M for MSD). However, regarding unseen L_2, L_1 and L_0 attacks, CCAT outperforms TRADES with 52.2%, 58.8% and 23% compared to 70.9%, 96.9% and 36.9% in terms of RErr. Similarly, CCAT outperforms MSD. This is surprising, as MSD trains on both L_2 and L_1 attacks with smaller ϵ, while CCAT does not. Only against larger L_∞ adversarial examples with $\epsilon = 0.06$, TRADES reduces RErr from 92.4% (CCAT) to 81%. Similar to AT, TRADES also generalizes better to L_2, L_1 or L_0 on MNIST, while MSD is not able to compete. Overall, compared to MSD and TRADES, the robustness obtained by CCAT generalizes better to previously unseen attacks. We also note that, on MNIST, CCAT outperforms the robust Analysis-by-Synthesis (ABS) approach of [SRBB19] w.r.t. L_∞, L_2, and L_0 attacks.

MNIST:	**Err** ↓ in %		*confidence-thresholded* **RErr** ↓ for τ@99%TPR						**FPR** ↓	**Err** ↓
	(clean) $\tau = 0$	(clean) 99%TPR	L_∞ $\epsilon = 0.3$	L_∞ $\epsilon = 0.4$	L_2 $\epsilon = 3$	L_1 $\epsilon = 18$	L_0 $\epsilon = 15$	adv. frames	distal	corrupted MNIST-C
	(seen)	(seen)	seen	**unseen**	**unseen**	**unseen**	**unseen**	**unseen**	**unseen**	**unseen**
NORMAL	0.4	0.1	100.0	100.0	100.0	100.0	92.3	87.7	100.0	32.8
AT-50%	0.5	**0.0**	**1.7**	100.0	81.5	24.6	23.9	73.7	100.0	12.6
AT-100%	0.5	**0.0**	**1.7**	100.0	84.8	21.3	**13.9**	62.3	100.0	17.6
CCAT	**0.3**	0.1	7.4	**11.9**	**0.3**	**1.8**	14.8	**0.2**	**0.0**	**5.7**
* MSD	1.8	0.9	34.3	98.9	59.2	55.9	66.4	8.8	100.0	6.0
* TRADES	0.5	0.1	4.0	99.9	44.3	9.0	35.5	**0.2**	100.0	7.9

SVHN:	**Err** ↓ in %		*confidence-thresholded* **RErr** ↓ for τ@99%TPR						**FPR** ↓
	(clean) $\tau = 0$	(clean) 99%TPR	L_∞ $\epsilon = 0.03$	L_∞ $\epsilon = 0.06$	L_2 $\epsilon = 2$	L_1 $\epsilon = 24$	L_0 $\epsilon = 10$	adv. frames	distal
	(seen)	(seen)	seen	**unseen**	**unseen**	**unseen**	**unseen**	**unseen**	**unseen**
NORMAL	3.6	2.6	99.9	100.0	100.0	100.0	83.7	78.7	87.1
AT-50%	3.4	2.5	56.0	88.4	99.4	99.5	73.6	33.6	86.3
AT-100%	5.9	4.6	48.3	87.1	99.5	99.8	89.4	26.0	81.0
CCAT	**2.9**	**2.1**	**39.1**	**53.1**	**29.0**	**31.7**	**3.5**	**3.7**	**0.0**
* LID	3.3	2.2	91.0	93.1	92.2	90.0	41.6	89.8	8.6
* MAHA	3.3	2.2	73.0	79.5	78.1	67.5	41.5	9.9	**0.0**

CIFAR10:	**Err** ↓ in %		*confidence-thresholded* **RErr** ↓ for τ@99%TPR						**FPR** ↓	**Err** ↓
	(clean) $\tau = 0$	(clean) 99%TPR	L_∞ $\epsilon = 0.03$	L_∞ $\epsilon = 0.06$	L_2 $\epsilon = 2$	L_1 $\epsilon = 24$	L_0 $\epsilon = 10$	adv. frames	distal	corrupted CIFAR10-C
	(seen)	(seen)	seen	**unseen**	**unseen**	**unseen**	**unseen**	**unseen**	**unseen**	**unseen**
NORMAL	8.3	7.4	100.0	100.0	100.0	100.0	84.7	96.7	83.3	12.3
AT-50%	16.6	15.5	62.7	93.7	98.4	98.4	74.4	78.7	75.0	16.2
AT-100%	19.4	18.3	59.9	90.3	98.3	98.0	72.3	79.6	72.5	19.6
CCAT	10.1	6.7	68.4	92.4	**52.2**	**58.8**	**23.0**	**66.1**	**0.0**	**8.5**
* MSD	18.4	17.6	53.2	89.4	88.5	68.6	39.2	82.6	76.7	19.3
* TRADES	15.2	13.2	**43.5**	**81.0**	70.9	96.9	36.9	72.1	76.2	15.0
* AT-Madry	13.0	11.7	45.1	84.5	98.7	97.8	42.3	73.3	78.5	12.9
* LID	**6.4**	**4.9**	99.0	99.2	70.6	89.4	47.0	**66.1**	0.1	11.59
* MAHA	**6.4**	**4.9**	94.1	95.3	90.6	97.6	49.8	70.0	2.4	12.4

Table 7.2: **Main Results: Generalizing Robustness.** For L_∞, L_2, L_1, L_0 attacks and adversarial frames, we report per-example worst-case (confidence-thresholded) Err and RErr at 99%TPR across all attacks. ϵ is reported in the corresponding columns. For distal adversarial examples and corrupted examples, we report FPR and Err, respectively. L_∞ attacks with ϵ=0.3 on MNIST and $\epsilon = 0.03$ on SVHN/Cifar10 were used for training (seen). The remaining attacks were not encountered during training (**unseen**). CCAT outperforms AT and the other baselines regarding robustness against unseen attacks. * Pre-trained models with different architecture, LID/MAHA use the same model.

Detection Baselines: The detection methods LID and MAHA are outperformed by CCAT across all datasets and threat models. On SVHN, for example, MAHA obtains 73% RErr against the seen L_∞ attacks and 79.5%, 78.1%, 67.5% and 41.5% RErr for the unseen L_∞, L_2, L_1 and L_0 attacks. LID is consistently outperformed by MAHA on SVHN. This is striking, as we *only* used PGD-CE and PGD-Conf to attack these approaches and emphasizes the importance of training *adversarially* against an adaptive attack to successfully reject adversarial examples.

Robustness Against Unconventional Attacks: Against adversarial frames, robustness of AT reduces to 73.7% /62.3% RErr (AT-50%/100%), even on MNIST, while CCAT achieves 0.2%.

	MNIST: all unseen		SVHN: all unseen		CIFAR10: all unseen		CIFAR10: unseen except L_∞ with ϵ=0.06	
	FPR↓	RErr ↓	FPR↓	RErr ↓	FPR↓	RErr ↓	FPR↓	RErr ↓
Normal	99.3	100.0	95.9	100.0	93.0	100.0	93.0	100.0
AT-50%	99.3	100.0	96.2	99.9	84.1	99.2	84.1	99.2
AT-100%	99.2	100.0	93.7	99.9	81.0	98.6	81.1	98.7
CCAT	**23.4**	**23.9**	**57.5**	**61.1**	86.3	94.8	**69.1**	**77.6**
* MSD	97.0	99.2	–	–	**76.2**	**94.1**	75.6	93.5
* TRADES	99.3	99.9	–	–	82.8	97.4	82.7	97.3
* AT-Madry	–	–	–	–	87.6	98.9	87.6	98.9

Table 7.3: **Worst-Case Results Across Unseen Attacks:** We report the (per-example) worst-case, confidence-thresholded RErr and FPR across **all** unseen attacks on MNIST, SVHN and Cifar10. On Cifar10, we additionally present results for all attacks except L_∞ adversarial examples with larger $\epsilon = 0.06$ (indicated in blue). CCAT is able to outperform all baselines, including MSD and TRADES, significantly on MNIST and SVHN. On Cifar10, CCAT performs poorly on L_∞ adversarial examples with larger $\epsilon = 0.06$. However, excluding these adversarial examples, CCAT outperforms all baselines on Cifar10. * Pre-trained models with different architectures.

MSD, in contrast, is able to preserve robustness better with 8.8% RErr, which might be due to the L_2 and L_1 attacks seen during training. CCAT outperforms both approaches with 0.2% RErr, as does TRADES. On SVHN and Cifar10, however, CCAT outperforms all approaches, including TRADES, considering adversarial frames. Against distal adversarial examples, CCAT outperforms all approaches significantly, with 0% FPR, compared to the second-best of 72.5% for AT-100% on Cifar10. Only the detection baselines LID and MAHA are competitive, reaching close to 0% FPR. This means that CCAT is able to extrapolate low-confidence distributions to far-away regions of the input space. Finally, we consider corrupted examples (e.g., blur, noise, transforms etc.) where CCAT also improves results, i.e., mean Err across all corruptions. On Cifar10-C, for example, CCAT achieves 8.5% compared 12.9% for AT-Madry and 12.3% for normal training. On MNIST-C, only MSD yields a comparably low Err: 6% vs. 5.7% for CCAT.

Worst-Case Across Unseen Attacks: Table 7.3 reports *per-example* worst-case RErr and FPR for 99%TPR considering **all** unseen attacks. On MNIST and SVHN, RErr increases to nearly 100% for AT, both AT-50% and AT-100%. CCAT, in contrast, is able to achieve considerably lower RErr: 23.9% on MNIST and 61.1% on SVHN. Only on Cifar10, CCAT does not result in a significant improvement. All methods, including related work such as MSD and TRADES yield RErr of 94% or higher. However, this is mainly due to the poor performance of CCAT against large L_∞ adversarial examples with $\epsilon = 0.06$. Excluding these adversarial examples (right most table, indicated in blue) shows that RErr improves to 77.6% for CCAT, while RErr for the remaining methods remains nearly unchanged. Overall, these experiments emphasize that CCAT is able to generalize robustness to previously unseen attacks.

Improved Test Error: CCAT also outperforms AT regarding Err, coming close to that of normal training. On all datasets, *confidence-thresholded* Err for CCAT is better or equal than that of normal training. On Cifar10, only LID/MAHA achieve a better standard and confidence-thresholded Err using a ResNet-34 compared to our ResNet-20 for CCAT (21.2M vs. 4.3M weights). In total the performance of CCAT shows that the robustness-generalization trade-off can be improved significantly.

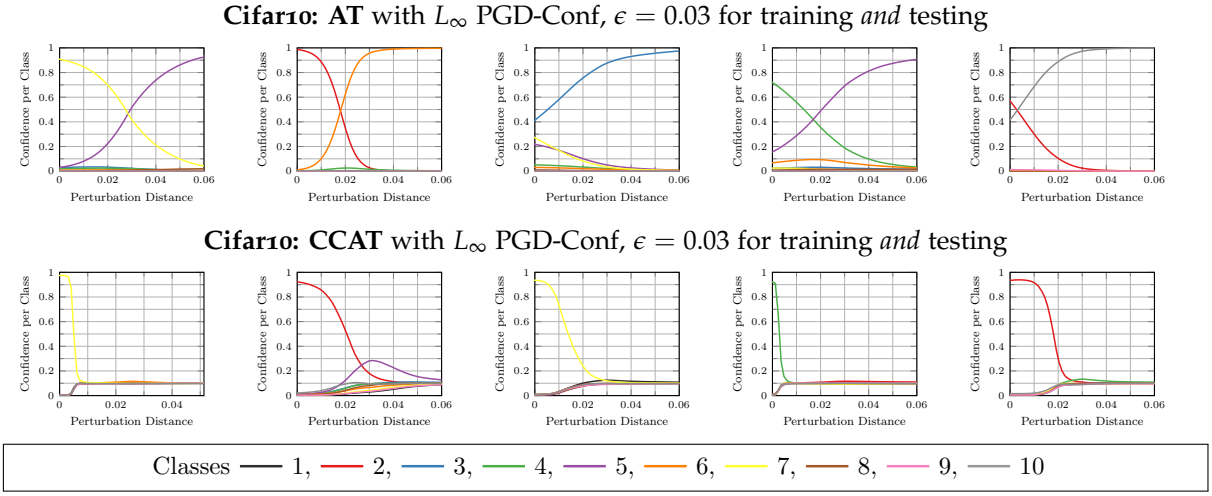


Figure 7.6: **Confidence Calibration:** We plot the probabilities for all ten classes along adversarial directions for AT and CCAT (with $\rho_{\text{pow}} = 10$). Adversarial examples were computed using our L_∞-PGD-Conf attack, c.f. Section 7.4.1: using projected gradient descent [MMS+18] the confidence of the adversarial examples is maximized for $T = 1000$ and L_∞-constraint $\epsilon = 0.03$. The robustness of AT does not generalize beyond the $\epsilon = 0.03$-ball as high confidence adversarial examples can be found for larger perturbations, whereas CCAT predicts close to uniform confidence after some transition phase allowing to easily detect adversarial examples.

7.4.5 Analysis

In the following, we present qualitative results showing that CCAT does indeed reduce confidence on adversarial examples. This also leads to more meaningful behavior in between test examples of different classes.

Confidence Along Adversarial Directions: In Figure 7.6, we plot the probabilities for all ten classes along an adversarial direction for the first 5 test examples on CIFAR10. We note that these directions do not necessarily correspond to successful or high-confidence adversarial examples. The adversarial examples were obtained using our L_∞ PGD-Conf attack with $T = 1000$ iterations and zero initialization for $\epsilon = 0.03$. For AT, we usually observe a change in predictions along these directions, some occurring within $\|\delta\|_\infty \leq \epsilon$, others occurring for $\|\delta\|_\infty > \epsilon$. However, AT always assigns high confidence. Thus, when allowing larger adversarial perturbations at test time, robustness of AT reduces significantly. For CCAT, in contrast, there are only few such cases; more often, the model achieves a near uniform prediction for small $\|\delta\|_\infty$ and extrapolates this behavior beyond the ϵ-ball used for training. Furthermore, this illustrates why using more iterations at test time with momentum and backtracking are necessary to find adversarial examples as the objective becomes more complex compared to AT.

Confidence Along Interpolation: In Figure 7.2, on MNIST, we additionally illustrate the advantage of CCAT with respect to the toy example in Proposition 2. Here, we consider the case where the ϵ-balls of two training or test examples (in different classes) overlap. As we show in Proposition 2, adversarial training is not able to handle such cases, resulting in the trade-off between accuracy in robustness reported in the literature [TSE+19, SHS19, RXY+19, ZYJ+19]. This is because adversarial training enforces high-confidence predictions on both ϵ-balls (corresponding to different classes), resulting in an obvious conflict. CCAT, in contrast, enforces uniform predictions throughout the largest parts of both ϵ-balls, resolving the conflict.

7.5 Conclusion

Adversarial training results in robust models against the threat model *seen* during training, e.g., L_∞ adversarial examples. However, generalization to *unseen* attacks such as other L_p adversarial examples or larger L_∞ perturbations is insufficient. We propose **confidence-calibrated adversarial training (CCAT)** which biases the model towards low confidence predictions on adversarial examples and beyond. Then, adversarial examples can easily be rejected based on their confidence. Trained exclusively on L_∞ adversarial examples, CCAT improves robustness against unseen threat models such as larger L_∞, L_2, L_1 and L_0 adversarial examples, adversarial frames, distal adversarial examples and corrupted examples. Additionally, accuracy is improved in comparison to adversarial training. We thoroughly evaluated CCAT using 7 different white-and black-box attacks with up to 50 random restarts and 5000 iterations. These attacks where adapted to CCAT by directly maximizing confidence. We reported worst-case robust test error, extended to our confidence-thresholded setting, across *all* attacks.

7.5.1 Discussion of Recent Results

Since the publication of our work [SHS20], CCAT has also been evaluated using novel attacks. Adaptive AutoAttack (AAA) [YBTV21] automatically searches for adaptive attacks, e.g., considering various attack types, hyperparameters and objectives. [Sch22], in contrast, adapts our PGD-Conf attack with an improved backtracking scheme. We briefly discuss results of both works, considering confidence-thresholded RErr against L_∞ adversarial examples:

[YBTV21] includes results for $\epsilon = 8/255 \approx 0.0314$, which is slightly larger than our $\epsilon = 0.03$ used during training. Compared to 68.4% RErr reported in Table 7.2, AAA obtains 60.46% using the "default" setting including 3 attacks. Searching for 8 attacks instead, RErr increases to 73.13%. We used the official code[6] to rerun this evaluation for $\epsilon = 0.03$ and obtain 65.6% RErr. Surprisingly, this is lower than our reported 68.4%, demonstrating that our evaluation in Section 7.1 is stronger than standard attacks. Re-training CCAT using a WRN-28-10 [ZK16] further reduces RErr to 54.5%. While [YBTV21] reports a runtime of 205 minutes against our ResNet-20, the search actually requires 659 minutes. On a NVIDIA™ Tesla® P40, considering the WRN-28-10, attack and search time exceed 1000 and 2500 minutes, respectively.

More recently, [Sch22] adapts our PGD-Conf using the Armijo-rule for backtracking [Arm66], requiring 9 additional forward passes per iteration. More importantly, [Sch22] evaluates 100 restarts for standard PGD-CE, our PGD-Conf and the proposed "Backtracking-PGD-Conf". All attacks are run with $T = 1000$ iterations and increase RErr significantly to 92.5%, suggesting very poor robustness. Interestingly, Backtracking-PGD-Conf with only $T = 100$ iterations obtains only 59.9% RErr, illustrating that 100 restarts with $T = 1000$ iterations *each* are necessary. While we did not reproduce these results with the more robust WRN-28-10, [Sch22] reports a considerable computational overhead for our ResNet-20: 650 minutes for PGD-CE and up to 2077 minutes for Backtracking-PGD-Conf.

Overall, these works emphasize the difficulty and computational complexity of attacking CCAT, as already highlighted in Section 7.4.3. While AAA [YBTV21] does *not* increase RErr, [Sch22] suggests that CCAT is significantly less robust under an extreme evaluation protocol with 100 restarts, 1000 iterations each. Both works, however, demonstrate that CCAT is not only difficult to attack, requiring more sophisticated attacks (e.g., more search time), but effective attacks are computationally expensive which can be sufficient in many practical applications.

[6] https://github.com/eth-sri/adaptive-auto-attack

8 Learning Optimal Conformal Classifiers

Contents

In this chapter, we switch focus from adversarial robustness to uncertainty estimation in general. But instead of quantifying uncertainty using the predicted confidence as proposed in Chapter 7, we allow the model to predict *sets* of classes instead. The number of predicted classes is an intuitive estimate of uncertainty and making sure that the true class is included with high probability leads to a statistical framework known as *conformal prediction* (CP).

The high accuracy on test data obtained using deep neural networks does generally *not* provide sufficient guarantees for safe deployment, especially in high-stake applications such as autonomous driving or medical diagnosis. CP addresses these issues by using the classifier's probability estimates to predict *confidence sets* containing the true class with a user-specified probability. However, using CP as a separate processing step after training prevents the underlying model from adapting to the prediction of confidence sets. Thus, we explore to differentiate through CP *during training* with the goal of training model with the conformal wrapper *end-to-end*. In our approach, **conformal training (ConfTr)**, we specifically "simulate" conformalization on mini-batches during training. We show that ConfTr outperforms state-of-the-art CP methods for classification by reducing the average confidence set size, called *inefficiency*. Moreover, it allows "shaping" the confidence sets predicted at test time, which is difficult for standard CP. On experiments with several datasets, we show ConfTr can influence how inefficiency is distributed across classes, or guide the composition of confidence sets in terms of the included classes, while retaining the guarantees offered by CP.

This chapter is based on [SDCD21]: As first author, David Stutz conducted all included experiments and was the main writer of the paper. The work was conducted while interning at DeepMind and presented during an invited talk at the International Seminar on Distribution-Free Statistics at UC Berkeley.

8.1 Introduction

In classification tasks, for input x, we approximate the posterior distribution over classes $y \in [K] := \{1, \dots, K\}$, denoted $\pi_y(x) \approx p(Y = y|X = x)$. Following Bayes' decision rule, the *single* class with the highest posterior probability is predicted. This way, deep neural networks $\pi_{\theta,y}(x)$ with parameters θ achieve impressive accuracy on held-out test sets. However, this does not *guarantee* safe deployment. *Conformal prediction (CP)* [VGS05] uses a post-training calibration step to *guarantee* a user-specified *coverage*: by allowing to predict confidence sets $C(X) \subseteq [K]$, CP guarantees the true class Y to be included with confidence level α, i.e. $p(Y \in C(X)) \geq 1 - \alpha$ when the calibration examples (X_i, Y_i), $i \in I_{\text{cal}}$ are drawn exchangeably from the test distribution. This is usually achieved in two steps: In the *prediction step*, so-called *conformity scores* (w.r.t. to a class $k \in [K]$) are computed to construct the confidence sets $C(X)$. During the *calibration step*, these conformity scores on the calibration set w.r.t. the true class Y_i are ranked to determine a cut-off threshold τ for the predicted probabilities $\pi_\theta(x)$ guaranteeing coverage $1 - \alpha$. This is called *marginal* coverage as it holds only unconditionally, i.e., the expectation is being taken not only w.r.t. (X, Y) but also over the distribution of all possible calibration sets, rather than w.r.t. the conditional distribution $p(Y|X)$.

CP not only provides marginal coverage, but it also outputs intuitive uncertainty estimates: larger confidence sets $|C(X)|$ generally convey higher uncertainty. Although CP is agnostic to details of the underlying model $\pi_\theta(x)$, the obtained uncertainty estimates depend strongly on the model's performance. If the underlying classifier is poor, CP results in too large and thus uninformative confidence sets. "Uneven" coverage is also a common issue, where lower coverage is achieved on more difficult classes. To address such problems, the threshold CP method of [SLW19] explicitly minimizes the expected confidence set size, referred to as *inefficiency*. [RSC20] and [CGD21] propose methods that perform favorably in terms of (approximate) conditional coverage. The *adaptive prediction sets (*APS*)* method of [RSC20] is further extended by [ABJM21] to return smaller confidence sets. These various objectives are typically achieved by changing the definition of the conformity scores, see Section 8.2 for some examples. In all cases, CP is used as a post-training calibration step. In contrast, our work does *not* focus on advancing CP itself, e.g., through new conformity scores, but develops a novel training procedure for the classifier π_θ.

Indeed, while the flexibility of CP regarding the underlying model appears attractive, it is also a severe limitation: Learning the model parameters θ is *not* informed about the post-hoc "conformalization", i.e., they are not tuned towards any specific objective such as reducing inefficiency. During training, the model will typically be trained to minimize cross-entropy loss. At test time, in contrast, it is used to obtain a set predictor $C(X)$ with specific properties such as low inefficiency. In concurrent work, [Bel21] addresses this issue by learning a set predictor $C(X)$ through thresholding logits: Classes with logits exceeding 1 are included in $C(X)$ and training aims to minimize inefficiency while targeting coverage $1 - \alpha$. In experiments using linear models only, this approach is shown to decrease inefficiency. However, [Bel21] ignores the crucial calibration step of CP during training and does *not* allow optimizing losses beyond marginal coverage or inefficiency. In contrast, our work subsumes [Bel21], but additionally considers the calibration step during training, which is crucial for further decreasing inefficiency. Furthermore, we aim to allow fine-grained control over class-conditional inefficiency or the composition of the confidence sets by allowing to optimize arbitrary losses defined on confidence sets.

Contributions: We propose **conformal training (ConfTr)**, a procedure allowing to train model and conformal wrapper *end-to-end*. This is achieved by developing smooth implementations

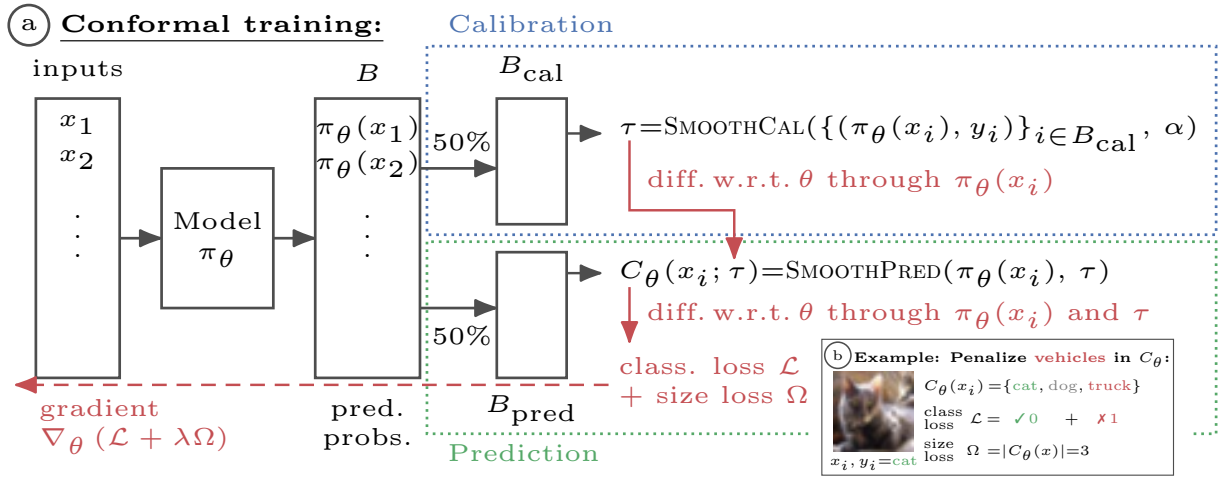


Figure 8.1: **Illustration of *conformal training (ConfTr)*:** *Conformal prediction (CP)* wraps any classifier $\pi_\theta(x)$ with parameters θ and constructs a confidence set C_θ with coverage guarantees, see text. We develop differentiable prediction and calibration implementations, SmoothCal and SmoothPred. During training, this allows ConfTr to "simulate" CP on each mini-batch B by calibrating on the first half B_{cal} and predicting confidence sets on the other half B_{pred} (c.f. ⓐ). This allows optimizing arbitrary losses on the predicted confidence sets, e.g., reducing average confidence set size (*inefficiency*) using a size loss Ω or penalizing specific classes from being included using a classification loss $\mathcal{L}$ (c.f. ⓑ).

of state-of-the-art CP methods. Specifically, on each mini-batch, ConfTr "simulates" conformalization, using half of the batch for calibration, and the other half for computing a loss on the predicted confidence sets, c.f. Figure 8.1 ⓐ. In experiments, ConfTr consistently reduces the inefficiency of recent CP methods, such as *threshold CP (*Thr*)* [SLW19] and APS [RSC20]. We further improve inefficiency over [Bel21], illustrating the importance of considering the calibration step during training. Using carefully constructed losses, ConfTr also allows us to "shape" the confidence sets obtained at test time: We can reduce *class-conditional* inefficiency or "coverage confusion", i.e., the likelihood of two or more classes being included in the same confidence sets, c.f. Figure 8.1 ⓑ. Generally, in contrast to [Bel21], ConfTr allows optimizing arbitrary losses on the predicted confidence sets. Because ConfTr is agnostic to the CP method used at test time, our work is complementary to most related work such that *any* advancement in terms of CP, e.g., improved conformity scores, are directly applicable to ConfTr, as well. Most importantly, ConfTr preserves the coverage guarantee obtained through CP.

8.2 Differentiable Conformal Predictors

We are interested in training the model π_θ end-to-end with the conformal wrapper in order to allow fine-grained control over the confidence sets $C(X)$. Before developing differentiable CP methods (Section 8.2.2), we review two conformal predictors for classification recently introduced in [SLW19] and [RSC20] (Section 8.2.1). These CP methods consist of two steps: for *prediction* (on the test set) we need to define the confidence sets $C_\theta(X; \tau)$ which depend on the model parameters θ through the predictions π_θ and where the threshold τ is determined during *calibration* on a held-out calibration set $(X_i, Y_i), i \in I_{\text{cal}}$ in order to obtain coverage.

CP Baseline Comparison by Ineff				
Dataset, α	ThrL	Thr	APS	RAPS
CIFAR10, 0.05	2.22	**1.64**	2.06	1.74
CIFAR10, 0.01	3.92	**2.93**	3.30	3.06
CIFAR100, 0.01	19.22	**10.63**	16.62	14.25

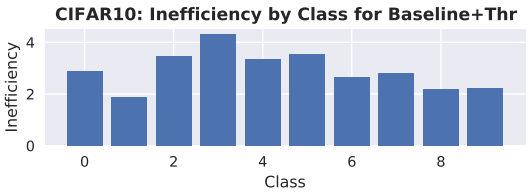


Figure 8.2: **Baseline CP Results on CIFAR:** *Left:* Inefficiency (Ineff, lower is better) for the CP methods discussed in Section 8.2. Coverage (Cover), omitted here, is empirically close to $1-\alpha$. Thr clearly outperforms all approaches w.r.t. inefficiency. *Right:* Inefficiency distribution across CIFAR10 classes (for α=0.01) is plotted, with more difficult classes yielding higher inefficiency.

8.2.1 Conformal Predictors

In this section, we detail two recent and popular CP methods and also discuss appropriate evaluation metrics as well as the provided coverage provided by CP in formal terms:

The **Threshold Conformal Predictor (Thr)** [SLW19] constructs the confidence sets by thresholding probabilities: $C_\theta(x;\tau) := \{k : \pi_{\theta,k}(x) =: E_\theta(x,k) \geq \tau\}$. Here, the subscript C_θ makes the dependence on the model π_θ and its parameters θ explicit. During calibration, τ is computed as the $\alpha(1+1/|I_{\text{cal}}|)$-quantile of the so-called conformity scores $E_\theta(x_i,y_i) = \pi_{\theta,y_i}(x_i)$. The conformity scores indicate, for each example, the threshold that ensures coverage. Marginal coverage of $(1-\alpha)$ is guaranteed on test examples. Thr is summarized in Algorithm 4 (left) and can also be applied on logits (ThrL) or log-probabilities (ThrLP) instead of probabilities.

Adaptive Prediction Sets (APS) [RSC20] constructs confidence sets based on the ordered probabilities. Specifically, $C_\theta(x;\tau) := \{k : E_\theta(x,k) \leq \tau\}$ with:

$$E_\theta(x,k) := \pi_{\theta,y^{(1)}}(x) + \ldots + \pi_{\theta,y^{(k-1)}}(x) + U\pi_{\theta,y^{(k)}}(x), \tag{8.1}$$

where $\pi_{\theta,y^{(1)}}(x) \geq \ldots \geq \pi_{\theta,y^{(K)}}(x)$ are the sorted probabilities and U is a uniform random variable in $[0,1]$ to break ties. Similar to Thr, the conformity scores $E_\theta(x_i,y_i)$ w.r.t. the true classes y_i are used for calibration, but the $(1-\alpha)(1+1/|I_{\text{cal}}|)$-quantile is required to ensure marginal coverage.

Evaluation Metrics: Performance of CP is then measured using two metrics: empirical and marginal **coverage (Cover)** as well as **inefficiency (Ineff)**. Letting I_{test} be a test set of size $|I_{\text{test}}|$, these metrics are computed as

$$\text{Cover} := \frac{1}{|I_{\text{test}}|}\sum_{i\in I_{\text{test}}} \delta[y_i \in C(x_i)] \quad \text{and} \quad \text{Ineff} := \frac{1}{|I_{\text{test}}|}\sum_{i\in I_{\text{test}}} |C(x_i)|, \tag{8.2}$$

where δ denotes the indicator function that is 1 when its argument is true and 0 otherwise. Due to the marginal coverage guarantee provided by CP, the empirical coverage, when averaged across many random calibration/test splits, is Cover $\approx 1-\alpha$. Thus, evaluation concentrates on inefficiency as the main metric to compare across CP methods and models where lower is generally better. Note that with *accuracy* we refer to the (top-1) accuracy with respect to the arg max-predictions, i.e., $\arg\max_k \pi_{\theta,k}(x)$, obtained by the underlying model π. As shown in Figure 8.2 (left), Thr clearly outperforms ThrL, APS and RAPS w.r.t. inefficiency, also averaged across random $I_{\text{cal}}/I_{\text{test}}$ splits.

Coverage Guarantee: We follow [RSC20] and briefly state the marginal coverage guarantee obtained by these CP methods in formal terms: Given that the learning algorithm used is

1: PREDICT($\pi_\theta(x)$, τ):
2: compute $E_\theta(x,k)$, $k\in[K]$
3: **return** $C_\theta(x;\tau) = \{k : E_\theta(x,k) \geq \tau\}$

1: SMOOTHPRED($\pi_\theta(x)$, τ, T=1):
2: compute $E_\theta(x,k)$, $k\in[K]$
3: **return** $C_{\theta,k}(x;\tau) = \sigma(^{(E_\theta(x,k)-\tau)}/_T)$, $k \in [K]$

1: CALIBRATE($\{(\pi_\theta(x_i), y_i\}_{i=1}^n$, α):
2: compute $E_\theta(x_i, y_i)$, i=1, . . . , n
3: **return** QUANTILE($\{E_\theta(x_i, y_i)\}$, $\alpha(1 + {}^1/_n)$)

1: SMOOTHCAL($\{(\pi_\theta(x_i), y_i\}_{i=1}^n$, α):
2: compute $E_\theta(x_i, y_i)$, i=1, . . . , n
3: **return** SMOOTHQTL($\{E_\theta(x_i, y_i)\}$, $\alpha(1+{}^1/_n)$)

Algorithm 4: **Smooth CP:** ***Left:*** PREDICT computes the conformity scores $E_\theta(x,k)$ for each $k\in[K]$ and constructs the confidence sets $C_\theta(x;\tau)$ by thresholding with τ. CALIBRATE determines the threshold τ as the $\alpha(1 + {}^1/_n)$-quantile of the conformity scores w.r.t. the true classes y_i on a calibration set $\{(x_i, y_i)\}$ of size $n{:=}|I_{\text{cal}}|$. THR and APS use different conformity scores. ***Right:*** SMOOTHPRED uses a sigmoid function σ for soft-thresholding to construct confidence sets. As detailed in Section 8.2.2, these can be understood as *soft* assignments. Smooth calibration, in SMOOTHCAL, essentially replaces the quantile computation using a smooth variant, denoted as SMOOTHQTL, based on smooth sorting [CTV19, BTBD20, Wil20].

invariant to permutations of the training examples, and the calibration examples $\{(X_i, Y_i)\}_{i\in I_{\text{cal}}}$ are exchangeably drawn from the same distribution as encountered at test time, the discussed CP approaches satisfy

$$p(Y \in C(X)) \geq 1 - \alpha \tag{8.3}$$

Moreover, this bound is near tight if the conformity scores E are almost surely distinct:

$$p(Y \in C(X)) \leq 1 - \alpha + \frac{1}{|I_{\text{cal}}| + 1}. \tag{8.4}$$

Unfortunately, there is generally no guarantee on conditional coverage, as this requires additional assumptions [RSC20]. However, class-conditional coverage can be obtained using THR [SLW19], which has also been shown to be the most efficient conformal predictor given a fixed model π_θ (i.e., minimizes inefficiency).

CP is intended to be used as a "wrapper" around the model π_θ. "Better" CP methods generally result in lower inefficiency for a *fixed* model π_θ. For example, following Figure 8.2 (left), regularized APS (RAPS) [ABJM21] recently showed how to improve inefficiency compared to APS by modifying the conformity score – without outperforming THR, however. Fine-grained control over inefficiency, e.g., conditioned on the class or the composition of the $C(X)$ is generally not possible. Integrating CP into the training procedure promises a higher degree of control, however, requires differentiable CP implementations.

8.2.2 Differentiable Prediction and Calibration Steps

Differentiating through CP involves differentiable prediction and calibration steps: We want $C_\theta(x;\tau)$ to be differentiable w.r.t. the predictions $\pi_\theta(x)$, and τ to be differentiable w.r.t. to the predictions $\pi_\theta(x_i)$, $i \in I_{\text{cal}}$ used for calibration. The latter involves differentiating through the quantile computation of Algorithm 4 (left). We emphasize that, ultimately, this allows us to differentiate through both calibration and prediction w.r.t. the *model parameters* θ, on which the predictions $\pi_\theta(x)$ and thus the conformity scores $E_\theta(x,k)$ depend. For THR, our smooth version is summarized in Algorithm 4 (right).

Prediction: For both THR and APS, prediction involves thresholding the conformity scores $E_\theta(x, k)$, which can be smoothed using the sigmoid function σ and a temperature parameter T:

$$C_{\theta,k}(x;\tau) := \sigma\left(^{(E_\theta(x,k)-\tau)}/_{T}\right) \quad \text{with} \quad \sigma(z) = {}^{1}/_{1+\exp(-z)}. \tag{8.5}$$

Essentially, $C_{\theta,k}(x;\tau) \in [0,1]$ represents a *soft* assignment of class k to the confidence set, i.e., can be interpreted as the probability of k being included. For $T \to 0$, the "hard" confidence set will be recovered, i.e., $C_{\theta,k}(x;\tau) = 1$ for $k \in C_\theta(x;\tau)$ and 0 otherwise. APS additionally involves sorting the probabilities, i.e., obtaining $\pi_{\theta,y^{(1)}}(x) \geq \ldots \geq \pi_{\theta,y^{(K)}}(x)$, see Equation (8.1). Here, any smooth sorting approach [CTV19, BTBD20, Wil20] can be used. These often come with a "dispersion" hyperparameter ϵ such that smooth sorting approximates "hard" sorting for $\epsilon \to 0$. The resulting $C_{\theta,k}(x;\tau)$ is differentiable w.r.t. the model predictions $\pi_\theta(x)$ and consequently w.r.t. its parameters θ.

Calibration: For calibration, two sub-tasks need to be addressed: a smooth way to compute the conformity score and a differentiable implementation to calculate quantiles. The former directly follows from the discussion above: for THR, the probabilities $\pi_\theta(x)$ are trivially differentiable, and for APS, the smooth conformity scores of Equation (8.1) are computed as detailed above in the prediction step. The latter, i.e., a differentiable quantile computation, can again be accomplished using smooth sorting. Overall, this results in the threshold τ being differentiable w.r.t. the predictions on the calibration examples $\{(\pi_\theta(x_i), y_i\}_{i \in I_{\text{cal}}}$ and the model's parameters θ as required.

As this approximation is using smooth operations, the coverage guarantee seems lost. However, in the limit of $T, \epsilon \to 0$ we recover the original non-smooth computations and the corresponding coverage guarantee. Thus, it is reasonable to assume that, in practice, we *empirically* obtain coverage close to $(1 - \alpha)$. We found that this is sufficient because these smooth variants are *only* used during training. At test time, we use the original (non-smooth) implementations and the coverage guarantee follows directly from [SLW19, RSC20].

8.3 CONFORMAL TRAINING: LEARNING CONFORMAL PREDICTION

The key idea of **conformal training (ConfTr)** is to "simulate" CP during training, i.e., performing both calibration and prediction steps on each mini-batch. This is accomplished using the differentiable versions of THR or APS introduced in Section 8.2.2. ConfTr can be viewed as a generalization of [Bel21] that just differentiates through the prediction step with a fixed threshold, without considering the crucial calibration step, as we discuss later in Section 8.3.4. ConfTr is summarized in detail in Algorithm 5.

8.3.1 Conformal Training by Optimizing Inefficiency

ConfTr performs (differentiable) CP on each mini-batch during stochastic gradient descent (SGD) training. In particular, as illustrated in Figure 8.1 ⓐ, we split each mini-batch B in half: the first half is used for calibration, B_{cal}, and the second one for prediction and loss computation, B_{pred}. That is, on B_{cal}, we calibrate τ by computing the $\alpha(1 + {}^{1}/_{|B_{\text{cal}}|})$-quantile of the conformity scores in a differentiable manner. It is important to note that we compute $C_\theta(x_i;\tau)$ only for $i \in B_{\text{pred}}$ and *not* for $i \in B_{\text{cal}}$. Then, in expectation across mini-batches and large enough $|B_{\text{cal}}|$, CP guarantees coverage $1 - \alpha$ on B_{pred} for $T, \epsilon \to 0$. In practice, assuming empirical coverage close to $1 - \alpha$, irrespective of the performance of the model π_θ, we only

ConformalTraining(α, λ=1):
for mini-batch B **do**
  randomly split batch $B_{\text{cal}} \uplus B_{\text{pred}} = B$
  {"On-the-fly" calibration on B_{cal}:}
  $\tau = \text{SmoothCal}(\{(\pi_\theta(x_i), y_i)\}_{i \in B_{\text{cal}}}, \alpha)$
  {Prediction only on $i \in B_{\text{pred}}$:}
  $C_\theta(x_i; \tau) = \text{SmoothPred}(\pi_\theta(x_i), \tau)$
  {*Optional* classification loss:}
  $\mathcal{L}_B = 0$ or $\sum_{i \in B_{\text{pred}}} \mathcal{L}(C_\theta(x_i; \tau), y_i)$
  $\Omega_B = \sum_{i \in B_{\text{pred}}} \Omega(C_\theta(x_i; \tau))$
  $\Delta = \nabla_\theta 1/|B_{\text{pred}}|(\mathcal{L}_B + \lambda \Omega_B)$
  update parameters θ using Δ
end for

Algorithm 5: **Conformal Training (ConfTr)** calibrates on a part of each mini-batch, B_{cal}, in each training iteration. Thereby, we obtain coverage $1 - \alpha$ on the other part, B_{pred}, at least in expectation across many batches. Then, we intend to minimize inefficiency using a smooth *size loss* Ω on the other part of the mini-batch, B_{pred}, in order to obtain gradients to update the model's parameters θ. Optionally, an additional classification loss $\mathcal{L}$ can be used to "shape" the obtained confidence sets. Smooth implementations of calibration and prediction are used as detailed in Algorithm 4.

need to minimize inefficiency:

$$\min_\theta \log \mathbb{E}\left[\Omega(C_\theta(X; \tau))\right] \quad \text{with } \Omega(C_\theta(x; \tau)) = \max\left(0, \sum_{k=1}^{K} C_{\theta,k}(x; \tau) - \kappa\right). \tag{8.6}$$

We emphasize that ConfTr optimizes the model parameters θ on which the confidence sets C_θ depend through the model predictions π_θ. Here, Ω is a "smooth" *size loss* intended to minimize the expected inefficiency, i.e., $\mathbb{E}[|C_\theta(X; \tau)|]$, not to be confused with the statistic in Equation (8.2) used for evaluation. Remember that $C_{\pi,k}(x; \tau)$ can be understood as a soft assignment of class k to the confidence set $C_\theta(x; \tau)$. By default, we use $\kappa = 1$ in order to not penalize singletons. However, $\kappa \in \{0, 1\}$ can generally be treated as hyperparameter. After training, any CP method can be applied to calibrate τ on a held-out calibration set I_{cal} as usual, i.e., the thresholds τ obtained during training are *not* kept for testing. This preserves the coverage guarantee of CP.

8.3.2 Conformal Training with Classification Loss

In order to obtain more control over the composition of confidence sets $C_\theta(X; \tau)$ at test time, ConfTr can be complemented using a generic loss $\mathcal{L}$:

$$\min_\theta \log\left(\mathbb{E}\left[\mathcal{L}(C_\theta(X; \tau), Y) + \lambda \Omega(C_\theta(X; \tau))\right]\right). \tag{8.7}$$

While $\mathcal{L}$ can be any arbitrary loss defined directly on the confidence sets C_θ, we propose to use a "configurable" *classification loss* $\mathcal{L}_{\text{class}}$. This classification loss is intended to explicitly enforce coverage, i.e., make sure the true label Y is included in $C_\theta(X; \tau)$, and optionally penalize other classes k *not* to be included in C_θ, as illustrated in Figure 8.1 ⓑ. To this end, we define

$$\mathcal{L}_{\text{class}}(C_\theta(x; \tau), y) := \sum_{k=1}^{K} L_{y,k}\Big[\underbrace{C_{\theta,k}(x; \tau) \cdot \delta[y = k]}_{\text{enforce } y \text{ to be in } C} + \underbrace{(1 - C_{\theta,k}(x; \tau)) \cdot \delta[y \neq k]}_{\text{penalize class } k \neq y \textit{ not} \text{ to be in } C}\Big]. \tag{8.8}$$

As above, $C_{\theta,k}(x; \tau) \in [0, 1]$ such that $1 - C_\theta(x; \tau)$ can be understood as the likelihood of k not being in $C_\theta(x; \tau)$. In Equation (8.8), the first term is used to encourage coverage, while the

second term can be used to avoid predicting other classes. This is governed by the *loss matrix* L: For $L = I_K$, i.e., the identity matrix with K rows and columns, this loss simply enforces coverage (perfect coverage if $\mathcal{L}_{\text{class}} = 0$). However, setting any $L_{y,k} > 0$ for $y \neq k$ penalizes the model from including class k in confidence sets with ground truth y. Thus, cleverly defining L allows defining rather complex objectives, as we will explore next. ConfTr with (optional) classification loss is summarized in Algorithm 5.

8.3.3 Conformal Training with General and Application-Specific Losses

We consider several use cases motivated by medical diagnosis, e.g., breast cancer screening [MSG+20] or classification of dermatological conditions [LJE+20, RRA+22, JWG+21]. In skin condition classification, for example, predicting sets of classes, e.g., the top-k conditions, is already a common strategy for handling uncertainty. In these cases, we not only care about coverage guarantees but also desirable characteristics of the confidence sets. These constraints in terms of the predicted confidence sets can, however, be rather complicated and pose difficulties for standard CP. We explore several exemplary use cases to demonstrate the applicability of ConfTr, that are also relevant beyond the considered use cases in medical diagnosis.

First, we consider "shaping" class-conditional inefficiency, formally defined as

$$\text{Ineff}[Y = y] := \frac{1}{\sum_{i \in I_{\text{test}}} \delta[y_i = y]} \sum_{i \in I_{\text{test}}} \delta[y_i = y] |C(x_i)|. \tag{8.9}$$

Similarly, we can define inefficiency conditional on a *group* of classes. For example, we could reduce inefficiency on "low-risk" diseases at the expense of higher uncertainty on "high-risk" conditions. This can be thought of as re-allocating time spent by a doctor towards high-risk cases. Using ConfTr, we can manipulate group- or class-conditional inefficiency using a weighted size loss $\omega \cdot \Omega(C(X; \tau))$ with $\omega := \omega(Y)$ depending on the ground truth Y in Equation (8.6).

Next, we consider *which* classes are actually included in the confidence sets. CP itself does not enforce any constraints on the composition of the confidence sets. However, with ConfTr, we can penalize the "confusion" between pairs of classes: for example if two diseases are frequently confused by doctors, it makes sense to train models that avoid confidence sets that contain *both* diseases. To control such cases, we define the *coverage confusion matrix* as

$$\Sigma_{y,k} := \frac{1}{|I_{\text{test}}|} \sum_{i \in I_{\text{test}}} \delta[y_i = y \wedge k \in C(x_i)]. \tag{8.10}$$

The off-diagonals, i.e., $\Sigma_{y,k}$ for $y \neq k$, quantify how often class k is included in confidence sets with true class y. Reducing $\Sigma_{y,k}$ can be accomplished using a positive entry $L_{y,k} > 0$ in the classification loss of Equation (8.8).

Finally, we explicitly want to penalize "overlap" between groups of classes in confidence sets. For example, we may *not* want to concurrently include very high-risk conditions among low-risk ones in confidence sets, to avoid unwanted anxiety or tests for the patient. Letting $K_0 \uplus K_1$ be two disjoint sets of classes, we define *miscoverage* as

$$\text{MisCover}_{0 \to 1} = \frac{1}{\sum_{i \in I_{\text{test}}} \delta[y_i \in K_0]} \sum_{i \in I_{\text{test}}} \delta[y_i \in K_0 \wedge (\exists k \in K_1 : k \in C(x_i))]. \tag{8.11}$$

Reducing $\text{MisCover}_{0 \to 1}$ means avoiding classes K_1 being included in confidence sets of classes K_0. Again, we use $L_{y,k} > 0$ for $y \in K_0, k \in K_1$ to approach this problem. $\text{MisCover}_{1 \to 0}$ is

1: CoverageTraining(τ, λ) {*fixed* threshold}
2: **for** mini-batch B **do**
3: $C(x_i;\tau) :=$ SmoothPred($\pi_\theta(x_i)$, τ), $i \in B$
4: $\mathcal{L}_B := \sum_{i \in B} \mathcal{L}(C_\theta(x_i;\tau), y_i)$ {*not* optional}
5: $\Omega_B := \sum_{i \in B} \Omega(C_\theta(x_i;\tau))$
6: $\Delta := \nabla_\theta 1/|B|(\mathcal{L}_B + \lambda\Omega_B)$
7: update parameters θ using Δ
8: **end for**

Algorithm 6: **Coverage Training (CoverTr)**: As baseline, and compared to ConfTr in Algorithm 5, CoverTr simplifies training by not differentiating through the calibration step and avoiding splitting the batch B. However, fixing the threshold τ is problematic and training requires carefully balancing coverage and size loss, $\mathcal{L}$ and Ω.

defined analogously and measures the opposite, i.e., classes K_0 being included in confidence sets of K_1.

8.3.4 Conformal Training as Generalization of Coverage Training

As intermediate step towards ConfTr, we can also ignore the calibration step and just differentiate through the prediction step. This leads to a special case of ConfTr that we call *coverage training (CoverTr)* and subsumes concurrent work in [Bel21]. In the following, we discuss CoverTr as well as [Bel21] in detail and highlight several limitations and challenges that our ConfTr addresses.

Just differentiating through the prediction step, i.e., $C_\theta(X;\tau)$, without calibration can be accomplished by fixing the threshold τ. Then, π_θ essentially learns to produce probabilities that yield "good" confidence sets $C_\theta(X;\tau)$ for the chosen threshold τ. Following Algorithm 6, CoverTr computes $C_\theta(X;\tau)$ on each mini-batch using a fixed τ and smooth thresholding. The model's parameters θ are obtained by solving

$$\min_\theta \log\left(\mathbb{E}\left[\mathcal{L}(C_\theta(X;\tau), Y) + \lambda\Omega(C_\theta(X;\tau))\right]\right). \tag{8.12}$$

Here, $\mathcal{L}$ is the classification loss from Equation (8.8) and Ω the size loss from Equation (8.6). The classification loss has to ensure that the true label y is in the predicted confidence set $C_\theta(X;\tau)$ as the calibration step is missing. That is, in stark contrast to ConfTr, CoverTr strictly requires both classification and size loss during training. This is because using a fixed threshold τ yields trivial solutions for both classification and size loss when used in isolation: $\mathcal{L}$ is minimized for $C_\theta(X;\tau) = [K]$, while Ω is minimized for $C_\theta(X;\tau) = \emptyset$. Thus, balancing both terms in Equation (8.12) using λ is crucial during training. As with ConfTr, the threshold τ is re-calibrated at test time to obtain a coverage guarantee. Choosing τ for training, in contrast, can be difficult: First, τ will likely evolve during training as π_θ should get more and more accurate. Second, the general ballpark of reasonable thresholds τ depends on the dataset, model as well as CP method and is difficult to fix in advance.

In concurrent work [Bel21], referred to as Bel, the problem with fixing a threshold τ is partly circumvented by using ThrL during training, i.e., Thr on logits. As the logits are unbounded, the threshold can be chosen arbitrarily, e.g., $\tau = 1$. As Bel also follows the formulation of Equation (8.12), the approach can be seen as a special case of CoverTr. However, a less flexible *coverage loss* is used during training: Instead of $\mathcal{L}_{\text{class}}$, the loss is meant to enforce a specific coverage level $(1-\alpha)$ on each mini-batch. This is done using a squared loss on coverage:

$$\mathcal{L}_{\text{cov}} := \left[\left(\frac{1}{|B|}\sum_{i \in B} C_{\theta,y_i}(x_i;\tau)\right) - (1-\alpha)\right]^2 \tag{8.13}$$

for a mini-batch B of examples. In contrast to Equation (8.12), $\mathcal{L}_{\text{cov}}$ is applied per batch and not per example. For the size loss, [Bel21] uses $\kappa = 0$ in Equation (8.6). Besides *not* providing much control over the confidence sets, $\mathcal{L}_{\text{cov}}$ also encourages coverage $(1 - \alpha)$ instead of perfect coverage. Nevertheless, this approach is shown to improve inefficiency of THRL on various UCI datasets [DG17] using linear logistic regression models. We will show that this also generalizes – to some extent – to non-linear models and more complex datasets. Nevertheless, Bel is restricted to THRL which is outperformed significantly by both THR and APS as demonstrated in Figure 8.2. Thus, we will show that ConfTr consistently outperforms Bel in terms of inefficiency. Finally, the approach *cannot* be used for any of the studied use cases in Section 8.3.3.

Using CoverTr with THR and APS remains problematic. While we found $\tau \in [0.9, 0.99]$ for THR or $[-0.1, -0.01]$ for THRLP to work reasonably on some datasets, we generally had difficulties finding a fixed threshold τ that makes CoverTr easily trainable across various datasets. Moreover, as CoverTr requires balancing coverage $\mathcal{L}$ and size loss Ω, hyperparameter optimization is more complex compared to ConfTr. By extension, these problems also limit the applicability of Bel. Thus, we would ideally want to re-calibrate the threshold τ regularly, e.g., after each epoch or model update. Doing calibration on a larger, held-out calibration set, however, wastes valuable training examples and compute resources. Besides, re-calibration alone does not provide any useful information for model updates. Thus, ConfTr directly calibrates on each mini-batch and also differentiates through the calibration step itself to obtain meaningful gradients.

8.4 EXPERIMENTS

We present experiments in two parts: After introducing our experimental setup in Section 8.4.1, we first demonstrate that ConfTr can reduce inefficiency of THR and APS compared to CP applied to a baseline model trained using cross-entropy loss separately, see Section 8.4.2. The main results for this part can be found in Table 8.1, showing that we consistently outperform concurrent work of [Bel21]. Second, in Section 8.4.3, we show how ConfTr can be used to "shape" confidence sets, i.e., reduce class-conditional inefficiency for specific (groups of) classes or coverage confusion of two or more classes, while maintaining the marginal coverage guarantee. This is impossible using [Bel21] and rather difficult for standard CP.

8.4.1 Experimental Setup

Datasets and Splits: We consider MNIST [LBBH98], EMNIST [CATvS17], Fashion-MNIST [CATvS17], CIFAR [Kri09] as well as WineQuality [CCA+09] with a fixed split of training, calibration and test examples. For datasets providing a default training/test split, we take the last 10% of training examples as calibration set. For EMNIST, we consider a subset of the "byClass" split that contains $52 = 2 \cdot 26$ classes comprised of all lower and upper case letters. We take the first 122.8k examples split into 98.8k/5.2k/18.8k training/calibration/test examples. Finally, On WineQuality, we manually created training/calibration/test splits, roughly matching 70%/10%/20%. We use the "white wine" subset for WineQuality. In order to create a binary classification problem, wine with quality 6 or higher is categorized as "good wine" (class 1), and all other wine is categorized as "bad" (class 0) as done in [Bel21].

Models and Training: We consider a linear model on MNIST, 2-layer multi-layer perceptrons (MLPs) on WineQuality, EMNIST and Fashion-MNIST (256, 128, 64 hidden units, respectively)

Inefficiency ↓, ConfTr (trained w/ THRLP), $\alpha = 0.01$							
	THR				APS		
Dataset	Basel.	Bel	ConfTr	$+\mathcal{L}_{\text{class}}$	Basel.	ConfTr	$+\mathcal{L}_{\text{class}}$
MNIST	2.23	2.70	2.18	**2.11**	2.50	2.16	**2.14**
F-MNIST	2.05	1.90	1.69	**1.67**	2.36	1.82	**1.72**
EMNIST	2.66	3.48	2.66	**2.49**	4.23	**2.86**	2.87
CIFAR10	2.93	2.93	2.88	**2.84**	3.30	3.05	**2.93**
CIFAR100	10.63	10.91	10.78	**10.44**	16.62	12.99	**12.73**
WineQuality	1.76	1.77	1.75	**1.74**	1.79	1.82	**1.77**

Table 8.1: **Main Inefficiency Results**, comparing [Bel21] (Bel, trained with THRL) and ConfTr (trained with THRLP) using THR or APS at test time (with α=0.01). ConfTr results in a consistent improvement of inefficiency for both THR and APS. Training with $\mathcal{L}_{\text{class}}$ ($L = I_K$) generally works slightly better. On CIFAR, the inefficiency reduction is smaller compared to other datasets as ConfTr is trained on pre-trained ResNet features, and not trained from scratch, see text. On binary datasets such as WineQuality, THR and APS perform very similar and ConfTr is unable to improve significantly.

as well as ResNet-34/50 [HZRS16a] (4 and 64 base channels) on CIFAR10/100. In all cases, we use ReLU activations [NH10] and batch normalization [IS15b]. We train using stochastic gradient descent (SGD) with momentum 0.0005 and Nesterov gradients. The baseline models are trained with cross-entropy loss, while ConfTr follows Algorithm 5. Learning rate and batch size are optimized alongside the ConfTr hyperparameters using grid search. Except on CIFAR, see next paragraph, we do *not* use any data augmentation. Finally, we do *not* use Platt scaling [GPSW17] as used in [ABJM21].

Fine-Tuning on CIFAR: On CIFAR10 and CIFAR100, we train base ResNet-34/ResNet-50 models using AutoAugment [CZM+19] and Cutout [DT17] which are then fine-tuned using ConfTr. We only use 4 base channels for the ResNet-34 and 64 channels for the ResNet-50 as we focus on the results for CP at test time, without optimizing accuracy of the base model. For fine-tuning, the last layer is re-initialized and trained using the same data augmentation as applied for the base model, subject to the random training trials described below.

Random Training and Test Trials: For statistically meaningful results, we perform random *test* and *training* trials. Following common practice [ABJM21], we evaluate CP methods at test time using 10 random calibration/test splits. Metrics such as coverage and inefficiency are then empirically evaluated as the average across all test trials. Additionally, and in contrast to [Bel21], we consider random training trials: After hyperparameter optimization on all training examples, we train 10 models with the final hyperparameters on a new training set obtained by sampling the original one with up to 5 replacements. This means that we report, e.g., inefficiency as average over a total of $10 \cdot 10 = 100$ random training *and* test trials.

Appendix E.1 provides more details on the experimental setup, including key dataset statistics, the used hyperparameters on all datasets and how we perform random training trials when fine-tuning on CIFAR. The importance of performing random training *and* test trials is emphasized in Appendix E.2.

8.4.2 Reducing Inefficiency with Conformal Training

In the first part, we focus on the inefficiency reductions of ConfTr (with and without the classification loss $\mathcal{L}_{\text{class}}$ in Equation (8.8)) in comparison to the baseline and [Bel21] (Bel). After

MNIST: Ablation for CoverTr and ConfTr													
Method	Baseline			Bel		CoverTr				ConfTr			
Train				ThrL		Thr	APS	ThrLP		ThrLP	ThrLP	$+\mathcal{L}_{\text{class}}$	
Test	ThrL	Thr	APS	ThrL	Thr	Thr	APS	Thr	APS	Thr	APS	Thr	APS
Ineff	3.57	2.23	2.5	2.73	2.7	6.34	4.86	2.5	2.76	2.18	2.16	2.11	2.14
Acc	92.39	92.39	92.39	81.41	90.01	83.85	88.53	92.63	92.63	90.24	90.21	91.18	91.35

Table 8.2: **Ablation for CoverTr and ConfTr:** We report inefficiency and accuracy for Bel, CoverTr and ConfTr considering various CP methods for training and testing. Bel outperforms the baseline when using ThrL, but does not do so for Thr on MNIST. CoverTr with Thr or APS during training is challenging, resulting in high inefficiency (mainly due to large variation among training trials, c.f. Table E.3), justifying our choice of ThrLP for ConfTr. Also CoverTr is unable to improve over the Thr baseline.

summarizing the possible inefficiency reductions, we also discuss which CP method to use during training and how ConfTr can be used for ensembles and generalizes to lower α.

Main Results: In Table 8.1, we summarize the inefficiency reductions possible through ConfTr (trained with ThrLP) in comparison to Bel (trained with ThrL) and the baseline. Bel does *not* consistently improve inefficiency on all datasets. Specifically, on MNIST, EMNIST or CIFAR100, inefficiency actually *worsens*. Our ConfTr, in contrast, reduces inefficiency consistently, not only for Thr but also for APS. Here, improvements on CIFAR for Thr are generally less pronounced. This is likely because we train linear models on top of a pre-trained ResNet [HZRS16a] where features are not taking into account conformalization at test time. For APS, in contrast, improvements are still significant. Across all datasets, training with $\mathcal{L}_{\text{class}}$ generally performs slightly better, especially for datasets with many classes such as EMNIST (K=52) or CIFAR100 (K=100). Overall, ConfTr yields significant inefficiency reductions, independent of the CP method used at test time.

Conformal Predictors for Training: In Table 8.1, we specifically use ThrLP during training for ConfTr, irrespective of the CP method used at test time. This is counter-intuitive at first: when using, e.g., APS at test time, also using (smooth) APS during training seems more reasonable. However, we found training with Thr and APS to be difficult. This is highlighted in Table 8.2, showing inefficiency results for Bel, CoverTr and ConfTr with various CP methods during training on MNIST. We believe this to be caused by limited gradient flow as both Thr and APS are defined on the predicted probabilities instead of log-probabilities as used for ThrLP or in cross-entropy training. While Thr and ThrLP are clearly equivalent, re-formulating the conformity scores of APS in Equation (8.1) to use log-probabilities is left for future work. Similarly, Bel is trained using ThrL instead of Thr or APS. While training with Thr works on MNIST, it does *not* improve inefficiency. This, however, is due to requiring a fixed threshold τ=1 during training, which we found difficult to choose on most other datasets. This is due to the bounded range of the predicted probabilities ($\pi_{\theta,k}(x) \in [0,1]$) in contrast to logits. We believe that this contributes to the poor performance of Bel on some datasets. Finally, we found that Bel or ConfTr do not necessarily recover the accuracy of the baseline. For example, accuracy of ConfTr without $\mathcal{L}_{\text{class}}$ is roughly 2% lower than the baseline and for Bel with ThrL, it even reduces from 92.39% to 81.41%. This is despite ConfTr still reducing inefficiency.

Further Results: Table 8.3 includes additional results for ConfTr to "conformalize" ensembles on CIFAR10 (left) and with lower confidence levels α on EMNIST (right). In the first example, we consider applying CP to an ensemble of models. Ensemble CP methods such as [YK21] cannot improve Ineff over the best model of the ensemble, i.e., 3.10 for Thr. Instead, training

CIRFAR10: Ensemble Results			
Test	THR		
Method	(Models)	+MLP	+ConfTr
Avg. Ineff	3.10	2.40	**2.35**
Best Ineff	2.84	2.33	**2.30**

EMNIST: Confidence Levels		
Method	Basel.	ConfTr
Test	THR	
Ineff, α=0.005	4.10	**3.37**
Ineff, α=0.001	15.73	**13.65**

Table 8.3: **Ensemble Results and Lower Confidence Levels α:** *Left*: "Conformalization" of ensembles using a 2-layer MLP trained on logits, either normally or using ConfTr. The ensemble contains 18 models with accuracies in between 75.10 and 82.72%. Training a model on top of the ensemble clearly outperforms the best model of the ensemble; using ConfTr further boosts Ineff. *Right*: The inefficiency improvements of Table 8.1 generalize to lower confidence levels α on EMNIST, although ConfTr is trained with α=0.01.

an MLP on top of the ensemble's logits can improve Ineff to 2.40 and additionally using ConfTr to 2.35. The second example shows that ConfTr, trained for α=0.01, generalizes very well to significantly smaller confidence levels, e.g., α=0.001 on EMNIST. In fact, the improvement of ConfTr (without $\mathcal{L}_{\text{class}}$) in terms of inefficiency is actually more significant for lower confidence levels. We also found ConfTr to be very stable regarding hyperparameters. Only too small batch sizes (e.g., $|B|$=100 on MNIST) prevents convergence. This is likely because of too few examples ($|B_{\text{cal}}|$=50) for calibration with α=0.01 during training.

8.4.3 Conformal Training for Applications: Case Studies

For the second part, we focus on ConfTr trained with THRLP and evaluated using THR. We follow Section 8.3.3 and start by reducing class- or group-conditional inefficiency using ConfTr (without $\mathcal{L}_{\text{class}}$), before demonstrating reductions in coverage confusion of two or more classes and avoiding miscoverage between groups of classes (with $\mathcal{L}_{\text{class}}$). Because this level of control is not easily possible using Bel or standard CP, we concentrate on ConfTr only:

Shaping Conditional Inefficiency: We use ConfTr to reduce class-conditional inefficiency for specific classes or a group of classes, as defined in Equation (8.9). In Figure 8.2, inefficiency is shown to vary widely across classes: On CIFAR10, the more difficult class 3 ("cat") obtains higher inefficiency than to the easier class 1 ("automobile"). Thus, in Figure 8.3, we use ω=10 as described in Section 8.3.3 to reduce class- or group-conditional inefficiency. We report the *relative* change in percentage, showing that inefficiency reductions of 20% or more are possible

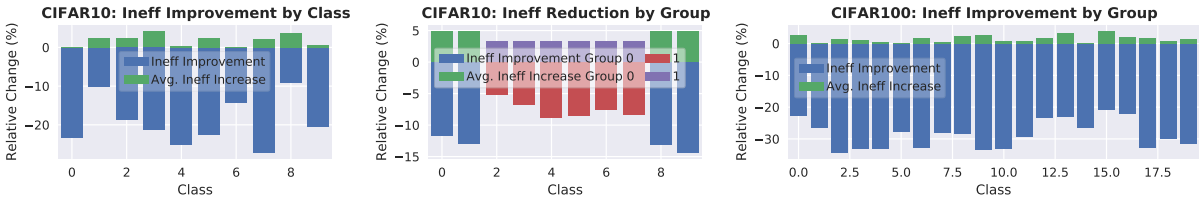


Figure 8.3: **Shaping Class-Conditional Inefficiency on CIFAR:** Possible inefficiency reductions, in percentage change, per class (blue) and the impact on the overall, average inefficiency across classes (green). *Left:* Significant inefficiency reductions are possible for all classes on CIFAR10. *Middle:* The same strategy applies to groups of classes, e.g., "vehicles" vs "animals", as well. *Right:* Similarly, on CIFAR100, we group classes by their coarse class (20 groups à 5 classes), allowing inefficiency improvements of more than 30% per individual group.

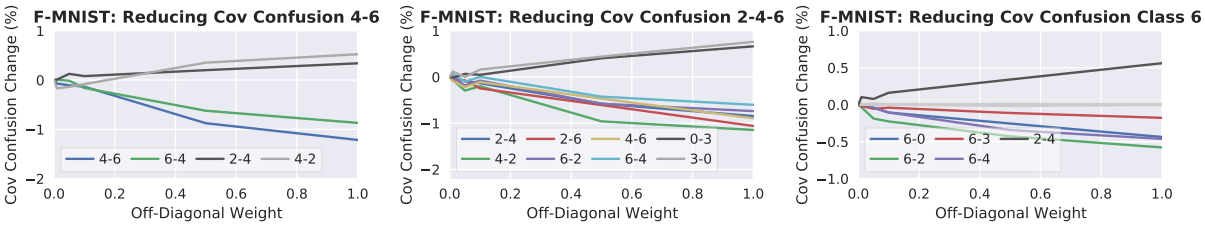


Figure 8.4: **Controlling Coverage Confusion:** Controlling coverage confusion using ConfTr with $\mathcal{L}_{\text{class}}$ and an increasing penalty $L_{y,k} > 0$ on Fashion-MNIST. For classes 4 and 6 ("coat" and "shirt"), coverage confusion $\Sigma_{y,k}$ and $\Sigma_{k,y}$ decreases significantly (blue and green). However, confusion of class 4 with class 2 ("pullover") might increase (gray). ConfTr can also reduce coverage confusion of multiple pairs of classes (e.g., additionally considering class 2). Instead, we can also penalize confusion for each pair (y,k), $k \in [K]$, e.g., $y=6$. Here, $L_{y,k} > 0$, but $L_{y,k} = 0$, i.e., Cover confusion is not reduced symmetrically.

for many classes, including "cat" on CIFAR10 (left, blue). This is also possible for two groups of classes, "vehicles" vs. "animals" (middle). However, these reductions usually come at the cost of a slight increase in average inefficiency across all classes (green). On CIFAR100, we consider 20 coarse classes, each containing 5 of the 100 classes (right). Again, significant inefficiency reductions per coarse class are possible. These observations generalize to all other considered datasets as well as different class groups.

Avoiding Coverage Confusion: Next, we use ConfTr to manipulate the coverage confusion matrix as defined in Equation (8.10). Specifically, we intend to reduce coverage confusion of selected sets of classes. Using a non-zero entry $L_{y,k} > 0$, $y \neq k$ in $\mathcal{L}_{\text{class}}$, as described in Section 8.3.3, Figure 8.4 (left) shows that coverage confusion can be reduced significantly for large enough $L_{y,k}$ on Fashion-MNIST: Considering classes 4 and 6 ("coat" and "shirt") confusion can be reduced by roughly 1%. However, as accuracy stays roughly the same and coverage is guaranteed, this comes at the cost of increasing coverage confusion for other class pairs, e.g., 2 ("pullover") and 4. ConfTr can also be used to reduce coverage confusion of multiple class pairs (middle) or a whole row in the coverage confusion matrix $\Sigma_{y,k}$ with fixed y

CIFAR10: $K_0 = 3$ ("cat") vs. $K_1 =$ Others
CIFAR100: $K_0 =$ "human-made vs. $K_1 =$ "natural"

	CIFAR10			CIFAR100		
		MisCover ↓			MisCover ↓	
Method	Ineff	0→1	1→0	Ineff	0→1	1→0
ConfTr	**2.84**	98.92	36.52	**10.44**	40.09	29.6
$L_{K_0,K_1}=1$	2.89	**91.60**	34.74	16.50	**15.77**	70.26
$L_{K_1,K_0}=1$	2.92	97.36	**26.43**	11.35	45.37	**17.56**

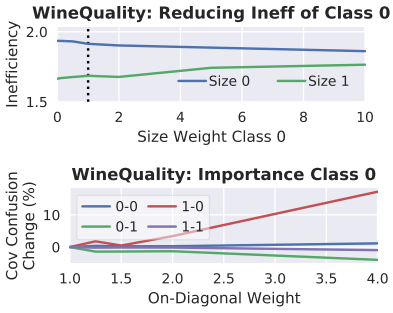


Figure 8.5: ***Left:*** **Reducing Miscoverage:** ConfTr allow reducing miscoverage on CIFAR. We consider $K_0 = \{3\}$ (i.e., "cat") vs. all other classes on CIFAR10 and "human-made" vs. "natural" on CIFAR100 ($|K_0| = 35$, $|K_1| = 65$, right). On CIFAR10, both $\text{MisCover}_{0\to1}$ and $\text{MisCover}_{1\to1}$ can be reduced significantly without large impact on inefficiency. For CIFAR100, in contrast, Ineff increases more significantly. ***Right:*** **Binary Class-Conditional Inefficiency and Coverage:** We plot inefficiency by class (top) and coverage confusion (bottom) on WineQuality. We can reduce inefficiency for class 0, the minority class, at the expense of higher inefficiency for class 1 and boost class-conditional coverage for class 0.

and $y \neq k \in [K]$. Figure 8.4 (right) shows the results for class 6: coverage confusion with, e.g., classes 0 ("t-shirt"), 2 or 4 (blue, green and violet) is reduced roughly 0.5% each at the cost of increased confusion of classes 2 and 4 (in gray). These experiments can be reproduced on other datasets, e.g., MNIST or CIFAR10 in Appendix E.7.

Reducing Miscoverage: We can also address unwanted "overlap" of two groups of classes using ConfTr and $\mathcal{L}_{\text{class}}$. In Figure 8.5 (left) we explicitly measure miscoverage as defined in Equation (8.11). First, on CIFAR10, we consider a singleton group $K_0 = \{3\}$ ("cat") and $K_1 = [K] \setminus \{3\}$: The ConfTr baseline $\text{MisCover}_{0 \to 1}$ tells us that 98.92% of confidence sets with true class 3 also contain other classes. Given an average inefficiency of 2.84 this is reasonable. Using $L_{3,k} = 1$, $k \neq 3$, this can be reduced to 91.6%. Vice-versa, the fraction of confidence sets of class $y \neq 3$ containing class 3 can be reduced from 36.52% to 26.43%. On CIFAR100, this also allows reducing overlap between "human-made" (35 classes) and "natural" (65 classes) things, e.g., $\text{MisCover}_{0 \to 1}$ reduces from 40.09% to 15.77%, at the cost of a slight increase in inefficiency. See Appendix E.8 for additional results.

Binary Datasets: Finally, in Figure 8.5 (right), we illustrate that the above conclusions generalize to the binary case: On WineQuality, we can control inefficiency of class 0 ("bad wine", minority class with $\sim$37% of examples) at the expense of increased inefficiency for class 1 ("good wine", top). Similarly, we can (empirically) improve class-conditional coverage for class 0 (bottom) or manipulate coverage confusion of both classes.

8.5 Conclusion

We introduced **conformal training (ConfTr)**, a novel method to train conformal predictors *end-to-end* with the underlying model. This addresses a major limitation of conformal prediction (CP) in practice: The model is fixed, leaving CP little to no control over the predicted confidence sets. In thorough experiments, we demonstrated that ConfTr can improve inefficiency of state-of-the-art CP methods such as Thr [SLW19] or APS [RSC20]. More importantly, motivated by medical diagnosis, we highlighted the ability of ConfTr to manipulate the predicted confidence sets in various ways. First, ConfTr can "shape" the class-conditional inefficiency distribution, i.e., reduce inefficiency on specific classes at the cost of higher inefficiency for others. Second, ConfTr allows us to control the coverage-confusion matrix by, e.g., reducing the probability of including classes other than the ground truth in confidence sets. Finally, this can be extended to explicitly reduce "overlap" between groups of classes in the predicted confidence sets. In all cases, ConfTr does *not* lose the (marginal) coverage guarantee provided by CP.

Conclusion and Future Work

9

Contents

Recent advances in deep learning enabled an astounding range of applications in computer vision and beyond. More and more of these concern high-stakes and security-critical applications including, e.g., autonomous driving or medical diagnosis among many others, where mistakes can have severe impact on human lives. These applications also come with their specific constraints for deployment, including space, time and power limitations, which are typically tackled using special-purpose hardware, so-called accelerators. This motivates a shift in evaluation from looking purely at accuracy to additionally considering robustness on manipulated or unexpected inputs and after quantization and deployment on potentially faulty hardware. Moreover, reliable uncertainty estimates are crucial to make informed decisions in such settings.

9.1 Key Insights and Conclusions

We addressed robustness and uncertainty in deep learning along three complementary research directions: First, in Part II, we improved our *understanding of adversarial examples and adversarial training*. We found that adversarial examples can easily fool models into incorrect classifications by leaving the underlying manifold of the data. However, the observed robustness-accuracy trade-off of adversarial training is not inherent and can be attributed to higher sample complexity. Moreover, when encouraging flatness in the robust loss landscape, adversarial training is very effective in improving adversarial robustness while avoiding severe robust overfitting. Second, in Part III, we considered *robustness against bit errors in quantized weights* in the context of deep neural network accelerators. Improving robustness not only improves security against attacks on the accelerator's memory but also allows reducing the memory's voltage. Usually, this leads to severe accuracy drops due to memory faults causing bit errors. Using our robust quantization, weight clipping regularization and random bit error training, the network is able to endure high bit error rates, allowing significant energy savings through low-voltage operation. Third, in Part IV, we contributed towards *improving robustness and uncertainty estimation*. Our confidence-calibrated adversarial training improves adversarial and corruption robustness as well as out-of-distribution detection by encouraging low-confidence predictions on perturbed inputs. In *conformal training*, in contrast, a conformal predictor that yields sets of classes is trained jointly with the underlying model. These confidence sets provide an intuitive notion of uncertainty and conformal training allows optimizing application-specific losses directly defined on the confidence sets while the performance guarantee obtained through conformal prediction is preserved.

Besides our work on robustness, Part I also considered the problem of *3D shape completion* of noisy point clouds as, e.g., obtained from single LiDAR views in autonomous driving. Unfortunately, ground truth shapes are extremely expensive to obtain which is why we proposed

a weakly-supervised approach based on recent 3D deep learning techniques. Specifically, we train a shape prior on synthetic shapes and *learn to* fit these to the actual observations.

In the following, we revisit the contributions of the individual chapters in more detail, before discussing future work in Section 9.2.

Part I, Deep Learning for 3D Shape Completion: In this first part, specifically Chapter 3, we introduced a new, weakly-supervised approach to 3D shape completion from noisy point clouds observations. Specifically, we leveraged a 3D variational auto-encoder to learn a latent space of shapes from synthetic data containing one or multiple object categories. Then, we formulated 3D shape completion as a maximum likelihood problem, which we amortized by essentially training a new encoder for the shape model. This is possible without access to ground truth shapes allowing to train our method on real-world datasets obtained from LiDAR sensors on autonomous vehicles or Kinect sensors in indoor robotics. Compared to related data-driven approaches, this allows fast inference at test time and, in contrast to other learning-based approaches, we do not require full supervision, resulting in better generalization.

Part II, Understanding Adversarial Examples and Training: In the second part, we focused on the robustness of deep neural networks against adversarial examples. Specifically, we studied why such adversarial perturbations exist and addressed both the robustness-accuracy trade-off and the robust overfitting phenomenon in adversarial training.

In Chapter 4, we studied adversarial examples in the context of the underlying, low-dimensional data manifold. Specifically, we showed that regular adversarial examples leave this manifold in an almost orthogonal direction. Furthermore, robustness against such "off-manifold" adversarial examples and generalization are not inherently contradictory. Instead, the robustness-accuracy trade-off can be explained by adversarial training requiring significantly more training data. We also constrained adversarial examples to the manifold, showing that robustness against such on-manifold adversarial examples is related to (clean) generalization.

In Chapter 5, we addressed the phenomenon of robust overfitting in adversarial training. To this end, we introduced two measures to quantify flatness in the robust loss landscape w.r.t. weight perturbations. This allowed us to show that robust overfitting is caused by converging to sharp minima. In fact, we revealed a clear relationship between robust flatness and adversarial robustness. That is, encouraging flatness throughout training avoids robust overfitting and is shown to improve overall adversarial robustness.

Part III, Improving Weight Robustness: In Chapter 6, we studied robustness against random and adversarial bit errors in (quantized) weights. This is of relevance in the context of deep neural network accelerators. Random bit errors occur when operating the accelerator's memory below its rated voltage in order to save energy. Combining a robust quantization scheme, with appropriate regularization and injecting bit errors during training, we were able to improve robustness considerably. In contrast to previous work, our approach generalizes well across different chips with their specific bit error patterns, various bit error rates and bit widths used for quantization. Adversarial bit errors, in contrast, are maliciously introduced through hardware- or software based attacks on the memory. Here, we propose a novel attack that is more efficient and effective than previous ones. Used during training, our attack allows improving adversarial bit error robustness which contributes towards more secure accelerators.

Part IV, Improving Adversarial Robustness and Uncertainty Estimation: In the fourth part, we considered adversarial robustness in the context of uncertainty estimation. This way, we addressed the inability of adversarial training to generalize robustness to unseen types of adversarial examples as well as its robustness-accuracy trade-off. Furthermore, as none of these approaches provide any guarantees beyond good empirical performance, we also considered conformal prediction. While conformal prediction is able to provide a guarantee on

the true label being included in predicted confidence sets, it is generally used as a post-training calibration step. In order to allow wide-spread adoption of conformal methods in deep learning, we worked towards integrating them better into nowadays training pipelines.

In Chapter 7, we proposed confidence-calibrated adversarial training which is able to generalize adversarial robustness to adversarial examples not seen during training as well as corrupted and out-of-distribution examples. To this end, training biases the model towards low-confidence predictions on adversarial examples. In practice, the model learns to extrapolate this behavior such that adversarial examples can be rejected based on their confidence. Thereby, it also improves accuracy significantly compared to standard adversarial training. We thoroughly evaluated our approach using multiple adaptive adversarial attacks and an adapted confidence-thresholded robust error.

In Chapter 8, we introduced conformal training, a new method to train conformal predictors and deep neural networks end-to-end. This addresses a significant limitation of conformal prediction which is generally applied *after* training. However, the training objective, e.g., cross-entropy loss, is not necessarily ideal for efficient conformal prediction. Moreover, the conformal predictor has little to no control over the composition of the predicted confidence sets. We showed that conformal training can reduce uncertainty, i.e., average confidence set size, significantly and allows optimizing application-specific losses defined directly on the confidence sets. This is possible while preserving the statistical guarantee of the conformal predictor by re-calibrating at test time.

9.2 Future Directions

In the following, we provide a discussion of potential future work regarding the main topics of this thesis. For example, we addressed various problems in adversarial robustness, including our understanding of adversarial examples and adversarial training. However, these and related problems remain difficult to solve:

Generative Models for Robustness: Since our work in Chapter 4, there has been significant progress in deep generative modeling. Recently, for example, generative adversarial networks [GPM+14b, RMC16, SGZ+16, ACB17], flow-based methods [RM15b, KSW16] or diffusion-based methods [SWMG15, SE19, HJA20, ND21, DN21] obtain impressive results. This also means that better low-dimensional representations and manifolds can be learned. As a result, there has been an increased interest in approaches similar to our on-manifold adversarial training, often subsumed in the larger context of adversarial data augmentation: [GQH+20] uses this idea to improve robustness against more realistic, real-world transformations of face images, while [PBZ+20] improve uncertainty estimation. Nevertheless, approximating the true manifold remains challenging and is the most significant limiting factor in larger-scale use of these techniques to boost generalization and robustness.

Robustness-Accuracy Trade-Off: While various approaches [BGH19, WCG+20], including ours from Chapter 7, improve or allow more control over the robustness-accuracy trade-off, adversarially trained models still lack state-of-the-art accuracy. Moreover, the trade-off also persists in provable defenses, e.g., [CRK19, SSY+20] or [WK18, ZCX+20]. Even significantly deeper models, with additional unlabeled training data [GQU+20], yield a significant reduction of clean accuracy. However, adversarial training against adversarial patches, for examples, does not incur a robustness-accuracy trade-off [RSS20]. These observations further support the hypothesis that conventional methods might not be able to address this problem, as suggested in Proposition 2, and re-thinking our notion of robustness might be beneficial.

Robust Overfitting in Adversarial Training: While [WXW20b, SSJF21] as well as our work in Chapter 5 try to address robust overfitting, it remains a significant obstacle in adversarial robustness. For example, to date, early stopping is still indispensable [RWK20]. While optimizing flatness is receiving much attention, also regarding clean generalization [FKMN21], it cannot avoid robust overfitting altogether [WXW20b]. These results indicate that additional factors besides flatness may be necessary for good robust generalization without overfitting. In clean generalization, [NBMS17] suggests that an additional measure of model complexity is required. This notion, however, has not been explored for robust generalization yet.

Robustness Against *Many* Threat Models: Besides our work in Chapter 7, several publications [TB19, MWK20, CH21b] address robustness against multiple threat models. However, this involves trade-offs: robustness against individual threat models reduces in order to generalize across threat models. Furthermore, these threat models are usually closely related, e.g., L_p for $p \in \{\infty, 2, 1\}$. We believe that robustness against a wider range of threat models is crucial for many applications, including adversarial patches, adversarial transformations, or semantic adversarial examples. Fine-tuning robust models as in [CH21b], could be a reasonable first step for obtaining robustness against *many*, *diverse* threat models.

Towards Practically Relevant Threat Models: While L_p-constrained adversarial examples are studied extensively, increased work on corrupted examples, adversarial patches, or transformations, etc. shows a trend to move towards more realistic and practically relevant threat models. Often, simple transformations, overlays, patches, filters or other accessible, off-the-shelf manipulation work well in practice, e.g., for circumventing automatic content moderation in social media. To this end, [WK21], for example, tries to *learn* more realistic perturbations. However, such settings are significantly more difficult to develop standard benchmarks for, which we believe to be essential for making progress.

Computationally-Limited Adversarial Robustness: Recent benchmarks demonstrate that proper adversarial robustness evaluation is computationally expensive. AutoAttack (AA) [CH20c] uses an ensemble of attacks, each with several hundreds if not thousands of iterations. Adaptive AA [YBTV21] additionally searches for appropriate objectives to be optimized. Against our method from Chapter 7, Adaptive AA additionally requires optimizing eight instead of only three attacks, increasing runtime from 108 to 205 minutes. For state-of-the-art Wide ResNets [ZK16], runtime can be even higher. These results indicate that *pure* worst-case adversarial robustness is an oversimplification. We believe that a shift towards a more fine-grained hierarchy of threat models by limiting the attackers computational resources is necessary as making it difficult "enough" for attackers could be sufficient for many applications.

Various parts of this thesis also considered robustness of deep neural networks against changes in their weights, e.g., in the context of accelerators that introduce bit errors in quantized weights or in order to measure flatness in the robust loss landscape. In the following, we provide an outlook on promising future work:

Understanding and Improving (Robust) Flatness Measures: In Chapter 5, we found that properly measuring flatness is challenging. For example, [WXW20b, FKMN21] suggest that optimizing *worst-case* flatness during training is required. However, we showed that average-case flatness is better for predicting good generalization. We also found that worst-case flatness is extremely difficult to estimate in practice, i.e., finding worst-case weight perturbations seems to be a challenging optimization problem. Due to the success of actively encouraging flatness [WXW20b, FKMN21], however, it seems critical to better understand, compute and compare these measures across varying models.

Robust Network Quantization: In Chapter 6, we also focused on proper quantization of neural networks for improving robustness against bit errors. Surprisingly, implementation details can have a significant impact on robustness, while being negligible for good accuracy. Despite extensive work on network quantization [Guo18], it is generally viewed solely in the context of its approximation quality, i.e., quantization errors. As bit error robustness is important for deployment, we think that future work should not only focus on developing more robust quantization schemes but also work on discussing the underlying theory.

Bit Error Robustness on Real Accelerators: While Chapter 6 provides empirical results on *profiled* bit errors, obtained from real chips used in [CSC+19], there still remains a gap to actually measuring robustness on real chips. For example, it is impossible to know the exact memory layout and data movement in advance [WES19, YCES17]. This also makes it difficult to estimate obtained energy savings or evaluate bit errors in activations. Furthermore, profiling is very expensive and usually constrained to few memory arrays per chip. We believe that standardized and more realistic benchmarks are required for this problem to be addressed by a wider community within (robust) deep learning.

Weight Robustness for Different Hardware Architectures: Apart from digital accelerators, analog hardware is receiving increased attention [SHES17, HGP19]. In analog devices, values are not represented as bit patterns but mapped to continuous, physical quantities such as voltage or current. This is often combined with in-memory computation [HGP19]. This has the potential to reduce data movement and energy consumption significantly, but leads to inherently noisy operations. In practice, this can be modeled by perturbed weights [ZKMW20, ICZ+20, FQ21, ICZ+21]. Similarly, non-volatile memory is used, e.g., for binary neural networks [YBG+21]. But the reliability of the memory depends not only on voltage but also on temperature. In all of these settings, robust deep neural networks play a crucial role for energy-efficient inference. However, these problems are, to the best of our knowledge, not receiving sufficient attention, especially outside the hardware communities.

Batch Normalization: While batch normalization [IS15a] is still poorly understood [BGSW18, STIM18], it is ubiquitous in deep learning. More importantly, its relationship to adversarial or corruption robustness is largely unclear [GGT+19, SWGG20, BZK20, BZKK21]. In Chapter 8, we additionally found that batch normalization reduces robustness to weight perturbations significantly. Instead, we resorted to group normalization [WH18], suggesting that the batch/dataset statistics play a key role in robustness settings. Moreover, it remains unclear how to properly adjust the batch normalization statistics when weights are subject to perturbations. Properly evaluating and understanding batch normalization in these settings could have tremendous impact on a wide range of architectures relying on it.

Finally, we also made contributions in the context of conformal prediction. Our *conformal training* integrates conformal predictors with nowadays deep learning pipelines, opening up many interesting directions of future work:

Addressing Limitations of Conformal Training: Besides Chapter 8, there is concurrent work [Bel20, Bel21, BMW+22] aiming to integrate conformal prediction into training. However, these approaches have several drawbacks compared to conformal training. For example, [Bel21] does not consider the calibration step during training. In contrast, [BMW+22] expresses the actual calibration step as learning problem. In the end, both require to optimize a combination of efficiency and coverage loss. All approaches, including ours, are limited in terms of model complexity. While we are able to train deeper models than [Bel21, BMW+22], we also fine-tuned pre-trained models on CIFAR [Kri09]. However, enabling conformal training *from scratch* on large-sale datasets is key prerequisite for making conformal prediction more accessible.

Conformal Prediction with Application-Specific Guarantees: Standard conformal prediction has limited control over the exact composition of confidence sets. We addressed this problem by allowing to (empirically) optimize application-specific losses on the confidence sets. In contrast to conformal prediction, [BAL+21] is able to provide guarantees on arbitrary losses, i.e., the coverage guarantee is replaced with a probabilistic upper bound on a loss. This allows more flexibility regarding applications such that integrating conformal training with [BAL+21] is a natural next step. Initial experiments show that the calibration and prediction steps of [BAL+21] can be made differentiable, even if both become computationally more expensive compared to standard conformal prediction.

Conformal Prediction in Adversarial Settings: The recent interest in conformal prediction also sparked some early work on adversarially robust conformal prediction [GWDR22]. While the considered threat models are very limited compared to adversarial training, this opens a novel direction of research towards provable robustness. Where an empirical robustness of, e.g., 90% or above on CIFAR is impossible to obtain with current methods, conformal prediction could provide user-specified *adversarial coverage* by allowing to predict multiple classes. However, [GWDR22] also shows that coverage guarantees on adversarial examples are very difficult to obtain. Nevertheless, we think that combining high coverage guarantee on unperturbed examples with high *empirical* coverage on adversarial examples can already be useful in many applications.

List of Algorithms

List of Figures

List of Tables

Bibliography

[AAB+15] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems. https://www.tensorflow.org/, 2015. Software available from tensorflow.org. Cited on page 267.

[AAG19] Rima Alaifari, Giovanni S. Alberti, and Tandri Gauksson. Adef: an iterative algorithm to construct adversarial deformations. In *Proc. of the International Conference on Learning Representations (ICLR)*, 2019. Cited on pages 14, 17, and 58.

[AB21] Anastasios N. Angelopoulos and Stephen Bates. A gentle introduction to conformal prediction and distribution-free uncertainty quantification. *arXiv.org*, abs/2107.07511, 2021. Cited on page 23.

[ABB+17] Laurent Amsaleg, James Bailey, Dominique Barbe, Sarah M. Erfani, Michael E. Houle, Vinh Nguyen, and Milos Radovanovic. The vulnerability of learning to adversarial perturbation increases with intrinsic dimensionality. In *Proc. of the IEEE Workshop on Information Forensics and Security*, 2017. Cited on page 19.

[ABC+18] Yossi Adi, Carsten Baum, Moustapha Cissé, Benny Pinkas, and Joseph Keshet. Turning your weakness into a strength: Watermarking deep neural networks by backdooring. In *USENIX Security Symposium*, pages 1615–1631, 2018. Cited on page 15.

[ABC+21] Anastasios N. Angelopoulos, Stephen Bates, Emmanuel J. Candès, Michael I. Jordan, and Lihua Lei. Learn then test: Calibrating predictive algorithms to achieve risk control. *arXiv.org*, abs/2110.01052, 2021. Cited on page 23.

[ABJM21] Anastasios Nikolas Angelopoulos, Stephen Bates, Michael Jordan, and Jitendra Malik. Uncertainty sets for image classifiers using conformal prediction. In *Proc. of the International Conference on Learning Representations (ICLR)*, 2021. Cited on pages 7, 14, 23, 136, 139, and 145.

[Abr74] Milton Abramowitz. *Handbook of Mathematical Functions, With Formulas, Graphs, and Mathematical Tables*. Dover Publications, 1974. Cited on page 33.

[ABvB+20] Milad Alizadeh, Arash Behboodi, Mart van Baalen, Christos Louizos, Tijmen Blankevoort, and Max Welling. Gradient ℓ_1 regularization for quantization robustness. In *Proc. of the International Conference on Learning Representations (ICLR)*, 2020. Cited on pages 14, 20, and 21.

[ABZJ21] Anastasios N. Angelopoulos, Stephen Bates, Tijana Zrnic, and Michael I. Jordan. Private prediction sets. *arXiv.org*, abs/2102.06202, 2021. Cited on page 23.

[AC18] Anish Athalye and Nicholas Carlini. On the robustness of the CVPR 2018 white-box adversarial example defenses. *arXiv.org*, abs/1804.03286, 2018. Cited on pages 18, 50, and 116.

[ACB17] Martín Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *Proc. of the International Conference on Machine Learning (ICML)*, 2017. Cited on page 153.

[ACFH20] Maksym Andriushchenko, Francesco Croce, Nicolas Flammarion, and Matthias Hein. Square attack: A query-efficient black-box adversarial attack via random search. In *Proc. of the European Conference on Computer Vision (ECCV)*, 2020. Cited on pages 16, 117, 124, and 127.

[ACW18] Anish Athalye, Nicholas Carlini, and David A. Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *Proc. of the International Conference on Machine Learning (ICML)*, 2018. Cited on pages 16, 50, 57, 63, 116, and 230.

[AD19a] Jonathan Aigrain and Marcin Detyniecki. Detecting adversarial examples and other misclassifications in neural networks by introspection. *arXiv.org*, abs/1905.09186, 2019. Cited on page 19.

[AD19b] Jonathan Aigrain and Marcin Detyniecki. Improving robustness without sacrificing accuracy with patch gaussian augmentation. In *Proc. of the International Conference on Machine Learning (ICML) Workshops*, 2019. Cited on page 22.

[ADMG18] Panos Achlioptas, Olga Diamanti, Ioannis Mitliagkas, and Leonidas J. Guibas. Learning representations and generative models for 3d point clouds. In *Proc. of the International Conference on Learning Representations (ICLR)*, 2018. Cited on page 13.

[AF20] Maksym Andriushchenko and Nicolas Flammarion. Understanding and improving fast adversarial training. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. Cited on page 18.

[AF21] Sahar Abdelnabi and Mario Fritz. What's in the box: Deflecting adversarial attacks by randomly deploying adversarially-disjoint models. In *Proc. of the ACM Conference on Computer and Communications Security (CCS) Workshops*, 2021. Cited on page 17.

[AFG+19] Shruti Agarwal, Hany Farid, Yuming Gu, Mingming He, Koki Nagano, and Hao Li. Protecting world leaders against deep fakes. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. Cited on page 15.

[AGGC18] Arash AziziMazreah, Yongbin Gu, X. Gu, and L. Chen. Tolerating soft errors in deep learning accelerators with reliable on-chip memory designs. *IEEE International Conference on Networking, Architecture and Storage (NAS)*, 2018. Cited on page 22.

[AGL+17] Dan Alistarh, Demjan Grubic, Jerry Li, Ryota Tomioka, and Milan Vojnovic. QSGD: communication-efficient SGD via gradient quantization and encoding. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017. Cited on page 21.

[AH19] Maksym Andriushchenko and Matthias Hein. Provably robust boosted decision stumps and trees against adversarial attacks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. Cited on page 14.

[AHFD21] Ahmed Aldahdooh, Wassim Hamidouche, Sid Ahmed Fezza, and Olivier Déforges. Adversarial example detection for DNN models: A review. *arXiv.org*, abs/2105.00203, 2021. Cited on page 19.

[AL21] Zeyuan Allen-Zhu and Yuanzhi Li. Feature purification: How adversarial training performs robust deep learning. In *IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, 2021. Cited on page 17.

[AM18] Naveed Akhtar and Ajmal S. Mian. Threat of adversarial attacks on deep learning in computer vision: A survey. *IEEE Access*, 6, 2018. Cited on page 15.

[AME+14] Mathieu Aubry, Daniel Maturana, Alexei Efros, Bryan Russell, and Josef Sivic. Seeing 3D chairs: exemplar part-based 2D-3D alignment using a large dataset of CAD models. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014. Cited on page 12.

[AMH20] Maximilian Augustin, Alexander Meinke, and Matthias Hein. Adversarial robustness on in- and out-distribution improves explainability. In *Proc. of the European Conference on Computer Vision (ECCV)*, 2020. Cited on page 15.

[AMO12] Sameer Agarwal, Keir Mierle, and Others. Ceres solver. http://ceres-solver.org, 2012. Cited on page 45.

[ANKT19] Nilesh A Ahuja, Ibrahima Ndiour, Trushant Kalyanpur, and Omesh Tickoo. Probabilistic modeling of deep features for out-of-distribution and adversarial detection. *arXiv.org*, abs/1909.11786, 2019. Cited on page 22.

[AO20] Abdullah Al-Dujaili and Una-May O'Reilly. Sign bits are all you need for black-box attacks. In *Proc. of the International Conference on Learning Representations (ICLR)*, 2020. Cited on page 16.

[APH+21] Moloud Abdar, Farhad Pourpanah, Sadiq Hussain, Dana Rezazadegan, Li Liu, Mohammad Ghavamzadeh, Paul W. Fieguth, Xiaochun Cao, Abbas Khosravi, U. Rajendra Acharya, Vladimir Makarenkov, and Saeid Nahavandi. A review of uncertainty quantification in deep learning: Techniques, applications and challenges. *Information Fusion*, 76:243–297, 2021. Cited on page 14.

[APS94] C. Alippi, V. Piuri, and M. Sami. Sensitivity to errors in artificial neural networks: a behavioral approach. *IEEE International Symposium on Circuits and Systems (ISCAS)*, 6, 1994. Cited on page 21.

[APT+21] Motasem Alfarra, Juan Camilo Perez, Ali Thabet, Adel Bibi, Philip Torr, and Bernard Ghanem. Combating adversaries with anti-adversaries. In *Proc. of the International Conference on Machine Learning (ICML) Workshops*, 2021. Cited on page 18.

[Arm66] Larry Armijo. Minimization of functions having lipschitz continuous first partial derivatives. *Pacific Journal of Mathematics*, 16(1):1–3, 1966. Cited on page 134.

[ASS+18] Eman Ahmed, Alexandre Saint, Abd El Rahman Shabayek, Kseniya Cherenkova, Rig Das, Gleb Gusev, Djamila Aouada, and Björn E. Ottersten. Deep learning advances on different 3d data representations: A survey. *arXiv.org*, abs/1808.01462, 2018. Cited on page 13.

[ASZ20] Elahe Arani, Fahad Sarfraz, and Bahram Zonooz. Adversarial concurrent training: Optimizing robustness and accuracy trade-off of deep neural networks. In *Proc. of the British Machine Vision Conference (BMVC)*, 2020. Cited on page 19.

[AUH+19] Jean-Baptiste Alayrac, Jonathan Uesato, Po-Sen Huang, Alhussein Fawzi, Robert Stanforth, and Pushmeet Kohli. Are labels required for improving adversarial robustness? In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. Cited on pages 18, 67, 73, and 239.

[AY01] Charu C. Aggarwal and Philip S. Yu. Outlier detection for high dimensional data. In *Proc. of the ACM International Conference on Management of Data (SIGMOD)*, 2001. Cited on page 14.

[BAL+21] Stephen Bates, Anastasios Angelopoulos, Lihua Lei, Jitendra Malik, and Michael I. Jordan. Distribution-free, risk-controlling prediction sets. *Journal of the ACM*, 68(6):43:1–43:34, 2021. Cited on pages 7, 23, and 156.

[BBSZ20] Maria-Florina Balcan, A. Blum, Dravyansh Sharma, and Hongyang Zhang. On the power of abstention and data-driven decision making for adversarial robustness. *arXiv.org*, abs/2010.06154, 2020. Cited on page 18.

[BCC+21] Sebastian Buschjäger, Jian-Jia Chen, Kuan-Hsun Chen, Mario Günzel, Christian Hakert, Katharina Morik, Rodion Novkin, Lukas Pfahler, and Mikail Yayla. Margin-maximization in binarized neural networks for optimizing bit error tolerance. In *Proc. of the Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2021. Cited on page 20.

[BCKW15] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural network. In *Proc. of the International Conference on Machine Learning (ICML)*, JMLR Workshop and Conference Proceedings, 2015. Cited on page 14.

[BCL+20] Jiawang Bai, B. Chen, Y. Li, Dongxian Wu, Weiwei Guo, Shutao Xia, and E. Yang. Targeted attack for deep hashing based retrieval. In *Proc. of the European Conference on Computer Vision (ECCV)*, 2020. Cited on page 16.

[BCL+21] Stephen Bates, J. Candès, Lihua Lei, Yaniv Romano, and Matteo Sesia. Testing for outliers with conformal p-values. *arXiv.org*, abs/2104.08279, 2021. Cited on page 23.

[BCLS13] S.Y. Bao, M. Chandraker, Yuanqing Lin, and S. Savarese. Dense object reconstruction with semantic priors. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013. Cited on pages 12, 28, and 46.

[BCM+13] Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Srndic, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. Evasion attacks against machine learning at test time. In *Proc. of the European Conference on Machine Learning and Knowledge Discovery in Databases (ECML PKDD)*, 2013. Cited on pages 1, 4, and 14.

[BCM17] Arjun Nitin Bhagoji, Daniel Cullina, and Prateek Mittal. Dimensionality reduction as a defense against evasion attacks on machine learning classifiers. *arXiv.org*, abs/1704.02654, 2017. Cited on pages 17 and 19.

[BCRT20] Rina Foygel Barber, J. Candès, Aaditya Ramdas, and Ryan J. Tibshirani. The limits of distribution-free conditional predictive inference. *Information and Inference: A Journal of the IMA*, 10(2):455–482, 2020. Cited on page 23.

[BCRT21] Rina Foygel Barber, J. Candès, Aaditya Ramdas, and Ryan J. Tibshirani. Predictive inference with the jackknife+. *The Annals of Statistics*, 49(1):486 – 507, 2021. Cited on page 23.

[BCZ+18] Tom B. Brown, Nicholas Carlini, Chiyuan Zhang, Catherine Olsson, Paul Christiano, and Ian Goodfellow. Unrestricted adversarial examples. *arXiv.org*, abs/1809.08352, 2018. Cited on page 50.

[BDK19] Thomas Brunner, Frederik Diehl, and Alois Knoll. Copy and paste: A simple but effective initialization method for black-box adversarial attacks. *arXiv.org*, abs/1906.06086, 2019. Cited on page 16.

[Bel20] Anthony Bellotti. Constructing normalized nonconformity measures based on maximizing predictive efficiency. In Alexander Gammerman, Vladimir Vovk, Zhiyuan Luo, Evgueni N. Smirnov, Giovanni Cherubin, and Marco Christini, editors, *Proc. of the Symposium on Conformal and Probabilistic Prediction and Applications (COPA)*, 2020. Cited on page 155.

[Bel21] Anthony Bellotti. Optimized conformal classification using gradient descent approximation. *arXiv.org*, abs/2105.11255, 2021. Cited on pages 136, 137, 140, 143, 144, 145, 155, and 269.

[BFH+18] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs. http://github.com/google/jax, 2018. Cited on page 267.

[BFS18] Apratim Bhattacharyya, Mario Fritz, and Bernt Schiele. Long-term on-board prediction of people in traffic scenes under uncertainty. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. Cited on page 1.

[BG08] Eta S. Berner and Mark L. Graber. Overconfidence as a cause of diagnostic error in medicine. *The American Journal of Medicine*, 121(5, Supplement):S2–S23, 2008. Cited on page 1.

[BGBK21] Lukas Bieringer, Kathrin Grosse, Michael Backes, and Katharina Krombholz. Mental models of adversarial machine learning. *arXiv.org*, abs/2105.03726, 2021. Cited on page 13.

[BGH19] Yogesh Balaji, Tom Goldstein, and Judy Hoffman. Instance adaptive adversarial training: Improved accuracy tradeoffs in neural nets. *arXiv.org*, abs/1910.08051, 2019. Cited on pages 18, 19, and 153.

[BGSW18] Johan Bjorck, Carla P. Gomes, Bart Selman, and Kilian Q. Weinberger. Understanding batch normalization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018. Cited on pages 19 and 155.

[BHJ+18] Jakub Breier, Xiaolu Hou, Dirmanto Jap, Lei Ma, Shivam Bhasin, and Yang Liu. Practical fault attack on deep neural networks. In *Proc. of the ACM Conference on Computer and Communications Security (CCS)*, 2018. Cited on page 89.

[BHLS17] Arjun Nitin Bhagoji, Warren He, Bo Li, and Dawn Song. Exploring the space of black-box attacks on deep neural networks. *arXiv.org*, abs/1712.09491, 2017. Cited on page 50.

[BHLS18] Arjun Nitin Bhagoji, Warren He, Bo Li, and Dawn Song. Practical black-box attacks on deep neural networks using efficient query mechanisms. In *Proc. of the European Conference on Computer Vision (ECCV)*, 2018. Cited on page 16.

[BHN19] Solon Barocas, Moritz Hardt, and Arvind Narayanan. *Fairness and Machine Learning*. fairmlbook.org, 2019. http://www.fairmlbook.org. Cited on page 15.

[BHP+21] Oliver Bryniarski, Nabeel Hingun, Pedro Pachuca, Vincent Wang, and Nicholas Carlini. Evading adversarial example detection defenses with orthogonal projected gradient descent. *arXiv.org*, abs/2106.15023, 2021. Cited on pages 7 and 19.

[BHV14] Vineeth Balasubramanian, Shen-Shyang Ho, and Vladimir Vovk. *Conformal prediction for reliable machine learning: theory, adaptations and applications*. Newnes, 2014. Cited on page 23.

[Bis93] C. M. Bishop. Novelty detection and neural network validation. In Stan Gielen and Bert Kappen, editors, *Proc. of the International Conference on Artificial Neural Networks (ICANN)*, 1993. Cited on page 14.

[BKM16] D. M. Blei, A. Kucukelbir, and J. D. McAuliffe. Variational inference: A review for statisticians. *arXiv.org*, 1601.00670, 2016. Cited on page 31.

[BLRW16] André Brock, Theodore Lim, James M. Ritchie, and Nick Weston. Generative and discriminative voxel modeling with convolutional neural networks. *arXiv.org*, 1608.04236, 2016. Cited on pages 13 and 28.

[BLT+20] Rudy Bunel, Jingyue Lu, Ilker Turkaslan, Philip H. S. Torr, Pushmeet Kohli, and M. Pawan Kumar. Branch and bound for piecewise linear neural network verification. *Journal of Machine Learning Research (JMLR)*, 21:42:1–42:39, 2020. Cited on page 18.

[BLZ+20] A. Bui, Trung Le, He Zhao, Paul Montague, Olivier deVel, Tamas Abraham, and Dinh Q. Phung. Improving adversarial robustness by enforcing local and global compactness. In *Proc. of the European Conference on Computer Vision (ECCV)*, 2020. Cited on page 18.

[BM92] Paul J. Besl and Neil D. McKay. A method for registration of 3-d shapes. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 14(2):239–256, 1992. Cited on pages 12 and 39.

[BMAH22] Julian Bitterwolf, Alexander Meinke, Maximilian Augustin, and Matthias Hein. Revisiting out-of-distribution detection: A simple baseline is surprisingly effective. https://openreview.net/forum?id=-BTmxCddppP, 2022. Cited on page 22.

[BMH20] Julian Bitterwolf, Alexander Meinke, and Matthias Hein. Certifiably adversarially robust detection of out-of-distribution data. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. Cited on page 22.

[BMR+17] Tom B. Brown, Dandelion Mané, Aurko Roy, Martín Abadi, and Justin Gilmer. Adversarial patch. *arXiv.org*, abs/1712.09665, 2017. Cited on pages 16 and 17.

[BMW+22] Yu Bai, Song Mei, Huan Wang, Yingbo Zhou, and Caiming Xiong. Efficient and differentiable conformal prediction with general function classes. In *Proc. of the International Conference on Learning Representations (ICLR)*, 2022. Cited on page 155.

[BNaL12] Battista Biggio, Blaine Nelson, and avel Laskov. Poisoning attacks against support vector machines. In *Proc. of the International Conference on Machine Learning (ICML)*, 2012. Cited on page 15.

[BNL11] Battista Biggio, Blaine Nelson, and Pavel Laskov. Support vector machines under adversarial label noise. In *Proc. of the Asian Conference on Machine Learning (ACML)*, 2011. Cited on page 15.

[BNS19] Ron Banner, Yury Nahshan, and Daniel Soudry. Post training 4-bit quantization of convolutional networks for rapid-deployment. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. Cited on page 21.

[BR18] Battista Biggio and Fabio Roli. Wild patterns: Ten years after the rise of adversarial machine learning. In *Proc. of the ACM Conference on Computer and Communications Security (CCS)*, 2018. Cited on page 15.

[BRA+19] Vincent Ballet, Xavier Renard, Jonathan Aigrain, Thibault Laugel, Pascal Frossard, and Marcin Detyniecki. Imperceptible adversarial attacks on tabular data. *arXiv.org*, abs/1911.03274, 2019. Cited on page 16.

[BRB18] Wieland Brendel, Jonas Rauber, and Matthias Bethge. Decision-based adversarial attacks: Reliable attacks against black-box machine learning models. In *Proc. of the International Conference on Learning Representations (ICLR)*, 2018. Cited on page 16.

[BRK+19] Wieland Brendel, Jonas Rauber, Matthias Kümmerer, Ivan Ustyuzhaninov, and Matthias Bethge. Accurate, reliable and fast robustness evaluation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. Cited on page 16.

[BRRG18] Jacob Buckman, Aurko Roy, Colin Raffel, and Ian Goodfellow. Thermometer encoding: One hot way to resist adversarial examples. In *Proc. of the International Conference on Learning Representations (ICLR)*, 2018. Cited on page 17.

[BTBD20] Mathieu Blondel, Olivier Teboul, Quentin Berthet, and Josip Djolonga. Fast differentiable sorting and ranking. In *Proc. of the International Conference on Machine Learning (ICML)*, 2020. Cited on pages 139 and 140.

[BTT+18] Rudy Bunel, Ilker Turkaslan, Philip H. S. Torr, Pushmeet Kohli, and Pawan Kumar Mudigonda. A unified view of piecewise linear neural network verification. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018. Cited on page 18.

[BVvD+17] Babak Ehteshami Bejnordi, Mitko Veta, Paul Johannes van Diest, Bram van Ginneken, Nico Karssemeijer, Geert J. S. Litjens, Jeroen A. van der Laak, Meyke Hermsen, Quirine F Manson, Maschenka C. A. Balkenhol, Oscar G. F. Geessink, Nikolaos Stathonikos, Marcory Crf van Dijk, Peter Bult, Francisco Beca, Andrew H. Beck, Dayong Wang, Aditya Khosla, Rishab Gargeya, Humayun Irshad, Aoxiao Zhong, Qi Dou, Quanzheng Li, Hao Chen, Huang Lin, Pheng-Ann Heng, Christian Hass, Elia Bruni, Quincy Wong, Ugur Halici, Mustafa Ümit Öner, Rengul Cetin-Atalay, Matt Berseth, Vitali Khvatkov, A F Vylegzhanin, Oren Z. Kraus, Muhammad Shaban, Nasir M. Rajpoot, Ruqayya Awan, Korsuk Sirinukunwattana, Talha Qaiser, Yee-Wah Tsang, David Tellez, Jonas Annuscheit, Peter Hufnagl, Mira Valkonen, Kimmo Kartasalo, Leena Latonen, Pekka Ruusuvuori, Kaisa Liimatainen, Shadi Albarqouni, Bharti Mungal, Ami George, Stefanie Demirci, Nassir Navab, Seiryo Watanabe, Shigeto Seno, Yoichi Takenaka, Hideo Matsuda, Hady Ahmady Phoulady, Vassili A. Kovalev, Alexander Kalinovsky, Vitali Liauchuk, Gloria Bueno, M. del Milagro Fernández-Carrobles, Ismael Serrano, Oscar Deniz, Daniel Racoceanu, and Rui Venâncio. Diagnostic assessment of deep learning algorithms for detection of lymph node metastases in women with breast cancer. *Journal of the American Medical Association (JAMA)*, 318:2199–2210, 2017. Cited on page 267.

[BYW+20] Saifullahi Aminu Bello, Shangshu Yu, Cheng Wang, Jibril Muhmmad Adam, and Jonathan Li. Review: Deep learning on 3d point clouds. *Remote Sensing*, 12(11):1729, 2020. Cited on page 13.

[BZK20] Philipp Benz, C. Zhang, and I. Kweon. Batch normalization increases adversarial vulnerability: Disentangling usefulness and robustness of model features. *arXiv.org*, abs/2010.03316, 2020. Cited on page 155.

[BZKK21] Philipp Benz, Chaoning Zhang, Adil Karjauv, and In So Kweon. Revisiting batch normalization for improving corruption robustness. In *Proc. of the IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2021. Cited on page 155.

[CAB+19] Nicholas Carlini, Anish Athalye, Nicolas Papernotand Wieland Brendel, Jonas Rauber, Dimitris Tsipras, Ian Goodfellow, Aleksander Madry, and Alexey Kurakin. On evaluating adversarial robustness. *arXiv.org*, abs/1902.06705, 2019. Cited on page 117.

[CAB+20] Ashutosh Chaubey, Nikhil Agrawal, Kavya Barnwal, Keerat Kaur Guliani, and Pramod Mehta. Universal adversarial perturbations: A survey. *arXiv.org*, abs/2005.08087, 2020. Cited on page 15.

[CAH19] Francesco Croce, Maksym Andriushchenko, and Matthias Hein. Provable robustness of relu networks via maximization of linear regions. In *Conference on Artificial Intelligence and Statistics (AISTATS)*, 2019. Cited on page 18.

[ÇAL+16] Özgün Çiçek, Ahmed Abdulkadir, Soeren S. Lienkamp, Thomas Brox, and Olaf Ronneberger. 3d u-net: Learning dense volumetric segmentation from sparse annotation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, 2016. Cited on page 39.

[CANK17] Moustapha M Cisse, Yossi Adi, Natalia Neverova, and Joseph Keshet. Houdini: Fooling deep structured visual and speech recognition models with adversarial examples. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017. Cited on pages 14 and 16.

[Car19] Nicholas Carlini. Is ami (attacks meet interpretability) robust to adversarial examples? *arXiv.org*, abs/1902.02322, 2019. Cited on page 18.

[CAS+20a] Francesco Croce, Maksym Andriushchenko, Vikash Sehwag, Nicolas Flammarion, Mung Chiang, Prateek Mittal, and Matthias Hein. Robustbench: a standardized adversarial robustness benchmark. *arXiv.org*, abs/2010.09670, 2020. Cited on pages 66, 74, 78, 79, and 241.

[CAS+20b] Francesco Croce, Maksym Andriushchenko, Naman D. Singh, Nicolas Flammarion, and Matthias Hein. Sparse-rs: a versatile framework for query-efficient sparse black-box adversarial attacks. *arXiv.org*, abs/2006.12834, 2020. Cited on page 16.

[CATvS17] Gregory Cohen, Saeed Afshar, Jonathan Tapson, and André van Schaik. EMNIST: an extension of MNIST to handwritten letters. *arXiv.org*, abs/1702.05373, 2017. Cited on pages 49, 50, 51, 54, 144, and 267.

[CBC+18] Fabio Carrara, Rudy Becarelli, Roberto Caldelli, Fabrizio Falchi, and Giuseppe Amato. Adversarial examples detection in features distance spaces. In *Proc. of the European Conference on Computer Vision (ECCV) Workshops*, 2018. Cited on page 19.

[CBD15] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. Binaryconnect: Training deep neural networks with binary weights during propagations. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2015. Cited on page 21.

[CBM+20] Maurizio Capra, Beatrice Bussolino, Alberto Marchisio, Guido Masera, Maurizio Martina, and Muhammad Shafique. Hardware and software optimizations for accelerating deep neural networks: Survey of current trends, challenges, and the road ahead. *IEEE Access*, 8:225134–225180, 2020. Cited on page 21.

[CC18] Robert M. Chesney and Danielle Keats Citron. Deep fakes: A looming challenge for privacy, democracy, and national security. *California Law Review*, 107, 2018. Cited on page 15.

[CCA+09] Paulo Cortez, António Cerdeira, Fernando Almeida, Telmo Matos, and José Reis. Modeling wine preferences by data mining from physicochemical properties. *Decision Support Systems*, 47(4):547–553, 2009. Cited on pages 144 and 267.

[CCC+08] Paolo Cignoni, Marco Callieri, Massimiliano Corsini, Matteo Dellepiane, Fabio Ganovelli, and Guido Ranzuglia. Meshlab: an open-source mesh processing tool. In *Eurographics Italian Chapter Conference*, 2008. Cited on page 35.

[CCS+17] Pratik Chaudhari, Anna Choromanska, Stefano Soatto, Yann LeCun, Carlo Baldassi, Christian Borgs, Jennifer T. Chayes, Levent Sagun, and Riccardo Zecchina. Entropy-sgd: Biasing gradient descent into wide valleys. In *Proc. of the International Conference on Learning Representations (ICLR)*, 2017. Cited on pages 19, 66, 67, 70, 73, 79, and 237.

[CDD19] Rob Cornish, George Deligiannidis, and Arnaud Doucet. Robust predictive uncertainty for neural networks via confidence densities. In *Proc. of the International Conference on Machine Learning (ICML) Workshops*, 2019. Cited on page 22.

[CDS+14] Tianshi Chen, Zidong Du, Ninghui Sun, Jia Wang, Chengyong Wu, Yunji Chen, and Olivier Temam. Diannao: a small-footprint high-throughput accelerator for ubiquitous machine-learning. In *Proc. of the International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2014. Cited on page 84.

[CES16] Yu-Hsin Chen, Joel S. Emer, and Vivienne Sze. Eyeriss: A spatial architecture for energy-efficient dataflow for convolutional neural networks. In *ACM/IEEE Annual International Symposium on Computer Architecture (ISCA)*, 2016. Cited on pages 6, 84, and 85.

[cff] C foreign function interface for python. https://cffi.readthedocs.io/en/latest/index.html. Cited on page 246.

[CFG+15] Angel X. Chang, Thomas A. Funkhouser, Leonidas J. Guibas, Pat Hanrahan, Qi-Xing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. Shapenet: An information-rich 3d model repository. *arXiv.org*, 1512.03012, 2015. Cited on pages 12, 27, 28, 31, 35, and 46.

[CG20] Jinghui Chen and Quanquan Gu. Rays: A ray searching method for hard-label adversarial attack. In *Proc. of the ACM International Conference on Knowledge Discovery & Data Mining*, 2020. Cited on page 16.

[CGAD22] Maxime Cauchois, Suyash Gupta, Alnur Ali, and John C. Duchi. Predictive inference with weak supervision. *arXiv.org*, abs/2201.08315, 2022. Cited on page 23.

[CGB+22] Francesco Croce, Sven Gowal, Thomas Brunner, Evan Shelhamer, Matthias Hein, and A. Taylan Cemgil. Evaluating the adversarial robustness of adaptive test-time defenses. *arXiv.org*, abs/2202.13711, 2022. Cited on page 18.

[CGD21] Maxime Cauchois, Suyash Gupta, and John C. Duchi. Knowing what you know: valid and validated confidence sets in multiclass and multilabel prediction. *Journal of Machine Learning Research (JMLR)*, 22:81:1–81:42, 2021. Cited on pages 23 and 136.

[CGDK20] A. Taylan Cemgil, Sumedh Ghaisas, Krishnamurthy (Dj) Dvijotham, and Pushmeet Kohli. Adversarially robust representations with smooth encoders. In *Proc. of the International Conference on Learning Representations (ICLR)*, 2020. Cited on page 17.

[CGG+20] Ping-Yeh Chiang, Jonas Geiping, Micah Goldblum, Tom Goldstein, Renkun Ni, Steven Reich, and Ali Shafahi. Witchcraft: Efficient PGD attacks with random step size. In *Proc. of the IEEE Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020. Cited on page 16.

[CH19] Francesco Croce and Matthias Hein. Sparse and imperceivable adversarial attacks. In *Proc. of the IEEE International Conference on Computer Vision (ICCV)*, 2019. Cited on pages 16, 90, and 127.

[CH20a] Francesco Croce and Matthias Hein. Minimally distorted adversarial examples with a fast adaptive boundary attack. In *Proc. of the International Conference on Machine Learning (ICML)*, 2020. Cited on page 16.

[CH20b] Francesco Croce and Matthias Hein. Provable robustness against all adversarial l_p-perturbations for $p \geq 1$. In *Proc. of the International Conference on Learning Representations (ICLR)*, 2020. Cited on page 18.

[CH20c] Francesco Croce and Matthias Hein. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *Proc. of the International Conference on Machine Learning (ICML)*, 2020. Cited on pages 8, 16, 17, 18, 68, 70, 72, 154, 241, and 242.

[CH21a] TaiYu Cheng and Masanori Hashimoto. Minimizing energy of DNN training with adaptive bit-width and voltage scaling. In *IEEE International Symposium on Circuits and Systems (ISCAS)*, 2021. Cited on pages 6 and 20.

[CH21b] Francesco Croce and Matthias Hein. Adversarial robustness against multiple l_p-threat models at the price of one and how to quickly fine-tune robust models to another threat model. *arXiv.org*, abs/2105.12508, 2021. Cited on page 154.

[CH21c] Francesco Croce and Matthias Hein. Mind the box: l_1-apgd for sparse adversarial attacks on image classifiers. In *Proc. of the International Conference on Machine Learning (ICML)*, 2021. Cited on page 16.

[CHY22] Jacob Clarysse, Julia Hörmann, and Fanny Yang. Why adversarial training can hurt robust accuracy. *arXiv.org*, abs/2203.02006, 2022. Cited on page 19.

[CJW20] Jianbo Chen, Michael I. Jordan, and Martin J. Wainwright. Hopskipjumpattack: A query-efficient decision-based attack. In *Proc. of the IEEE Symposium on Security and Privacy*, 2020. Cited on page 16.

[CKF11] Ronan Collobert, Koray Kavukcuoglu, and Clément Farabet. Torch7: A matlab-like environment for machine learning. In *Advances in Neural Information Processing Systems (NeurIPS) Workshops*, 2011. Cited on pages 38 and 45.

[CKZ+18] Xiaozhi Chen, Kaustav Kundu, Yukun Zhu, Huimin Ma, Sanja Fidler, and Raquel Urtasun. 3d object proposals using stereo imagery for accurate object class detection. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 40(5):1259–1272, 2018. Cited on page 36.

[CL96] Brian Curless and Marc Levoy. A volumetric method for building complex models from range images. In *ACM Trans. on Graphics (SIGGRAPH)*, 1996. Cited on page 37.

[CLC+19] Minhao Cheng, Thong Le, Pin-Yu Chen, Huan Zhang, Jinfeng Yi, and Cho-Jui Hsieh. Query-efficient hard-label black-box attack: An optimization-based approach. In *Proc. of the International Conference on Learning Representations (ICLR)*, 2019. Cited on page 16.

[CLC+20] Minhao Cheng, Qi Lei, Pin-Yu Chen, Inderjit S. Dhillon, and Cho-Jui Hsieh. CAT: customized adversarial training for improved robustness. *arXiv.org*, abs/2002.06789, 2020. Cited on pages 18 and 120.

[CLE+19] Nicholas Carlini, Chang Liu, Úlfar Erlingsson, Jernej Kos, and Dawn Song. The secret sharer: Evaluating and testing unintended memorization in neural networks. In *USENIX Security Symposium*, 2019. Cited on page 15.

[CLL+17] Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Song. Targeted backdoor attacks on deep learning systems using data poisoning. *arXiv.org*, abs/1712.05526, 2017. Cited on page 15.

[CLS18] Qi-Zhi Cai, Chang Liu, and Dawn Song. Curriculum adversarial training. In *Proc. of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 3740–3747, 2018. Cited on page 18.

[CLW+20] J. Chen, Yixuan Li, X. Wu, Yingyu Liang, and S. Jha. Robust out-of-distribution detection in neural networks. *arXiv.org*, abs/2003.09711, 2020. Cited on page 22.

[CLW+21] Jiefeng Chen, Yixuan Li, Xi Wu, Yingyu Liang, and Somesh Jha. ATOM: robustifying out-of-distribution detection using outlier mining. In *Proc. of the European Conference on Machine Learning and Knowledge Discovery in Databases (ECML PKDD)*, 2021. Cited on page 22.

[CLZ21] Zhuotong Chen, Qianxiao Li, and Zheng Zhang. Towards robust neural networks via close-loop control. In *Proc. of the International Conference on Learning Representations (ICLR)*, 2021. Cited on page 18.

[CM99] S. Cavalieri and O. Mirabella. A novel learning algorithm which improves the partial fault tolerance of multilayer neural networks. *Neural networks: the official journal of the International Neural Network Society*, 12 1, 1999. Cited on page 22.

[CMLZ16] Xiangli Chen, Mathew Monfort, Anqi Liu, and Brian D. Ziebart. Robust covariate shift regression. In *Conference on Artificial Intelligence and Statistics (AISTATS)*, 2016. Cited on page 23.

[CMMR94] Ching-Tai Chiu, Kishan Mehrotra, Chilukuri K. Mohan, and Sanjay Ranka. Training techniques to obtain fault-tolerant neural networks. In *Annual International Symposium on Fault-Tolerant Computing*, 1994. Cited on pages 14 and 21.

[CMS+22] Francesco Crecchi, Marco Melis, Angelo Sotgiu, Davide Bacciu, and Battista Biggio. FADER: fast adversarial example rejection. *Neurocomputing*, 470:257–268, 2022. Cited on page 22.

[CMZK20] Lin Chen, Yifei Min, Mingrui Zhang, and Amin Karbasi. More data can expand the generalization gap between adversarially robust and standard models. In *Proc. of the International Conference on Machine Learning (ICML)*, 2020. Cited on page 19.

[CRC+22] Jiefeng Chen, Jayaram Raghuram, Jihye Choi, Xi Wu, Yingyu Liang, and Somesh Jha. Revisiting adversarial robustness of classifiers with a reject option. In *Proc. of the Conference on Artificial Intelligence (AAAI) Workshops*, 2022. Cited on page 18.

[CRH20] Francesco Croce, Jonas Rauber, and Matthias Hein. Scaling up the randomized gradient-free adversarial attack reveals overestimation of robustness using established attacks. *International Journal of Computer Vision (IJCV)*, 128(4):1028–1046, 2020. Cited on page 16.

[CRK19] Jeremy M. Cohen, Elan Rosenfeld, and J. Zico Kolter. Certified adversarial robustness via randomized smoothing. In *Proc. of the International Conference on Machine Learning (ICML)*, 2019. Cited on pages 18 and 153.

[CRS+19] Yair Carmon, Aditi Raghunathan, Ludwig Schmidt, John C. Duchi, and Percy Liang. Unlabeled data improves adversarial robustness. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. Cited on pages 18, 65, 66, 67, 73, 79, and 239.

[CS19] Safa Cicek and Stefano Soatto. Input and weight space smoothing for semi-supervised learning. In *Proc. of the IEEE International Conference on Computer Vision (ICCV) Workshops*, 2019. Cited on page 67.

[CSC+19] Nandhini Chandramoorthy, Karthik Swaminathan, Martin Cochet, Arun Paidimarri, Schuyler Eldridge, Rajiv V. Joshi, Matthew M. Ziegler, Alper Buyuktosunoglu, and Pradip Bose. Resilient low voltage accelerators for high energy efficiency. In *IEEE International Symposium on High Performance Computer Architecture (HPCA)*, 2019. Cited on pages 6, 20, 84, 85, 86, 87, 91, 111, and 155.

[CSC+20] Minhao Cheng, Simranjit Singh, Patrick H. Chen, Pin-Yu Chen, Sijia Liu, and Cho-Jui Hsieh. Sign-opt: A query-efficient hard-label adversarial attack. In *Proc. of the International Conference on Learning Representations (ICLR)*, 2020. Cited on page 16.

[CSG20] Gilad Cohen, Guillermo Sapiro, and Raja Giryes. Detecting adversarial samples using influence functions and nearest neighbors. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. Cited on page 19.

[CSK17] Nicholas Cheney, Martin Schrimpf, and Gabriel Kreiman. On the robustness of convolutional neural networks to internal architecture and weight perturbations. *arXiv.org*, abs/1703.08245, 2017. Cited on page 21.

[CT11] Fatih Calakli and Gabriel Taubin. SSD: smooth signed distance surface reconstruction. *Computer Graphics Forum*, 30(7):1993–2002, 2011. Cited on page 12.

[CTOF20] Alvin Chan, Yi Tay, Yew-Soon Ong, and Jie Fu. Jacobian adversarially regularized networks for robustness. In *Proc. of the International Conference on Learning Representations (ICLR)*, 2020. Cited on page 17.

[CTV19] Marco Cuturi, Olivier Teboul, and Jean-Philippe Vert. Differentiable ranking and sorting using optimal transport. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. Cited on pages 139 and 140.

[cup] Cupy: A numpy-compatible array library accelerated by cuda. https://cupy.dev/. Cited on page 246.

[CW16] Nicholas Carlini and David A. Wagner. Defensive distillation is not robust to adversarial examples. *arXiv.org*, abs/1607.04311, 2016. Cited on page 17.

[CW17a] Nicholas Carlini and David Wagner. Adversarial examples are not easily detected: Bypassing ten detection methods. *arXiv.org*, abs/1705.07263, 2017. Cited on pages 7, 17, and 19.

[CW17b] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *Proc. of the IEEE Symposium on Security and Privacy*, 2017. Cited on pages 16, 53, 62, 63, 128, 228, and 229.

[CW17c] Nicholas Carlini and David A. Wagner. Magnet and "efficient defenses against adversarial attacks" are not robust to adversarial examples. *arXiv.org*, abs/1711.08478, 2017. Cited on page 18.

[CW18] Nicholas Carlini and David A. Wagner. Audio adversarial examples: Targeted attacks on speech-to-text. In *Proc. of the IEEE Symposium on Security and Privacy*, 2018. Cited on page 14.

[CWV+18] Jungwook Choi, Zhuo Wang, Swagath Venkataramani, Pierce I-Jen Chuang, Vijayalakshmi Srinivasan, and Kailash Gopalakrishnan. PACT: parameterized clipping activation for quantized neural networks. *arXiv.org*, abs/1805.06085, 2018. Cited on pages 21 and 106.

[CYG+17] Kevin K. Chang, Abdullah Giray Yaalikçi, Saugata Ghose, Aditya Agrawal, Niladrish Chatterjee, Abhijith Kashyap, Donghyuk Lee, Mike O'Connor, Hasan Hassan, and Onur Mutlu. Understanding reduced-voltage operation in modern DRAM devices: Experimental characterization, analysis, and mechanisms. *Proc. of the ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems (POMACS)*, 1(1), 2017. Cited on page 20.

[CYZF20] Dingfan Chen, Ning Yu, Yang Zhang, and Mario Fritz. Gan-leaks: A taxonomy of membership inference attacks against generative models. In *Proc. of the ACM Conference on Computer and Communications Security (CCS)*, 2020. Cited on page 15.

[CZ19] Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. Cited on page 13.

[CZC+17] Hongge Chen, Huan Zhang, Pin-Yu Chen, Jinfeng Yi, and Cho-Jui Hsieh. Show-and-fool: Crafting adversarial examples for neural image captioning. *arXiv.org*, abs/1712.02051, 2017. Cited on page 16.

[CZHW20] Weilun Chen, Zhaoxiang Zhang, Xiaolin Hu, and Baoyuan Wu. Boosting decision-based black-box adversarial attacks with random sign flip. In *Proc. of the European Conference on Computer Vision (ECCV)*, 2020. Cited on page 16.

[CZL+21] Tianlong Chen, Zhenyu Zhang, Sijia Liu, Shiyu Chang, and Zhangyang Wang. Robust overfitting may be mitigated by properly learned smoothening. In *Proc. of the International Conference on Learning Representations (ICLR)*, 2021. Cited on page 19.

[CZM+19] Ekin D. Cubuk, Barret Zoph, Dandelion Mané, Vijay Vasudevan, and Quoc V. Le. Autoaugment: Learning augmentation strategies from data. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. Cited on pages 65, 67, 73, 79, 97, 145, 237, and 268.

[CZS+17] Pin-Yu Chen, Huan Zhang, Yash Sharma, Jinfeng Yi, and Cho-Jui Hsieh. ZOO: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In *Proc. of the ACM Workshop on Artificial Intelligence and Security*, 2017. Cited on page 16.

[DCP+21] Yinpeng Dong, Shuyu Cheng, Tianyu Pang, Hang Su, and Jun Zhu. Query-efficient black-box adversarial attacks guided by a transfer-based prior. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 2021. Cited on page 16.

[DDM+04] Nilesh N. Dalvi, Pedro M. Domingos, Mausam, Sumit K. Sanghai, and Deepak Verma. Adversarial classification. In *Proc. of the ACM International Conference on Knowledge Discovery & Data Mining*, 2004. Cited on page 14.

[DFC+15] Zidong Du, Robert Fasthuber, Tianshi Chen, Paolo Ienne, Ling Li, Tao Luo, Xiaobing Feng, Yunji Chen, and Olivier Temam. Shidiannao: shifting vision processing closer to the sensor. In *ACM/IEEE Annual International Symposium on Computer Architecture (ISCA)*, 2015. Cited on page 84.

[DFD+15] Jiachao Deng, Yuntan Fang, Zidong Du, Ying Wang, Huawei Li, Olivier Temam, Paolo Ienne, David Novo, Xiaowei Li, Yunji Chen, and Chengyong Wu. Retraining-based timing error mitigation for hardware neural networks. In *Proc. of the Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2015. Cited on page 22.

[DFY+21] Yinpeng Dong, Qi-An Fu, Xiao Yang, Wenzhao Xiang, Tianyu Pang, Hang Su, Jun Zhu, Jiayu Tang, Yuefeng Chen, Xiaofeng Mao, Yuan He, Hui Xue, Chao Li, Ye Liu, Qilong Zhang, Lianli Gao, Yunrui Yu, Xitong Gao, Zhe Zhao, Daquan Lin, Jiadong Lin, Chuanbiao Song, Zihao Wang, Zhennan Wu, Yang Guo, Jiequan Cui, Xiaogang Xu, and Pengguang Chen. Adversarial attacks on ML defense models competition. *arXiv.org*, abs/2110.08042, 2021. Cited on page 17.

[DG17] Dheeru Dua and Casey Graff. UCI machine learning repository. http://archive.ics.uci.edu/ml, 2017. Cited on pages 144 and 267.

[DHHR20] Edgar Dobriban, Hamed Hassani, David Hong, and Alexander Robey. Provable tradeoffs in adversarially robust classification. *arXiv.org*, abs/2006.05161, 2020. Cited on page 19.

[DL21] Frank Dellaert and Yen-Chen Lin. Neural volume rendering: Nerf and beyond. *arXiv.org*, abs/2101.05204, 2021. Cited on page 13.

[DLJ+20] Xinshuai Dong, Hong Liu, Rongrong Ji, Liujuan Cao, Qixiang Ye, Jianzhuang Liu, and Qi Tian. Api-net: Robust generative classifier via a single discriminator. In *Proc. of the European Conference on Computer Vision (ECCV)*, 2020. Cited on page 17.

[DLP+18] Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Hang Su, Jun Zhu, Xiaolin Hu, and Jianguo Li. Boosting adversarial attacks with momentum. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. Cited on pages 16, 73, 97, 124, and 228.

[DLS21a] Chengyu Dong, Liyuan Liu, and Jingbo Shang. Data profiling for adversarial training: On the ruin of problematic data. *arXiv.org*, abs/2102.07437, 2021. Cited on page 19.

[DLS21b] Chengyu Dong, Liyuan Liu, and Jingbo Shang. Double descent in adversarial training: An implicit label noise perspective. *arXiv.org*, abs/2110.03135, 2021. Cited on page 19.

[DLT+18] Hanjun Dai, Hui Li, Tian Tian, Xin Huang, Lin Wang, Jun Zhu, and Le Song. Adversarial attack on graph structured data. In *Proc. of the International Conference on Machine Learning (ICML)*, 2018. Cited on page 16.

[DMM18] Beranger Dumont, Simona Maggio, and Pablo Montalvo. Robustness of rotation-equivariant networks to adversarial perturbations. *arXiv.org*, abs/1802.06627, 2018. Cited on page 17.

[DMM21] Sihui Dai, Saeed Mahloujifar, and Prateek Mittal. Parameterizing activation functions for adversarial robustness. *arXiv.org*, abs/2110.05626, 2021. Cited on page 18.

[DMP+18] Ambra Demontis, Marco Melis, Maura Pintor, Matthew Jagielski, Battista Biggio, Alina Oprea, Cristina Nita-Rotaru, and Fabio Roli. On the intriguing connections of regularization, input gradients and transferability of evasion and poisoning attacks. *arXiv.org*, abs/1809.02861, 2018. Cited on page 16.

[DMP+19] Ambra Demontis, Marco Melis, Maura Pintor, Matthew Jagielski, Battista Biggio, Alina Oprea, Cristina Nita-Rotaru, and Fabio Roli. Why do adversarial attacks transfer? explaining transferability of evasion and poisoning attacks. In *USENIX Security Symposium*, 2019. Cited on page 16.

[DN10] Dmitry Devetyarov and Ilia Nouretdinov. Prediction with confidence based on a random forest classifier. In *Proc. of the International Conference on Artificial Intelligence Applications and Innovations (IFIP)*, 2010. Cited on page 23.

[DN19] Angela Dai and Matthias Nießner. Scan2mesh: From unstructured range scans to 3d meshes. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. Cited on page 13.

[DN21] Prafulla Dhariwal and Alex Nichol. Diffusion models beat gans on image synthesis. In *Proc. of the International Conference on Learning Representations (ICLR)*, 2021. Cited on page 153.

[DPA+14] Zidong Du, K. Palem, L. Avinash, O. Temam, Yunji Chen, and Chengyong Wu. Leveraging the error resilience of machine-learning applications for designing highly energy efficient accelerators. *Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2014. Cited on page 22.

[DPBB17] Laurent Dinh, Razvan Pascanu, Samy Bengio, and Yoshua Bengio. Sharp minima can generalize for deep nets. In *Proc. of the International Conference on Machine Learning (ICML)*, 2017. Cited on pages 19, 67, 70, 71, 72, and 102.

[DPRR13] A. Dame, V.A. Prisacariu, C.Y. Ren, and I. Reid. Dense reconstruction using 3D object shape priors. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013. Cited on pages 3, 12, 28, and 46.

[DPSZ19] Yinpeng Dong, Tianyu Pang, Hang Su, and Jun Zhu. Evading defenses to transferable adversarial examples by translation-invariant attacks. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. Cited on page 16.

[DQN17] Angela Dai, Charles Ruizhongtai Qi, and Matthias Nießner. Shape completion using 3d-encoder-predictor cnns and shape synthesis. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. Cited on pages 3, 12, 13, 27, 28, 29, 38, 39, 41, 43, and 46.

[DS20] Jacob Dumford and Walter J. Scheirer. Backdooring convolutional neural networks via targeted weight perturbations. In *Proc. of the IEEE International Joint Conference on Biometrics, IJCB 2017*, pages 1–9. IEEE, 2020. Cited on pages 14, 15, and 21.

[dSBB+18] Jacson Rodrigues Correia da Silva, Rodrigo Ferreira Berriel, Claudine Badue, Alberto Ferreira de Souza, and Thiago Oliveira-Santos. Copycat CNN: stealing knowledge by persuading confession with random non-labeled data. In *International Joint Conference on Neural Networks (IJCNN)*, 2018. Cited on page 15.

[DSLH20] Gavin Weiguang Ding, Yash Sharma, Kry Yik Chau Lui, and Ruitong Huang. MMA training: Direct input space margin maximization through adversarial training. In *Proc. of the International Conference on Learning Representations (ICLR)*, 2020. Cited on page 18.

[DSSC08] John C. Duchi, Shai Shalev-Shwartz, Yoram Singer, and Tushar Chandra. Efficient projections onto the l_1-ball for learning in high dimensions. In *Proc. of the International Conference on Machine Learning (ICML)*, 2008. Cited on page 127.

[DT17] Terrance Devries and Graham W. Taylor. Improved regularization of convolutional neural networks with cutout. *arXiv.org*, abs/1708.04552, 2017. Cited on pages 97, 145, 237, and 268.

[DVK98] Dipti Deodhare, M. Vidyasagar, and S. Sathiya Keerthi. Synthesis of fault-tolerant feedforward neural networks using minimax optimization. *IEEE Trans. on Neural Networks (TNN)*, 9(5):891–900, 1998. Cited on pages 14 and 21.

[DVK17] Finale Doshi-Velez and Been Kim. Towards a rigorous science of interpretable machine learning. *arXiv.org*, abs/1702.08608, 2017. Cited on page 15.

[DWJ19] Gavin Weiguang Ding, Luyu Wang, and Xiaomeng Jin. advertorch v0.1: An adversarial robustness toolbox based on pytorch. *arXiv.org*, abs/1902.07623, 2019. Cited on page 17.

[Dwo06] Cynthia Dwork. Differential privacy. In *International Colloquium on Automata, Languages and Programming (ICALP)*, 2006. Cited on page 15.

[DWR18] Robin Dunn, Larry Wasserman, and Aaditya Ramdas. Distribution-free prediction sets with random effects. *arXiv.org*, 2018. Cited on page 23.

[DXY+22] Yinpeng Dong, Ke Xu, Xiao Yang, Tianyu Pang, Zhijie Deng, Hang Su, and Jun Zhu. Exploring memorization in adversarial training. In *Proc. of the International Conference on Learning Representations (ICLR)*, 2022. Cited on page 19.

[DZGZ21] Zhun Deng, Linjun Zhang, Amirata Ghorbani, and James Zou. Improving adversarial robustness via unlabeled out-of-domain data. In *Conference on Artificial Intelligence and Statistics (AISTATS)*, 2021. Cited on page 17.

[ECW20] Hasan Ferit Eniser, Maria Christakis, and Valentin Wüstholz. RAID: randomized adversarial-input detection for neural networks. *arXiv.org*, abs/2002.02776, 2020. Cited on page 19.

[EF15] David Eigen and Rob Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *Proc. of the IEEE International Conference on Computer Vision (ICCV)*, 2015. Cited on page 37.

[EGS19] Gamaleldin F. Elsayed, Ian J. Goodfellow, and Jascha Sohl-Dickstein. Adversarial reprogramming of neural networks. In *Proc. of the International Conference on Learning Representations (ICLR)*, 2019. Cited on page 15.

[EIA18] Logan Engstrom, Andrew Ilyas, and Anish Athalye. Evaluating and understanding the robustness of adversarial logit pairing. *arXiv.org*, abs/1807.10272, 2018. Cited on page 18.

[EIS+19] Logan Engstrom, Andrew Ilyas, Hadi Salman, Shibani Santurkar, and Dimitris Tsipras. Robustness (python library). https://github.com/MadryLab/robustness, 2019. Cited on page 17.

[EPF14] David Eigen, Christian Puhrsch, and Rob Fergus. Depth map prediction from a single image using a multi-scale deep network. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2014. Cited on page 37.

[ESL16] Francis Engelmann, Jörg Stückler, and Bastian Leibe. Joint object pose estimation and shape reconstruction in urban street scenes using 3D shape priors. In *Proc. of the German Conference on Pattern Recognition (GCPR)*, 2016. Cited on pages 3, 12, 27, 28, 29, 37, 39, 41, and 46.

[ESL17] Francis Engelmann, Jörg Stückler, and Bastian Leibe. SAMP: shape and motion priors for 4d vehicle reconstruction. In *Proc. of the IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 400–408, 2017. Cited on page 46.

[ETSM17] Logan Engstrom, Dimitris Tsipras, Ludwig Schmidt, and Aleksander Madry. A rotation and a translation suffice: Fooling CNNs with simple transformations. *arXiv.org*, abs/1712.02779, 2017. Cited on pages 14, 17, and 58.

[EUD18] Stefan Elfwing, Eiji Uchibe, and Kenji Doya. Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. *Neural Networks*, 107, 2018. Cited on pages 67, 73, 237, and 240.

[FBR21] Shai Feldman, Stephen Bates, and Yaniv Romano. Improving conditional coverage via orthogonal quantile regression. *arXiv.org*, abs/2106.00394, 2021. Cited on page 23.

[FCSG17] Reuben Feinman, Ryan R Curtin, Saurabh Shintre, and Andrew B Gardner. Detecting adversarial samples from artifacts. *arXiv.org*, abs/1703.00410, 2017. Cited on page 19.

[FH13] Yasutaka Furukawa and Carlos Hernandez. Multi-view stereo: A tutorial. *Foundations and Trends in Computer Graphics and Vision*, 9(1-2):1–148, 2013. Cited on pages 3 and 27.

[FKMB17] Volker Fischer, Mummadi Chaithanya Kumar, Jan Hendrik Metzen, and Thomas Brox. Adversarial examples for semantic image segmentation. *Proc. of the International Conference on Learning Representations (ICLR) Workshops*, 2017. Cited on page 14.

[FKMN21] Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. Sharpness-aware minimization for efficiently improving generalization. In *Proc. of the International Conference on Learning Representations (ICLR)*, 2021. Cited on pages 14, 19, and 154.

[FMAJB16] Michael Firman, Oisin Mac Aodha, Simon Julier, and Gabriel J. Brostow. Structured prediction of unobserved voxels from a single depth image. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. Cited on pages 3, 12, and 46.

[FMDF16] Alhussein Fawzi, Seyed-Mohsen Moosavi-Dezfooli, and Pascal Frossard. Robustness of classifiers: from adversarial to random noise. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2016. Cited on page 231.

[FQ21] Omobayode Fagbohungbe and Lijun Qian. Benchmarking inference performance of deep learning models on analog devices. In *International Joint Conference on Neural Networks (IJCNN)*, 2021. Cited on pages 20 and 155.

[FSG17] Haoqiang Fan, Hao Su, and Leonidas J. Guibas. A point set generation network for 3d object reconstruction from a single image. *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. Cited on pages 12, 13, 28, and 46.

[FSJB21] Adam Fisch, Tal Schuster, Tommi S. Jaakkola, and Regina Barzilay. Few-shot conformal prediction with auxiliary tasks. In *Proc. of the International Conference on Machine Learning (ICML)*, 2021. Cited on page 23.

[FSJB22] Adam Fisch, Tal Schuster, Tommi Jaakkola, and Regina Barzilay. Conformal prediction sets with limited false positives. *arXiv.org*, abs/2202.07650, 2022. Cited on page 23.

[FSP+06] Prahlad Fogla, Monirul I. Sharif, Roberto Perdisci, Oleg M. Kolesnikov, and Wenke Lee. Polymorphic blending attacks. In Angelos D. Keromytis, editor, *USENIX Security Symposium*. USENIX Association, 2006. Cited on page 14.

[FSTF16] Alhussein Fawzi, Horst Samulowitz, Deepak S. Turaga, and Pascal Frossard. Adaptive data augmentation for image classification. In *Proc. of the IEEE International Conference on Image Processing (ICIP)*, 2016. Cited on page 58.

[FXMY14] Jiashi Feng, Huan Xu, Shie Mannor, and Shuicheng Yan. Robust logistic regression and classification. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2014. Cited on page 14.

[FZT19] Farzan Farnia, Jesse M. Zhang, and David Tse. Generalizable adversarial training via spectral normalization. In *Proc. of the International Conference on Learning Representations (ICLR)*, 2019. Cited on page 66.

[GAGM15] Saurabh Gupta, Pablo Andrés Arbeláez, Ross B. Girshick, and Jitendra Malik. Aligning 3D models to RGB-D images of cluttered scenes. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. Cited on pages 12, 29, 39, 41, 43, and 46.

[Gal16] Yarin Gal. *Uncertainty in Deep Learning*. PhD thesis, University of Cambridge, 2016. Cited on page 22.

[GaWH+21] Yulan Guo, anyun Wang, Qingyong Hu, Hao Liu, Li Liu, and Mohammed Bennamoun. Deep learning for 3d point clouds: A survey. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 43(12):4338–4364, 2021. Cited on page 13.

[GAZ19] Amirata Ghorbani, Abubakar Abid, and James Y. Zou. Interpretation of neural networks is fragile. In *Proc. of the Conference on Artificial Intelligence (AAAI)*, 2019. Cited on page 15.

[GB10a] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Conference on Artificial Intelligence and Statistics (AISTATS)*, 2010. Cited on page 38.

[GB10b] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Conference on Artificial Intelligence and Statistics (AISTATS)*, 2010. Cited on page 52.

[GBY+18] Leilani H. Gilpin, David Bau, Ben Z. Yuan, Ayesha Bajwa, Michael A. Specter, and Lalana Kagal. Explaining explanations: An overview of interpretability of machine learning. In *Proc. of the IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, 2018. Cited on page 15.

[GCJ+21] Washington Garcia, Pin-Yu Chen, Somesh Jha, Scott Clouse, and Kevin R. B. Butler. Hard-label manifolds: Unexpected advantages of query efficiency for finding on-manifold adversarial examples. *arXiv.org*, abs/2103.03325, 2021. Cited on page 17.

[GCP+09] Zheng Guo, Andrew Carlson, Liang-Teck Pang, Kenneth Duong, Tsu-Jae King Liu, and Borivoje Nikolic. Large-scale SRAM variability characterization in 45 nm CMOS. *IEEE Journal of Solid-State Circuits*, 44(11), 2009. Cited on page 84.

[GCV+19] Kyle Genova, Forrester Cole, Daniel Vlasic, Aaron Sarna, William T. Freeman, and Thomas A. Funkhouser. Learning shape templates with structured implicit functions. In *Proc. of the IEEE International Conference on Computer Vision (ICCV)*, 2019. Cited on page 13.

[GDAT18] Alexander Goncharenko, Andrey Denisov, Sergey Alyamkin, and Evgeny Terentev. Fast adjustable threshold for uniform neural network quantization. *arXiv.org*, abs/1812.07872, 2018. Cited on page 21.

[GDS+18] Sven Gowal, Krishnamurthy Dvijotham, Robert Stanforth, Rudy Bunel, Chongli Qin, Jonathan Uesato, Relja Arandjelovic, Timothy A. Mann, and Pushmeet Kohli. On the effectiveness of interval bound propagation for training verifiably robust models. *arXiv.org*, abs/1810.12715, 2018. Cited on page 18.

[GE18] Izhak Golan and Ran El-Yaniv. Deep anomaly detection using geometric transformations. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018. Cited on page 22.

[GFFG20] Micah Goldblum, Liam Fowl, Soheil Feizi, and Tom Goldstein. Adversarially robust distillation. In *Proc. of the Conference on Artificial Intelligence (AAAI)*, 2020. Cited on page 17.

[GFK+18] Thibault Groueix, Matthew Fisher, Vladimir G. Kim, Bryan C. Russell, and Mathieu Aubry. A papier-mâché approach to learning 3d surface generation. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. Cited on page 12.

[GFRG16] Rohit Girdhar, David F. Fouhey, Mikel Rodriguez, and Abhinav Gupta. Learning a predictable and generative vector representation for objects. In *Proc. of the European Conference on Computer Vision (ECCV)*, 2016. Cited on pages 12, 28, 40, and 46.

[GG14] Samuel Gershman and Noah D. Goodman. Amortized inference in probabilistic reasoning. In *Proc. of the Annual Meeting of the Cognitive Science Society*, 2014. Cited on pages 29 and 32.

[GG15] Fatma Güney and Andreas Geiger. Displets: Resolving stereo ambiguities using object knowledge. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. Cited on pages 12 and 36.

[GG16] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *Proc. of the International Conference on Machine Learning (ICML)*, pages 1050–1059, 2016. Cited on page 14.

[GGT+19] Angus Galloway, Anna Golubeva, Thomas Tanay, Medhat Moussa, and Graham W. Taylor. Batch normalization is a cause of adversarial vulnerability. *arXiv.org*, abs/1905.02161, 2019. Cited on pages 100 and 155.

[GGY+19] Chuan Guo, Jacob R. Gardner, Yurong You, Andrew Gordon Wilson, and Kilian Q. Weinberger. Simple black-box adversarial attacks. In *Proc. of the International Conference on Machine Learning (ICML)*, 2019. Cited on page 16.

[GKB+19] Shrikanth Ganapathy, John Kalamatianos, Bradford M. Beckmann, Steven Raasch, and Lukasz G. Szafaryn. Killi: Runtime fault classification to deploy low voltage caches without MBIST. In *IEEE International Symposium on High Performance Computer Architecture (HPCA)*, 2019. Cited on pages 14, 20, 87, and 88.

[GKD+21] Amir Gholami, Sehoon Kim, Zhen Dong, Zhewei Yao, Michael W. Mahoney, and Kurt Keutzer. A survey of quantization methods for efficient neural network inference. *arXiv.org*, abs/2103.13630, 2021. Cited on page 21.

[GKGL20] Siddhant Garg, Adarsh Kumar, Vibhor Goel, and Yingyu Liang. Can adversarial weight perturbations inject neural backdoors. In Mathieu d'Aquin, Stefan Dietze, Claudia Hauff, Edward Curry, and Philippe Cudré-Mauroux, editors, *Proc. of the ACM International Conference on Information and Knowledge Management (CIKM)*, 2020. Cited on page 15.

[GKKR17] Shrikanth Ganapathy, John Kalamatianos, Keith Kasprak, and Steven Raasch. On characterizing near-threshold SRAM failures in FinFET technology. In *ACM/ESDA/IEEE Design Automation Conference (DAC)*, 2017. Cited on pages 14, 20, 84, 85, 87, and 88.

[GLU12] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? The KITTI vision benchmark suite. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012. Cited on pages 1, 2, 3, 12, 27, 28, 30, 35, and 46.

[GMD+18] Timon Gehr, Matthew Mirman, Dana Drachsler-Cohen, Petar Tsankov, Swarat Chaudhuri, and Martin T. Vechev. AI2: safety and robustness certification of neural networks with abstract interpretation. In *Proc. of the IEEE Symposium on Security and Privacy*, pages 3–18, 2018. Cited on page 18.

[GMF+18] Justin Gilmer, Luke Metz, Fartash Faghri, Samuel S. Schoenholz, Maithra Raghu, Martin Wattenberg, and Ian Goodfellow. Adversarial spheres. *Proc. of the International Conference on Learning Representations (ICLR) Workshops*, 2018. Cited on pages 4, 5, 17, 50, 57, and 63.

[GMP+17] Kathrin Grosse, Praveen Manoharan, Nicolas Papernot, Michael Backes, and Patrick McDaniel. On the (statistical) detection of adversarial examples. *arXiv.org*, abs/1702.06280, 2017. Cited on page 19.

[GMVP21] Diego Gragnaniello, Francesco Marra, Luisa Verdoliva, and Giovanni Poggi. Perceptual quality-preserving black-box attack against deep learning image classifiers. *Pattern Recognition Letters*, 147:142–149, 2021. Cited on page 16.

[GP18] Jia Guo and Miodrag Potkonjak. Watermarking deep neural networks for embedded systems. In Iris Bahar, editor, *Proc. of the International Conference on Computer-Aided Design*, page 133. ACM, 2018. Cited on page 15.

[GPC+16] Varun Gulshan, Lily Peng, Marc Coram, Martin C. Stumpe, Derek Wu, Arunachalam Narayanaswamy, Subhashini Venugopalan, Kasumi Widner, Tom Madams, Jorge Cuadros, Ramasamy Kim, Rajiv Raman, Philip C. Nelson, Jessica L. Mega, and Dale R. Webster. Development and Validation of a Deep Learning Algorithm for Detection of Diabetic Retinopathy in Retinal Fundus Photographs. *Journal of the American Medical Association (JAMA)*, 316(22):2402–2410, 12 2016. Cited on page 1.

[GPM+14a] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2014. Cited on page 12.

[GPM+14b] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2014. Cited on page 153.

[GPSB18] Kathrin Grosse, David Pfaff, Michael T. Smith, and Michael Backes. The limitations of model uncertainty in adversarial settings. *arXiv.org*, abs/1812.02606, 2018. Cited on page 14.

[GPSW17] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On calibration of modern neural networks. In *Proc. of the International Conference on Machine Learning (ICML)*, 2017. Cited on pages 22 and 145.

[GQB19] Ian Goodfellow, Yao Qin, and David Berthelot. Evaluation methodology for attacks against confidence thresholding models. https://openreview.net/forum?id=H1gopiA9tQ, 2019. Cited on page 119.

[GQH+20] Sven Gowal, Chongli Qin, Po-Sen Huang, A. Taylan Cemgil, Krishnamurthy Dvijotham, Timothy A. Mann, and Pushmeet Kohli. Achieving robustness in the wild via adversarial mixing with disentangled representations. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. Cited on pages 17 and 153.

[GQU+20] Sven Gowal, Chongli Qin, Jonathan Uesato, Timothy A. Mann, and Pushmeet Kohli. Uncovering the limits of adversarial training against norm-bounded adversarial examples. *arXiv.org*, abs/2010.03593, 2020. Cited on pages 18, 67, 73, 153, and 236.

[GRA+21] Kartik Gupta, Amir Rahimi, Thalaiyasingam Ajanthan, Thomas Mensink, Cristian Sminchisescu, and Richard Hartley. Calibration of neural networks using splines. In *Proc. of the International Conference on Learning Representations (ICLR)*, 2021. Cited on page 22.

[GRCvdM18] Chuan Guo, Mayank Rana, Moustapha Cissé, and Laurens van der Maaten. Countering adversarial images using input transformations. In *Proc. of the International Conference on Learning Representations (ICLR)*, 2018. Cited on page 17.

[GRF+20] Yue Gao, Harrison Rosenberg, Kassem Fawaz, Somesh Jha, and Justin Hsu. Analyzing accuracy loss in randomized smoothing defenses. *arXiv.org*, abs/2003.01595, 2020. Cited on page 19.

[GRM+19] Robert Geirhos, Patricia Rubisch, Claudio Michaelis, Matthias Bethge, Felix A. Wichmann, and Wieland Brendel. Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness. In *Proc. of the International Conference on Learning Representations (ICLR)*, 2019. Cited on page 14.

[GRW+21] Sven Gowal, Sylvestre-Alvise Rebuffi, Olivia Wiles, Florian Stimberg, Dan Andrei Calian, and Timothy Mann. Improving robustness using generated data. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. Cited on page 18.

[GSS15] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *Proc. of the International Conference on Learning Representations (ICLR)*, 2015. Cited on pages 4, 15, 16, 17, 18, 50, 53, 116, and 231.

[GSS22] Yong Guo, David Stutz, and Bernt Schiele. Improving corruption and adversarial robustness by enhancing weak subnets. *arXiv.org*, abs/2201.12765, 2022. Cited on page 10.

[GTA+21] Jakob Gawlikowski, Cedrique Rovile Njieutcheu Tassi, Mohsin Ali, Jongseok Lee, Matthias Humt, Jianxiang Feng, Anna M. Kruspe, Rudolph Triebel, Peter Jung, Ribana Roscher, Muhammad Shahzad, Wen Yang, Richard Bamler, and Xiao Xiang Zhu. A survey of uncertainty in deep neural networks. *arXiv.org*, abs/2107.03342, 2021. Cited on page 14.

[Guo18] Yunhui Guo. A survey on methods and theories of quantized neural networks. *arXiv.org*, abs/1808.04752, 2018. Cited on pages 21 and 155.

[GUQ+19] Sven Gowal, Jonathan Uesato, Chongli Qin, Po-Sen Huang, Timothy A. Mann, and Pushmeet Kohli. An alternative surrogate loss for pgd-based adversarial testing. *arXiv.org*, abs/1910.09338, 2019. Cited on page 16.

[GvGS16] Hayit Greenspan, Bram van Ginneken, and Ronald M. Summers. Guest editorial deep learning in medical imaging: Overview and future promise of an exciting new technique. *IEEE Transactions on Medical Imaging*, 35(5):1153–1159, 2016. Cited on page 1.

[GVV98] Alexander Gammerman, Volodya Vovk, and Vladimir Vapnik. Learning by transduction. In *Proc. of the Conference on Uncertainty in Artificial Intelligence (UAI)*, 1998. Cited on pages 7, 14, 22, and 23.

[GWDR22] Asaf Gendler, Tsui-Wei Weng, Luca Daniel, and Yaniv Romano. Adversarially robust conformal prediction. In *Proc. of the International Conference on Learning Representations (ICLR)*, 2022. Cited on page 156.

[GWK17] Zhitao Gong, Wenlu Wang, and Wei-Shinn Ku. Adversarial and clean data are not twins. *arXiv.org*, abs/1704.04960, 2017. Cited on page 19.

[GXY+20] Dou Goodman, Hao Xin, Wang Yang, Wu Yuesheng, Xiong Junfeng, and Zhang Huan. Advbox: a toolbox to generate adversarial examples that fool neural networks. *arXiv.org*, 2001.05574, 2020. Cited on page 17.

[HA04] Victoria J. Hodge and Jim Austin. A survey of outlier detection methodologies. *Artificial Intelligence Review*, 22(2):85–126, 2004. Cited on page 14.

[HA17] Matthias Hein and Maksym Andriushchenko. Formal guarantees on the robustness of a classifier against adversarial manipulation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017. Cited on page 17.

[HAB19] Matthias Hein, Maksym Andriushchenko, and Julian Bitterwolf. Why relu networks yield high-confidence predictions far away from the training data and how to mitigate the problem. *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. Cited on pages 7, 22, 117, and 127.

[Hay20] Jamie Hayes. Provable trade-offs between private & robust machine learning. *arXiv.org*, abs/2006.04622, 2020. Cited on page 15.

[HBK22] Jamie Hayes, Borja Balle, and M. Pawan Kumar. Learning to be adversarially robust and differentially private. *arXiv.org*, abs/2201.02265, 2022. Cited on page 15.

[HCNB20] Tom Hennigan, Trevor Cai, Tamara Norman, and Igor Babuschkin. Haiku: Sonnet for JAX. http://github.com/deepmind/dm-haiku, 2020. Cited on page 268.

[HCS+17] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Quantized neural networks: Training neural networks with low precision weights and activations. *Journal of Machine Learning Research (JMLR)*, 18, 2017. Cited on page 21.

[HD19] Dan Hendrycks and Thomas G. Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. In *Proc. of the International Conference on Learning Representations (ICLR)*, 2019. Cited on pages 1, 14, and 126.

[Hes96] Tom Heskes. Practical confidence and prediction intervals. In *Advances in Neural Information Processing Systems (NeurIPS)*, 1996. Cited on page 14.

[HG16] Dan Hendrycks and Kevin Gimpel. Bridging nonlinearities and stochastic regularizers with gaussian error linear units. *arXiv.org*, abs/1606.08415, 2016. Cited on pages 67, 73, 237, and 240.

[HG17a] Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. In *Proc. of the International Conference on Learning Representations (ICLR)*, 2017. Cited on pages 7, 14, and 22.

[HG17b] Dan Hendrycks and Kevin Gimpel. Early methods for detecting adversarial images. In *Proc. of the International Conference on Learning Representations (ICLR)*, 2017. Cited on page 19.

[HGP19] Wilfried Haensch, Tayfun Gokmen, and Ruchir Puri. The next generation of deep learning hardware: Analog computing. *Proceedings of the IEEE*, 107(1), 2019. Cited on pages 20 and 155.

[HGXL21] Jiaxing Huang, Dayan Guan, Aoran Xiao, and Shijian Lu. RDA: robust domain adaptation via fourier adversarial attacking. In *Proc. of the IEEE International Conference on Computer Vision (ICCV)*, 2021. Cited on page 16.

[HHS20] Le-Ha Hoang, M. Hanif, and Muhammad Shafique. Ft-clipact: Resilience analysis of deep neural networks and improving their fault tolerance using clipped activation. *Proc. of the Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2020. Cited on page 22.

[HJA20] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. Cited on page 153.

[HJF+21] Marton Havasi, Rodolphe Jenatton, Stanislav Fort, Jeremiah Zhe Liu, Jasper Snoek, Balaji Lakshminarayanan, Andrew Mingbo Dai, and Dustin Tran. Training independent subnetworks for robust prediction. In *Proc. of the International Conference on Learning Representations (ICLR)*, 2021. Cited on page 14.

[HKR+20] Xiaowei Huang, Daniel Kroening, Wenjie Ruan, James Sharp, Youcheng Sun, Emese Thamo, Min Wu, and Xinping Yi. A survey of safety and trustworthiness of deep neural networks: Verification, testing, adversarial attack and defence, and interpretability. *Computer Science Review*, 37, 2020. Cited on pages 1, 13, and 15.

[HLB21] Xian-Feng Han, Hamid Laga, and Mohammed Bennamoun. Image-based 3d object reconstruction: State-of-the-art and trends in the deep learning era. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 43(5):1578–1604, 2021. Cited on page 11.

[HLH+17] Xiaoguang Han, Zhen Li, Haibin Huang, Evangelos Kalogerakis, and Yizhou Yu. High-resolution shape completion using deep neural networks for global structure and local geometry inference. In *Proc. of the IEEE International Conference on Computer Vision (ICCV)*, pages 85–93, 2017. Cited on pages 12, 13, 28, and 46.

[HLM19] Dan Hendrycks, Kimin Lee, and Mantas Mazeika. Using pre-training can improve model robustness and uncertainty. In *Proc. of the International Conference on Machine Learning (ICML)*, 2019. Cited on page 18.

[HLOB21] Joong-won Hwang, Youngwan Lee, Sungchan Oh, and Yuseok Bae. Adversarial training with stochastic weight average. In *Proc. of the IEEE International Conference on Image Processing (ICIP)*, 2021. Cited on page 66.

[HLR21] Duhun Hwang, Eunjung Lee, and Wonjong Rhee. Aid-purifier: A light auxiliary network for boosting adversarial defense. *arXiv.org*, abs/2107.06456, 2021. Cited on page 18.

[HMC+20] Dan Hendrycks, Norman Mu, Ekin Dogus Cubuk, Barret Zoph, Justin Gilmer, and Balaji Lakshminarayanan. Augmix: A simple data processing method to improve robustness and uncertainty. In *Proc. of the International Conference on Learning Representations (ICLR)*, 2020. Cited on pages 14 and 22.

[HMD19] Dan Hendrycks, Mantas Mazeika, and Thomas G. Dietterich. Deep anomaly detection with outlier exposure. In *Proc. of the International Conference on Learning Representations (ICLR)*, 2019. Cited on pages 7 and 22.

[HMKS19] Dan Hendrycks, Mantas Mazeika, Saurav Kadavath, and Dawn Song. Using self-supervised learning can improve model robustness and uncertainty. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. Cited on pages 14, 18, 22, 65, 66, 67, 73, and 238.

[HP18] Hossein Hosseini and Radha Poovendran. Semantic adversarial examples. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 1614–1619, 2018. Cited on page 17.

[HP20] Aminul Huq and Mst. Tasnim Pervin. Adversarial attacks and defense on texts: A survey. *arXiv.org*, abs/2005.14108, 2020. Cited on pages 14 and 16.

[HPG+08] Vincent Huard, Chittoor R. Parthasarathy, C. Guérin, T. Valentin, E. Pion, M. Mammasse, Nicolas Planes, and L. Camus. Nbti degradation: From transistor to sram arrays. *IEEE International Reliability Physics Symposium*, 2008. Cited on page 20.

[HPK01] Mark Handley, Vern Paxson, and Christian Kreibich. Network intrusion detection: Evasion, traffic normalization, and end-to-end protocol semantics. In *USENIX Security Symposium*, 2001. Cited on page 14.

[HPW18] Yotam Hechtlinger, Barnabás Póczos, and Larry A. Wasserman. Cautious deep learning. *arXiv.org*, abs/1805.09460, 2018. Cited on page 23.

[HRFS16] Seyyed Hossein HasanPour, Mohammad Rouhani, Mohsen Fayyaz, and Mohammad Sabokrou. Lets keep it simple, using simple architectures to outperform deeper and more complex architectures. *arXiv.org*, abs/1608.06037, 2016. Cited on pages 97 and 247.

[HRL+20] Zhezhi He, Adnan Siraj Rakin, Jingtao Li, Chaitali Chakrabarti, and Deliang Fan. Defending and harnessing the bit-flip based adversarial weight attack. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. Cited on pages 6, 21, 84, 86, 87, and 110.

[HRY19] Judy Hoffman, Daniel A. Roberts, and Sho Yaida. Robust learning with jacobian regularization. *arXiv.org*, abs/1908.02729, 2019. Cited on page 17.

[HS97] S. Hochreiter and J. Schmidhuber. Flat minima. *Neural Computation*, 9, 1997. Cited on page 67.

[HS21] Muhammad Abdullah Hanif and Muhammad Shafique. DNN-Life: An energy-efficient aging mitigation framework for improving the lifetime of on-chip weight memories in deep neural network hardware architectures. In *Proc. of the Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2021. Cited on page 20.

[HSP14] Christian Haene, Nikolay Savinov, and Marc Pollefeys. Class specific 3d object shape priors using surface normals. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014. Cited on pages 3, 12, 28, and 46.

[HTM17] Christian Hane, Shubham Tulsiani, and Jitendra Malik. Hierarchical surface prediction for 3d object reconstruction. In *Proc. of the International Conference on 3D Vision (3DV)*, 2017. Cited on page 13.

[HW21] Eyke Hüllermeier and Willem Waegeman. Aleatoric and epistemic uncertainty in machine learning: an introduction to concepts and methods. *Machine Learning*, 110(3):457–506, 2021. Cited on page 22.

[HWC+17] Warren He, James Wei, Xinyun Chen, Nicholas Carlini, and Dawn Song. Adversarial example defense: Ensembles of weak defenses are not strong. In *USENIX Security Symposium Workshops*, 2017. Cited on page 17.

[HWE+21] Hanxun Huang, Yisen Wang, Sarah Monazam Erfani, Quanquan Gu, James Bailey, and Xingjun Ma. Exploring architectural ingredients of adversarially robust deep neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. Cited on page 18.

[HZRS15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proc. of the IEEE International Conference on Computer Vision (ICCV)*, 2015. Cited on page 97.

[HZRS16a] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. Cited on pages 62, 63, 72, 97, 128, 145, 146, 230, 235, and 267.

[HZRS16b] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. Cited on page 247.

[IAMB17] Daniel Jiwoong Im, Sungjin Ahn, Roland Memisevic, and Yoshua Bengio. Denoising criterion for variational auto-encoding framework. In *Proc. of the Conference on Artificial Intelligence (AAAI)*, pages 2059–2065, 2017. Cited on pages 29, 30, 31, 34, and 46.

[ICNK17] Anastasia Ioannidou, Elisavet Chatzilari, Spiros Nikolopoulos, and Ioannis Kompatsiaris. Deep learning advances in computer vision with 3d data: A survey. *ACM Computing Surveys*, 50(2):20:1–20:38, 2017. Cited on page 13.

[ICZ+20] Berivan Isik, Kristy Choi, Xin Zheng, H.-S. Philip Wong, Stefano Ermon, Tsachy Weissman, and Armin Alaghi. Noisy neural network compression for analog storage devices. In *Advances in Neural Information Processing Systems (NeurIPS) Workshops*, 2020. Cited on pages 20 and 155.

[ICZ+21] Berivan Isik, Kristy Choi, Xin Zheng, Tsachy Weissman, Stefano Ermon, H.-S. Philip Wong, and Armin Alaghi. Neural network compression for noisy storage devices. *arXiv.org*, abs/2102.07725, 2021. Cited on pages 20 and 155.

[IEAL18] Andrew Ilyas, Logan Engstrom, Anish Athalye, and Jessy Lin. Black-box adversarial attacks with limited queries and information. In *Proc. of the International Conference on Machine Learning (ICML)*, 2018. Cited on pages 16, 50, 117, 124, and 127.

[IEM19] Andrew Ilyas, Logan Engstrom, and Aleksander Madry. Prior convictions: Black-box adversarial attacks with bandits and priors. In *Proc. of the International Conference on Learning Representations (ICLR)*, 2019. Cited on page 16.

[IJA+17] Andrew Ilyas, Ajil Jalal, Eirini Asteri, Constantinos Daskalakis, and Alexandros G. Dimakis. The robust manifold defense: Adversarial training using generative models. *arXiv.org*, abs/1712.09196, 2017. Cited on pages 17, 50, and 63.

[IPG+18] Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry P. Vetrov, and Andrew Gordon Wilson. Averaging weights leads to wider optima and better generalization. In *Proc. of the Conference on Uncertainty in Artificial Intelligence (UAI)*, 2018. Cited on pages 19, 67, 73, and 236.

[IS15a] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proc. of the International Conference on Machine Learning (ICML)*, 2015. Cited on pages 21, 37, 38, 97, 155, and 247.

[IS15b] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proc. of the International Conference on Machine Learning (ICML)*, 2015. Cited on pages 52, 53, 71, 72, 89, 92, 95, 97, 145, 230, and 235.

[IST+19] Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Logan Engstrom, Brandon Tran, and Aleksander Madry. Adversarial examples are not bugs, they are features. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. Cited on page 17.

[JBC+19] Jörn-Henrik Jacobsen, Jens Behrmann, Nicholas Carlini, Florian Tramèr, and Nicolas Papernot. Exploiting excessive invariance caused by norm-bounded adversarial robustness. *arXiv.org*, abs/1903.10484, 2019. Cited on page 118.

[JBZB19] Jörn-Henrik Jacobsen, Jens Behrmann, Richard S. Zemel, and Matthias Bethge. Excessive invariance causes adversarial vulnerability. In *Proc. of the International Conference on Learning Representations (ICLR)*, 2019. Cited on page 118.

[JCB+20] Matthew Jagielski, Nicholas Carlini, David Berthelot, Alex Kurakin, and Nicolas Papernot. High accuracy and high fidelity extraction of neural networks. In *USENIX Security Symposium*, 2020. Cited on page 15.

[JCCW20] Ziyu Jiang, Tianlong Chen, Ting Chen, and Zhangyang Wang. Robust pre-training by adversarial contrastive learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. Cited on page 18.

[JDV+14] Rasmus Ramsbøl Jensen, Anders Lindbjerg Dahl, George Vogiatzis, Engil Tola, and Henrik Aanæs. Large scale multi-view stereopsis evaluation. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014. Cited on page 37.

[JG18a] Daniel Jakubovitz and Raja Giryes. Improving DNN robustness to adversarial attacks using jacobian regularization. In *Proc. of the European Conference on Computer Vision (ECCV)*, 2018. Cited on page 17.

[JG18b] Jinyuan Jia and Neil Zhenqiang Gong. Attriguard: A practical defense against attribute inference attacks via adversarial machine learning. In *USENIX Security Symposium*. USENIX Association, 2018. Cited on page 15.

[JGBG20] Joel Janai, Fatma Güney, Aseem Behl, and Andreas Geiger. Foundations and trends in computer graphics and vision. *Foundations and Trends in Computer Graphics and Vision*, 12(1-3):1–308, 2020. Cited on pages 1, 2, and 3.

[JHHM21] Syed Mohammad Asad Hassan Jafri, Hasan Hassan, Ahmed Hemani, and Onur Mutlu. Refresh triggered computation: Improving the energy efficiency of convolutional neural network accelerators. *ACM Transactions on Architecture and Code Optimization*, 18(1):2:1–2:29, 2021. Cited on page 20.

[JHS19] Byunggill Joe, Sung Ju Hwang, and Insik Shin. Learning to disentangle robust and vulnerable features for adversarial detection. *arXiv.org*, abs/1909.04311, 2019. Cited on page 22.

[JK21] Florian Jaeckle and M. Pawan Kumar. Generating adversarial examples with graph neural networks. In *Proc. of the Conference on Uncertainty in Artificial Intelligence (UAI)*, 2021. Cited on page 16.

[JKC+18] Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew G. Howard, Hartwig Adam, and Dmitry Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. Cited on pages 21, 89, 91, 92, and 105.

[JM19] Brett Jefferson and Carlos Ortiz Marrero. Robustness metrics for real-world adversarial examples. *arXiv.org*, abs/1911.10435, 2019. Cited on page 17.

[JMGD19] Matt Jordan, Naren Manoj, Surbhi Goel, and Alexandros G. Dimakis. Quantifying perceptual distortion of adversarial examples. *arXiv.org*, 2019. Cited on page 16.

[JNM+20] Yiding Jiang, Behnam Neyshabur, Hossein Mobahi, Dilip Krishnan, and Samy Bengio. Fantastic generalization measures and where to find them. In *Proc. of the International Conference on Learning Representations (ICLR)*, 2020. Cited on pages 5, 14, 19, 66, 67, and 71.

[JOP+01] Eric Jones, Travis Oliphant, Pearu Peterson, et al. SciPy: Open source scientific tools for Python, 2001. Cited on page 35.

[JPK+21] Jongheon Jeong, Sejun Park, Minkyu Kim, Heung-Chang Lee, Do-Guk Kim, and Jinwoo Shin. Smoothmix: Training confidence-calibrated smoothed classifiers for certified robustness. *arXiv.org*, abs/2111.09277, 2021. Cited on page 18.

[JSH20] Adel Javanmard, Mahdi Soltanolkotabi, and Hamed Hassani. Precise tradeoffs in adversarial training for linear regression. In *Proc. of the Conference on Learning Theory (COLT)*, 2020. Cited on page 19.

[JSW20] Ahmadreza Jeddi, Mohammad Javad Shafiee, and Alexander Wong. A simple fine-tuning is all you need: Towards robust deep learning via adversarial fine-tuning. *arXiv.org*, abs/2012.13628, 2020. Cited on page 18.

[JSZ+19] Guoqing Jin, Shiwei Shen, Dongming Zhang, Feng Dai, and Yongdong Zhang. APE-GAN: adversarial perturbation elimination with GAN. In *Proc. of the IEEE Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019. Cited on page 17.

[JSZK15] Max Jaderberg, Karen Simonyan, Andrew Zisserman, and Koray Kavukcuoglu. Spatial transformer networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2015. Cited on pages 51 and 58.

[JWG+21] Ayush Jain, David H. Way, Vishakha Gupta, Yi Gao, Guilherme de Oliveira Marinho, Jay Hartford, R. Sayres, K. Kanada, C. Eng, Kunal Nagpal, K. Desalvo, Greg S Corrado, Lily H. Peng, Dale R. Webster, R. C. Dunn, David Coz, Susan J. Huang, Yun Liu, Peggy Bui, and Yuan Liu. Development and assessment of an artificial intelligence-based tool for skin condition diagnosis by primary care physicians and nurse practitioners in teledermatology practices. *Journal of the American Medical Association (JAMA)*, 4 4, 2021. Cited on page 142.

[JZJ+18] Yujie Ji, Xinyang Zhang, Shouling Ji, Xiapu Luo, and Ting Wang. Model reuse attacks on deep learning systems. In *Proc. of the ACM Conference on Computer and Communications Security (CCS)*, 2018. Cited on pages 14, 15, and 21.

[KAEP12] Oliver Kroemer, Heni Ben Amor, Marco Ewerton, and Jan Peters. Point cloud completion using extrusions. In *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, 2012. Cited on page 12.

[KB15a] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proc. of the International Conference on Learning Representations (ICLR)*, 2015. Cited on page 38.

[KB15b] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proc. of the International Conference on Learning Representations (ICLR)*, 2015. Cited on pages 52, 53, 70, 228, and 229.

[KBD+17] Guy Katz, Clark W. Barrett, David L. Dill, Kyle Julian, and Mykel J. Kochenderfer. Reluplex: An efficient SMT solver for verifying deep neural networks. In *Proc. of the International Conference on Computer Aided Verification (CAV)*, 2017. Cited on page 18.

[KCF20] Masahiro Kato, Zhenghang Cui, and Yoshihiro Fukuhara. ATRO: adversarial training with a rejection option. *arXiv.org*, abs/2010.12905, 2020. Cited on page 18.

[KDK+14] Yoongu Kim, Ross Daly, Jeremie Kim, Chris Fallin, Ji-Hye Lee, Donghyuk Lee, Chris Wilkerson, Konrad Lai, and Onur Mutlu. Flipping bits in memory without accessing them: An experimental study of DRAM disturbance errors. In *ACM/IEEE Annual International Symposium on Computer Architecture (ISCA)*, 2014. Cited on pages 6, 14, 20, 21, 84, 86, and 89.

[KFS18] Jernej Kos, Ian Fischer, and Dawn Song. Adversarial examples for generative models. In *Proc. of the IEEE Symposium on Security and Privacy Workshops*, 2018. Cited on page 16.

[KGB17] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. Adversarial machine learning at scale. In *Proc. of the International Conference on Learning Representations (ICLR)*, 2017. Cited on page 236.

[KGB+18] Alexey Kurakin, Ian J. Goodfellow, Samy Bengio, Yinpeng Dong, Fangzhou Liao, Ming Liang, Tianyu Pang, Jun Zhu, Xiaolin Hu, Cihang Xie, Jianyu Wang, Zhishuai Zhang, Zhou Ren, Alan L. Yuille, Sangxia Huang, Yao Zhao, Yuzhe Zhao, Zhonglin Han, Junjiajia Long, Yerkebulan Berdibekov, Takuya Akiba, Seiya Tokui, and Motoki Abe. Adversarial attacks and defences competition. *arXiv.org*, abs/1804.00097, 2018. Cited on page 17.

[KGC+18] Daniel S. Kermany, Michael Goldbaum, Wenjia Cai, Carolina C.S. Valentim, Huiying Liang, Sally L. Baxter, Alex McKeown, Ge Yang, Xiaokang Wu, Fangbing Yan, Justin Dong, Made K. Prasadha, Jacqueline Pei, Magdalene Y.L. Ting, Jie Zhu, Christina Li, Sierra Hewett, Jason Dong, Ian Ziyar, Alexander Shi, Runze Zhang, Lianghong Zheng, Rui Hou, William Shi, Xin Fu, Yaou Duan, Viet A.N. Huu, Cindy Wen, Edward D. Zhang, Charlotte L. Zhang, Oulan Li, Xiaobo Wang, Michael A. Singer, Xiaodong Sun, Jie Xu, Ali Tafreshi, M. Anthony Lewis, Huimin Xia, and Kang Zhang. Identifying medical diagnoses and treatable diseases by image-based deep learning. *Cell*, 172(5):1122–1131.e9, 2018. Cited on page 1.

[KH13] Michael M. Kazhdan and Hugues Hoppe. Screened poisson surface reconstruction. *ACM Trans. on Graphics (SIGGRAPH)*, 32(3):29, 2013. Cited on page 12.

[KH18] Marc Khoury and Dylan Hadfield-Menell. On the geometry of adversarial examples. *arXiv.org*, abs/1811.00525, 2018. Cited on pages 16, 62, 66, 117, 124, and 127.

[KHH20a] Agustinus Kristiadi, Matthias Hein, and Philipp Hennig. Being bayesian, even just a bit, fixes overconfidence in relu networks. In *Proc. of the International Conference on Machine Learning (ICML)*, 2020. Cited on pages 14 and 22.

[KHH20b] Agustinus Kristiadi, Matthias Hein, and Philipp Hennig. Fixing asymptotic uncertainty of bayesian neural networks with infinite relu features. *arXiv.org*, abs/2010.02709, 2020. Cited on page 14.

[KHH20c] Agustinus Kristiadi, Matthias Hein, and Philipp Hennig. Fixing asymptotic uncertainty of bayesian neural networks with infinite relu features. *arXiv.org*, abs/2010.02709, 2020. Cited on page 14.

[KHH21] Agustinus Kristiadi, Matthias Hein, and Philipp Hennig. Learnable uncertainty under laplace approximations. In *Proc. of the Conference on Uncertainty in Artificial Intelligence (UAI)*, 2021. Cited on page 14.

[KHM+18a] Sung Kim, Patrick Howe, Thierry Moreau, Armin Alaghi, Luis Ceze, and Visvesh Sathe. MATIC: learning around errors for efficient low-voltage neural network accelerators. In *Proc. of the Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2018. Cited on pages 6, 20, 84, 85, 86, 87, 88, 91, 97, 103, and 111.

[KHM+18b] Sung Kim, Patrick Howe, Thierry Moreau, Armin Alaghi, Luis Ceze, and Visvesh S. Sathe. Energy-efficient neural network acceleration in the presence of bit-level memory errors. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 65-I(12):4285–4298, 2018. Cited on page 20.

[KHNY21] Jaeyeon Kim, Binh-Son Hua, Duc Thanh Nguyen, and Sai-Kit Yeung. Minimal adversarial examples for deep learning on 3d point clouds. In *arXiv.org*, 2021. Cited on page 14.

[KJP+18] Ilya Kostrikov, Zhongshi Jiang, Daniele Panozzo, Denis Zorin, and Joan Bruna. Surface networks. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. Cited on page 13.

[KL17] Roman Klokov and Victor S. Lempitsky. Escape from cells: Deep kd-networks for the recognition of 3d point cloud models. In *Proc. of the IEEE International Conference on Computer Vision (ICCV)*, 2017. Cited on page 13.

[KLFG20] Aounon Kumar, Alexander Levine, Soheil Feizi, and Tom Goldstein. Certifying confidence via randomized smoothing. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. Cited on page 18.

[KLL21] Hoki Kim, Woojin Lee, and Jaewook Lee. Understanding catastrophic overfitting in single-step adversarial training. In *Proc. of the Conference on Artificial Intelligence (AAAI)*, 2021. Cited on page 18.

[KM21] Peilin Kang and Seyed-Mohsen Moosavi-Dezfooli. Understanding catastrophic overfitting in adversarial training. *arXiv.org*, abs/2105.02942, 2021. Cited on page 18.

[KMN+17] Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. In *Proc. of the International Conference on Learning Representations (ICLR)*, 2017. Cited on pages 5, 19, 66, 67, 70, 71, and 72.

[KMS19] Michael Klachko, Mohammad Reza Mahmoodi, and Dmitri B. Strukov. Improving noise tolerance of mixed-signal neural networks. In *International Joint Conference on Neural Networks (IJCNN)*, 2019. Cited on page 22.

[KNL+20] Ram Shankar Siva Kumar, Magnus Nyström, John Lambert, Andrew Marshall, Mario Goertzel, Andi Comissoneru, Matt Swann, and Sharon Xia. Adversarial machine learning-industry perspectives. In *Proc. of the IEEE Symposium on Security and Privacy Workshops*, 2020. Cited on page 13.

[KOY+19] Skanda Koppula, Lois Orosa, Abdullah Giray Yaglikçi, Roknoddin Azizi, Taha Shahroodi, Konstantinos Kanellopoulos, and Onur Mutlu. EDEN: enabling energy-efficient, high-performance deep neural network inference using approximate DRAM. In *Proc. of the Annual IEEE/ACM International Symposium on Microarchitecture*, pages 166–181, 2019. Cited on pages 6, 20, 85, 86, 87, 88, 91, 97, 103, 111, 250, and 251.

[KPQ21] Sanjay Kariyappa, Atul Prakash, and Moinuddin K. Qureshi. MAZE: data-free model stealing attack using zeroth-order gradient estimation. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. Cited on page 15.

[KR19] Michael Kearns and Aaron Roth. *The ethical algorithm: The science of socially aware algorithm design*. Oxford University Press, 2019. Cited on pages 13 and 15.

[Kri09] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009. Cited on pages 72, 97, 126, 144, 155, and 267.

[Kri18] Raghuraman Krishnamoorthi. Quantizing deep convolutional networks for efficient inference: A whitepaper. *arXiv.org*, abs/1806.08342, 2018. Cited on pages 21, 91, and 105.

[KSA+21] Iryna Korshunova, David Stutz, Alexander A. Alemi, Olivia Wiles, and Sven Gowal. A closer look at the adversarial robustness of information bottleneck models. In *Proc. of the International Conference on Machine Learning (ICML) Workshops*, 2021. Cited on page 10.

[KSB21] Benjamin Kompa, Jasper Snoek, and Andrew L Beam. Second opinion needed: communicating uncertainty in medical machine learning. *NPJ Digital Medicine*, 4(1):1–6, 2021. Cited on page 22.

[KSDT21] Qiyu Kang, Yang Song, Qinxu Ding, and Wee Peng Tay. Stable neural ode with lyapunov-stable equilibrium points for defending against adversarial attacks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. Cited on page 18.

[KSH+19] Daniel Kang, Yi Sun, Dan Hendrycks, Tom Brown, and Jacob Steinhardt. Testing robustness against unforeseen adversaries. *arXiv.org*, abs/1908.08016, 2019. Cited on page 116.

[KSJ19] Beomsu Kim, Junghoon Seo, and Taegyun Jeon. Bridging adversarial robustness and gradient interpretability. *arXiv.org*, abs/1903.11626, 2019. Cited on page 15.

[KSW16] Diederik P. Kingma, Tim Salimans, and Max Welling. Improving variational inference with inverse autoregressive flow. *arXiv.org*, abs/1606.04934, 2016. Cited on page 153.

[KTCM15] Abhishek Kar, Shubham Tulsiani, João Carreira, and Jitendra Malik. Category-specific object reconstruction from a single image. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. Cited on page 12.

[KTH20] Minseon Kim, Jihoon Tack, and Sung Ju Hwang. Adversarial self-supervised contrastive learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. Cited on page 18.

[KTP+20] Kalpesh Krishna, Gaurav Singh Tomar, Ankur P. Parikh, Nicolas Papernot, and Mohit Iyyer. Thieves on sesame street! model extraction of bert-based apis. In *Proc. of the International Conference on Learning Representations (ICLR)*, 2020. Cited on page 15.

[KW14a] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. *Proc. of the International Conference on Learning Representations (ICLR)*, 2014. Cited on pages 12, 29, 30, 31, 32, and 46.

[KW14b] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *Proc. of the International Conference on Learning Representations (ICLR)*, 2014. Cited on page 53.

[KW20] Jungeum Kim and Xiao Wang. Sensible adversarial learning. https://openreview.net/forum?id=rJlf_RVKwr, 2020. Cited on page 18.

[KZG18] Danny Karmon, Daniel Zoran, and Yoav Goldberg. Lavan: Localized and visible adversarial noise. In *Proc. of the International Conference on Machine Learning (ICML)*, 2018. Cited on page 16.

[LA11] Alvin J. Law and Daniel G. Aliaga. Single viewpoint model completion of symmetric objects for digital inspection. *Computer Vision and Image Understanding (CVIU)*, 115(5):603–610, 2011. Cited on page 12.

[LAG+19] Mathias Lécuyer, Vaggelis Atlidakis, Roxana Geambasu, Daniel Hsu, and Suman Jana. Certified robustness to adversarial examples with differential privacy. In *Proc. of the IEEE Symposium on Security and Privacy*, 2019. Cited on page 15.

[LAJ19] Guang-He Lee, David Alvarez-Melis, and Tommi S. Jaakkola. Towards robust, locally linear deep networks. In *Proc. of the International Conference on Learning Representations (ICLR)*, 2019. Cited on page 18.

[LBBH98] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proc. of the IEEE*, 86(11):2278–2324, 1998. Cited on pages 50, 58, 97, 118, 126, 128, 144, and 267.

[LBG+18] Alex Lamb, Jonathan Binas, Anirudh Goyal, Dmitriy Serdyuk, Sandeep Subramanian, Ioannis Mitliagkas, and Yoshua Bengio. Fortified networks: Improving the robustness of deep networks by modeling the manifold of hidden representations. *arXiv.org*, abs/1804.02485, 2018. Cited on page 17.

[LC87] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *ACM Trans. on Graphics (SIGGRAPH)*, 1987. Cited on pages 13 and 35.

[LCC20] Ziquan Liu, Yufei Cui, and Antoni B. Chan. Improve generalization and robustness of neural networks via weight scale shifting invariant regularizations. *arXiv.org*, abs/2008.02965, 2020. Cited on page 17.

[LCLS17] Yanpei Liu, Xinyun Chen, Chang Liu, and Dawn Song. Delving into transferable adversarial examples and black-box attacks. *Proc. of the International Conference on Learning Representations (ICLR)*, 2017. Cited on pages 16, 50, and 229.

[LCWC19] Bai Li, Changyou Chen, Wenlin Wang, and Lawrence Carin. On norm-agnostic robustness of adversarial training. *arXiv.org*, abs/1905.06455, 2019. Cited on page 116.

[LCY+20] Kibok Lee, Zhuoyuan Chen, Xinchen Yan, Raquel Urtasun, and Ersin Yumer. Shapeadv: Generating shape-aware adversarial 3d point clouds. *arXiv.org*, abs/2005.11626, 2020. Cited on page 14.

[LCZH17] Xuanqing Liu, Minhao Cheng, Huan Zhang, and Cho-Jui Hsieh. Towards robust neural networks via random self-ensemble. *arXiv.org*, abs/1712.00673, 2017. Cited on page 17.

[LDG18] Yiyi Liao, Simon Donne, and Andreas Geiger. Deep marching cubes: Learning explicit surface representations. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. Cited on page 13.

[LDGN15] Yangyan Li, Angela Dai, Leonidas J. Guibas, and Matthias Nießner. Database-assisted object retrieval for real-time 3d reconstruction. *Computer Graphics Forum*, 34(2):435–446, 2015. Cited on pages 12, 28, and 46.

[LDX+17] Hao Li, Soham De, Zheng Xu, Christoph Studer, Hanan Samet, and Tom Goldstein. Training quantized nets: A deeper understanding. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems (NeurIPS)*, 2017. Cited on page 21.

[LF19] Cassidy Laidlaw and Soheil Feizi. Playing it safe: Adversarial robustness with an abstain option. *arXiv.org*, abs/1911.11253, 2019. Cited on page 18.

[LF20] Klas Leino and Matt Fredrikson. Stolen memories: Leveraging model memorization for calibrated white-box membership inference. In *USENIX Security Symposium*, 2020. Cited on page 15.

[LHL15] Chunchuan Lyu, Kaizhu Huang, and Hai-Ning Liang. A unified gradient regularization family for adversarial examples. In *IEEE International Conference on Data Mining*, 2015. Cited on page 17.

[LHS14] M. Lee, Kyuyeon Hwang, and Wonyong Sung. Fault tolerance analysis of digital feed-forward deep neural networks. *Proc. of the IEEE Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2014. Cited on page 21.

[LJC+21] Jie Li, Rongrong Ji, Peixian Chen, Baochang Zhang, Xiaopeng Hong, Ruixin Zhang, Shaoxin Li, Jilin Li, Feiyue Huang, and Yongjian Wu. Aha! adaptive history-driven attack for decision-based black-box models. In *Proc. of the IEEE International Conference on Computer Vision (ICCV)*, 2021. Cited on page 16.

[LJE+20] Yuan Liu, Ayush Jain, Clara Eng, David H. Way, Kang Lee, Peggy Bui, Kimberly Kanada, Guilherme de Oliveira Marinho, Jessica Gallegos, Sara Gabriele, Vishakha Gupta, Nalini Singh, Vivek Natarajan, Rainer Hofmann-Wellenhof, Gregory S. Corrado, Lily H. Peng, Dale R. Webster, Dennis Ai, Susan Huang, Yun Liu, R. Carter Dunn, and David Coz. A deep learning system for differential diagnosis of skin diseases. *Nature Medicine*, 26:900–908, 2020. Cited on page 142.

[LJL+20] Jie Li, Rongrong Ji, Hong Liu, Jianzhuang Liu, Bineng Zhong, Cheng Deng, and Qi Tian. Projection & probability-driven black-box attack. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. Cited on page 16.

[LK21] Jingyue Lu and M. Pawan Kumar. Improving local effectiveness for global robust training. *arXiv.org*, abs/2110.14030, 2021. Cited on page 18.

[LKB+17] Geert Litjens, Thijs Kooi, Babak Ehteshami Bejnordi, Arnaud Arindra Adiyoso Setio, Francesco Ciompi, Mohsen Ghafoorian, Jeroen A.W.M. van der Laak, Bram van Ginneken, and Clara I. Sánchez. A survey on deep learning in medical image analysis. *Medical Image Analysis (MIA)*, 42:60–88, 2017. Cited on page 1.

[LL17] Xin Li and Fuxin Li. Adversarial examples detection in deep networks with convolutional filter statistics. In *Proc. of the IEEE International Conference on Computer Vision (ICCV)*, pages 5775–5783, 2017. Cited on page 19.

[LLD+18] Fangzhou Liao, Ming Liang, Yinpeng Dong, Tianyu Pang, Xiaolin Hu, and Jun Zhu. Defense against adversarial attacks using high-level representation guided denoiser. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. Cited on page 19.

[LLL+20] Wei-An Lin, Chun Pong Lau, Alexander Levine, Rama Chellappa, and Soheil Feizi. Dual manifold adversarial robustness: Defense against lp and non-lp adversarial attacks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. Cited on page 17.

[LLLS18a] Kimin Lee, Honglak Lee, Kibok Lee, and Jinwoo Shin. Training confidence-calibrated classifiers for detecting out-of-distribution samples. In *Proc. of the International Conference on Learning Representations (ICLR)*, 2018. Cited on page 22.

[LLLS18b] Kimin Lee, Kibok Lee, Honglak Lee, and Jinwoo Shin. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 7167–7177, 2018. Cited on pages 7, 19, 22, 117, 124, 126, and 128.

[LLS17] Shiyu Liang, Yixuan Li, and R. Srikant. Principled detection of out-of-distribution examples in neural networks. *arXiv.org*, abs/1706.02690, 2017. Cited on page 22.

[LLS18] Shiyu Liang, Yixuan Li, and R. Srikant. Enhancing the reliability of out-of-distribution image detection in neural networks. In *Proc. of the International Conference on Learning Representations (ICLR)*, 2018. Cited on pages 19 and 22.

[LLS+21] Chun Pong Lau, Jiang Liu, Hossein Souri, Wei-An Lin, Soheil Feizi, and Rama Chellappa. Interpolated joint space adversarial training for robust and generalizable defenses. *arXiv.org*, abs/2112.06323, 2021. Cited on page 17.

[LLWH19] Xuanqing Liu, Yao Li, Chongruo Wu, and Cho-Jui Hsieh. Adv-bnn: Improved adversarial defense through robust bayesian neural network. In *Proc. of the International Conference on Learning Representations (ICLR)*, 2019. Cited on page 18.

[LLWT15] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proc. of the IEEE International Conference on Computer Vision (ICCV)*, 2015. Cited on pages 49, 51, and 54.

[LM05] Daniel Lowd and Christopher Meek. Adversarial learning. In *Proc. of the ACM International Conference on Knowledge Discovery & Data Mining*, 2005. Cited on page 14.

[LM09] Matthew J. Leotta and Joseph L. Mundy. Predicting high resolution image edges with a generic, adaptive, 3-d vehicle model. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009. Cited on page 12.

[LMA+18] Yingqi Liu, Shiqing Ma, Yousra Aafer, Wen-Chuan Lee, Juan Zhai, Weihang Wang, and Xiangyu Zhang. Trojaning attack on neural networks. In *Annual Network and Distributed System Security Symposium*, 2018. Cited on page 15.

[LMHD14] Yen-Liang Lin, Vlad I. Morariu, Winston H. Hsu, and Larry S. Davis. Jointly optimizing 3d model fitting and fine-grained classification. In *Proc. of the European Conference on Computer Vision (ECCV)*, 2014. Cited on page 12.

[LML+21] Yao Li, Martin Renqiang Min, Thomas Lee, Wenchao Yu, Erik Kruus, Wei Wang, and Cho-Jui Hsieh. Towards robustness of deep neural networks via regularization. In *Proc. of the IEEE International Conference on Computer Vision (ICCV)*, 2021. Cited on page 18.

[LPB17] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 6405–6416, 2017. Cited on page 14.

[LRB+16] Iro Laina, Christian Rupprecht, Vasileios Belagiannis, Federico Tombari, and Nassir Navab. Deeper depth prediction with fully convolutional residual networks. In *Proc. of the International Conference on 3D Vision (3DV)*, 2016. Cited on page 37.

[LRH+21] Jingtao Li, Adnan Siraj Rakin, Zhezhi He, Deliang Fan, and Chaitali Chakrabarti. RADAR: run-time adversarial weight attack detection and accuracy recovery. In *Proc. of the Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2021. Cited on page 21.

[LRW13] Jing Lei, Alessandro Rinaldo, and Larry Wasserman. A conformal prediction approach to explore functional data. *Annals of Mathematics and Artificial Intelligence*, 74:29–43, 2013. Cited on page 23.

[LSF21] Cassidy Laidlaw, Sahil Singla, and Soheil Feizi. Perceptual adversarial robustness: Defense against unseen threat models. In *Proc. of the International Conference on Learning Representations (ICLR)*, 2021. Cited on page 16.

[LSFF17] Jiajun Lu, Hussein Sibai, Evan Fabry, and David Forsyth. No need to worry about adversarial examples in object detection in autonomous vehicles. *arXiv.org*, abs/1707.03501, 2017. Cited on pages 14 and 16.

[LSJ+17] Lubor Ladicky, Olivier Saurer, SoHyeon Jeong, Fabio Maninchedda, and Marc Pollefeys. From point clouds to mesh using regression. In *Proc. of the IEEE International Conference on Computer Vision (ICCV)*, 2017. Cited on page 13.

[LSK19] Juncheng Li, Frank R. Schmidt, and J. Zico Kolter. Adversarial camera stickers: A physical camera-based attack on deep learning systems. In *Proc. of the International Conference on Machine Learning (ICML)*, 2019. Cited on page 17.

[LSL+20] Chen Liu, Mathieu Salzmann, Tao Lin, Ryota Tomioka, and Sabine Süsstrunk. On the loss landscape of adversarial training: Identifying challenges and how to overcome them. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. Cited on page 18.

[LSLW16] Anders Boesen Lindbo Larsen, Søren Kaae Sønderby, Hugo Larochelle, and Ole Winther. Autoencoding beyond pixels using a learned similarity metric. In *Proc. of the International Conference on Machine Learning (ICML)*, 2016. Cited on pages 52, 54, 56, and 63.

[LSPJ20] Tao Lin, Sebastian U. Stich, Kumar Kshitij Patel, and Martin Jaggi. Don't use large mini-batches, use local SGD. In *Proc. of the International Conference on Learning Representations (ICLR)*, 2020. Cited on page 67.

[LTA16] Darryl Dexu Lin, Sachin S. Talathi, and V. Sreekanth Annapureddy. Fixed point quantization of deep convolutional networks. In *Proc. of the International Conference on Machine Learning (ICML)*, 2016. Cited on pages 21 and 84.

[LTHL21] Yao Li, Tongyi Tang, Cho-Jui Hsieh, and Thomas C. M. Lee. Detecting adversarial examples with bayesian neural network. *arXiv.org*, abs/2105.08620, 2021. Cited on page 19.

[LTL+19] Hsueh-Ti Derek Liu, Michael Tao, Chun-Liang Li, Derek Nowrouzezahrai, and Alec Jacobson. Beyond pixel norm-balls: Parametric adversaries using an analytically differentiable renderer. In *Proc. of the International Conference on Learning Representations (ICLR)*, 2019. Cited on page 17.

[LV20] Yi Li and Nuno Vasconcelos. Background data resampling for outlier-aware classification. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. Cited on page 22.

[LVKB19] Alex Lamb, Vikas Verma, Juho Kannala, and Yoshua Bengio. Interpolated adversarial training: Achieving robust neural networks without sacrificing too much accuracy. In *Proc. of the ACM Workshop on Artificial Intelligence and Security*, 2019. Cited on page 18.

[LWB21] Yue Li, Hongxia Wang, and Mauro Barni. A survey of deep neural network watermarking techniques. *Neurocomputing*, 461:171–193, 2021. Cited on page 15.

[LWJC20] Bai Li, Shiqi Wang, Suman Jana, and Lawrence Carin. Towards understanding fast adversarial training. *arXiv.org*, abs/2006.03089, 2020. Cited on page 18.

[LWL+19] Rundong Li, Yan Wang, Feng Liang, Hongwei Qin, Junjie Yan, and Rui Fan. Fully quantized network for object detection. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. Cited on pages 21, 89, and 92.

[LWL+20] Aishan Liu, Jiakai Wang, Xianglong Liu, Bowen Cao, Chongzhi Zhang, and Hang Yu. Bias-based universal adversarial patch attack for automatic check-out. In *Proc. of the European Conference on Computer Vision (ECCV)*, 2020. Cited on page 17.

[LXTG18] Hao Li, Zheng Xu, G. Taylor, and T. Goldstein. Visualizing the loss landscape of neural nets. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018. Cited on pages 66, 67, 69, 71, 72, and 233.

[LYF17] Jerry Liu, Fisher Yu, and Thomas A. Funkhouser. Interactive 3d modeling with a generative adversarial network. In *Proc. of the International Conference on 3D Vision (3DV)*, 2017. Cited on pages 12, 40, and 46.

[LZLY17] Yujia Liu, Weiming Zhang, Shaohua Li, and Nenghai Yu. Enhanced attacks on defensively distilled deep neural networks. *arXiv.org*, abs/1711.05934, 2017. Cited on pages 16 and 18.

[LZP+20] Shaohui Liu, Yinda Zhang, Songyou Peng, Boxin Shi, Marc Pollefeys, and Zhaopeng Cui. DIST: rendering deep implicit signed distance function with differentiable sphere tracing. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. Cited on page 13.

[LZS+18] Cong Liao, Haoti Zhong, Anna Cinzia Squicciarini, Sencun Zhu, and David J. Miller. Backdoor embedding in convolutional neural network models via invisible perturbation. *arXiv.org*, abs/1808.10307, 2018. Cited on page 15.

[LZZ+19] Jiayang Liu, Weiming Zhang, Yiwei Zhang, Dongdong Hou, Yujia Liu, Hongyue Zha, and Nenghai Yu. Detection based defense against adversarial examples from the steganalysis point of view. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. Cited on page 19.

[MAA+16] Paul Merolla, Rathinakumar Appuswamy, John V. Arthur, Steven K. Esser, and Dharmendra S. Modha. Deep neural networks are robust to weight binarization and other non-linear distortions. *arXiv.org*, abs/1606.01981, 2016. Cited on pages 14, 20, 21, 84, and 91.

[MAT+18] Marius Mosbach, Maksym Andriushchenko, Thomas Alexander Trost, Matthias Hein, and Dietrich Klakow. Logit pairing methods can fool gradient-based attacks. *arXiv.org*, abs/1810.12042, 2018. Cited on page 18.

[MBH21] Alexander Meinke, Julian Bitterwolf, and Matthias Hein. Provably robust detection of out-of-distribution data (almost) for free. *arXiv.org*, abs/2106.04260, 2021. Cited on page 22.

[McD18] Mark D. McDonnell. Training wide residual networks for deployment using a single bit for each weight. In *Proc. of the International Conference on Learning Representations (ICLR)*, 2018. Cited on page 92.

[MCW+21] Chengzhi Mao, Mia Chiquier, Hao Wang, Junfeng Yang, and Carl Vondrick. Adversarial attacks are reversible with natural supervision. In *Proc. of the IEEE International Conference on Computer Vision (ICCV)*, 2021. Cited on page 18.

[MDI19] Abhishek Murthy, Himel Das, and Md. Ariful Islam. Robustness of neural networks to parameter quantization. *arXiv.org*, abs/1903.10672, 2019. Cited on pages 14, 20, 21, 84, and 91.

[MDR+21] Matthias Minderer, Josip Djolonga, Rob Romijnders, Frances Hubis, Xiaohua Zhai, Neil Houlsby, Dustin Tran, and Mario Lucic. Revisiting the calibration of modern neural networks. *arXiv.org*, abs/2106.07998, 2021. Cited on page 22.

[MFF16] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: A simple and accurate method to fool deep neural networks. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. Cited on page 16.

[MFFF17] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. Universal adversarial perturbations. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. Cited on page 17.

[MFM21] Thibault Maho, Teddy Furon, and Erwan Le Merrer. Surfree: A fast surrogate-free black-box attack. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. Cited on page 16.

[MG15] Moritz Menze and Andreas Geiger. Object scene flow for autonomous vehicles. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. Cited on page 30.

[MG19] Norman Mu and Justing Gilmer. Mnist-c: A robustness benchmark for computer vision. *Proc. of the International Conference on Machine Learning (ICML) Workshops*, 2019. Cited on page 126.

[MGB19] Konda Reddy Mopuri, Aditya Ganeshan, and R. Venkatesh Babu. Generalizable data-free objective for crafting universal adversarial perturbations. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 41(10), 2019. Cited on page 17.

[MGFB17] Jan Hendrik Metzen, Tim Genewein, Volker Fischer, and Bastian Bischoff. On detecting adversarial perturbations. In *Proc. of the International Conference on Learning Representations (ICLR)*, 2017. Cited on page 19.

[MGV18] Matthew Mirman, Timon Gehr, and Martin T. Vechev. Differentiable abstract interpretation for provably robust neural networks. In *Proc. of the International Conference on Machine Learning (ICML)*, pages 3575–3583, 2018. Cited on page 18.

[MH20] Alexander Meinke and Matthias Hein. Towards neural networks that provably know when they don't know. In *Proc. of the International Conference on Learning Representations (ICLR)*, 2020. Cited on page 22.

[Mis20] Diganta Misra. Mish: A self regularized non-monotonic activation function. In *Proc. of the British Machine Vision Conference (BMVC)*, 2020. Cited on pages 67, 73, 237, and 240.

[MKH19] Rafael Müller, Simon Kornblith, and Geoffrey E. Hinton. When does label smoothing help? In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. Cited on page 14.

[MKW+21] Jeet Mohapatra, Ching-Yun Ko, Lily Weng, Pin-Yu Chen, Sijia Liu, and Luca Daniel. Hidden cost of randomized smoothing. In *Conference on Artificial Intelligence and Statistics (AISTATS)*, 2021. Cited on page 19.

[MLW+18] Xingjun Ma, Bo Li, Yisen Wang, Sarah M. Erfani, Sudanthi N. R. Wijewickrema, Grant Schoenebeck, Dawn Song, Michael E. Houle, and James Bailey. Characterizing adversarial subspaces using local intrinsic dimensionality. In *Proc. of the International Conference on Learning Representations (ICLR)*, 2018. Cited on pages 7, 19, 22, 117, 124, 126, and 128.

[MMF19] Apostolos Modas, Seyed-Mohsen Moosavi-Dezfooli, and Pascal Frossard. Sparsefool: A few pixels make a big difference. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. Cited on page 16.

[MMS+18] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *Proc. of the International Conference on Learning Representations (ICLR)*, 2018. Cited on pages 4, 15, 16, 18, 50, 53, 66, 68, 86, 89, 96, 116, 117, 118, 121, 125, 126, 127, 128, 133, 227, 228, 229, 231, and 260.

[MOG+20] Kit Murdock, David Oswald, Flavio D. Garcia, Jo Van Bulck, Daniel Gruss, and Frank Piessens. Plundervolt: Software-based fault injection attacks against intel sgx. In *Proc. of the IEEE Symposium on Security and Privacy*, 2020. Cited on pages 6, 14, 20, 21, 84, 86, and 89.

[MON+19] Lars M. Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. Cited on page 13.

[MPJ+19] Mateusz Michalkiewicz, Jhony K. Pontes, Dominic Jack, Mahsa Baktashmotlagh, and Anders P. Eriksson. Deep level sets: Implicit surface representations for 3d shape inference. *arXiv.org*, abs/1901.06802, 2019. Cited on page 13.

[MPT20] Erwan Le Merrer, Patrick Pérez, and Gilles Trédan. Adversarial frontier stitching for remote neural network watermarking. *Neural Computing and Applications*, 32(13):9233–9244, 2020. Cited on page 15.

[MPYW20] Sina Mohseni, Mandar Pitale, J. B. S. Yadawa, and Zhangyang Wang. Self-supervised learning for generalizable out-of-distribution detection. In *Proc. of the Conference on Artificial Intelligence (AAAI)*, 2020. Cited on page 22.

[MS14] Lu Ma and Gabe Sibley. Unsupervised dense object discovery, detection, tracking and reconstruction. In *Proc. of the European Conference on Computer Vision (ECCV)*, 2014. Cited on page 12.

[MSG+20] Scott Mayer McKinney, Marcin Sieniek, Varun Godbole, Jonathan Godwin, Natasha Antropova, Hutan Ashrafian, Trevor Back, Mary Chesus, Greg Corrado, Ara Darzi, Mozziyar Etemadi, Florencia Garcia-Vicente, Fiona J. Gilbert, Mark D. Halling-Brown, Demis Hassabis, Sunny Jansen, Alan Karthikesalingam, Christopher J. Kelly, Dominic King, Joseph R. Ledsam, David S. Melnick, Hormuz Mostofi, Lily H. Peng, Joshua Jay Reicher, Bernardino Romera-Paredes, Richard Sidebottom, Mustafa Suleyman, Daniel Tse,

Kenneth C. Young, Jeffrey De Fauw, and Shravya Shetty. International evaluation of an ai system for breast cancer screening. *Nature*, 577:89–94, 2020. Cited on page 142.

[MSH21] Divyam Madaan, Jinwoo Shin, and Sung Ju Hwang. Learning to generate noise for multi-attack robustness. In *Proc. of the International Conference on Machine Learning (ICML)*, 2021. Cited on page 18.

[MST18] Seyed-Mohsen Moosavi-Dezfooli, Ashish Shrivastava, and Oncel Tuzel. Divide, denoise, and defend against adversarial attacks. *arXiv.org*, abs/1802.06806, 2018. Cited on page 17.

[MTP+21] Mateusz Michalkiewicz, Stavros Tsogkas, Sarah Parisot, Mahsa Baktashmotlagh, Anders P. Eriksson, and Eugene Belilovsky. Learning compositional shape priors for few-shot 3d reconstruction. *arXiv.org*, abs/2106.06440, 2021. Cited on page 12.

[MUDV17] Bert Moons, Roel Uytterhoeven, Wim Dehaene, and Marian Verhelst. DVAFS: trading computational accuracy for energy through dynamic-voltage-accuracy-frequency-scaling. In *Proc. of the Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2017. Cited on pages 6 and 20.

[MWK20] Pratyush Maini, Eric Wong, and J. Zico Kolter. Adversarial robustness against the union of multiple perturbation models. *Proc. of the International Conference on Machine Learning (ICML)*, 2020. Cited on pages 7, 18, 116, 117, 126, 128, and 154.

[MWYA19] Shuntaro Miyazato, Xueting Wang, Toshihiko Yamasaki, and Kiyoharu Aizawa. Reinforcing the robustness of a deep neural network to adversarial examples by using color quantization of training image data. In *Proc. of the IEEE International Conference on Image Processing (ICIP)*, 2019. Cited on page 17.

[NADL19] Adam Noack, Isaac Ahern, Dejing Dou, and Boyang Li. Does interpretability of neural networks imply adversarial robustness? *arXiv.org*, abs/1912.03430, 2019. Cited on page 15.

[Nak19] Preetum Nakkiran. Adversarial robustness may be at odds with simplicity. *arXiv.org*, abs/1901.00532, 2019. Cited on page 19.

[NBC+08] Blaine Nelson, Marco Barreno, Fuching Jack Chi, Anthony D. Joseph, Benjamin I. P. Rubinstein, Udam Saini, Charles Sutton, J. Doug Tygar, and Kai Xia. Exploiting machine learning to subvert your spam filter. In *USENIX Security Symposium*, 2008. Cited on page 14.

[NBMS17] Behnam Neyshabur, Srinadh Bhojanapalli, David McAllester, and Nati Srebro. Exploring generalization in deep learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017. Cited on pages 5, 19, 66, 67, 70, 72, and 154.

[ND21] Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *Proc. of the International Conference on Machine Learning (ICML)*, 2021. Cited on page 153.

[NDS+21] Vedant Nanda, Samuel Dooley, Sahil Singla, Soheil Feizi, and John P. Dickerson. Fairness through robustness: Investigating robustness disparity in deep learning. In *ACM Conference on Fairness, Accountability, and Transparency (FAccT)*, 2021. Cited on page 15.

[NDZ+19] Jeremy Nixon, Michael W. Dusenberry, Linchuan Zhang, Ghassen Jerfel, and Dustin Tran. Measuring calibration in deep learning. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2019. Cited on page 22.

[ner] Nervana neural network distiller. https://github.com/nervanasystems/distiller. Cited on page 21.

[NG17] Aran Nayebi and Surya Ganguli. Biologically inspired protection of deep networks from adversarial attacks. *arXiv.org*, abs/1703.09202, 2017. Cited on page 17.

[NG21] Michael Niemeyer and Andreas Geiger. GIRAFFE: representing scenes as compositional generative neural feature fields. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. Cited on page 13.

[NH10] Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proc. of the International Conference on Machine Learning (ICML)*, 2010. Cited on page 145.

[NHL20] Jay Nandy, Wynne Hsu, and Mong-Li Lee. Approximate manifold defense against multiple adversarial perturbations. In *International Joint Conference on Neural Networks (IJCNN)*, 2020. Cited on page 17.

[NHT+16] Duc Thanh Nguyen, Binh-Son Hua, Minh-Khoi Tran, Quang-Hieu Pham, and Sai-Kit Yeung. A field model for repairing 3d shapes. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. Cited on pages 12, 28, and 46.

[NIH+11] Richard A. Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J. Davison, Pushmeet Kohli, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *Proc. of the International Symposium on Mixed and Augmented Reality (ISMAR)*, 2011. Cited on page 37.

[NK17] Nina Narodytska and Shiva Prasad Kasiviswanathan. Simple black-box adversarial attacks on deep neural networks. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2017. Cited on pages 16, 117, 124, and 127.

[NKM18] Taesik Na, Jong Hwan Ko, and Saibal Mukhopadhyay. Cascade adversarial machine learning regularized with a unified embedding. In *Proc. of the International Conference on Learning Representations (ICLR)*, 2018. Cited on page 18.

[NMOG20] Michael Niemeyer, Lars M. Mescheder, Michael Oechsle, and Andreas Geiger. Differentiable volumetric rendering: Learning implicit 3d representations without 3d supervision. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. Cited on page 13.

[NMT+19a] Eric T. Nalisnick, Akihiro Matsukawa, Yee Whye Teh, Dilan Görür, and Balaji Lakshminarayanan. Do deep generative models know what they don't know? In *Proc. of the International Conference on Learning Representations (ICLR)*, 2019. Cited on page 14.

[NMT+19b] Eric T. Nalisnick, Akihiro Matsukawa, Yee Whye Teh, Dilan Görür, and Balaji Lakshminarayanan. Hybrid models with deep and invertible features. In *Proc. of the International Conference on Machine Learning (ICML)*, 2019. Cited on page 14.

[NO15] Khanh Nguyen and Brendan O'Connor. Posterior calibration and exploratory analysis for natural language processing models. In *Proc. of the Conference on Empirical Methods in Natural Language Processing*, 2015. Cited on page 22.

[NST+18] Maria-Irina Nicolae, Mathieu Sinn, Minh Ngoc Tran, Beat Buesser, Ambrish Rawat, Martin Wistuba, Valentina Zantedeschi, Nathalie Baracaldo, Bryant Chen, Heiko Ludwig, Ian Molloy, and Ben Edwards. Adversarial robustness toolbox v1.2.0. *arXiv.org*, 1807.01069, 2018. Cited on page 17.

[NSY92] Chalapathy Neti, Michael H. Schneider, and Eric D. Young. Maximally fault tolerant neural networks. *IEEE Trans. on Neural Networks (TNN)*, 3(1):14–23, 1992. Cited on pages 14 and 21.

[NvBBW19] Markus Nagel, Mart van Baalen, Tijmen Blankevoort, and Max Welling. Data-free quantization through weight equalization and bias correction. In *Proc. of the IEEE International Conference on Computer Vision (ICCV)*, 2019. Cited on page 101.

[nvd] NVIDIA Deep Learning Accelerator. http://nvdla.org/. Cited on pages 6 and 84.

[nvr] Nvidia tensorrt. https://developer.nvidia.com/tensorrt. Cited on page 21.

[NW17] Charlie Nash and Christopher K. I. Williams. The shape variational autoencoder: A deep generative model of part-segmented 3d objects. *Eurographics Symposium on Geometry Processing (SGP)*, 36(5):1–12, 2017. Cited on page 12.

[NWC+11] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. Reading digits in natural images with unsupervised feature learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2011. Cited on page 126.

[NXS12] Liangliang Nan, Ke Xie, and Andrei Sharf. A search-classify approach for cluttered indoor scene understanding. *ACM TG*, 31(6):137:1–137:10, November 2012. Cited on pages 3, 12, 28, and 46.

[NYC15] Anh Nguyen, Jason Yosinski, and Jeff Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. Cited on pages 14 and 17.

[OAFS18] Seong Joon Oh, Max Augustin, Mario Fritz, and Bernt Schiele. Towards reverse-engineering black-box neural networks. *ICLR*, 2018. Cited on pages 15 and 16.

[OSF19] Tribhuvanesh Orekondy, Bernt Schiele, and Mario Fritz. Knockoff nets: Stealing functionality of black-box models. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. Cited on page 15.

[OSF20] Tribhuvanesh Orekondy, Bernt Schiele, and Mario Fritz. Prediction poisoning: Towards defenses against DNN model stealing attacks. In *Proc. of the International Conference on Learning Representations (ICLR)*, 2020. Cited on page 15.

[PBZ+20] Kanil Patel, William Beluch, Dan Zhang, Michael Pfeiffer, and Bin Yang. On-manifold adversarial data augmentation improves uncertainty calibration. In *Proc. of the International Conference on Pattern Recognition (ICPR)*, 2020. Cited on pages 17 and 153.

[PCCT14] Marco A. F. Pimentel, David A. Clifton, Lei A. Clifton, and Lionel Tarassenko. A review of novelty detection. *Signal Processing*, 99:215–249, 2014. Cited on page 14.

[PDS+21] Maura Pintor, Luca Demetrio, Angelo Sotgiu, Giovanni Manca, Ambra Demontis, Nicholas Carlini, Battista Biggio, and Fabio Roli. Indicators of attack failure: Debugging and improving optimization of adversarial examples. In *Proc. of the International Conference on Machine Learning (ICML) Workshops*, 2021. Cited on page 18.

[Pea19] Judea Pearl. The seven tools of causal inference, with reflections on machine learning. *Communications of the ACM*, 62(3):54–60, 2019. Cited on page 15.

[PEM01] Georgios Papadopoulos, Peter J. Edwards, and Alan F. Murray. Confidence estimation methods for neural networks: a practical comparison. *IEEE Trans. on Neural Networks (TNN)*, 12(6):1278–1287, 2001. Cited on page 14.

[PFS+19] Jeong Joon Park, Peter Florence, Julian Straub, Richard A. Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. Cited on page 13.

[PGC+17] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *Advances in Neural Information Processing Systems (NeurIPS) Workshops*, 2017. Cited on pages 92, 128, 236, and 246.

[Piz07] Zygmunt Pizlo. Human perception of 3d shapes. In *Proc. of the International Conference on Computer Analysis of Images and Patterns (CAIP)*, 2007. Cited on page 27.

[Piz10] Zygmunt Pizlo. *3D shape: Its unique place in visual perception*. MIT Press, 2010. Cited on pages 27 and 28.

[PKY18] Eunhyeok Park, Dongyoung Kim, and Sungjoo Yoo. Energy-efficient neural network accelerator based on outlier-aware low-precision computation. In *ACM/IEEE Annual International Symposium on Computer Architecture (ISCA)*, 2018. Cited on pages 20, 21, and 111.

[PL21] Geon Yeong Park and Sang Wan Lee. Reliably fast adversarial training via latent adversarial perturbation. In *Proc. of the IEEE International Conference on Computer Vision (ICCV)*, 2021. Cited on pages 16 and 18.

[PLY+22] Tianyu Pang, Min Lin, Xiao Yang, Junyi Zhu, and Shuicheng Yan. Robustness and accuracy could be reconcilable by (proper) definition. *arXiv.org*, abs/2202.10103, 2022. Cited on pages 18 and 19.

[PMG+05] Mark Pauly, Niloy J. Mitra, Joachim Giesen, Markus H. Gross, and Leonidas J. Guibas. Example-based 3d scan completion. In *Eurographics Symposium on Geometry Processing (SGP)*, 2005. Cited on page 12.

[PMG16] Nicolas Papernot, Patrick D. McDaniel, and Ian J. Goodfellow. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *arXiv.org*, abs/1605.07277, 2016. Cited on page 16.

[PMG+17] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z. Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning. In *Proc. of the ACM on Asia Conference on Computer and Communications Security (AsiaCCS)*. ACM, 2017. Cited on pages 16 and 229.

[PMG+18a] Aaditya Prakash, Nick Moran, Solomon Garber, Antonella DiLillo, and James A. Storer. Deflecting adversarial attacks with pixel deflection. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. Cited on page 17.

[PMG+18b] Aaditya Prakash, Nick Moran, Solomon Garber, Antonella DiLillo, and James A. Storer. Protecting JPEG images against adversarial attacks. In *Proc. of the IEEE Data Compression Conference*, 2018. Cited on page 17.

[PMJ+16] Nicolas Papernot, Patrick D. McDaniel, Somesh Jha, Matt Fredrikson, Z. Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. In *Proc. of the IEEE Symposium on Security and Privacy*, 2016. Cited on page 16.

[PMPP18] Julien Pérolat, Mateusz Malinowski, Bilal Piot, and Olivier Pietquin. Playing the game of universal adversarial perturbations. *arXiv.org*, abs/1809.07802, 2018. Cited on page 18.

[PMSW18] Nicolas Papernot, Patrick D. McDaniel, Arunesh Sinha, and Michael P. Wellman. Sok: Security and privacy in machine learning. In *Proc. of the IEEE European Symposium on Security and Privacy (EuroS&P)*, 2018. Cited on page 15.

[PMW+08] Mark Pauly, Niloy J. Mitra, Johannes Wallner, Helmut Pottmann, and Leonidas J. Guibas. Discovering structural regularity in 3d geometry. *ACM Trans. on Graphics*, 27(3):43:1–43:11, 2008. Cited on page 12.

[PMW+16] Nicolas Papernot, Patrick D. McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a defense to adversarial perturbations against deep neural networks. In *Proc. of the IEEE Symposium on Security and Privacy*, 2016. Cited on page 17.

[PR21] Aleksandr Podkopaev and Aaditya Ramdas. Distribution-free uncertainty quantification for classification under label shift. In *Proc. of the Conference on Uncertainty in Artificial Intelligence (UAI)*, 2021. Cited on page 23.

[PRBB21] Maura Pintor, Fabio Roli, Wieland Brendel, and Battista Biggio. Fast minimum-norm adversarial attacks through adaptive norm constraints. *arXiv.org*, abs/2102.12827, 2021. Cited on page 16.

[PRSW21] Aristotelis-Angelos Papadopoulos, Mohammad Reza Rajati, Nazim Shaikh, and Jiamian Wang. Outlier exposure with confidence control for out-of-distribution detection. *Neurocomputing*, 441, 2021. Cited on page 22.

[PS18] Rama Chellappa Pouya Samangouei, Maya Kabkab. Defense-GAN: Protecting classifiers against adversarial attacks using generative models. *Proc. of the International Conference on Learning Representations (ICLR)*, 2018. Cited on pages 17, 50, and 63.

[PSCvdH21] Guansong Pang, Chunhua Shen, Longbing Cao, and Anton van den Hengel. Deep learning for anomaly detection: A review. *ACM Computing Surveys*, 54(2):38:1–38:38, 2021. Cited on page 14.

[PSG+15] Bojan Pepik, Michael Stark, Peter V. Gehler, Tobias Ritschel, and Bernt Schiele. 3d object class detection in the wild. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–10, 2015. Cited on page 12.

[PSR13] Victor Prisacariu, Alex Segal, and Ian Reid. Simultaneous monocular 2d segmentation, 3d pose recovery and 3d reconstruction. In *Proc. of the Asian Conference on Computer Vision (ACCV)*, 2013. Cited on page 12.

[PTSS21] Ameya D. Patil, Michael Tuttle, Alexander G. Schwing, and Naresh R. Shanbhag. Robustifying ℓ_∞ adversarial training to the union of perturbation models. *arXiv.org*, abs/2105.14710, 2021. Cited on page 18.

[PVG+11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research (JMLR)*, 12:2825–2830, 2011. Cited on page 260.

[PYD+20] Tianyu Pang, Xiao Yang, Yinpeng Dong, Taufik Xu, Jun Zhu, and Hang Su. Boosting adversarial training with hypersphere embedding. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. Cited on page 18.

[PYD+21] Tianyu Pang, Xiao Yang, Yinpeng Dong, Hang Su, and Jun Zhu. Bag of tricks for adversarial training. In *Proc. of the International Conference on Learning Representations (ICLR)*, 2021. Cited on pages 18, 67, 73, and 236.

[PYV18] Eunhyeok Park, Sungjoo Yoo, and Peter Vajda. Value-aware quantization for training and inference of neural networks. In Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss, editors, *Proc. of the European Conference on Computer Vision (ECCV)*, 2018. Cited on page 101.

[PYXW19] Vinay Uday Prabhu, Dian Ang Yap, Joyce Xu, and J. Whaley. Understanding adversarial robustness through loss landscape geometries. *arXiv.org*, abs/1907.09061, 2019. Cited on pages 18 and 69.

[PZH+21] Tianyu Pang, Huishuai Zhang, Di He, Yinpeng Dong, Hang Su, Wei Chen, Jun Zhu, and Tie-Yan Liu. Adversarial training with rectified rejection. *arXiv.org*, abs/2105.14785, 2021. Cited on page 18.

[Qaz96] Cazhaow S. Qazaz. *Bayesian error bars for regression*. PhD thesis, Aston University, Birmingham, UK, 1996. Cited on page 14.

[QCSLS09] Joaquin Quiñonero-Candela, Masashi Sugiyama, Neil D Lawrence, and Anton Schwaighofer. *Dataset shift in machine learning*. Mit Press, 2009. Cited on page 14.

[QMG+19] Chongli Qin, James Martens, Sven Gowal, Dilip Krishnan, Krishnamurthy Dvijotham, Alhussein Fawzi, Soham De, Robert Stanforth, and Pushmeet Kohli. Adversarial robustness through local linearization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. Cited on page 236.

[QSMG17] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. Cited on page 13.

[QYSG17] Charles R Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017. Cited on page 13.

[QZZ+20] Han Qiu, Yi Zeng, Qinkai Zheng, Tianwei Zhang, Meikang Qiu, and Gérard Memmi. Mitigating advanced adversarial attacks with more advanced gradient obfuscation techniques. *arXiv.org*, abs/2005.13712, 2020. Cited on pages 16 and 17.

[RB19] Andras Rozsa and Terrance E. Boult. Improved adversarial robustness by reducing open space risk via tent activations. *arXiv.org*, abs/1908.02435, 2019. Cited on page 17.

[RBSB18] Anurag Ranjan, Timo Bolkart, Soubhik Sanyal, and Michael J. Black. Generating 3d faces using convolutional mesh autoencoders. In *Proc. of the European Conference on Computer Vision (ECCV)*, 2018. Cited on page 13.

[RCK18] Bita Darvish Rouhani, Huili Chen, and Farinaz Koushanfar. Deepsigns: A generic watermarking framework for ip protection of deep learning models. *arXiv.org*, abs/1804.00750, 2018. Cited on page 15.

[RCSM20] Rebecca Roelofs, Nicholas Cain, Jonathon Shlens, and Michael C. Mozer. Mitigating bias in calibration error estimation. *arXiv.org*, abs/2012.08668, 2020. Cited on page 22.

[RD18] Andrew Slavin Ross and Finale Doshi-Velez. Improving the adversarial robustness and interpretability of deep neural networks by regularizing their input gradients. In *Proc. of the Conference on Artificial Intelligence (AAAI)*, 2018. Cited on page 17.

[RDS+15] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael S. Bernstein, Alexander C. Berg, and Fei-Fei Li. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. Cited on pages 50 and 60.

[REM+16] Danilo Jimenez Rezende, S. M. Ali Eslami, Shakir Mohamed, Peter W. Battaglia, Max Jaderberg, and Nicolas Heess. Unsupervised learning of 3d structure from images. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2016. Cited on pages 12, 28, and 46.

[RFB15] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, 2015. Cited on page 39.

[RGB16] Andras Rozsa, Manuel Günther, and Terrance E. Boult. Are accuracy and robustness correlated. In *Proc. of the International Conference on Machine Learning and Applications (ICMLA)*, 2016. Cited on pages 5, 19, 50, and 61.

[RGB17] Andras Rozsa, Manuel Günther, and Terrance E. Boult. Adversarial robustness: Softmax versus openmax. In *Proc. of the British Machine Vision Conference (BMVC)*, 2017. Cited on page 16.

[RGC+21a] Sylvestre-Alvise Rebuffi, Sven Gowal, Dan A. Calian, Florian Stimberg, Olivia Wiles, and Timothy A. Mann. Fixing data augmentation to improve adversarial robustness. *arXiv.org*, abs/2103.01946, 2021. Cited on page 18.

[RGC+21b] Sylvestre-Alvise Rebuffi, Sven Gowal, Dan Andrei Calian, Florian Stimberg, Olivia Wiles, and Timothy Mann. Data augmentation can improve robustness. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. Cited on page 18.

[RGPA21] Jérôme Rony, Eric Granger, Marco Pedersoli, and Ismail Ben Ayed. Augmented lagrangian adversarial attacks. In *Proc. of the IEEE International Conference on Computer Vision (ICCV)*, 2021. Cited on page 16.

[RGT+15] Jason Rock, Tanmay Gupta, Justin Thorsen, JunYoung Gwak, Daeyun Shin, and Derek Hoiem. Completing 3d object shape from one depth image. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. Cited on pages 12, 28, and 46.

[RHF19a] Adnan Siraj Rakin, Zhezhi He, and Deliang Fan. Bit-flip attack: Crushing neural network with progressive bit search. In *Proc. of the IEEE International Conference on Computer Vision (ICCV)*, 2019. Cited on pages 6, 21, 84, 86, 89, 98, 108, 109, and 111.

[RHF19b] Adnan Siraj Rakin, Zhezhi He, and Deliang Fan. Parametric noise injection: Trainable randomness to improve deep neural network robustness against adversarial attack. *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. Cited on page 18.

[RHF20] Adnan Siraj Rakin, Zhezhi He, and Deliang Fan. TBT: targeted neural network attack with bit trojan. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. Cited on pages 14 and 21.

[RHG16] Daniel Ritchie, Paul Horsfall, and Noah D. Goodman. Deep amortized inference for probabilistic programs. *arXiv.org*, 1610.05735, 2016. Cited on page 12.

[RHL+20] Adnan Siraj Rakin, Zhezhi He, Jingtao Li, Fan Yao, Chaitali Chakrabarti, and Deliang Fan. T-BFA: targeted bit-flip adversarial weight attack. *arXiv.org*, abs/2007.12336, 2020. Cited on pages 6, 21, 84, 86, 89, 97, 109, and 110.

[RK20] David Rolnick and Konrad P. Kording. Reverse-engineering deep relu networks. In *Proc. of the International Conference on Machine Learning (ICML)*, 2020. Cited on pages 15 and 16.

[RKH19] Kevin Roth, Yannic Kilcher, and Thomas Hofmann. The odds are odd: A statistical test for detecting adversarial examples. In *Proc. of the International Conference on Machine Learning (ICML)*, pages 5498–5507, 2019. Cited on page 19.

[RKV+21] Lukas Ruff, Jacob R. Kauffmann, Robert A. Vandermeulen, Grégoire Montavon, Wojciech Samek, Marius Kloft, Thomas G. Dietterich, and Klaus-Robert Müller. A unifying review of deep and shallow anomaly detection. *Proceedings of the IEEE*, 109(5):756–795, 2021. Cited on page 14.

[RLF+19] Jie Ren, Peter J. Liu, Emily Fertig, Jasper Snoek, Ryan Poplin, Mark A. DePristo, Joshua V. Dillon, and Balaji Lakshminarayanan. Likelihood ratios for out-of-distribution detection. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 14680–14691, 2019. Cited on page 22.

[RLT+20] Martin Rünz, Kejie Li, Meng Tang, Lingni Ma, Chen Kong, Tanner Schmidt, Ian D. Reid, Lourdes Agapito, Julian Straub, Steven Lovegrove, and Richard A. Newcombe. Frodo: From detections to 3d objects. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. Cited on page 12.

[RLWFM17] Mihaela Rosca, Balaji Lakshminarayanan, David Warde-Farley, and Shakir Mohamed. Variational approaches for auto-encoding generative adversarial networks. *arXiv.org*, abs/1706.04987, 2017. Cited on pages 52, 54, 56, and 63.

[RM15a] Danilo Jimenez Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *Proc. of the International Conference on Machine Learning (ICML)*, 2015. Cited on page 12.

[RM15b] Danilo Jimenez Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *Proc. of the International Conference on Machine Learning (ICML)*, 2015. Cited on page 153.

[RMC16] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *Proc. of the International Conference on Learning Representations (ICLR)*, 2016. Cited on page 153.

[RMD21] Rahul Rade and Seyed-Mohsen Moosavi-Dezfooli. Helper-based adversarial training: Reducing excessive margin to achieve a better accuracy vs. robustness trade-off. In *Proc. of the International Conference on Machine Learning (ICML) Workshops*, 2021. Cited on pages 18 and 19.

[RNH+09] Benjamin I. P. Rubinstein, Blaine Nelson, Ling Huang, Anthony D. Joseph, Shing-hon Lau, Satish Rao, Nina Taft, and J. D. Tygar. ANTIDOTE: understanding and defending against poisoning of anomaly detectors. In *Proc. ACM SIGCOMM Internet Measurement Conference (IMC)*, 2009. Cited on page 15.

[RNR20] Arash Rahnama, André T. Nguyen, and Edward Raff. Robust design of deep neural networks against adversarial attacks based on lyapunov theory. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. Cited on page 17.

[ROF21] Shadi Rahimian, Tribhuvanesh Orekondy, and Mario Fritz. Differential privacy defenses and sampling attacks for membership inference. In *Proc. of the ACM Workshop on Artificial Intelligence and Security*, 2021. Cited on page 15.

[RORF16] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *Proc. of the European Conference on Computer Vision (ECCV)*, 2016. Cited on page 21.

[RPC19] Yaniv Romano, Evan Patterson, and Emmanuel J. Candès. Conformalized quantile regression. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. Cited on page 23.

[RRA+22] Abhijit Guha Roy, Jie Ren, Shekoofeh Azizi, Aaron Loh, Vivek Natarajan, Basil Mustafa, Nick Pawlowski, Jan Freyberg, Yuan Liu, Zachary Beaver, Nam Vo, Peggy Bui, Samantha Winter, Patricia MacWilliams, Gregory S. Corrado, Umesh Telang, Yun Liu, A. Taylan Cemgil, Alan Karthikesalingam, Balaji Lakshminarayanan, and Jim Winkens. Does your dermatology classifier know what it doesn't know? detecting the long-tail of unseen conditions. *Medical Image Analysis*, 75:102274, 2022. Cited on page 142.

[RRLM18] Md. Atiqur Rahman, Tanzila Rahman, Robert Laganière, and Noman Mohammed. Membership inference attack against differentially private deep learning model. *Transactions on Data Privacy*, 11(1):61–79, 2018. Cited on page 15.

[RSC20] Yaniv Romano, Matteo Sesia, and Emmanuel J. Candès. Classification with valid and adaptive coverage. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. Cited on pages 7, 14, 23, 136, 137, 138, 139, 140, and 149.

[RSCC17] Rajeev Ranjan, Swami Sankaranarayanan, Carlos D. Castillo, and Rama Chellappa. Improving network robustness against adversarial attacks with compact convolution. *arXiv.org*, abs/1712.00699, 2017. Cited on page 17.

[RSJK18] Bita Darvish Rouhani, Mohammad Samragh, Tara Javidi, and Farinaz Koushanfar. Towards safe deep learning: Unsupervised defense against generic adversarial attacks. https://openreview.net/forum?id=HyI6s40a-, 2018. Cited on page 19.

[RSS20] Sukrut Rao, David Stutz, and Bernt Schiele. Adversarial training against location-optimized adversarial patches. In *Proc. of the European Conference on Computer Vision (ECCV) Workshops*, 2020. Cited on pages 10, 17, and 153.

[RUBG17] Gernot Riegler, Ali Osman Ulusoy, Horst Bischof, and Andreas Geiger. OctNetFusion: Learning depth fusion from data. In *Proc. of the International Conference on 3D Vision (3DV)*, 2017. Cited on pages 12, 13, 28, 35, 37, and 46.

[Rud19] Cynthia Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1(5):206–215, 2019. Cited on page 15.

[RUG17] Gernot Riegler, Ali Osman Ulusoy, and Andreas Geiger. Octnet: Learning deep 3d representations at high resolutions. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. Cited on page 13.

[RWA+16] Brandon Reagen, Paul N. Whatmough, Robert Adolf, Saketh Rama, Hyunkwang Lee, Sae Kyu Lee, José Miguel Hernández-Lobato, Gu-Yeon Wei, and David M. Brooks. Minerva: Enabling low-power, highly-accurate deep neural network accelerators. In *ACM/IEEE Annual International Symposium on Computer Architecture (ISCA)*, 2016. Cited on pages 6, 20, 84, 85, and 111.

[RWK20] Leslie Rice, Eric Wong, and J. Zico Kolter. Overfitting in adversarially robust deep learning. In *Proc. of the International Conference on Machine Learning (ICML)*, 2020. Cited on pages 4, 15, 18, 19, 66, 68, 74, 75, 79, and 154.

[RXY+19] Aditi Raghunathan, Sang Michael Xie, Fanny Yang, John C. Duchi, and Percy Liang. Adversarial training can hurt generalization. *arXiv.org*, abs/1906.06032, 2019. Cited on pages 19, 116, 120, and 133.

[RXY+20] Aditi Raghunathan, Sang Michael Xie, Fanny Yang, John C. Duchi, and Percy Liang. Understanding and mitigating the tradeoff between robustness and accuracy. In *Proc. of the International Conference on Machine Learning (ICML)*, 2020. Cited on page 19.

[RYL+21] Adnan Siraj Rakin, Li Yang, Jingtao Li, Fan Yao, Chaitali Chakrabarti, Yu Cao, Jae-sun Seo, and Deliang Fan. RA-BNN: constructing robust & accurate binary neural network to simultaneously defend adversarial bit-flip attack and improve accuracy. *arXiv.org*, abs/2103.13813, 2021. Cited on page 21.

[RZBB20] Jonas Rauber, Roland Zimmermann, Matthias Bethge, and Wieland Brendel. Foolbox native: Fast adversarial attacks to benchmark the robustness of machine learning models in pytorch, tensorflow, and jax. *Journal of Open Source Software*, 5(53):2607, 2020. Cited on page 17.

[SB20] Vivek B. S. and R. Venkatesh Babu. Single-step adversarial training with dropout scheduling. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. Cited on page 18.

[SBC+20] Kihyuk Sohn, David Berthelot, Nicholas Carlini, Zizhao Zhang, Han Zhang, Colin Raffel, Ekin Dogus Cubuk, Alexey Kurakin, and Chun-Liang Li. Fixmatch: Simplifying semi-supervised learning with consistency and confidence. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. Cited on page 239.

[SBS17] Sungho Shin, Yoonho Boo, and Wonyong Sung. Fixed-point optimization of deep neural networks with adaptive step size retraining. In *Proc. of the IEEE Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017. Cited on page 21.

[SC18] Yash Sharma and Pin-Yu Chen. Attacking the madry defense model with l_1-based adversarial examples. In *Proc. of the International Conference on Learning Representations (ICLR) Workshops*, 2018. Cited on page 116.

[SCD+06] Steven M. Seitz, Brian Curless, James Diebel, Daniel Scharstein, and Richard Szeliski. A comparison and evaluation of multi-view stereo reconstruction algorithms. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2006. Cited on page 11.

[SCET20] Fnu Suya, Jianfeng Chi, David Evans, and Yuan Tian. Hybrid batch attacks: Finding black-box adversarial examples with limited queries. In *USENIX Security Symposium*, 2020. Cited on page 16.

[Sch22] Christian Schlarmann. Confidence-calibrated adversarial training and out-of-distribution detection. Master's thesis, University of Tübingen, February 2022. Cited on page 134.

[SCHS21a] David Stutz, Nandhini Chandramoorthy, Matthias Hein, and Bernt Schiele. Bit error robustness for energy-efficient DNN accelerators. In *Proc. of Machine Learning and Systems (MLSys)*, 2021. Cited on pages 2, 9, 10, 66, 67, 73, 79, and 83.

[SCHS21b] David Stutz, Nandhini Chandramoorthy, Matthias Hein, and Bernt Schiele. Random and adversarial bit error robustness: Energy-efficient and secure DNN accelerators. *arXiv.org*, abs/2104.08323, 2021. Cited on pages 2, 9, 10, and 83.

[SCYE17] Vivienne Sze, Yu-Hsin Chen, Tien-Ju Yang, and Joel S. Emer. Efficient processing of deep neural networks: A tutorial and survey. *Proceedings of the IEEE*, 105(12), 2017. Cited on pages 1, 20, 21, and 84.

[SDCD21] David Stutz, Krishnamurthy Dvijotham, Ali Taylan Cemgil, and Arnaud Doucet. Learning optimal conformal classifiers. In *Proc. of the International Conference on Learning Representations (ICLR)*, 2021. Cited on pages 2, 9, 10, and 135.

[SDYT09] Romeil Sandhu, Samuel Dambreville, Anthony J. Yezzi, and Allen Tannenbaum. Non-rigid 2d-3d pose estimation and 2d image segmentation. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009. Cited on page 12.

[SDYT11] Romeil Sandhu, Samuel Dambreville, Anthony J. Yezzi, and Allen Tannenbaum. A nonrigid kernel-based framework for 2d-3d pose estimation and 2d image segmentation. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 33(6):1098–1115, 2011. Cited on page 12.

[SE19] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. Cited on page 153.

[SeSM19] Liwei Song, eza Shokri, and Prateek Mittal. Membership inference attacks against adversarially robust deep learning models. In *Proc. of the IEEE Symposium on Security and Privacy Workshops*, pages 50–56, 2019. Cited on page 15.

[SF20] Sahil Singla and Soheil Feizi. Second-order provable defenses against adversarial attacks. In *Proc. of the International Conference on Machine Learning (ICML)*, 2020. Cited on page 18.

[SFCH12] Chao-Hui Shen, Hongbo Fu, Kang Chen, and Shi-Min Hu. Structure recovery by part assembly. *ACM Trans. on Graphics*, 31(6):180:1–180:11, 2012. Cited on page 12.

[SFD+14] Frank Seide, Hao Fu, Jasha Droppo, Gang Li, and Dong Yu. 1-bit stochastic gradient descent and its application to data-parallel distributed training of speech DNNs. In *Annual Conference of the International Speech Communication Association*, 2014. Cited on page 21.

[SG18a] Lewis Smith and Yarin Gal. Understanding measures of uncertainty for adversarial example detection. In *Proc. of the Conference on Uncertainty in Artificial Intelligence (UAI)*, 2018. Cited on page 19.

[SG18b] David Stutz and Andreas Geiger. Learning 3d shape completion from laser scan data with weak supervision. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. Cited on pages 8, 27, and 29.

[SG20] David Stutz and Andreas Geiger. Learning 3D shape completion under weak supervision. *International Journal of Computer Vision (IJCV)*, 128(5):1162–1181, 2020. Cited on pages 2, 8, 10, and 27.

[SGF16] Abhishek Sharma, Oliver Grau, and Mario Fritz. Vconv-dae: Deep volumetric shape learning without object labels. In *Proc. of the European Conference on Computer Vision (ECCV) Workshops*, 2016. Cited on pages 12, 28, 40, and 46.

[SGHG19] Ali Shafahi, Amin Ghiasi, Furong Huang, and Tom Goldstein. Label smoothing and logit squeezing: A replacement for adversarial training? *arXiv.org*, abs/1910.11585, 2019. Cited on page 17.

[SGM+18] Gagandeep Singh, Timon Gehr, Matthew Mirman, Markus Püschel, and Martin T. Vechev. Fast and effective robustness certification. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 10825–10836, 2018. Cited on page 18.

[SGOS+18] Carl-Johann Simon-Gabriel, Yann Ollivier, Bernhard Schölkopf, Léon Bottou, and David Lopez-Paz. Adversarial vulnerability of neural networks increases with input dimension. *arXiv.org*, abs/1802.01421, 2018. Cited on page 17.

[SGZ+16] Tim Salimans, Ian J. Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2016. Cited on page 153.

[SHES17] Yichen Shen, Nicholas Christopher Harris, Dirk R. Englund, and Marin Soljacic. Deep learning with coherent nanophotonic circuits. *Berkeley Symposium on Energy Efficient Electronic Systems & Steep Transistors Workshop (E3S)*, 2017. Cited on pages 20 and 155.

[Shi00] Hidetoshi Shimodaira. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of statistical planning and inference*, 90(2):227–244, 2000. Cited on page 23.

[SHJU17] Thilo Strauss, Markus Hanselmann, Andrej Junginger, and Holger Ulmer. Ensemble methods as a defense to adversarial perturbations against deep neural networks. *arXiv.org*, abs/1709.03423, 2017. Cited on page 17.

[SHM21] Changhao Shi, Chester Holtz, and Gal Mishne. Online adversarial purification based on self-supervised learning. In *Proc. of the International Conference on Learning Representations (ICLR)*, 2021. Cited on page 18.

[SHN+18] Ali Shafahi, W. Ronny Huang, Mahyar Najibi, Octavian Suciu, Christoph Studer, Tudor Dumitras, and Tom Goldstein. Poison frogs! targeted clean-label poisoning attacks on neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018. Cited on page 15.

[SHN+19] Shunsuke Saito, Zeng Huang, Ryota Natsume, Shigeo Morishima, Hao Li, and Angjoo Kanazawa. Pifu: Pixel-aligned implicit function for high-resolution clothed human digitization. In *Proc. of the IEEE International Conference on Computer Vision (ICCV)*, 2019. Cited on page 13.

[SHS19] David Stutz, Matthias Hein, and Bernt Schiele. Disentangling adversarial robustness and generalization. *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. Cited on pages 2, 8, 10, 49, 116, 118, 120, and 133.

[SHS20] David Stutz, Matthias Hein, and Bernt Schiele. Confidence-calibrated adversarial training: Generalizing to unseen attacks. In *Proc. of the International Conference on Machine Learning (ICML)*, 2020. Cited on pages 2, 9, 10, 73, 97, 115, and 134.

[SHS21] David Stutz, Matthias Hein, and Bernt Schiele. Relating adversarially robust generalization to flat minima. In *Proc. of the IEEE International Conference on Computer Vision (ICCV)*, 2021. Cited on pages 2, 9, 10, and 65.

[SHT20] Yucheng Shi, Yahong Han, and Qi Tian. Polishing decision-based adversarial noise with a customized sampling. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. Cited on page 16.

[SKAG15] Minhyuk Sung, Vladimir G. Kim, Roland Angst, and Leonidas J. Guibas. Data-driven structural priors for shape completion. *ACM Trans. on Graphics*, 34(6):175:1–175:11, 2015. Cited on pages 3 and 12.

[SKC13] Frank Steinbrucker, Christian Kerl, and Daniel Cremers. Large-scale multi-resolution surface reconstruction from rgb-d sequences. In *Proc. of the IEEE International Conference on Computer Vision (ICCV)*, 2013. Cited on page 32.

[SKC+20] Philip Sperl, Ching-Yu Kao, Peng Chen, Xiao Lei, and Konstantin Böttinger. DLA: dense-layer-analysis for adversarial example detection. In *Proc. of the IEEE European Symposium on Security and Privacy (EuroS&P)*, 2020. Cited on page 19.

[SKM+20] Mayank Singh, Nupur Kumari, P. Mangla, Abhishek Sinha, V. Balasubramanian, and Balaji Krishnamurthy. Attributional robustness training using input-gradient spatial alignment. In *Proc. of the European Conference on Computer Vision (ECCV)*, 2020. Cited on pages 15 and 16.

[SKN+18] Yang Song, Taesup Kim, Sebastian Nowozin, Stefano Ermon, and Nate Kushman. Pixeldefend: Leveraging generative models to understand and defend against adversarial examples. *Proc. of the International Conference on Learning Representations (ICLR)*, 2018. Cited on pages 17 and 50.

[SLH+21] Zheyan Shen, Jiashuo Liu, Yue He, Xingxuan Zhang, Renzhe Xu, Han Yu, and Peng Cui. Towards out-of-distribution generalization: A survey. *arXiv.org*, abs/2108.13624, 2021. Cited on page 14.

[SLK21] Fatemeh Sheikholeslami, Ali Lotfi, and J. Zico Kolter. Provably robust classification of adversarial examples with detection. In *Proc. of the International Conference on Learning Representations (ICLR)*, 2021. Cited on page 18.

[SLR+19] Hadi Salman, Jerry Li, Ilya P. Razenshteyn, Pengchuan Zhang, Huan Zhang, Sébastien Bubeck, and Greg Yang. Provably robust deep learning via adversarially trained smoothed classifiers. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. Cited on page 18.

[SLW19] Mauricio Sadinle, Jing Lei, and Larry Wasserman. Least ambiguous set-valued classifiers with bounded error levels. *Journal of the American Statistical Association (JASA)*, 114(525):223–234, 2019. Cited on pages 23, 136, 137, 138, 139, 140, and 149.

[SM17] Edward J. Smith and David Meger. Improved adversarial systems for 3d object generation and reconstruction. In *Proc. of the Conference on Robot Learning (CoRL)*, 2017. Cited on pages 12, 28, 40, and 46.

[SMB21] Adi Shamir, Odelia Melamed, and Oriel BenShmuel. The dimpled manifold model of adversarial examples in machine learning. *arXiv.org*, abs/2106.10151, 2021. Cited on page 17.

[SMH+22] Vikash Sehwag, Saeed Mahloujifar, Tinashe Handina, Sihui Dai, Chong Xiang, Mung Chiang, and Prateek Mittal. Robust learning meets generative models: Can proxy distributions improve adversarial robustness? In *Proc. of the International Conference on Learning Representations (ICLR)*, 2022. Cited on page 18.

[SMK21] Jacob M. Springer, Melanie Mitchell, and Garrett T. Kenyon. Adversarial perturbations are not so weird: Entanglement of robust and non-robust features in neural network classifiers. *arXiv.org*, abs/2102.05110, 2021. Cited on page 17.

[SN20] Samuel Henrique Silva and Peyman Najafirad. Opportunities and challenges in deep learning adversarial robustness: A survey. *arXiv.org*, abs/2007.00753, 2020. Cited on page 15.

[SNG+19] Ali Shafahi, Mahyar Najibi, Amin Ghiasi, Zheng Xu, John P. Dickerson, Christoph Studer, Larry S. Davis, Gavin Taylor, and Tom Goldstein. Adversarial training for free! In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. Cited on page 18.

[SNX+18] Ali Shafahi, Mahyar Najibi, Zheng Xu, John P. Dickerson, Larry S. Davis, and Tom Goldstein. Universal adversarial training. *arXiv.org*, abs/1811.11304, 2018. Cited on page 18.

[SO20] Chandramouli Shama Sastry and Sageev Oore. Detecting out-of-distribution examples with gram matrices. In *Proc. of the International Conference on Machine Learning (ICML)*, 2020. Cited on page 22.

[SOF+19] Jasper Snoek, Yaniv Ovadia, Emily Fertig, Balaji Lakshminarayanan, Sebastian Nowozin, D. Sculley, Joshua V. Dillon, Jie Ren, and Zachary Nado. Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. Cited on pages 15 and 22.

[SOY+20] Behzad Salami, Erhan Baturay Onural, Ismail Emir Yuksel, Fahrettin Koc, Oguz Ergin, Adrián Cristal Kestelman, Osman S. Unsal, Hamid Sarbazi-Azad, and Onur Mutlu. An experimental study of reduced-voltage operation in modern fpgas for neural network acceleration. In *IEEE/IFIP International Conference on Dependable Systems and Networks*, 2020. Cited on page 20.

[SPS+18] Hardik Sharma, Jongse Park, Naveen Suda, Liangzhen Lai, Benson Chau, Joon Kyung Kim, Vikas Chandra, and Hadi Esmaeilzadeh. Bit fusion: Bit-level dynamically composable architecture for accelerating deep neural networks. In *ACM/IEEE Annual International Symposium on Computer Architecture (ISCA)*, 2018. Cited on page 84.

[SRBB19] Lukas Schott, Jonas Rauber, Matthias Bethge, and Wieland Brendel. Towards the first adversarially robust neural network model on MNIST. In *Proc. of the International Conference on Learning Representations (ICLR)*, 2019. Cited on pages 17, 50, 57, 63, and 130.

[SRPR20] Saima Sharmin, Nitin Rathi, Priyadarshini Panda, and K. Roy. Inherent adversarial robustness of deep spiking neural networks: Effects of discrete input encoding and non-linear activations. In *Proc. of the European Conference on Computer Vision (ECCV)*, 2020. Cited on page 17.

[SRR20] Sanchari Sen, Balaraman Ravindran, and Anand Raghunathan. EMPIR: ensembles of mixed precision deep networks for increased robustness against adversarial attacks. In *Proc. of the International Conference on Learning Representations (ICLR)*, 2020. Cited on page 17.

[SRS17] Congzheng Song, Thomas Ristenpart, and Vitaly Shmatikov. Machine learning models that remember too much. In *Proc. of the ACM Conference on Computer and Communications Security (CCS)*, pages 587–601, 2017. Cited on page 15.

[SSC20] Ali Shahin Shamsabadi, Ricardo Sánchez-Matilla, and Andrea Cavallaro. Colorfool: Semantic adversarial colorization. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. Cited on page 17.

[SSH15] Wonyong Sung, Sungho Shin, and Kyuyeon Hwang. Resiliency of deep neural networks under quantization. *arXiv.org*, abs/1511.06488, 2015. Cited on pages 14, 20, 21, 84, 85, 91, 101, and 111.

[SSJF21] Vasu Singla, Sahil Singla, David Jacobs, and Soheil Feizi. Low curvature activations reduce overfitting in adversarial training. *arXiv.org*, abs/2102.07861, 2021. Cited on pages 19, 66, 67, 154, and 237.

[SSKE18] Yang Song, Rui Shu, Nate Kushman, and Stefano Ermon. Constructing unrestricted adversarial examples with generative models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018. Cited on pages 17 and 50.

[SSRD19] Adi Shamir, Itay Safran, Eyal Ronen, and Orr Dunkelman. A simple explanation for the existence of adversarial examples with small hamming distance. *arXiv.org*, abs/1901.10861, 2019. Cited on page 17.

[SSSS17] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *Proc. of the IEEE Symposium on Security and Privacy*, 2017. Cited on page 15.

[SST+18] Ludwig Schmidt, Shibani Santurkar, Dimitris Tsipras, Kunal Talwar, and Aleksander Madry. Adversarially robust generalization requires more data. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018. Cited on pages 19, 62, and 66.

[SSWK21] Satya Narayan Shukla, Anit Kumar Sahu, Devin Willmott, and J. Zico Kolter. Simple and efficient hard label black-box adversarial attacks in low query budget regimes. In *Proc. of the ACM International Conference on Knowledge Discovery & Data Mining*, 2021. Cited on page 16.

[SSY+20] Hadi Salman, Mingjie Sun, Greg Yang, Ashish Kapoor, and J. Zico Kolter. Denoised smoothing: A provable defense for pretrained classifiers. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. Cited on page 153.

[SSZ21] Reza Shokri, Martin Strobel, and Yair Zick. On the privacy risks of model explanations. In *Proc. of the AAAI/ACM Conference on AI, Ethics, and Society (AIES)*, 2021. Cited on page 15.

[STIM18] Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry. How does batch normalization help optimization? In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018. Cited on pages 19 and 155.

[Stu17] David Stutz. Learning shape completion from bounding boxes with cad shape priors. Master's thesis, RWTH Aachen University, September 2017. Cited on pages 8, 27, and 29.

[SUC18] Behzad Salami, O. Unsal, and A. Cristal. On the resilience of rtl nn accelerators: Fault characterization and mitigation. *International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD)*, 2018. Cited on page 22.

[SV08] Glenn Shafer and Vladimir Vovk. A tutorial on conformal prediction. *Journal of Machine Learning Research (JMLR)*, 9:371–421, 2008. Cited on page 23.

[SVI+16] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. Cited on pages 67, 73, 102, 120, 236, 248, and 249.

[SW19] Chawin Sitawarin and David A. Wagner. On the robustness of deep k-nearest neighbors. In *Proc. of the IEEE Symposium on Security and Privacy*, 2019. Cited on page 14.

[SW20] Chawin Sitawarin and David A. Wagner. Minimum-norm adversarial examples on KNN and KNN based models. In *Proc. of the IEEE Symposium on Security and Privacy Workshops*, pages 34–40. IEEE, 2020. Cited on page 14.

[SWGG20] Manli Shu, Zuxuan Wu, Micah Goldblum, and Tom Goldstein. Prepare for the worst: Generalizing across domain shifts with adversarial batch normalization. *arXiv.org*, abs/2009.08965, 2020. Cited on page 155.

[SWMG15] Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *Proc. of the International Conference on Machine Learning (ICML)*, 2015. Cited on page 153.

[SWS+16] Gopalakrishnan Srinivasan, Parami Wijesinghe, Syed Shakib Sarwar, Akhilesh Jaiswal, and Kaushik Roy. Significance driven hybrid 8t-6t SRAM for energy-efficient synaptic storage in artificial neural networks. In *Proc. of the Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2016. Cited on pages 6 and 20.

[SWW+20] Shawn Shan, Emily Wenger, Bolun Wang, Bo Li, Haitao Zheng, and Ben Y. Zhao. Gotta catch'em all: Using honeypots to catch adversarial attacks on neural networks. In *Proc. of the ACM Conference on Computer and Communications Security (CCS)*, 2020. Cited on page 19.

[SX14] Shuran Song and Jianxiong Xiao. Sliding shapes for 3D object detection in depth images. In *Proc. of the European Conference on Computer Vision (ECCV)*, 2014. Cited on page 12.

[SYO+18] Sining Sun, Ching-Feng Yeh, Mari Ostendorf, Mei-Yuh Hwang, and Lei Xie. Training augmentation with adversarial examples for robust speech recognition. In *Annual Conference of the International Speech Communication Association*, 2018. Cited on page 16.

[SZ15] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *Proc. of the International Conference on Learning Representations (ICLR)*, 2015. Cited on pages 63, 97, and 230.

[SZB+21] Ilia Shumailov, Yiren Zhao, Daniel Bates, Nicolas Papernot, Robert D. Mullins, and Ross Anderson. Sponge examples: Energy-latency attacks on neural networks. In *Proc. of the IEEE European Symposium on Security and Privacy (EuroS&P)*, 2021. Cited on page 15.

[SZC+18] Dong Su, Huan Zhang, Hongge Chen, Jinfeng Yi, Pin-Yu Chen, and Yupeng Gao. Is robustness the cost of accuracy? - A comprehensive study on the robustness of 18 deep image classification models. In *Proc. of the European Conference on Computer Vision (ECCV)*, 2018. Cited on pages 5, 19, 49, 50, 51, 60, 61, and 63.

[SZS+14] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *Proc. of the International Conference on Learning Representations (ICLR)*, 2014. Cited on pages 1, 4, 14, 16, 17, 49, 50, 53, 66, 89, 116, 118, and 231.

[TB19] Florian Tramèr and Dan Boneh. Adversarial training and robustness for multiple perturbations. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. Cited on pages 7, 18, 116, and 154.

[TBCR19] Ryan J. Tibshirani, Rina Foygel Barber, Emmanuel J. Candès, and Aaditya Ramdas. Conformal prediction under covariate shift. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems (NeurIPS)*, pages 2526–2536, 2019. Cited on page 23.

[TCB+19] Sunil Thulasidasan, Gopinath Chennupati, Jeff A. Bilmes, Tanmoy Bhattacharya, and Sarah Michalak. On mixup training: Improved calibration and predictive uncertainty for deep neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. Cited on pages 14 and 22.

[TCC+20] Li-Huang Tsai, Shih-Chieh Chang, Yu-Ting Chen, Jia-Yu Pan, Wei Wei, and Da-Cheng Juan. Calibrated batchnorm: Improving robustness against noisy weights in neural networks. *arXiv.org*, abs/2007.03230, 2020. Cited on page 21.

[TDB17] M. Tatarchenko, A. Dosovitskiy, and T. Brox. Octree generating networks: Efficient convolutional architectures for high-resolution 3d outputs. In *Proc. of the IEEE International Conference on Computer Vision (ICCV)*, 2017. Cited on page 13.

[TFT+20] Ayush Tewari, Ohad Fried, Justus Thies, Vincent Sitzmann, Stephen Lombardi, Kalyan Sunkavalli, Ricardo Martin-Brualla, Tomas Simon, Jason M. Saragih, Matthias Nießner, Rohit Pandey, Sean Ryan Fanello, Gordon Wetzstein, Jun-Yan Zhu, Christian Theobalt, Maneesh Agrawala, Eli Shechtman, Dan B. Goldman, and Michael Zollhöfer. State of the art on neural rendering. *Computer Graphics Forum*, 39(2):701–727, 2020. Cited on page 13.

[TG16] Thomas Tanay and Lewis Griffin. A boundary tilting persepective on the phenomenon of adversarial examples. *arXiv.org*, abs/1608.07690, 2016. Cited on pages 4, 17, 50, and 63.

[TG17] César Torres-Huitzil and Bernard Girau. Fault and error tolerance in neural networks: A review. *IEEE Access*, 5, 2017. Cited on page 21.

[THYC21a] Yu-Lin Tsai, Chia-Yi Hsu, Chia-Mu Yu, and Pin-Yu Chen. Formalizing generalization and robustness of neural networks to weight perturbations. *arXiv.org*, abs/2103.02200, 2021. Cited on page 21.

[THYC21b] Yu-Lin Tsai, Chia-Yi Hsu, Chia-Mu Yu, and Pin-Yu Chen. Non-singular adversarial robustness of neural networks. In *Proc. of the IEEE Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021. Cited on page 21.

[TMJS20] Jihoon Tack, Sangwoo Mo, Jongheon Jeong, and Jinwoo Shin. CSI: novelty detection via contrastive learning on distributionally shifted instances. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. Cited on page 22.

[TMWP21] Jean-Baptiste Truong, Pratyush Maini, Robert J. Walls, and Nicolas Papernot. Data-free model extraction. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. Cited on page 15.

[TOZ+11] Andrea Tagliasacchi, Matt Olson, Hao Zhang, Ghassan Hamarneh, and Daniel Cohen-Or. VASE: volume-aware surface evolution for surface reconstruction from incomplete point clouds. *Computer Graphics Forum*, 30(5), 2011. Cited on page 12.

[TPG+17] Florian Tramèr, Nicolas Papernot, Ian J. Goodfellow, Dan Boneh, and Patrick D. McDaniel. The space of transferable adversarial examples. *arXiv.org*, abs/1704.03453, 2017. Cited on page 16.

[TSE+18] Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. Robustness may be at odds with accuracy. *arXiv.org*, abs/1805.12152, 2018. Cited on pages 61 and 159.

[TSE+19] Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. Robustness may be at odds with accuracy. In *Proc. of the International Conference on Learning Representations (ICLR)*, 2019. Cited on pages 4, 5, 15, 18, 19, 49, 50, 51, 60, 61, 63, 116, 120, and 133.

[TSS17] Adrian Tang, Simha Sethumadhavan, and Salvatore J. Stolfo. CLKSCREW: exposing the perils of security-oblivious energy management. In *USENIX Security Symposium*, 2017. Cited on pages 20, 84, and 87.

[TTC+19] Chun-Chen Tu, Pai-Shun Ting, Pin-Yu Chen, Sijia Liu, Huan Zhang, Jinfeng Yi, Cho-Jui Hsieh, and Shin-Ming Cheng. Autozoom: Autoencoder-based zeroth order optimization method for attacking black-box neural networks. In *Proc. of the Conference on Artificial Intelligence (AAAI)*, 2019. Cited on page 16.

[TTD+21] Sunil Thulasidasan, Sushil Thapa, Sayera Dhaubhadel, Gopinath Chennupati, Tanmoy Bhattacharya, and Jeff A. Bilmes. An effective baseline for robustness to distributional shift. In *Proc. of the International Conference on Machine Learning and Applications (ICMLA)*, 2021. Cited on page 22.

[TTV16] Pedro Tabacof, Julia Tavares, and Eduardo Valle. Adversarial images for variational autoencoders. *arXiv.org*, abs/1612.00155, 2016. Cited on page 16.

[TW05] Sebastian Thrun and Ben Wegbreit. Shape from symmetry. In *Proc. of the IEEE International Conference on Computer Vision (ICCV)*, pages 1824–1831, 2005. Cited on page 12.

[TZJ+16] Florian Tramèr, Fan Zhang, Ari Juels, Michael K. Reiter, and Thomas Ristenpart. Stealing machine learning models via prediction apis. In *USENIX Security Symposium*, 2016. Cited on page 15.

[TZLD21] Jinyu Tian, Jiantao Zhou, Yuanman Li, and Jia Duan. Detecting adversarial examples from sensitivity inconsistency of spatial-transform domain. In *Proc. of the Conference on Artificial Intelligence (AAAI)*, 2021. Cited on page 19.

[UOKvdO18] Jonathan Uesato, Brendan O'Donoghue, Pushmeet Kohli, and Aäron van den Oord. Adversarial risk and the dangers of evaluating against weak attacks. In *Proc. of the International Conference on Machine Learning (ICML)*, 2018. Cited on page 16.

[Var19] Kush R. Varshney. Trustworthy machine learning and artificial intelligence. *XRDS*, 25(3):26–29, 2019. Cited on pages 1, 13, and 15.

[vASTG20] Joost van Amersfoort, Lewis Smith, Yee Whye Teh, and Yarin Gal. Uncertainty estimation using a single deep deterministic neural network. In *Proc. of the International Conference on Machine Learning (ICML)*, 2020. Cited on page 22.

[vdMH08] Laurens van der Maaten and Geoffrey Hinton. Visualizing high-dimensional data using t-sne. *Journal of Machine Learning Research (JMLR)*, 9:2579–2605, 2008. Cited on page 40.

[VDR+17] Jacob Varley, Chad DeChant, Adam Richardson, Joaquín Ruales, and Peter K. Allen. Shape completion enabled robotic grasping. In *Proc. of the IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2017. Cited on pages 28, 39, and 46.

[VGS05] Vladimir Vovk, Alex Gammerman, and Glenn Shafer. *Algorithmic Learning in a Random World*. Springer-Verlag, Berlin, Heidelberg, 2005. Cited on pages 7, 14, 22, 23, and 136.

[VJZ+18] Apoorv Vyas, Nataraj Jammalamadaka, Xia Zhu, Dipankar Das, Bharat Kaul, and Theodore L. Willke. Out-of-distribution detection using an ensemble of self supervised leave-out classifiers. In *Proc. of the European Conference on Computer Vision (ECCV)*, 2018. Cited on page 22.

[Vov12] Vladimir Vovk. Conditional validity of inductive conformal predictors. In *Proc. of the Asian Conference on Machine Learning (ACML)*, 2012. Cited on page 23.

[Vov13] Vladimir Vovk. Cross-conformal predictors. *Annals of Mathematics and Artificial Intelligence*, 74:9–28, 2013. Cited on page 23.

[VWA+19] Juozas Vaicenavicius, David Widmann, Carl R. Andersson, Fredrik Lindsten, Jacob Roll, and Thomas B. Schön. Evaluating model calibration in classification. In *Conference on Artificial Intelligence and Statistics (AISTATS)*, 2019. Cited on page 22.

[WBH19] Hongzhi Wang, Mohamed Jaward Bah, and Mohamed Hammad. Progress in outlier detection techniques: A survey. *IEEE Access*, 7:107964–108000, 2019. Cited on page 14.

[WBR+20] Jim Winkens, Rudy Bunel, Abhijit Guha Roy, Robert Stanforth, Vivek Natarajan, Joseph R. Ledsam, Patricia MacWilliams, Pushmeet Kohli, Alan Karthikesalingam, Simon Kohl, A. Taylan Cemgil, S. M. Ali Eslami, and Olaf Ronneberger. Contrastive training for improved out-of-distribution detection. *arXiv.org*, abs/2007.05566, 2020. Cited on page 22.

[WBSS04] Zhou Wang, Alan C. Bovik, Hamid R. Sheikh, and Eero P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Trans. on Image Processing (TIP)*, 13(4):600–612, 2004. Cited on page 16.

[WCC+21] Boxi Wu, Jinghui Chen, Deng Cai, Xiaofei He, and Quanquan Gu. Do wider neural networks really help adversarial robustness? In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. Cited on page 18.

[WCG+20] Haotao Wang, Tianlong Chen, Shupeng Gui, Ting-Kuei Hu, Ji Liu, and Zhangyang Wang. Once-for-all adversarial training: In-situ tradeoff between robustness and accuracy for free. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. Cited on pages 18, 19, and 153.

[WCY20] Weitao Wag, Jiansheng Chen, and Ming-Hsuan Yang. Adversarial training with bidirectional likelihood regularization for visual classification. In *Proc. of the European Conference on Computer Vision (ECCV)*, 2020. Cited on page 18.

[WES19] Yannan Nellie Wu, Joel S. Emer, and Vivienne Sze. Accelergy: An architecture-level energy estimation methodology for accelerator designs. In David Z. Pan, editor, *Proc. of the International Conference on Computer-Aided Design*, 2019. Cited on pages 89 and 155.

[WG18] Binghui Wang and Neil Zhenqiang Gong. Stealing hyperparameters in machine learning. In *Proc. of the IEEE Symposium on Security and Privacy*, 2018. Cited on pages 15 and 16.

[WGS+21] Olivia Wiles, Sven Gowal, Florian Stimberg, Sylvestre-Alvise Rebuffi, Ira Ktena, Krishnamurthy Dvijotham, and A. Taylan Cemgil. A fine-grained analysis on distribution shift. *arXiv.org*, abs/2110.11328, 2021. Cited on page 14.

[WGU+21] Johannes Welbl, Amelia Glaese, Jonathan Uesato, Sumanth Dathathri, John Mellor, Lisa Anne Hendricks, Kirsty Anderson, Pushmeet Kohli, Ben Coppin, and Po-Sen Huang. Challenges in detoxifying language models. In *Findings of the Association for Computational Linguistics: EMNLP*, 2021. Cited on page 15.

[WH18] Yuxin Wu and Kaiming He. Group normalization. In *Proc. of the European Conference on Computer Vision (ECCV)*, pages 3–19, 2018. Cited on pages 21, 89, 95, 97, 155, and 247.

[WH19] Bram Wallace and Bharath Hariharan. Few-shot generalization for single-image 3d reconstruction via priors. In *Proc. of the IEEE International Conference on Computer Vision (ICCV)*, 2019. Cited on page 12.

[WHK20] Bryan Wilder, Eric Horvitz, and Ece Kamar. Learning to complement humans. In *Proc. of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2020. Cited on page 267.

[WHY+17] Weiyue Wang, Qiangui Huang, Suya You, Chao Yang, and Ulrich Neumann. Shape inpainting using 3d generative adversarial network and recurrent convolutional networks. In *Proc. of the IEEE International Conference on Computer Vision (ICCV)*, 2017. Cited on pages 28, 39, and 46.

[Wil20] John H Williamson. Differentiable parallel approximate sorting networks. https://johnhw.github.io/differentiable_sorting/index.md.html, 2020. Cited on pages 139 and 140.

[WJC18] Yizhen Wang, Somesh Jha, and Kamalika Chaudhuri. Analyzing the robustness of nearest neighbors to adversarial examples. In *Proc. of the International Conference on Machine Learning (ICML)*, 2018. Cited on page 17.

[WJM+20] Yeming Wen, Ghassen Jerfel, Rafael Muller, Michael W Dusenberry, Jasper Snoek, Balaji Lakshminarayanan, and Dustin Tran. Improving calibration of batchensemble with data augmentation. In *Proc. of the International Conference on Machine Learning (ICML) Workshops*, volume 6, 2020. Cited on page 14.

[WJM+21] Yeming Wen, Ghassen Jerfel, Rafael Muller, Michael W. Dusenberry, Jasper Snoek, Balaji Lakshminarayanan, and Dustin Tran. Combining ensembles and data augmentation can harm your calibration. In *Proc. of the International Conference on Learning Representations (ICLR)*, 2021. Cited on page 14.

[WK18] Eric Wong and J. Zico Kolter. Provable defenses against adversarial examples via the convex outer adversarial polytope. In *Proc. of the International Conference on Machine Learning (ICML)*, 2018. Cited on pages 18 and 153.

[WK21] Eric Wong and J. Zico Kolter. Learning perturbation sets for robust machine learning. In *Proc. of the International Conference on Learning Representations (ICLR)*, 2021. Cited on pages 17 and 154.

[WL16] Dilin Wang and Qiang Liu. Learning to draw samples: With application to amortized MLE for generative adversarial learning. *arXiv.org*, 1611.01722, 2016. Cited on page 12.

[WLG+17] Peng-Shuai Wang, Yang Liu, Yu-Xiao Guo, Chun-Yu Sun, and Xin Tong. O-cnn: Octree-based convolutional neural networks for 3d shape analysis. In *ACM Trans. on Graphics (SIGGRAPH)*, 2017. Cited on page 13.

[WLS+15] Thomas Whelan, Stefan Leutenegger, Renato F. Salas-Moreno, Ben Glocker, and Andrew J. Davison. Elasticfusion: Dense SLAM without A pose graph. In *Proc. of Robotics: Science and Systems (RSS)*, 2015. Cited on page 36.

[WLT20] Peng-Shuai Wang, Yang Liu, and Xin Tong. Deep octree-based cnns with output-guided skip connections for 3d shape and scene completion. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2020. Cited on pages 12 and 13.

[WLX+20] Shu-Fan Wang, Ningyi Liao, L. Xiang, Nanyang Ye, and Q. Zhang. Achieving adversarial robustness via sparsity. *arXiv.org*, abs/2009.05423, 2020. Cited on page 17.

[WRK20] Eric Wong, Leslie Rice, and J. Zico Kolter. Fast is better than free: Revisiting adversarial training. In *Proc. of the International Conference on Learning Representations (ICLR)*, 2020. Cited on pages 16, 18, and 236.

[WRTK19] Stefan Webb, Tom Rainforth, Yee Whye Teh, and M. Pawan Kumar. A statistical approach to assessing neural network robustness. In *Proc. of the International Conference on Learning Representations (ICLR)*, 2019. Cited on page 18.

[WRV+20] Florian Wenzel, Kevin Roth, Bastiaan S. Veeling, Jakub Swiatkowski, Linh Tran, Stephan Mandt, Jasper Snoek, Tim Salimans, Rodolphe Jenatton, and Sebastian Nowozin. How good is the bayes posterior in deep neural networks really? In *Proc. of the International Conference on Machine Learning (ICML)*, 2020. Cited on page 14.

[WSK+15] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. Cited on pages 12, 13, 27, 28, 29, 31, 35, and 46.

[WSMK18] Eric Wong, Frank R. Schmidt, Jan Hendrik Metzen, and J. Zico Kolter. Scaling provable adversarial defenses. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018. Cited on page 18.

[WSS+21] Devin Willmott, Anit Kumar Sahu, Fatemeh Sheikholeslami, Filipe Condessa, and J. Zico Kolter. You only query once: Effective black box adversarial attacks with minimal repeated queries. *arXiv.org*, abs/2102.00029, 2021. Cited on page 16.

[WSTJ20] Florian Wenzel, Jasper Snoek, Dustin Tran, and Rodolphe Jenatton. Hyperparameter ensembles for robustness and uncertainty quantification. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. Cited on page 14.

[WT11] Max Welling and Yee Whye Teh. Bayesian learning via stochastic gradient langevin dynamics. In *Proc. of the International Conference on Machine Learning (ICML)*, 2011. Cited on page 14.

[WTB20] Yeming Wen, Dustin Tran, and Jimmy Ba. Batchensemble: an alternative approach to efficient ensemble and lifelong learning. In *Proc. of the International Conference on Learning Representations (ICLR)*, 2020. Cited on page 14.

[WW04] Gregory L. Wittel and Shyhtsun Felix Wu. On attacking statistical spam filters. In *Proc. of the Conference on Email and Anti-Spam (CEAS)*, 2004. Cited on page 14.

[WW22] Hongjun Wang and Yisen Wang. Self-ensemble adversarial training for improved robustness. In *Proc. of the International Conference on Learning Representations (ICLR)*, 2022. Cited on page 18.

[WWHX20] Haohan Wang, Xindi Wu, Zeyi Huang, and Eric P. Xing. High-frequency component helps explain the generalization of convolutional neural networks. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. Cited on page 17.

[WWX+20] Dongxian Wu, Yisen Wang, Shu-Tao Xia, James Bailey, and Xingjun Ma. Skip connections matter: On the transferability of adversarial examples generated with resnets. In *Proc. of the International Conference on Learning Representations (ICLR)*, 2020. Cited on page 16.

[WWZ+18] Siyue Wang, Xiao Wang, Pu Zhao, Wujie Wen, David R. Kaeli, Peter Chin, and Xue Lin. Defensive dropout for hardening deep neural networks under adversarial attacks. In *Proc. of the International Conference on Computer-Aided Design*, pages 71:1–71:8, 2018. Cited on page 17.

[WXW20a] Xunguang Wang, Ship Peng Xu, and Eric Ke Wang. Initializing perturbations in multiple directions for fast adversarial training. *arXiv.org*, abs/2005.07606, 2020. Cited on page 18.

[WXW20b] Dongxian Wu, Shu-Tao Xia, and Yisen Wang. Adversarial weight perturbation helps robust generalization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. Cited on pages 5, 14, 18, 19, 65, 66, 67, 69, 70, 71, 77, 79, 154, 235, and 237.

[WZC+20] Yihan Wang, H. Zhang, H. Chen, D. Boning, and C. Hsieh. On ℓ_p-norm robustness of ensemble stumps and trees. *arXiv.org*, abs/2010.07344, 2020. Cited on page 14.

[WZL+20] Tsui-Wei Weng, Pu Zhao, Sijia Liu, Pin-Yu Chen, Xue Lin, and Luca Daniel. Towards certificated model robustness against weight perturbations. In *Proc. of the Conference on Artificial Intelligence (AAAI)*, 2020. Cited on page 21.

[WZX+16a] Jiajun Wu, Chengkai Zhang, Tianfan Xue, Bill Freeman, and Josh Tenenbaum. Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2016. Cited on pages 12, 13, 28, and 40.

[WZX+16b] Jiajun Wu, Chengkai Zhang, Tianfan Xue, Bill Freeman, and Josh Tenenbaum. Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2016. Cited on page 73.

[WZY+20] Yisen Wang, Difan Zou, Jinfeng Yi, James Bailey, Xingjun Ma, and Quanquan Gu. Improving adversarial robustness requires revisiting misclassified examples. In *Proc. of the International Conference on Learning Representations (ICLR)*, 2020. Cited on pages 67, 73, 236, and 238.

[WZYS19] Xingxing Wei, Jun Zhu, Sha Yuan, and Hang Su. Sparse adversarial perturbations for videos. In *Proc. of the Conference on Artificial Intelligence (AAAI)*, 2019. Cited on page 16.

[XEQ18] Weilin Xu, David Evans, and Yanjun Qi. Feature squeezing: Detecting adversarial examples in deep neural networks. In *Annual Network and Distributed System Security Symposium*, 2018. Cited on page 17.

[XKSG16] Jun Xie, Martin Kiefel, Ming-Ting Sun, and Andreas Geiger. Semantic instance annotation of street scenes by 3d to 2d label transfer. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. Cited on page 30.

[XLG+21] Tao Xiang, Hangcheng Liu, Shangwei Guo, Tianwei Zhang, and Xiaofeng Liao. Local black-box adversarial attacks: A query efficient approach. *arXiv.org*, abs/2101.01032, 2021. Cited on page 16.

[XLL+21] Han Xu, Xiaorui Liu, Yaxin Li, Anil K. Jain, and Jiliang Tang. To be robust or to be fair: Towards fairness in adversarial training. In *Proc. of the International Conference on Machine Learning (ICML)*, 2021. Cited on page 15.

[XML+20] Han Xu, Yao Ma, Haochen Liu, Debayan Deb, Hui Liu, Jiliang Tang, and Anil K. Jain. Adversarial attacks and defenses in images, graphs and text: A review. *International Journal of Automation and Computing*, 17(2):151–178, 2020. Cited on page 15.

[XQL19] Chong Xiang, Charles R. Qi, and Bo Li. Generating 3d adversarial point clouds. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. Cited on pages 14 and 16.

[XRV17] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms. *arXiv.org*, abs/1708.07747, 2017. Cited on pages 49, 51, 54, and 267.

[XSG20] Zheng Xu, Ali Shafahi, and Tom Goldstein. Exploring model robustness with adaptive networks and improved adversarial training. *arXiv.org*, abs/2006.00387, 2020. Cited on page 17.

[XTCZ21] Qiuling Xu, Guanhong Tao, Siyuan Cheng, and Xiangyu Zhang. Towards feature space adversarial attack by style perturbation. In *Proc. of the Conference on Artificial Intelligence (AAAI)*, 2021. Cited on page 17.

[XWC+19] Qiangeng Xu, Weiyue Wang, Duygu Ceylan, Radomír Mech, and Ulrich Neumann. DISN: deep implicit surface network for high-quality single-view 3d reconstruction. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. Cited on page 13.

[XWZ+18] Cihang Xie, Jianyu Wang, Zhishuai Zhang, Zhou Ren, and Alan L. Yuille. Mitigating adversarial effects through randomization. *Proc. of the International Conference on Learning Representations (ICLR)*, 2018. Cited on page 17.

[XYA20] Zhisheng Xiao, Qing Yan, and Yali Amit. Likelihood regret: An out-of-distribution detection score for variational auto-encoder. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. Cited on page 22.

[XYL+19] Chaowei Xiao, Dawei Yang, Bo Li, Jia Deng, and Mingyan Liu. Meshadv: Adversarial meshes for visual recognition. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. Cited on page 16.

[XYL21] Jiancong Xiao, Liusha Yang, and Zhi-Quan Luo. Disentangling adversarial robustness in directions of the data manifold. https://openreview.net/forum?id=4mkxyuPcFt, 2021. Cited on page 17.

[XYP19] Joyce Xu, Dian Ang Yap, and Vinay Uday Prabhu. Understanding adversarial robustness through loss landscape geometries. *Proc. of the International Conference on Machine Learning (ICML) Workshops*, 2019. Cited on page 18.

[XZL+18] Chaowei Xiao, Jun-Yan Zhu, Bo Li, Warren He, Mingyan Liu, and Dawn Song. Spatially transformed adversarial examples. *Proc. of the International Conference on Learning Representations (ICLR)*, 2018. Cited on pages 14, 17, and 58.

[XZL+20] Kaidi Xu, Gaoyuan Zhang, S. Liu, Quanfu Fan, Mengshu Sun, H. Chen, Pin-Yu Chen, Yanzhi Wang, and X. Lin. Adversarial t-shirt! evading person detectors in a physical world. In *Proc. of the European Conference on Computer Vision (ECCV)*, 2020. Cited on page 17.

[XZZ+19] Cihang Xie, Zhishuai Zhang, Yuyin Zhou, Song Bai, Jianyu Wang, Zhou Ren, and Alan L. Yuille. Improving transferability of adversarial examples with input diversity. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. Cited on pages 16 and 229.

[YBG+21] Mikail Yayla, Sebastian Buschjager, Aniket Gupta, Jian-Jia Chen, Jörg Henkel, Katharina Morik, Kuan-Hsun Chen, and Hussam Amrouch. Fefet-based binarized neural networks under temperature-dependent bit errors. *IEEE Transactions on Computers*, 2021. Cited on pages 20 and 155.

[YBTV21] Chengyuan Yao, Pavol Bielik, Petar Tsankov, and Martin T. Vechev. Automated discovery of adaptive attacks on adversarial defenses. *arXiv.org*, abs/2102.11860, 2021. Cited on pages 7, 8, 17, 19, 134, and 154.

[YCES17] Tien-Ju Yang, Yu-Hsin Chen, Joel S. Emer, and Vivienne Sze. A method to estimate the energy consumption of deep neural networks. In *Asilomar Conference on Signals, Systems, and Computers (ACSSC)*, 2017. Cited on pages 89 and 155.

[YDH+20] Greg Yang, Tony Duan, J. Edward Hu, Hadi Salman, Ilya P. Razenshteyn, and Jerry Li. Randomized smoothing of all shapes and sizes. In *Proc. of the International Conference on Machine Learning (ICML)*, 2020. Cited on page 18.

[YFST18] Yaoqing Yang, Chen Feng, Yiru Shen, and Dong Tian. Foldingnet: Point cloud auto-encoder via deep grid deformation. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. Cited on page 13.

[YGL+18] Zhewei Yao, Amir Gholami, Qi Lei, Kurt Keutzer, and Michael W. Mahoney. Hessian-based analysis of large batch training and robustness to adversaries. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018. Cited on pages 67 and 71.

[YHL21] Jongmin Yoon, Sung Ju Hwang, and Juho Lee. Adversarial purification with score-based generative models. In *Proc. of the International Conference on Machine Learning (ICML)*, 2021. Cited on page 18.

[YHZL19] Xiaoyong Yuan, Pan He, Qile Zhu, and Xiaolin Li. Adversarial examples: Attacks and defenses for deep learning. *IEEE Trans. Neural Netw. Learning Syst.*, 30(9), 2019. Cited on page 15.

[YK21] Yachong Yang and A. Kuchibhotla. Finite-sample efficient conformal prediction. *arXiv.org*, abs/2104.13871, 2021. Cited on pages 23 and 146.

[YKM+20] Lior Yariv, Yoni Kasten, Dror Moran, Meirav Galun, Matan Atzmon, Ronen Basri, and Yaron Lipman. Multiview neural surface reconstruction by disentangling geometry and appearance. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. Cited on page 13.

[YLD11] Dong Yu, Jinyu Li, and Li Deng. Calibration of confidence measures in speech recognition. *IEEE Transactions on Audio, Speech, and Language Processing*, 19(8):2461–2473, 2011. Cited on page 22.

[YLF+18] Lequan Yu, Xianzhi Li, Chi-Wing Fu, Daniel Cohen-Or, and Pheng-Ann Heng. Pu-net: Point cloud upsampling network. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. Cited on page 13.

[YLL+19] Hang Yu, Aishan Liu, Xianglong Liu, Jichen Yang, and Chongzhi Zhang. Towards noise-robust neural networks via progressive adversarial training. *arXiv.org*, abs/1909.04839, 2019. Cited on page 18.

[YLS+19] Dong Yin, Raphael Gontijo Lopes, Jon Shlens, Ekin Dogus Cubuk, and Justin Gilmer. A fourier perspective on model robustness in computer vision. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. Cited on page 17.

[YLW+18] F. Yu, C. Liu, Yanzhi Wang, Liang Zhao, and X. Chen. Interpreting adversarial robustness: A view from decision surface in input space. *arXiv.org*, abs/1810.00144, 2018. Cited on pages 17, 18, and 69.

[YMH21] Maksym Yatsura, Jan Hendrik Metzen, and Matthias Hein. Meta-learning the search distribution of black-box random search based adversarial attacks. *arXiv.org*, abs/2111.01714, 2021. Cited on page 16.

[YRF20] Fan Yao, Adnan Siraj Rakin, and Deliang Fan. Deephammer: Depleting the intelligence of deep neural networks through targeted chain of bit flips. In *USENIX Security Symposium*, 2020. Cited on page 21.

[YRM+19] Bo Yang, Stefano Rosa, Andrew Markham, Niki Trigoni, and Hongkai Wen. Dense 3d object reconstruction from a single depth view. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 41(12):2820–2834, 2019. Cited on pages 3, 12, 27, 28, 29, 35, 36, 39, 45, and 46.

[YRZ+20a] Yao-Yuan Yang, Cyrus Rashtchian, Hongyang Zhang, Ruslan Salakhutdinov, and Kamalika Chaudhuri. Adversarial robustness through local lipschitzness. *arXiv.org*, abs/2003.02460, 2020. Cited on page 17.

[YRZ+20b] Yao-Yuan Yang, Cyrus Rashtchian, Hongyang Zhang, Russ R. Salakhutdinov, and Kamalika Chaudhuri. A closer look at accuracy vs. robustness. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. Cited on page 19.

[YSAF21] Ning Yu, Vladislav Skripniuk, Sahar Abdelnabi, and Mario Fritz. Artificial fingerprinting for generative models: Rooting deepfake attribution in training data. In *Proc. of the IEEE International Conference on Computer Vision (ICCV)*, 2021. Cited on page 15.

[YWW+17] Bo Yang, Hongkai Wen, Sen Wang, Ronald Clark, Andrew Markham, and Niki Trigoni. 3d object reconstruction from a single depth view with adversarial learning. In *Proc. of the IEEE International Conference on Computer Vision (ICCV) Workshops*, 2017. Cited on pages 12, 18, and 39.

[YZ18] Nanyang Ye and Zhanxing Zhu. Bayesian adversarial learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018. Cited on page 18.

[YZLL21] Jingkang Yang, Kaiyang Zhou, Yixuan Li, and Ziwei Liu. Generalized out-of-distribution detection: A survey. *arXiv.org*, abs/2110.11334, 2021. Cited on page 14.

[ZAG19] Daniel Zügner, Amir Akbarnejad, and Stephan Günnemann. Adversarial attacks on neural networks for graph data. In *Proc. of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2019. Cited on page 16.

[Zak01] Radoslaw R. Zakrzewski. Verification of a trained neural network accuracy. In *International Joint Conference on Neural Networks (IJCNN)*, 2001. Cited on page 18.

[ZCR19] Tianhang Zheng, Changyou Chen, and Kui Ren. Distributionally adversarial attack. In *Proc. of the Conference on Artificial Intelligence (AAAI)*, 2019. Cited on page 16.

[ZCS+19] Huan Zhang, Hongge Chen, Zhao Song, Duane Boning, inderjit dhillon, and Cho-Jui Hsieh. The limitations of adversarial training and the blind-spot attack. In *Proc. of the International Conference on Learning Representations (ICLR)*, 2019. Cited on page 18.

[ZCWL20] Pu Zhao, Pin-Yu Chen, Siyue Wang, and Xue Lin. Towards query-efficient black-box adversary with zeroth-order natural gradient descent. In *Proc. of the Conference on Artificial Intelligence (AAAI)*, 2020. Cited on page 16.

[ZCX+20] Huan Zhang, Hongge Chen, Chaowei Xiao, Sven Gowal, Robert Stanforth, Bo Li, Duane S. Boning, and Cho-Jui Hsieh. Towards stable and efficient training of verifiably robust neural networks. In *Proc. of the International Conference on Learning Representations (ICLR)*, 2020. Cited on pages 19 and 153.

[ZDS18] Zhengli Zhao, Dheeru Dua, and Sameer Singh. Generating natural adversarial examples. *Proc. of the International Conference on Learning Representations (ICLR)*, 2018. Cited on pages 50 and 57.

[ZFV20] Gianluca Zeni, Matteo Fontana, and Simone Vantini. Conformal prediction: a unified review of theory and new challenges. *arXiv.org*, abs/2005.07972, 2020. Cited on page 23.

[ZGJ+18] Jialong Zhang, Zhongshu Gu, Jiyong Jang, Hui Wu, Marc Ph. Stoecklin, Heqing Huang, and Ian Molloy. Protecting intellectual property of deep neural networks with watermarking. In *Proc. of the ACM on Asia Conference on Computer and Communications Security (AsiaCCS)*, 2018. Cited on page 15.

[ZH18] Zhihao Zheng and Pengyu Hong. Robust detection of adversarial attacks by modeling the intrinsic properties of deep neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018. Cited on page 19.

[ZIE+18] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. Cited on page 16.

[ZK16] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *Proc. of the British Machine Vision Conference (BMVC)*, 2016. Cited on pages 78, 97, 128, 134, and 154.

[ZKMW20] Chuteng Zhou, Prad Kadambi, Matthew Mattina, and Paul N. Whatmough. Noisy machines: Understanding noisy neural networks and enhancing robustness to analog hardware errors using distillation. *arXiv.org*, abs/2001.04974, 2020. Cited on pages 20 and 155.

[ZKS+18] Tom Zahavy, Bingyi Kang, Alex Sivak, Jiashi Feng, Huan Xu, and Shie Mannor. Ensemble robustness and generalization of stochastic deep learning algorithms. In *Proc. of the International Conference on Learning Representations (ICLR) Workshops*, 2018. Cited on page 17.

[ZKTX12] Yan Zhou, Murat Kantarcioglu, Bhavani M. Thuraisingham, and Bowei Xi. Adversarial support vector machine learning. In *Proc. of the ACM International Conference on Knowledge Discovery & Data Mining*, 2012. Cited on page 14.

[ZKX18] Yan Zhou, Murat Kantarcioglu, and Bowei Xi. Breaking transferability of adversarial samples with randomness. *arXiv.org*, abs/1805.04613, 2018. Cited on page 17.

[ZL19] Yuchen Zhang and Percy Liang. Defending against whitebox adversarial attacks via randomized discretization. In *Conference on Artificial Intelligence and Statistics (AISTATS)*, 2019. Cited on page 17.

[ZLL20a] Zhengyu Zhao, Zhuoran Liu, and Martha A. Larson. Adversarial color enhancement: Generating unrestricted adversarial images by optimizing a color filter. In *Proc. of the British Machine Vision Conference (BMVC)*, 2020. Cited on page 17.

[ZLL20b] Zhengyu Zhao, Zhuoran Liu, and Martha A. Larson. Towards large yet imperceptible adversarial image perturbations with perceptual color distance. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. Cited on page 16.

[ZMCF18] Yiren Zhou, Seyed-Mohsen Moosavi-Dezfooli, Ngai-Man Cheung, and Pascal Frossard. Adaptive quantization for deep neural network. In Sheila A. McIlraith and Kilian Q. Weinberger, editors, *Proc. of the Conference on Artificial Intelligence (AAAI)*, 2018. Cited on page 101.

[ZMS+21] Yao Zhu, Jiacheng Ma, Jiacheng Sun, Zewei Chen, Rongxin Jiang, and Zhenguo Li. Towards understanding the generative capability of adversarially robust classifiers. In *Proc. of the IEEE International Conference on Computer Vision (ICCV)*, 2021. Cited on page 18.

[ZNR17] Valentina Zantedeschi, Maria-Irina Nicolae, and Ambrish Rawat. Efficient defenses against adversarial attacks. In *Proc. of the ACM Workshop on Artificial Intelligence and Security*, 2017. Cited on page 17.

[ZNZ+16] Shuchang Zhou, Zekun Ni, Xinyu Zhou, He Wen, Yuxin Wu, and Yuheng Zou. Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. *arXiv.org*, abs/1606.06160, 2016. Cited on pages 21 and 92.

[ZPA+15] Shuai Zheng, Victor Adrian Prisacariu, Melinos Averkiou, Ming-Ming Cheng, Niloy J. Mitra, Jamie Shotton, Philip H. S. Torr, and Carsten Rother. Object proposal estimation in depth images using compact 3d shape manifolds. In *Proc. of the German Conference on Pattern Recognition (GCPR)*, 2015. Cited on page 12.

[ZSA19] Wei Emma Zhang, Quan Z. Sheng, and Ahoud Abdulrahmn F. Alhazmi. Generating textual adversarial examples for deep learning models: A survey. *arXiv.org*, abs/1901.06796, 2019. Cited on pages 14 and 16.

[ZSS14] Muhammad Zeeshan Zia, Michael Stark, and Konrad Schindler. Are cars just 3d boxes? jointly estimating the 3d shape of multiple objects. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3678–3685, 2014. Cited on page 12.

[ZSSS13] M.Z. Zia, M. Stark, B. Schiele, and K. Schindler. Detailed 3D representations for object recognition and modeling. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 35(11):2608–2623, November 2013. Cited on page 12.

[ZST+18] Bohan Zhuang, Chunhua Shen, Mingkui Tan, Lingqiao Liu, and Ian D. Reid. Towards effective low-bitwidth convolutional neural networks. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. Cited on pages 21, 85, 101, and 111.

[ZSW+10] Qian Zheng, Andrei Sharf, Guowei Wan, Yangyan Li, Niloy J. Mitra, Daniel Cohen-Or, and Baoquan Chen. Non-local scan consolidation for 3d urban scenes. *ACM Trans. on Graphics*, 29(4):94:1–94:9, 2010. Cited on page 12.

[ZW19] Haichao Zhang and Jianyu Wang. Defense against adversarial attacks using feature scattering-based adversarial training. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. Cited on page 18.

[ZWC+18] Huan Zhang, Tsui-Wei Weng, Pin-Yu Chen, Cho-Jui Hsieh, and Luca Daniel. Efficient neural network robustness certification with general activation functions. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 4944–4953, 2018. Cited on page 18.

[ZWG+19] Pu Zhao, Siyue Wang, Cheng Gongye, Yanzhi Wang, Yunsi Fei, and Xue Lin. Fault sneaking attack: a stealthy framework for misleading deep neural networks. In *ACM/ESDA/IEEE Design Automation Conference (DAC)*, 2019. Cited on page 21.

[ZXH+20] Jingfeng Zhang, Xilie Xu, Bo Han, Gang Niu, Lizhen Cui, Masashi Sugiyama, and Mohan S. Kankanhalli. Attacks which do not kill training make adversarial learning stronger. In *Proc. of the International Conference on Machine Learning (ICML)*, 2020. Cited on pages 16, 18, 19, 73, and 238.

[ZYJ+19] Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric P. Xing, Laurent El Ghaoui, and Michael I. Jordan. Theoretically principled trade-off between robustness and accuracy. In *Proc. of the International Conference on Machine Learning (ICML)*, 2019. Cited on pages 16, 18, 65, 66, 67, 70, 73, 79, 116, 117, 120, 126, 128, 133, and 237.

[ZYYH18] Dongqing Zhang, Jiaolong Yang, Dongqiangzi Ye, and Gang Hua. Lq-nets: Learned quantization for highly accurate and compact deep neural networks. In *Proc. of the European Conference on Computer Vision (ECCV)*, 2018. Cited on page 101.

[ZZL+19] Dinghuai Zhang, Tianyuan Zhang, Yiping Lu, Zhanxing Zhu, and Bin Dong. You only propagate once: Accelerating adversarial training via maximal principle. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. Cited on page 18.

[ZZM21] Yaowei Zheng, Richong Zhang, and Yongyi Mao. Regularizing neural networks via adversarial model perturbation. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. Cited on page 67.

[ZZMJ21] Bojia Zi, Shihao Zhao, Xingjun Ma, and Yu-Gang Jiang. Revisiting adversarial robustness distillation: Robust soft labels make student better. In *Proc. of the IEEE International Conference on Computer Vision (ICCV)*, 2021. Cited on page 18.

[ZZN+21] Jingfeng Zhang, Jianing Zhu, Gang Niu, Bo Han, Masashi Sugiyama, and Mohan Kankanhalli. Geometry-aware instance-reweighted adversarial training. In *Proc. of the International Conference on Learning Representations (ICLR)*, 2021. Cited on page 18.

[ZZRP19] Michal Zajac, Konrad Zolna, Negar Rostamzadeh, and Pedro O. Pinheiro. Adversarial framing for image and video classification. In *Proc. of the Conference on Artificial Intelligence (AAAI) Workshops*, 2019. Cited on pages 16, 117, and 127.

A Disentangling Adversarial Robustness and Generalization

A.1 On-Manifold Adversarial Examples

In Figure A.1, we show additional examples of regular and on-manifold adversarial examples, complementing the examples in Figure 4.3. On FONTS, both using the true and the approximated manifold, on-manifold adversarial examples reflect the underlying invariances of the data, i.e., the transformations employed in the generation process. This is in contrast to the corresponding regular adversarial examples and their (seemingly) random noise patterns. We note that regular and on-manifold adversarial examples can best be distinguished based on their difference to the original test image – although both are perceptually close to the original image. Similar observations hold on MNIST and Fashion. However, especially on Fashion and CelebA, the discrepancy between true images and on-manifold adversarial examples becomes visible. This is the "cost" of approximating the underlying manifold using VAE-GANs.

A.2 L_2 and Transfer Attacks

In Section 4.2, we primarily focus on the L_∞ white-box attack by Madry et al. [MMS+18]. Here, we further consider the L_2 variant, which, given image x with label y and classifier f, maximizes the cross-entropy loss, i.e.,

$$\max_\delta \mathcal{L}(f(x+\delta), y) \text{ s.t. } \|\delta\|_2 \leq \epsilon, \tilde{x}_i \in [0,1], \tag{A.1}$$

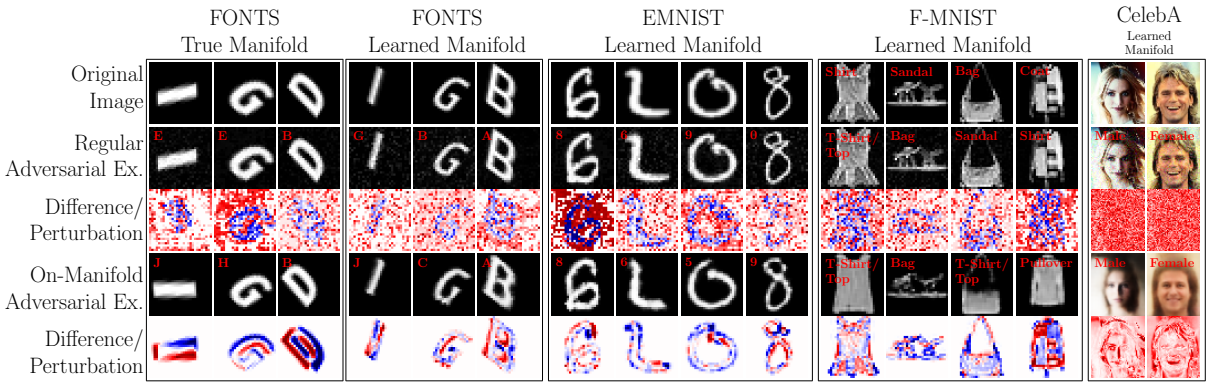


Figure A.1: **Additional Qualitative Examples:** Regular and on-manifold adversarial examples on FONTS, MNIST, Fashion and CelebA. On FONTS, the manifold is known; on the other datasets, class manifolds have been approximated using VAE-GANs. Notice that the crafted on-manifold adversarial examples correspond to meaningful manipulations of the image – as long as the learned class-manifolds are good approximations. This can best be seen considering the (normalized) difference images (or the magnitude thereof for CelebA).

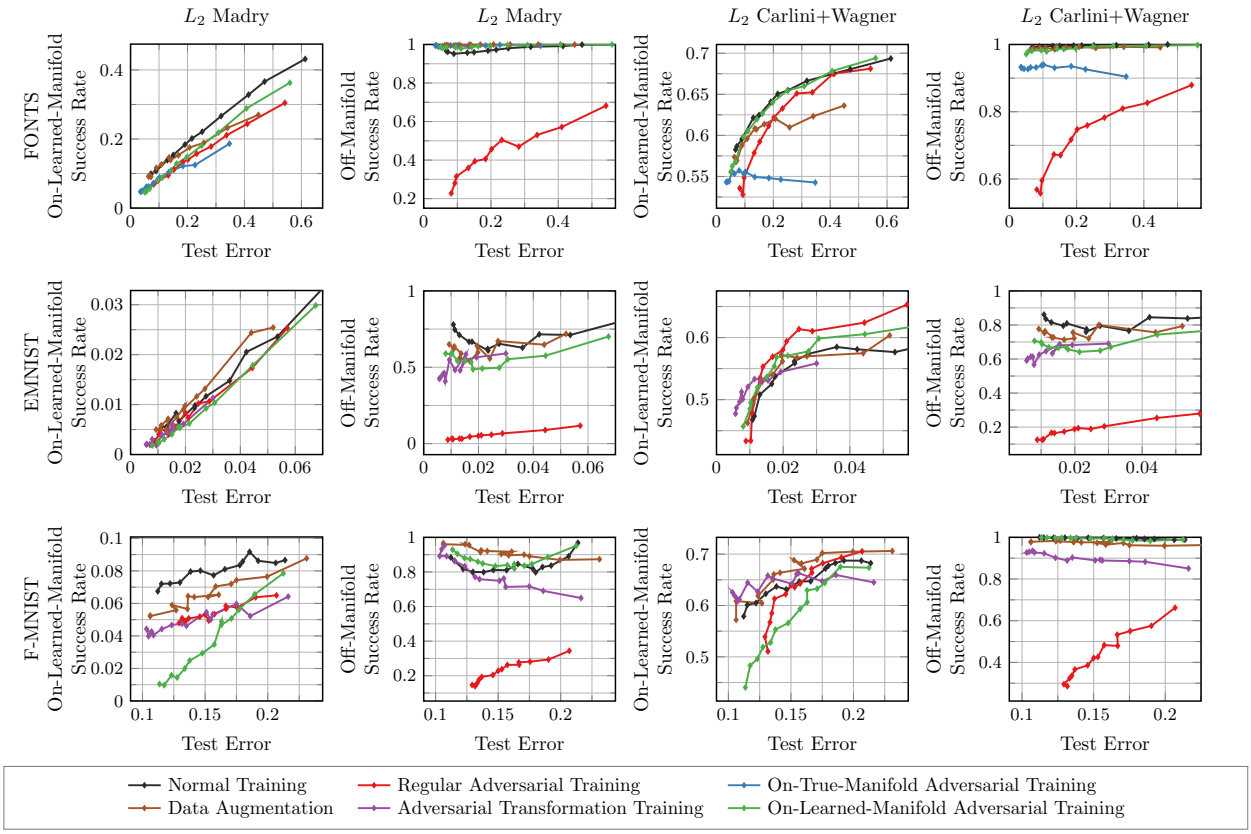


Figure A.2: **Results for L_2 Adversarial Examples:** L_2 attacks of Madry et al. [MMS+18] and Carlini and Wagner [CW17b] on FONTS, MNIST and Fashion. In all cases, we plot regular or on-manifold success rate against test error. Independent of the attack, we can confirm that on-manifold robustness is strongly related to generalization, while regular robustness is independent of generalization.

to obtain an adversarial example $\tilde{x} = x + \delta$. We use $\epsilon = 1.5$ for regular adversarial examples and $\epsilon = 0.3$ for on-manifold adversarial examples. For optimization, we utilize projected ADAM [KB15b]: after each iteration, $\tilde{x}$ is projected onto the L_2-ball of radius ϵ using

$$\tilde{x}' = \tilde{x} \cdot \max\left(1, \frac{\epsilon}{\|\tilde{x}\|_2}\right) \tag{A.2}$$

and clipped to $[0,1]$. We use a learning rate of 0.005 and we note that ADAM includes momentum, as suggested in [DLP+18]. Optimization stops as soon as the label changes, or runs for a maximum of 40 iterations. The perturbation δ is initialized randomly as follows:

$$\delta = u\epsilon\frac{\delta'}{\|\delta'\|_2}, \quad \delta' \sim \mathcal{N}(0, I), u \sim U(0,1). \tag{A.3}$$

Here, $U(0,1)$ refers to the uniform distribution over $[0,1]$. This results in δ being in the ϵ-ball and uniformly distributed over distance and direction. Note that this is in contrast to sampling uniformly w.r.t. the volume of the ϵ-ball. The same procedure applies to the L_∞ attack where the projection onto the ϵ-ball is achieved by clipping. The attack can also be used to obtain on-manifold adversarial examples, as described in Section 4.2.3. Then, optimization is done over the perturbation ζ in latent space, with constraint $\|\zeta\|_2 \leq \eta$. The adversarial example is

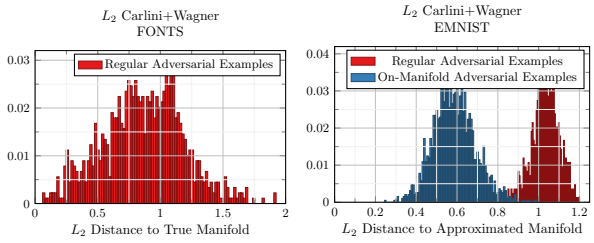


Figure A.3: **Off-Manifold Carlini+Wagner Adversarial Examples:** Distance of Carlini+Wagner adversarial examples to the true, on FONTS (left), or approximated, on MNIST (right), manifold. As before, we show normalized histograms of the L_2 distance of adversarial examples to their projections onto the manifold.

obtained as $\tilde{x} = \text{dec}(z + \zeta)$ with z being the latent code of image x and dec being the true or approximated generative model, i.e., decoder.

We also consider the L_2 white box attack by Carlini and Wagner [CW17b]. Instead of directly maximizing the training loss, Carlini and Wagner propose to use a surrogate objective on the classifier's logits l_y:

$$F(\tilde{x}, y) = \max(-\kappa, l_y(\tilde{x}) - \max_{y' \neq y} l_{y'}(\tilde{x})). \tag{A.4}$$

Compared to the training loss, which might be close to zero for a well-trained network, F is argued to provide more useful gradients [CW17b]. Then,

$$\min_\delta F(x + \delta, y) + \lambda \|\delta\|_2 \text{ s.t. } \tilde{x}_i \in [0, 1] \tag{A.5}$$

is minimized by reparameterizing δ in terms of $\delta = 1/2(\tanh(\omega) + 1) - x$ in order to ensure the image-constraint, i.e., $\tilde{x}_i \in [0, 1]$. In practice, we empirically chose $\kappa = 1.5$, use 120 iterations of ADAM [KB15b] with learning rate 0.005 and $\lambda = 1$. Again, this attack can be used to obtain on-manifold adversarial examples, as well.

As black-box attack we transfer L_∞ Madry adversarial examples from a held out model, as previously done in [PMG+17, LCLS17, XZZ+19]. The held out transfer model is trained normally, i.e., without any data augmentation or adversarial training, on 10k training images for 20 epochs. The success rate of these transfer attacks is computed with respect to images that are correctly classified by both the transfer model and the target model.

Figure A.2 shows results on FONTS, MNIST and Fashion considering both L_2 attacks, i.e., Madry et al. [MMS+18] and Carlini and Wagner [CW17b]. In contrast to the L_∞ Madry attack, we observe generally lower success rates. Nevertheless, we can observe a clear relationship between on-manifold success rate and test error. The exact form of this relationship, however, depends on the attack. For the L_2 Madry attack, the relationships seems to be mostly linear (especially on FONTS and MNIST), while it seems non-linear for the L_2 Carlini and Wagner attack. Furthermore, the independence of regular robustness and generalization can be confirmed, i.e., regular success rate is roughly constant when test error varies – again, with the exception of regular adversarial training. Finally, for completeness, in Figure A.3, we illustrate that the Carlini+Wagner L_2 adversarial examples also leave the manifold.

In Figure A.4, we also consider the black-box case, i.e., without access to the target model. While both observations from above can be confirmed, especially on FONTS and MNIST, the results are significantly less pronounced. This is mainly due to the significantly lower success rate of transfer attacks – both regarding regular and on-manifold adversarial examples. Especially on MNIST and Fashion, success rate may reduce from previously 80% or higher to 10% or lower. This might also explain the high variance on MNIST and Fashion regarding regular robustness. Overall, we demonstrate that our claims can be confirmed in both white- and black-box settings as well as using different attacks [CW17b, MMS+18] and norms.

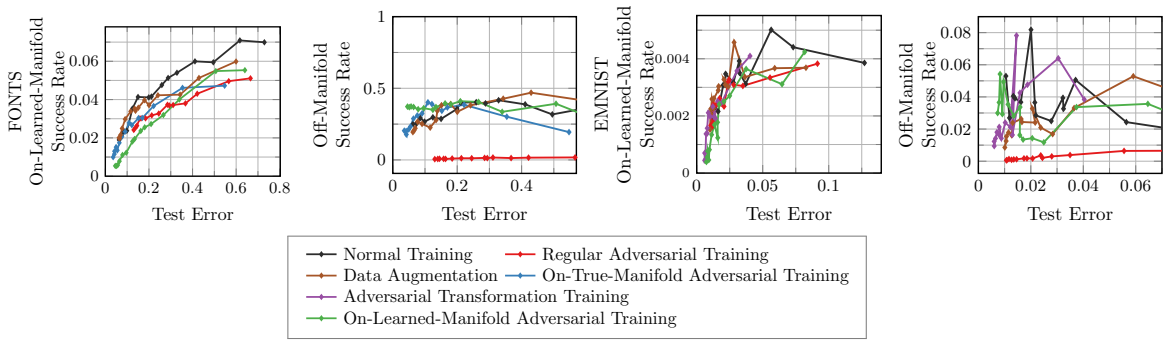


Figure A.4: **Results with Transfer Attacks** on FONTS, MNIST and Fashion. We show on-manifold (left) and regular success rate (right) plotted against test error. In spite of significantly lower success rates, transfer attacks also confirm the strong relationship between on-manifold success rate and test error, while – at least on FONTS and MNIST– regular success rate is independent of test error.

A.3 Influence of Network Architecture

Also in relation to the discussion in Section 4.2.4 and 4.2.5, Figure A.5 shows results on FONTS, MNIST and Fashion using multi-layer perceptrons instead of convolutional neural networks. Specifically, we consider a network with 4 hidden layers, using 128 hidden units each. Each layer is followed by ReLU activations and batch normalization [IS15b]; training strategy, however, remains unchanged. Both of our claims, i.e., that on-manifold robustness is essentially generalization, but regular robustness is independent of generalization, can be confirmed. Especially regarding the latter, results are more pronounced using multi-layer perceptrons: except for regular adversarial training, success rate stays nearly constant at 100% irrespective of test error. Overall, these results suggest that our claims generally hold for the class of (deep) neural networks, irrespective of architectural details.

In order to further validate our claims, we also consider variants of two widely used, state-of-the-art architectures: ResNet-13 [HZRS16a] and VGG [SZ15]. For VGG, however, we removed the included dropout layers. The main reason is that randomization might influence robustness, e.g., see [ACW18]. Additionally, we only use 2 stages of model A, see [SZ15], in

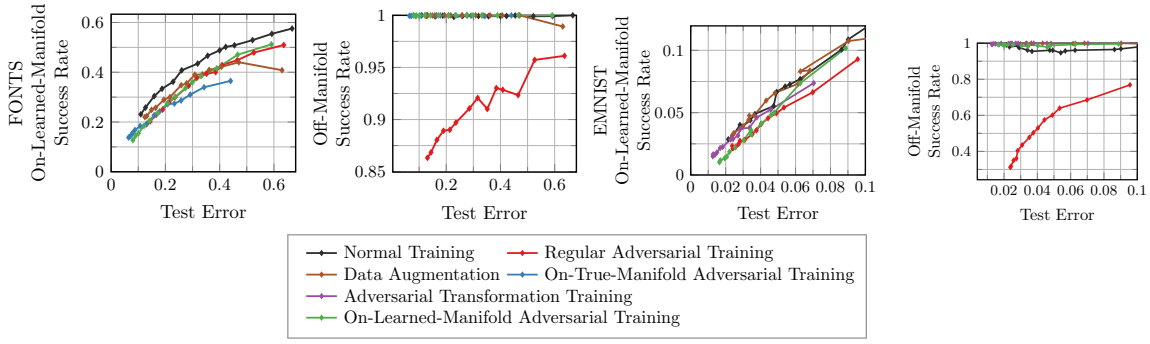


Figure A.5: **Results for Multi-Layer Perceptrons:** Experiments with multilayer-perceptrons on FONTS, MNIST and Fashion. We plot on-manifold (left) or regular success rate (right) against test error. On-manifold robustness is strongly related to generalization, while regular robustness seems mostly independent of generalization.

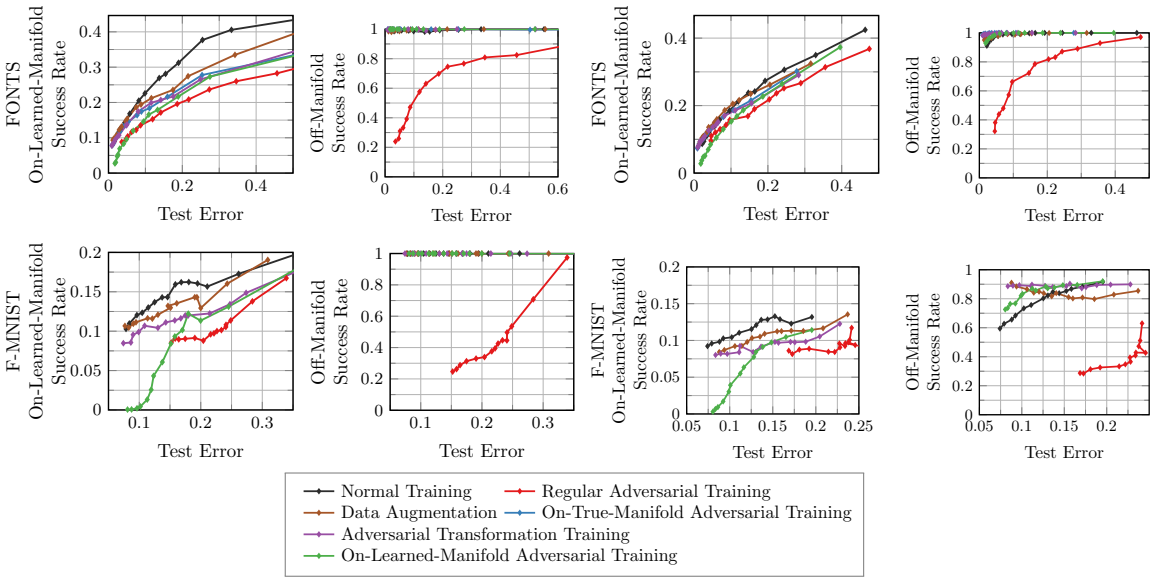


Figure A.6: **Results with ResNet-13 and VGG:** Experiments with ResNet-13 (left) and VGG (right) on FONTS and Fashion. We plot on-manifold (left) or regular success rate (right) against test error. As in Figure A.5, our claims can be confirmed for these network architectures, as well.

order to deal with the significantly lower resolution of 28×28 on FONTS, MNIST and Fashion. Finally, we only use 1024 hidden units in the fully connected layers. Figure A.6 shows results on FONTS and Fashion (which are significantly more difficult than MNIST) confirming our claims.

A.4 Baselines and Adversarial Training Variants

In Section 4.2, we consider the adversarial training variant by Madry et al. [MMS+18], i.e.,

$$\min_w \sum_{n=1}^{N} \max_{\|\delta\|_\infty \leq \epsilon} \mathcal{L}(f(x_n + \delta; w), y_n), \tag{A.6}$$

where f is the classifier with weights w, $\mathcal{L}$ is the cross-entropy loss and x_n, y_n are training images and labels. In contrast to [MMS+18], we train on 50% clean and 50% adversarial examples [SZS+14, GSS15]. The inner optimization problem is run for full 40 iterations without early stopping. Here, we additionally consider the *full variant*, i.e., training on 100% adversarial examples, and the *weak variant*, i.e., stopping the inner optimization problem as soon as the label changes. Additionally, we consider random perturbations as baseline, i.e., choosing the perturbations δ uniformly at random without any optimization. The same variants and baselines apply to on-manifold adversarial training and adversarial transformation training.

In Section 4.2.6, we observed that different training strategies might exhibit different robustness-generalization characteristics. For example, regular adversarial training renders the learning problem harder: in addition to the actual task, the network has to learn (seemingly) random but adversarial noise directions leaving the manifold. In Figure A.7, we first show that training on randomly perturbed examples (instead of adversarially perturbed ones) is not effective, neither in image space nor in latent space. This result highlights the difference between random and adversarial noise, as also discussed in [FMDF16]. For regular adversarial

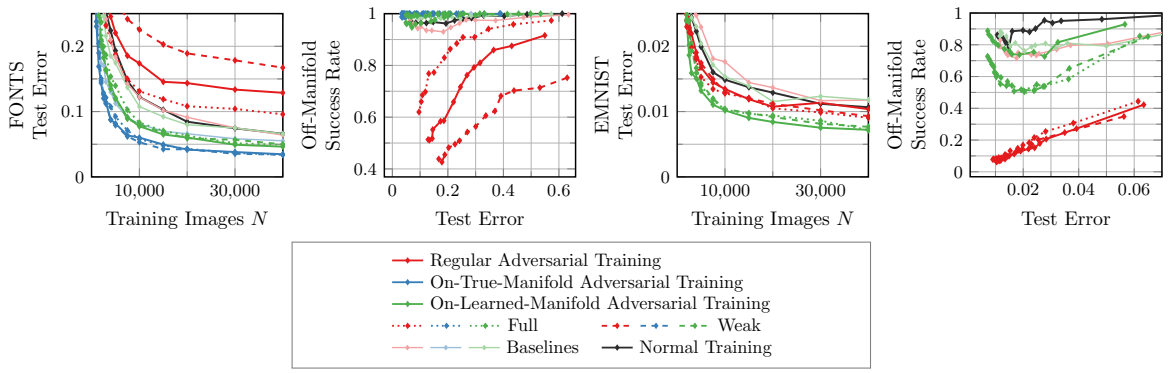


Figure A.7: **Various Adversarial Training Variants** and baselines on FONTS and MNIST. For adversarial training, we consider the *full variant*, i.e., training on 100% adversarial examples, and the *weak variant*, i.e., stopping the inner optimization problem of off-manifold adversarial training as soon as the first adversarial example is found. For regular adversarial training, the strength of the adversary determines the robustness-generalization trade-off. For on-manifold adversarial training, the ideal strength depends on the approximation quality of the used VAE-GANs.

training, the strength of the adversary primarily influences the robustness-generalization trade-off. For example, the weak variant increases generalization while reducing robustness. Note that this effect also depends on the difficulty of the task, e.g., FONTS is considerably more difficult than MNIST. For on-manifold adversarial training, in contrast, the different variants have very little effect; generalization is influenced only slightly, while regular robustness is – as expected – not influenced.

B Relating Adversarially Robust Generalization to Flat Minima

B.1 Visualization Details and Discussion

Discussion of [LXTG18]: Originally, [LXTG18] uses a per-filter normalization instead of our per-layer normalization. Specifically, this means

$$\hat{v}^{(l,i)} = \frac{v^{(l)}}{\|v^{(l,i)}\|_2} \|w^{(l,i)}\|_2 \quad \text{for layer } l, \text{ filter } i, \tag{B.1}$$

instead of the normalization used in Section 5.2.2:

$$\hat{v}^{(i)} = \frac{v^{(l)}}{\|v^{(l)}\|_2} \|w^{(l)}\|_2 \quad \text{for layer } l. \tag{B.2}$$

Furthermore, [LXTG18] does not consider changes in the biases or batch normalization parameters. Instead, we also normalize the biases as above and take them into account for visualization (but not the batch normalization parameters). More importantly, [LXTG18] considers only (clean) Loss, while we focus on RLoss. Figure B.1 shows that the difference between filter-wise and layer-wise normalization has little impact in visually judging flatness. Generally, filter-wise normalization makes the RLoss landscape "look" flatter. However, this is mainly because the absolute step size, i.e., $\|\hat{v}\|_2$, is smaller compared to layer-wise normalization: for our AT baseline, this is (on average) $\|\hat{v}\|_2 \approx 33.13$ for layer-wise and $\|\hat{v}\|_2 \approx 21.49$ for filter-wise normalization.

B.2 Ablation for Flatness Measures

Standard Deviation in Average-Case Flatness: In Figure B.2 (left), the x-axis plots the standard deviation in our average-case flatness measure (in RLoss). Note that the standard

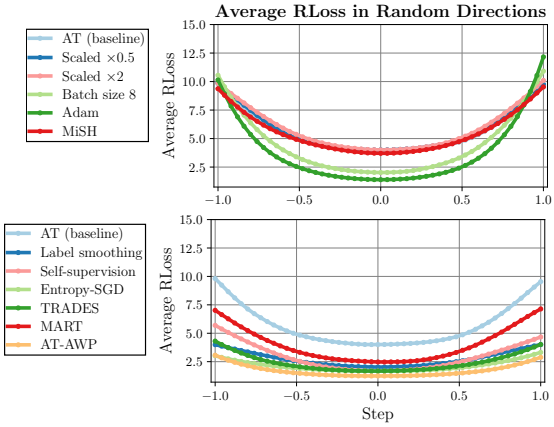


Figure B.1: **Filter-Wise Normalization:** Compared to the RLoss landscape visualizations in Figure 5.4, using per-layer normalization, we follow [LXTG18] and use filter-wise normalization in Equation (B.1). Again, we plot mean RLoss across 10 random directions. However, this does not change results significantly, flatness remains difficult to judge and compare in an objective way. Filter-wise normalization, however, "looks" generally flatter.

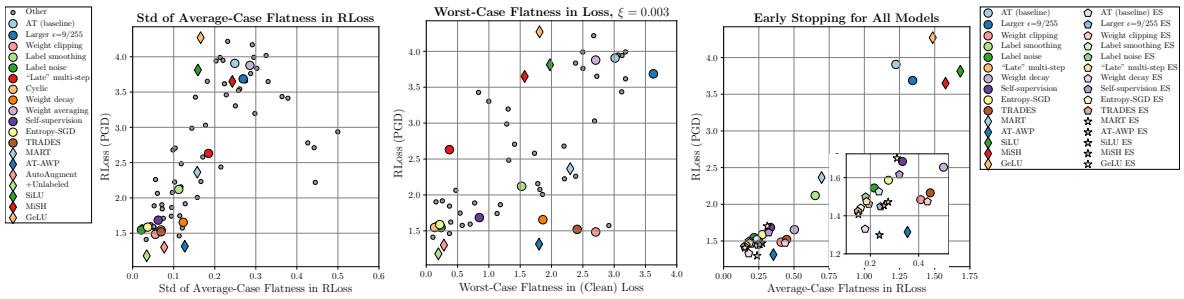


Figure B.2: **Left: Standard Deviation of Average-Case Flatness:** We plot RLoss (y-axis) against the standard deviation (std) in our average-case flatness measure (x-axis). Note that the standard-deviation is due to the random weight perturbations ν in the average-case flatness computation. Interestingly, more robust methods are not only flatter, but our average-case flatness measure also has lower standard deviation. **Middle: Worst-Case Flatness in (Clean) Loss:** As worst-case flatness in the *clean* Loss landscape also mirrors robust overfitting, we plot RLoss against worst-case flatness in Loss. Even though flatness is measured considering clean Loss, many methods improving robustness (i.e., lower RLoss) exhibit surprisingly good flatness. **Right: Early Stopping for all Models:** RLoss vs. average-case flatness for all models where early stopping improves adversarial robustness. For example, this is not the case for AutoAugment or AT with unlabeled examples. Across all models, early stopping improves both robustness and flatness. For clarity we provide a zoomed-in plot for the lower left corner.

deviation originates in the random samples ν used to calculate average-case flatness. First of all, standard deviation tends to be small (i.e., ≤ 0.3) across almost all models. This means that our findings, i.e., the strong correlation between flatness and RLoss, is supported by low standard deviation. More importantly, the standard deviation *reduces* for particularly robust methods.

Worst-Case Flatness on *Clean* Loss: During training worst-case flatness on clean Loss also seems to correlate with robust overfitting. Thus, in Figure B.2 (middle), we plot RLoss against worst-case flatness of Loss, showing that there is no clear relationship across models. Nevertheless, many methods improving adversarial robustness also result in flatter minima in the clean loss landscape. This is sensible as RLoss is generally an upper bound for (clean) Loss. On the other hand, flatness in Loss is *not* discriminative enough to clearly distinguish between robust and less robust models.

Ablation for $B_\xi(w)$: For computing our average- and worst-case flatness measures (in RLoss), we considered various sizes of neighborhoods in weight space, i.e. $B_\xi(w)$ for different ξ. Figure B.3 considers $\xi \in \{0.25, 0.5, 0.75, 1\}$ for average-case flatness (top) and $\xi \in \{0.00075, 0.001, 0.003, 0.005\}$ for worst-case flatness (bottom). In both cases, we plot RLoss (y-axis) against flatness in RLoss (y-axis). Average-case flatness using small $\xi = 0.25$ results in significantly smaller values, between 0 and 0.4, i.e., the increase in RLoss in random weight directions is rather small. Still, the relationship between adversarial robustness and flatness is clearly visible. The same holds for larger $\xi \in \{0.75, 1\}$. Worst-case flatness generally gives a less clear picture regarding the relationship between robustness and flatness. Additionally, for larger $\xi \in \{0.003, 0.005\}$, variance seems to increase such that this relationship becomes less pronounced. In contrast to average-case flatness, the variance is not induced by the 10 restarts, but caused by training itself. Indeed, re-training our AT baseline leads to a worst-case flatness in RLoss of 5.1, a significant reduction from 6.49 as obtained for our original baseline. Overall, however, the observations from Section 5.3.2 can be confirmed using different sizes of the neighborhood $B_\xi(w)$.

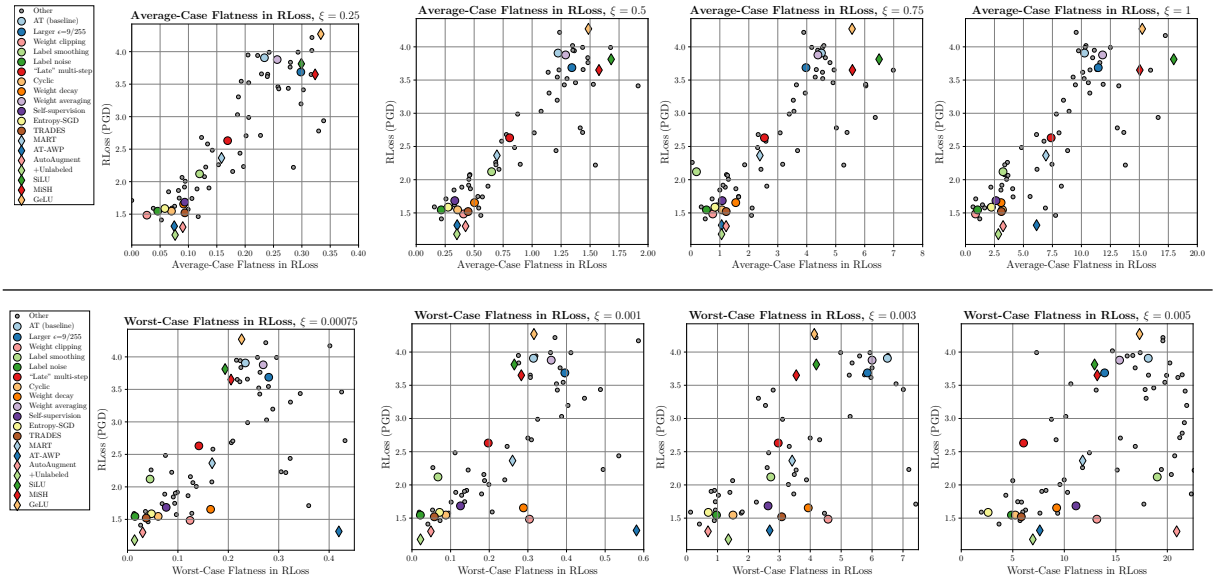
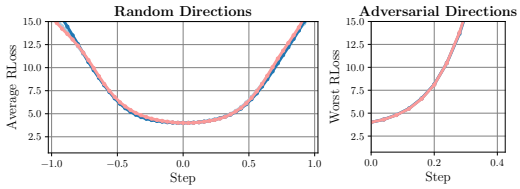


Figure B.3: **Flatness in RLoss, Ablation for $B_\xi(w)$:** RLoss (y-axis) plotted against average-case (top) and worst-case (bottom) flatness in RLoss (x-axis). **Top:** We consider $\xi \in \{0.25, 0.5, 0.75, 1\}$ for average-case flatness. The clear relationship between adversarial robustness, i.e., low RLoss, and flatness shown for $\xi = 0.5$ in Figure 5.9 can be reproduced for all cases. **Bottom:** For worst-case flatness, we consider $\xi \in \{0.00075, 0.001, 0.003, 0.005\}$. When chosen too large, e.g., $\xi = 0.005$, however, variance seems to increase, making the relationship less clear. For small ξ, e.g., $\xi = 0.00075$, the correlation between robustness and flatness is pronounced, except for a few outliers, including AT-AWP [WXW20b].



Model	Robustness		Flatness		
	RErr ↓ (test)	RErr ↓ (train)	Worst *Loss*	Avg **RLoss**	Worst **RLoss**
Scaled ×0.5	60.9	8.4 (-52.5)	0.86	1.36	6.50
AT (baseline)	61.0	8.4 (-52.6)	0.86	1.21	6.48
Scaled ×2	61.0	8.3 (-52.7)	0.86	1.27	6.49

Figure B.4: **Flatness and Scale-Invariance. Left:** We plot average RLoss and worst RLoss along random and adversarial directions, for AT and its scaled variants, ×0.5 and ×2. Clearly, RLoss landscape looks nearly identical. **Right:** Robustness against PGD-20 on train and test examples, as well as average- and worst-case flatness measures on RLoss. For completeness, we also include worst-case flatness on clean Loss. All of these measures are nearly invariant to scaling. The shown differences can be attributed to randomness in computing these measures.

B.3 Scaling Networks and Scale-Invariance

Figure B.4 shows experiments supporting the claim that our flatness measures are scale-invariant. As before, we scaled weights *and* biases of *all* convolutional layers in our adversarially trained ResNet-18 [HZRS16a] by factor $s \in \{0.5, 2\}$. Note that all convolutional layers in the ResNet are followed by batch normalization layers [IS15b]. Thus, the effect of scaling is essentially "canceled out", i.e., these convolutional layers are scale-invariant. Thus, the prediction stays roughly constant. Figure B.4 (left) shows RLoss landscape visualizations for AT and its scaled variants in random and adversarial weight directions. Clearly, scaling AT has negligible impact on the RLoss landscape in both cases. Figure B.4 (right) shows that our flatness measures remain invariant, as well.

B.4 Methods

In the following, we briefly elaborate on the individual methods considered in our experiments.

Learning Rate Schedules: Besides our default, multistep learning rate schedule (learning rate 0.05, reduced by factor 0.1 after epochs 60, 90, and 120), we followed [PYD+21] and implemented the following learning rate schedules: First, simply using a constant learning rate of 0.05. Second, only two "late" learning rate reductions at epochs 140 and 145, as done in [QMG+19]. Third, using a cyclic learning rate, interpolating between a learning rate of 0.2 and 0 for 30 epochs per cycle, as, e.g., done in [WRK20]. We consider training for up to 4 cycles (= 120 epochs). These learning rate schedules are available as part of PyTorch [PGC+17].

Label Smoothing: In [SVI+16], label smoothing is introduced as regularization to improve (clean) generalization by *not* enforcing one-hot labels in the cross-entropy loss. Instead, for label y and $K = 10$ classes, a target distribution $p \in [0,1]^K$ (subject to $\sum_i p_i = 1$) with $p_y = 1 - \tau$ (correct label) and $p_i = \tau/K-1$ for $i \neq y$ (all other labels) is enforced. During AT, we only apply label smoothing for the weight update, not for PGD. We consider $\tau \in \{0.1, 0.2, 0.3\}$.

Label Noise: Instead of explicitly enforcing a "smoothed" target distribution, we also consider injecting label noise during training. In each batch, we sample random labels for a fraction of τ of the examples. Note that the labels are sampled uniformly across all $K = 10$ classes. Thus, in expectation, the enforced target distribution is $p_y = 1 - \tau + \tau/K$ and $p_i = \tau - \tau/K / K-1$. As a result, this is equivalent to label smoothing with $\tau = \tau - \tau/K$. In contrast to label smoothing, this distribution is not enforced explicitly in the cross-entropy loss. As above, adversarial examples are computed against the true labels (without label noise) and label noise is injected for the weight update. We consider $\tau \in \{0.1, 0.2, 0.3, 0.4, 0.5\}$. While label smoothing does not further improve adversarial robustness for $\tau > 0.3$, label noise proved very effective in avoiding robust overfitting, which is why we also consider $\tau = 0.4$ or 0.5.

Weight Averaging: Follow [GQU+20, IPG+18], we keep a "running" average $\bar{w}$ of the model's weights throughout training, updated in each iteration t as follows:

$$\bar{w}^{(t)} = \tau \bar{w}^{(t-1)} + (1 - \tau) w^{(t)} \tag{B.3}$$

where $w^{(t)}$ are the weights in iteration t *after* the gradient update. Weight averaging is motivated by finding the weights $\bar{w}$ in the center of the found local minimum. As, depending on the learning rate, training tends to oscillate, the average of the iterates is assumed to be close to the actual center of the minimum. In our experiments, we consider $\tau \in \{0.98, 0.985, 0.99, 0.9975\}$.

Weight Clipping: Following Chapter 6, we implement weight clipping by clipping the weights to $[-w_{\max}, w_{\max}]$ after each training iteration. We found that $w_{\max}$ can be chosen as small as 0.005, which we found to work particularly well. Larger $w_{\max}$ does *not* have significant impact on adversarial robustness for AT. As weight clipping improves weight robustness, we also expect weight clipping to improve flatness. We consider $w_{\max} \in \{0.005, 0.01, 0.025\}$.

Ignoring Incorrect Examples & Preventing Label Leaking: As robust overfitting in AT leads to large RLoss on incorrectly classified test examples, we investigate whether (a) *not* computing adversarial examples on incorrectly classified examples (during training) or (b) computing adversarial examples against the predicted (not true) label (during training) helps to mitigate robust overfitting. These changes can be interpreted as ablations of MART [WZY+20] and are easily implemented. Note that option (b) is essentially computing adversarial examples without label leaking [KGB17].

AutoAugment: In [CZM+19], an automatic procedure for finding data augmentation policies is proposed, so-called AutoAugment. We use the found CIFAR10 policy (c.f. [CZM+19], appendix), which includes quite extreme augmentations. For example, large translations are possible, rendering the image nearly completely uniform, only leaving few pixels at the border. In practice, AutoAugment usually prevents convergence and thus avoids overfitting. We further combine AutoAugment with CutOut [DT17] (using random 16×16 "cutouts"). We apply both AutoAugment and CutOut on top of our standard data augmentation, i.e., random flipping and cropping. We use publicly available PyTorch implementations[1].

Entropy-SGD: [CCS+17] explicitly encourages flatter minima by taking the so-called "local" entropy into account. As a result, Entropy-SGD not only finds "deep" minima (i.e., low loss values) but also flat ones. In practice, this is done using nested SGD: the inner loop approximates the local entropy using stochastic gradient Langevin dynamics (SGLD), the outer loop updates the weights. The number of inner iterations is denoted by L. While the original work [CCS+17] uses L in $[5, 20]$ on CIFAR10, we experiment with $L \in \{1, 2, 3, 5\}$. Note that, for fair comparison, we train for $150/L$ epochs. For details on the Entropy-SGD algorithm, we refer to [CCS+17]. Our implementation follows the official PyTorch implementation[2].

Activation functions: We consider three recently proposed activation functions: SiLU [EUD18], MiSH [Mis20] and GeLU [HG16]. These are defined as:

$$\text{(SiLU)} \quad x\sigma(x), \quad \text{(MiSH)} \quad x\tanh(\log(1+\exp(x))), \quad \text{(GeLU)} \quad x\sigma(1.702x). \tag{B.4}$$

with $\sigma(x) = 1/(1+\exp(-x))$. All of these activation functions can be seen as smooth versions of the ReLU activation. In [SSJF21], some of these activation functions are argued to avoid robust overfitting due to lower curvature compared to ReLU.

AT-AWP: AT with adversarial weight perturbations (AT-AWP) [WXW20b] computes adversarial weight perturbations *on top* of adversarial examples to further regularize training. This is similar to our worst-case flatness measure of RLoss, however, adversarial examples and adversarial weights are computed sequentially, not jointly, and only one iteration is used to compute adversarial weights. Specifically, after computing adversarial examples $\tilde{x} = x + \delta$, an adversarial weight perturbation v is computed by solving

$$\max_{v \in B_\xi(w)} \mathcal{L}(f(\tilde{x}; w+v), y) \tag{B.5}$$

using one iteration of gradient ascent with fixed step size of ξ. The gradient is normalized per layer as in Equation (B.2). We considered $\xi \in \{0.0005, 0.001, 0.005, 0.01, 0.015, 0.02\}$ and between 1 and 7 iterations with $\xi = 0.01$ and 1 iteration working best.

TRADES: [ZYJ+19] proposes an alternative formulation of AT that allows a better trade-off between adversarial robustness and (clean) accuracy. The loss to be minimized is

$$\mathcal{L}(f(x; w), y) + \lambda \max_{\|\delta\|_\infty \leq \epsilon} \mathrm{KL}(f(x; w), f(x+\delta; w)). \tag{B.6}$$

During training, adversarial examples are computed by maximizing the KL-divergence (instead of cross-entropy loss), i.e., using the second term in Equation (B.6). Commonly $\lambda = 6$ is chosen, however, we additionally tried $\lambda \in \{1, 3, 6, 9\}$. We follow the official implementation[3].

[1] https://github.com/DeepVoltaire/AutoAugment, https://github.com/uoguelph-mlrg/Cutout
[2] https://github.com/ucla-vision/entropy-sgd
[3] https://github.com/yaodongyu/TRADES

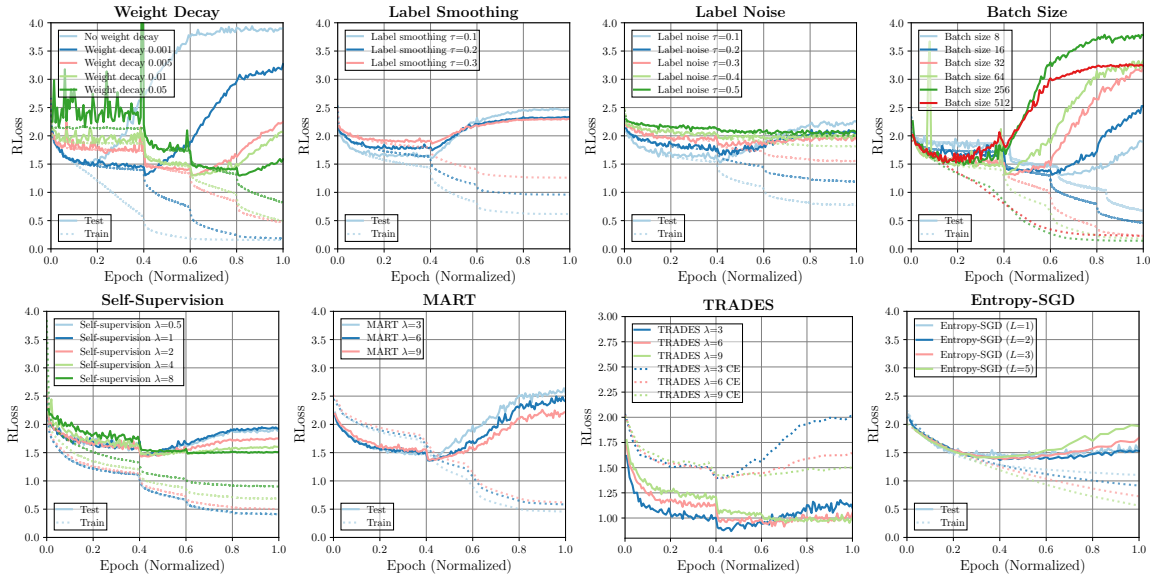


Figure B.5: **Training Curves for Varying Hyperparameters:** We plot RLoss for selected methods and hyperparameters to demonstrate the impact of hyperparameters on avoiding or reducing robust overfitting. Note that, for TRADES, we show both RLoss on adversarial examples computed by maximizing the KL-divergence in Equation (B.6) (solid) and on adversarial examples obtained by maximizing cross-entropy loss ("CE", dotted).

MART: [WZY+20] explicitly addresses the problem of incorrectly classified examples during training. First, the cross-entropy loss $\mathcal{L}$ for training is replaced using a binary cross-entropy loss $\mathcal{L}_{\text{bin}}$, i.e., classifying correct class vs. most-confident "other" class:

$$\mathcal{L}_{\text{bin}}(f(x;w),y) = -\log(f_y(x;w)) - \log(1 - \max_{y' \neq y} f_{y'}(x;w)). \tag{B.7}$$

Second, the KL-divergence used in TRADES in Equation (B.6) is combined with a confidence-based weight:

$$\mathcal{L}_{\text{bin}}(f(\tilde{x};w),y) + \lambda \text{KL}(f(x;w), f(\tilde{x};w))(1 - f_y(x;w)) \tag{B.8}$$

Adversarial examples are still computed by maximizing regular cross-entropy loss. We follow the official implementation[4].

PGD-τ: In [ZXH+20], a variant of PGD is proposed for AT: PGD-τ stops maximization τ iterations *after* the label flipped. This is supposed to find "friendlier" adversarial examples that can be used for AT. Note that $\tau = 0$ also does *not* compute adversarial examples on incorrectly classified training examples. We consider $\tau \in \{0, 1, 2, 3\}$.

Self-Supervision: Following [HMKS19], we implement AT using rotation-prediction as *additional* self-supervised task. Note, however, that no additional (unlabeled) training examples are used. Specifically, the following learning problem is tackled:

$$\max_{\|\delta\|_\infty \leq \epsilon} \mathcal{L}(f(x+\delta;w),y) + \lambda \max_{\|\delta\|_\infty \leq \epsilon} \mathcal{L}(f(\text{rot}(x+\delta,r);w),y_r) \tag{B.9}$$

for $r \in \{0, 90, 180, 270\}, y_r \in \{0, 1, 2, 3\}$ where $\text{rot}(x, r)$ rotates the training example x by r degrees. In practice, we split every batch in half: The first half uses the original training examples with correct labels. Examples in the second half are rotated randomly by $\{0, 90, 180, 270\}$

[4]https://github.com/YisenWang/MART

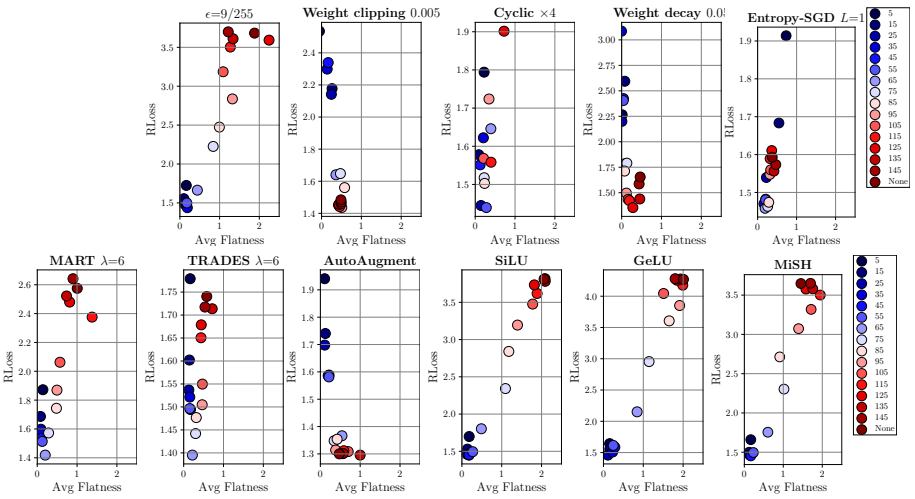


Figure B.6: **Flatness Throughout Training:** We plot RLoss against average-case flatness in RLoss for selected methods throughout training epochs. Early epochs are shown in dark blue, late epochs are shown in dark red. For cyclic learning rate, we show 4 cycles with a total of 120 epochs. For many methods not avoiding robust overfitting, flatness decreases alongside an increase in RLoss during overfitting. Using, e.g., AutoAugment, label noise or Entropy-SGD, in contrast, both effects are reduced.

degrees, and the labels correspond to the rotation (i.e., $\{0, 1, 2, 3\}$). Adversarial examples are computed against the true or rotation-based labels. Note that, in contrast to common practice [SBC+20], we do *not* predict all four possible rotations every batch, but just one randomly drawn per example. We still use 150 epochs in total. We consider $\lambda \in \{0.5, 1, 2, 4, 8\}$.

Additional Unlabeled Examples: As proposed in [CRS+19, AUH+19], we also consider additional, pseudo-labeled examples during training. We use the provided pseudo-labeled data from [CRS+19] and split each batch in half: using 50% original CIFAR10 training examples, and 50% pseudo-labeled training examples from [CRS+19]. We still use 150 epochs in total. We follow the official PyTorch implementation[5].

B.4.1 Training Curves

Figure B.5 shows (test) RLoss throughout training for selected methods and hyperparameters. Across all methods, we found that hyperparameters have a large impact on robust overfitting. For example, weight decay or smaller batch sizes can reduce and delay robust overfitting considerably if regularization is "strong" enough, i.e., large weight decay or low batch size (to induce more randomness). For the other methods, difference between hyperparameters is more subtle. However, across all cases, reduced overfitting generally goes hand in hand with higher RLoss on training examples, i.e., the robust generalization gap is reduced. This indicates that avoiding convergence on training examples plays an important role in avoiding robust overfitting. Training curves for all methods are shown in Figure B.8.

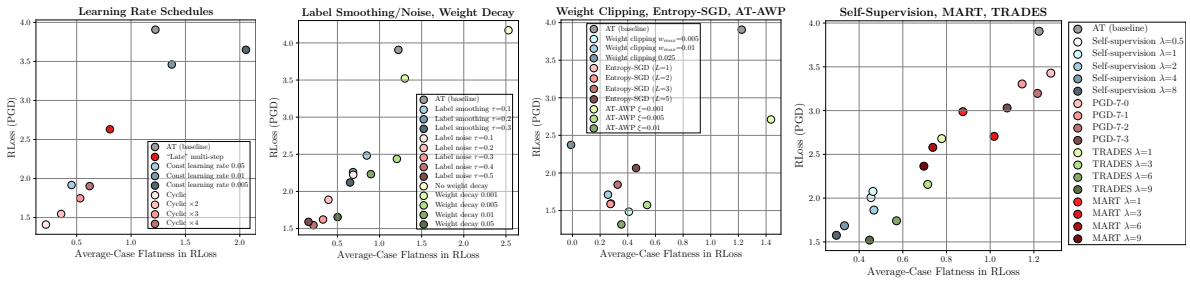


Figure B.7: **Robustness and Flatness for Varying Hyperparameters:** RLoss (y-axis) plotted against average-case flatness of RLoss (x-axis) for various groups of methods: learning rate schedules (left), label smoothing/noise and weight decay (middle left), weight clipping, Entropy-SGD and AT-AWP (middle right) as well as AT with self-supervision, MART and TRADES (right). As outlined in Section B.4, we considered multiple hyperparameter settings per method and show that favorable hyperparameters in terms of adversarial robustness also result in improved flatness. That is, in most cases, varying hyperparameters creates (roughly) a diagonal line in these plots. Interestingly, weight clipping can be considered an outlier: adversarial robustness improves while *average-case* flatness *reduces*.

B.4.2 Flatness

Flatness Throughout Training: Figure B.6 shows RLoss (y-axis) plotted against average-case flatness in RLoss (x-axis) throughout training, i.e., over epochs (dark blue to dark red), for methods not highlighted in Section 5.3. Strikingly, using higher $\epsilon = 9/255$ or alternative activation functions (SiLU [EUD18], GeLU [HG16] or MiSH [Mis20]) affect neither robust overfitting nor flatness significantly. Interestingly, label smoothing avoids sharper minima during overfitting, but does *not* avoid an increased RLoss. Methods that consistently reduce or avoid robust overfitting, e.g., weight clipping, label noise, strong weight decay or AutoAugment, avoid both the increase in RLoss and worse flatness. Clearly, our previous observations are confirmed: flatness usually reduces alongside RLoss in robust overfitting.

Flatness Across Hyperparameters: In Figure B.7, we consider flatness when changing hyperparameters of selected methods. As before, we plot RLoss (y-axis) against average-case flatness in RLoss (x-axis) for various groups of methods: learning rate schedules (first column), label smoothing/noise and weight decay (second column), methods explicitly improving flatness, i.e., weight clipping, Entropy-SGD and AT-AWP (third column), as well as self-supervision, MART and TRADES (fourth column). Except for weight clipping, hyperparameter settings with improved adversarial robustness also favor flatter minima. In most cases, this relationship follows a clear, diagonal line. For weight clipping, in contrast, the relationship is reversed: improved flatness reduces RLoss. Thus, Figure B.7 (fifth column) considers worst-case flatness in RLoss. Here, "stronger" weight clipping improves both robustness *and* flatness. This supports our conclusions: methods need at least "some kind" of flatness, average- or worst-case, in order to improve adversarial robustness.

[5]https://github.com/yaircarmon/semisup-adv

Model (sorted by RLoss on AA) (PGD = PGD-20, 10 restarts) (AA = AutoAttack [CH20c])	Test Robustness Err (test)	RErr (test) (PGD)	RErr (test) (AA)	Train Robustness Err (train)	RErr (train) (PGD)	RErr (train) (AA)	Flatness Avg RLoss	Worst RLoss
Carmon [CRS+19]	10.31	37.6	40.8	1.93	16.8	19.2	0.7	0.34
Engstrom [EIS+19]	12.97	45.3	49.2	6.71	33.1	36.3	0.23	0.51
Pang [PYD+20]	14.87	36.6	45.8	7.79	20.5	28.6	0.08	0.07
Wang [WZY+20]	12.5	37.1	42.8	8.07	24.8	32.1	0.61	0.34
Wong [WRK20]	16.66	54.4	57.6	11.86	44.9	49.2	0.3	0.16
Wu [WXW20b]	14.64	41.5	43.9	2.2	14.5	16.5	0.49	0.09
Zhang [ZYJ+19]	15.08	44.1	46.4	7.83	29.9	33.6	0.61	0.43
Zhang [ZXH+20]	15.48	43	47.2	4.85	26.3	30.1	0.51	0.13

Table B.1: **RobustBench [CAS+20a]: Err, RErr and Flatness in RLoss:** Err and RErr on train and test examples as well as average- and worst-case flatness in RLoss for pre-trained models from RobustBench. In contrast to Table B.2, the RobustBench models were obtained using early stopping.

B.5 Results in Tabular Form

Table B.2 reports the quantitative results from all our experiments. Besides flatness in RLoss, we also report both average- and worst-case flatness in (clean) Loss. We use $\xi = 0.5$ for average-case flatness and $\xi = 0.003$ for worst-case flatness. In Table B.2, methods are sorted (in ascending order) by RErr against AutoAttack [CH20c]. Additionally, we split all methods into four groups: good, average, poor and worse robustness at 57%, 60% and 62.8% RErr. These thresholds correspond roughly to the 30% and 70% percentile of all methods with RErr $\leq$ 62.8%. As our AT baseline obtains 62.8% RErr, we group all methods with higher RErr than 62.8% in worse robustness. Finally, Table B.1 report RErr and RLoss, together with our average- and worst-case flatness (of RLoss) measures for the evaluated, pre-trained models from RobustBench [CAS+20a].

Model	Test Robustness			Train Robustness			Early Stopping		Flatness			
(sorted by RErr on AA) (PGD = PGD-20, 10 restarts) (AA = AutoAttack [CH20c])	Err (test)	RErr (test) (PGD)	RErr (test) (AA)	Err (train)	RErr (train) (PGD)	RErr (train) (AA)	RErr (stop) (PGD)	RErr (stop) (AA)	Avg Loss	Worst Loss	Avg **RLoss**	Worst **RLoss**
+Unlabeled	16.96	45.9	48.9	12.6	38.6	43.2	45.3	48.9	0.12	4.64	0.32	1.2
Cyclic $\times 2$	19.66	51.2	53.6	7.64	32.3	35.4	51	53.6	0.09	3.93	0.35	1.5
AutoAugment	16.89	49.5	54.0	12.25	42.8	47.9	49.5	53.5	0.13	15.01	0.49	0.69
AT-AWP ξ=0.01	21.4	50.7	54.3	13.52	37.4	43.1	48.9	53.6	0.12	6.17	0.35	2.68
AT-AWP ξ=0.005	20.05	52.5	55	7.34	28.1	31.8	50.8	53.3	0.15	6.98	0.54	4.46
Label noise τ=0.4	20.56	52.4	55	9.66	32.8	36.8	51.2	54.8	0.11	3.95	0.21	0.96
TRADES λ=9	23.03	52.4	55	2.92	16.4	18.8	49.7	53	0.19	5.04	0.45	3.08
Cyclic $\times 3$	20.04	53.1	55.2	5.62	26.9	30.6	53.1	55.2	0.1	4.1	0.53	0.93
Cyclic	22.42	53.2	55.4	13.09	39.5	43.5	53.2	55.4	0.07	2.6	0.22	0.41
Label noise τ=0.5	22.71	51.3	55.4	15.04	40.4	45.5	51.3	55.4	0.09	0.43	0.16	0.13
Label noise τ=0.3	19.9	54.2	56.2	5.47	26.9	30	51.8	55.5	0.15	3.37	0.33	0.93
Weight clipping w_{max}=0.005	21.39	54.1	56.5	10.19	35.6	39	54.1	56.5	0.74	10.49	0.41	4.58
TRADES λ=6	21.68	55.3	56.7	1.74	13.5	15.8	50.1	53.4	0.21	5.12	0.57	2.26
Cyclic $\times 4$	19.85	55.2	56.9	4.01	23.1	26	55.1	56.9	0.16	6.65	0.62	0.8
Self-supervision λ=4	17.1	55.3	57.1	5.76	41.9	45	55.3	56.8	0.12	5.59	0.34	2.64
Adam	25.84	53.9	57.5	18.87	47.9	52.3	53.9	57.5	0.22	2.65	0.56	0.9
Entropy-SGD (L=2)	24.53	54.4	57.6	9.03	35.4	38.8	52.6	55.2	0.08	1.76	0.27	0.7
Self-supervision λ=1	15.9	56.9	58.1	1.48	28.3	31.6	55.9	57.5	0.12	6.98	0.46	3.87
Weight decay 0.05	19.32	56.2	58.1	5.03	29	32.8	52	54.8	0.12	5.77	0.51	3.94
Batch size 8	17.73	57.1	58.2	3.46	26.8	31.4	55.6	58.2	0.32	24.01	1.55	12.27
Entropy-SGD (L=1)	25.42	56	58.6	12.79	42.8	46.1	53.2	56.9	0.09	3.24	0.28	1.8
Self-supervision λ=0.5	16.16	58	58.6	1.26	28	30.7	56.7	58.3	0.1	6.48	0.45	3.29
AT-AWP ξ=0.001	18.75	57.3	58.7	1.34	15.1	18.3	52.1	54.6	0.34	20.42	1.44	13.82
Self-supervision λ=2	15.72	57.4	58.7	2.47	33.4	36.6	55.8	57.7	0.1	21.79	0.47	3.47
MART λ=9	22.06	57	58.8	3.86	16	22	50	55	0.18	8.08	0.7	3.42
Weight decay 0.01	18.52	57.2	58.9	2.06	20.1	23.2	51.7	55.3	0.25	16.46	0.9	7.19
Batch size 16	18.12	58.3	59	1.82	20.4	24.5	52.5	55.6	0.33	22.11	1.41	11.39
Self-supervision λ=8	19.6	56.6	59	12.08	50	53.3	56.6	58.6	0.11	3.59	0.29	1.76
TRADES λ=3	20.51	57.7	59.1	0.94	13.4	15.5	52.3	54.9	0.2	19.08	0.71	3.48
Weight decay 0.005	18.79	58.2	59.4	2.03	20.2	23.9	51.8	54.3	0.26	19.67	1.2	8.35
Label noise τ=0.2	19.45	57.5	59.5	2.34	18.8	22.2	50.2	53	0.18	9.79	0.39	1.4
MART λ=3	20.89	58.9	59.6	1.94	14.4	19.2	53.3	57.4	0.17	10.53	1.01	3.99
Weight clipping w_{max}=0.01	19.15	58	59.6	3.28	21.5	24.8	56.7	58.5	0.66	15.1	0.26	7.41
Learning rate 0.2	19.17	58.3	59.7	0.46	9.4	12.4	54.3	56.6	0.2	24.41	1.44	5.75
MiSH	19.29	58.9	59.8	0.06	4.5	5.3	51.8	53.7	0.25	10.04	1.58	3.55
"Late" multistep	20.63	58.5	59.8	1.6	16.4	18.4	54.2	57.8	0.17	5.24	0.81	2.96
SiLU	19.45	59.7	60	0.07	4.8	5.6	51.3	53.7	0.3	9.97	1.68	4.2
Weight averaging (τ=0.9975)	19.63	59.7	60	0.19	7.9	10	50.5	53	0.23	15.66	1.29	6
Weight clipping 0.025	18.91	59.2	60.4	0.73	12.5	15.6	52.1	54.9	0.32	17.4	0	8.61
Batch size 32	18.72	59.6	60.5	0.56	12	14.6	53.7	55.6	0.18	19.34	1.22	7.88
Entropy-SGD (L=3)	24	58.5	60.5	5.25	29.9	33.9	56.7	59.3	0.09	2.91	0.33	1.03
Label noise τ=0.1	19.39	59	60.8	1.12	14.1	17.5	51.9	55	0.2	16.75	0.69	3.55
Larger ϵ=9/255	21.3	60.4	60.9	0.47	8.9	11.1	51.3	53.8	0.21	10.26	1.34	5.85
Label smoothing τ=0.1	19.55	59.6	61	0.2	6.4	8.5	52.5	55	0.26	8.87	0.85	2.66
MART λ=6	21.51	58.7	61	3.21	16.1	20.8	49.2	54.7	0.18	13.52	0.74	3.17
Weight averaging (τ=0.98)	20.01	60.6	61	0.2	7.6	9.9	54.3	56.3	0.23	12.8	1.37	5.6
Weight decay 0.001	19.47	59.9	61	0.36	10.4	13.3	52	54.8	0.24	8.36	1.3	6.78
Batch size 64	19.06	60.5	61.1	0.3	9.2	11.1	51.2	54.4	0.18	23.13	1.14	5.96
GeLU	20.64	60.8	61.1	0.01	2.7	3.2	54.9	56.7	0.23	14.31	1.56	4.13
Label smoothing τ=0.3	19.41	59.4	61.2	0.27	5.7	8	51.1	54	0.29	18.42	0.65	2.72
MART λ=1	20.51	59.4	61.2	1.04	11.4	14.7	50.3	55.4	0.17	7.97	0.87	3.1
Weight averaging (τ=0.99)	20.41	60.3	61.4	0.19	7.8	9.6	51.7	54.2	0.22	6.12	1.44	4.98
Dropout	18.91	60.5	61.6	0.58	13	16.7	51.2	54.5	0.2	13.81	1.52	7.01
PGD-14	20.8	60.6	61.6	0.22	7.1	9.3	53.6	56.1	0.27	20.9	1.48	5.35
Entropy-SGD (L=5)	23.48	59.5	61.7	3.01	22.2	25.9	53.2	56.6	0.1	3.57	0.46	1.49
Ignore incorrect	18.4	60.5	61.8	0.06	6.3	9	54.4	56.4	0.21	14.65	1.68	5.93
Learning rate 0.1	19.23	61.1	61.9	0.26	8.9	11.5	51.9	54.2	0.21	17.63	1.23	5.26
TRADES λ=1	17.54	59.5	61.9	0.15	16.6	20.7	56.6	59.6	0.16	12.68	0.78	4.3
Weight averaging (τ=0.985)	20.27	61.7	62.3	0.18	7.4	9.4	55.9	58	0.22	15.66	1.35	6.51
Label smoothing τ=0.2	20.07	60.2	62.4	0.26	5.1	7.8	51.9	54.6	0.28	9.94	0.69	2.61
Prevent label leaking	18.38	62.1	62.4	0.38	8.6	10.8	55.3	57.7	0.22	14.62	1.48	6
AT (baseline)	20.2	61	62.8	0.33	8.5	10.7	52.3	54.6	0.21	21.05	1.22	6.49
Const learning rate 0.05	24.96	60.7	62.9	6.17	32.9	37.8	55.4	58.9	0.09	3.52	0.44	0.9
PGD-5	20.22	61.8	62.9	0.11	7.3	10.4	55.1	57.4	0.17	20.4	1.24	4.19
Batch size 256	20.86	62.6	63.3	0.28	8.2	10.3	56.9	58.4	0.3	11.22	1.35	8.33
PGD-7-3	17.17	61.7	63.3	0.08	19.5	25.2	51.3	58.8	0.17	7.4	1.08	5.29
Batch size 512	22.58	62.9	63.5	0.64	11	14.2	58.6	60	0.48	23.97	1.92	16.22
Learning rate 0.01	22.83	63	63.5	1.05	15.2	18	57.8	59.7	0.56	23.42	2.25	16.02
No weight decay	23.37	64.8	65.7	0.23	9.2	12.7	57.1	60.3	0.66	21.05	2.53	11.75
PGD-7-0	14.67	63.8	65.7	0.09	23.7	30	59.4	61.4	0.11	6.86	1.28	2.8
PGD-7-2	16.19	63.6	65.9	0.1	20.9	28.1	58.3	62.3	0.14	22.81	1.21	2.55
PGD-7-1	15.02	64.1	67.1	0.11	25.9	34.3	58.8	63.8	0.13	11.71	1.15	2.33
Const learning rate 0.01	25.87	66.7	67.4	0.67	18.5	20.7	58.4	61	0.33	15.09	1.37	8.27
Const learning rate 0.005	27.24	68.3	69.2	0.42	15.5	16.7	61.1	65.5	0.59	20.63	2.06	15.74

Table B.2: **Results: Err, RErr and Flatness in Loss and RLoss.** Err and RErr (PGD-20 and AutoAttack [CH20c]) on test and train examples, together with average- and worst-case flatness in (clean) Loss and RLoss. Methods sorted by (test) RErr against AutoAttack and split into good, average, poor and worse robustness at 57%, 60% and 62.8% RErr, see text.

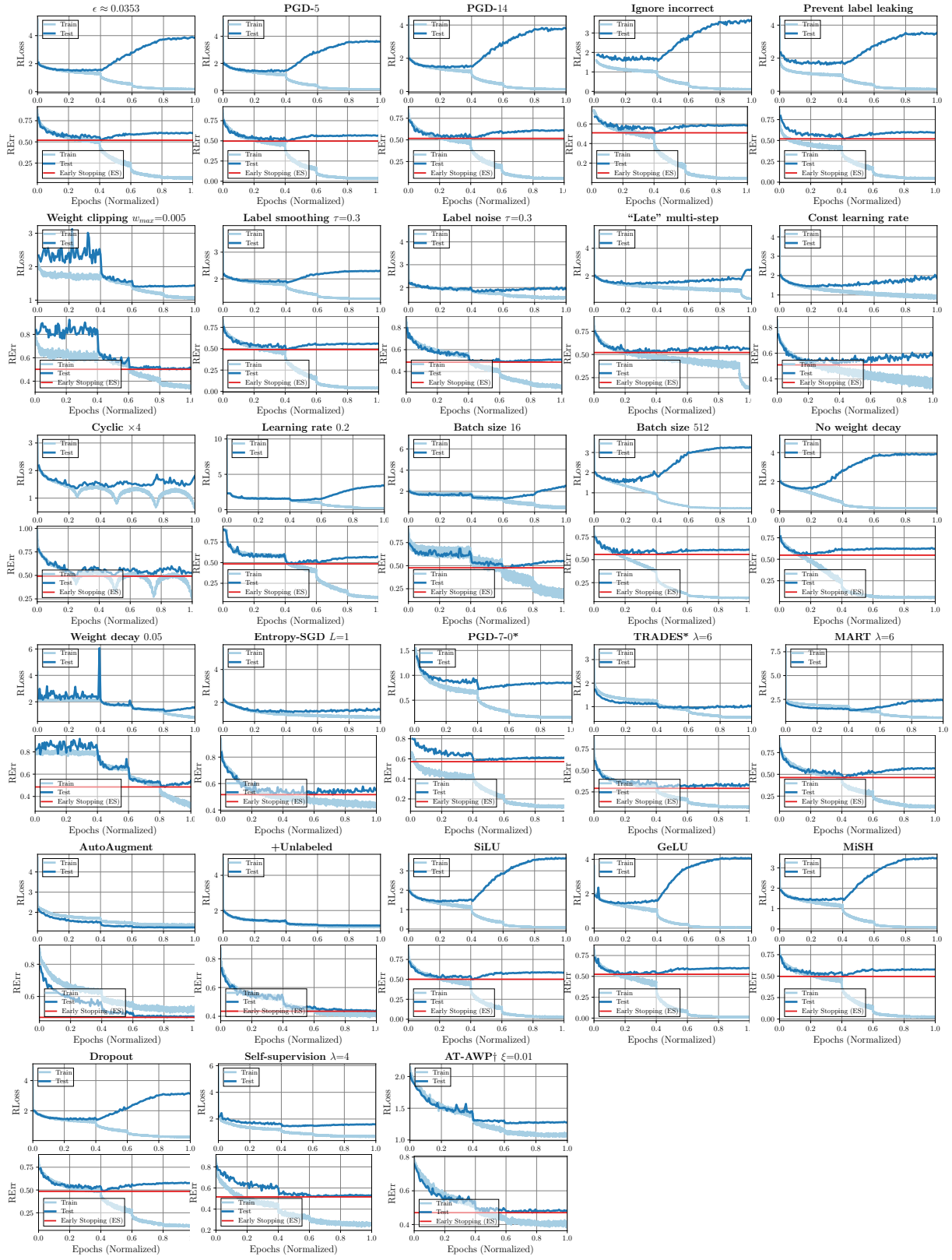


Figure B.8: **Training Curves:** Test and train RLoss (top) and RErr (bottom), including RErr for early stopping, for all considered methods with selected hyperparameters. * Train and test RLoss correspond to the attacks used during training, e.g., PGD-τ or maximizing KL-divergence for TRADES. † Reported RLoss corresponds to RLoss on adversarial examples *without* adversarial weights.

C Random and Adversarial Bit Error Robustness: Energy-Efficient and Secure Neural Network Accelerators

C.1 Complete Profiled Bit Error Statistics and Evaluation

Figure C.1 splits the bit error distributions for chips 1 and 2 into a 0-to-1 flip and a 1-to-0 bit flip map. The obtained maps, p_{1to} and p_{ot1}, contain per-bit flip probabilities for 1-to-0 and 0-to-1 bit flips. In this particular profiled chip, Figure C.1 (bottom), 0-to-1 flips are more likely. Similarly, Figure C.1 (right) shows that most 0-to-1 flips are actually persistent across time i.e., not random transient errors. The following table summarizing the key statistics of the profiled chips: the overall bit error rate p, the rate of 1-to-0 and 0-to-1 flips p_{1to} and p_{ot1}, and the rate of persistent errors p_{sa}, all in %:

Chip	p	p_{ot1}	p_{1to}	p_{sa}
1	2.744	1.27	1.47	1.223
	0.866	0.38	0.49	0.393
2	4.707	3.443	1.091	0.627
	1.01	0.82	0.19	0.105
	0.136	0.115	0.021	0.01
3	2.297	1.81	0.48	0.204
	0.597	0.496	0.0995	0.206

For evaluation, we assume that the deep neural network weights are mapped linearly onto the memory of these chips. The bit error maps are of size 8192 × 128 bits for chips 2 and 3 and

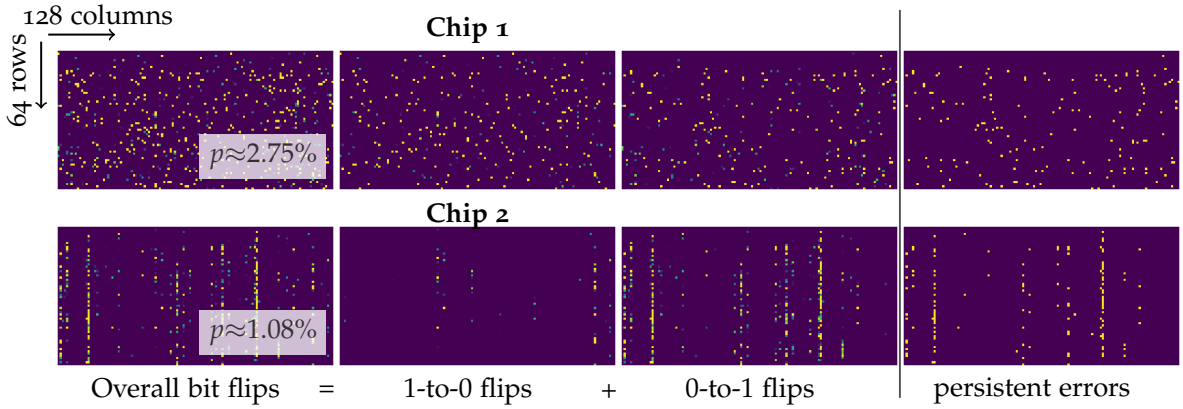


Figure C.1: **Low-Voltage Induced Bit Errors on Profiled Chips:** We break the the bit error distribution of chips 1 and 2 down into 1-to-0 and 0-to-1 bit flips. Additionally, we show that most of the bit errors are actually persistent and thus not subject to randomness. As before, we show a sub-array of size 64 × 128 from all profiled bit cells (i.e., across all SRAM arrays).

2048×128 bits for chip 1. Furthermore, to simulate various different mappings, we repeat this procedure with various offsets and compute average RErr across all mappings.

C.2 Quantization and Bit Manipulation in PyTorch

We implement "fake" fixed-point quantization for quantization-aware training and bit error injection directly in PyTorch [PGC+17]. Here, fake quantization means that computation is performed in floating point, but before doing a forward pass, the deep neural network is quantized and dequantized, i.e., $w_q = Q^{-1}(Q(w))$. Note that we quantize into *unsigned* 8 bit integers, irrespective of the target precision $m \leq 8$. To later induce random bit errors, the $8 - m$ most significant bits (MSBs) are masked for $m < 8$. Bit manipulation of unsigned 8 bit integers is then implemented in C/CUDA and interfaced to Python using CuPy [cup] or CFFI [cff]. These functions can directly operate on PyTorch tensors, allowing bit manipulation on the CPU as well as the GPU.

SimpleNet on **MNIST**	
Layer	Output Size N_C, N_H, N_W
Conv+GN+ReLU$*$	32, 28, 28
Conv+GN+ReLU$*$	64, 28, 28
Conv+GN+ReLU$*$	64, 28, 28
Conv+GN+ReLU$*$	64, 28, 28
Pool	64, 14, 14
Conv+GN+ReLU$*$	64, 14, 14
Conv+GN+ReLU$*$	64, 14, 14
Conv+GN+ReLU$*$	128, 14, 14
Pool	128, 7, 7
Conv+GN+ReLU$*$	256, 7, 7
Conv+GN+ReLU$*$	1024, 7, 7
Conv+GN+ReLU$*$	128, 7, 7
Pool	128, 3, 3
Conv+GN+ReLU$*$	128, 3, 3
Pool	128, 1, 1
FC	10
W	1,082,826

SimpleNet on **CIFAR10**	
Layer	Output Size N_C, N_H, N_W
Conv+GN+ReLU$*$	64, 32, 32
Conv+GN+ReLU$*$	128, 32, 32
Conv+GN+ReLU$*$	128, 32, 32
Conv+GN+ReLU$*$	128, 32, 32
Pool	128, 16, 16
Conv+GN+ReLU$*$	128, 16, 16
Conv+GN+ReLU$*$	128, 16, 16
Conv+GN+ReLU$*$	256, 16, 16
Pool	256, 8, 8
Conv+GN+ReLU$*$	256, 8, 8
Conv+GN+ReLU$*$	256, 8, 8
Pool	256, 4, 4
Conv+GN+ReLU$*$	512, 4, 4
Pool	512, 2, 2
Conv+GN+ReLU$*$	2048, 2, 2
Conv+GN+ReLU$*$	256, 2, 2
Pool	256, 1, 1
Conv+GN+ReLU$*$	256, 1, 1
Pool	256, 1, 1
FC	10
W	5,498,378

p on **MNIST**	
p in %	pmW, $m = 8$
Random Bit Errors	
10	866260
5	433130
1.5	129939
1	86626
0.5	43313

p on **CIFAR**	
p in %	pmW, $m = 8$
Random Bit Errors	
1	439870
0.5	219935
0.01	43987

Table C.1: **Architectures, Number of Weights W, Expected Number of Bit Errors:** *Left and Middle:* SimpleNet architectures used for MNIST and CIFAR10 with the corresponding output sizes, channels N_C, height N_H and width N_W, and the total number of weights W. We use group normalization *with* learnable scale/bias. *Right:* The number of expected bit errors for random bit errors, i.e., pmW. With $*$ we mark "blocks" of convolutional, normalization and ReLU layer after which we inject bit errors in activations in the corresponding experiments.

CIFAR10 SimpleNet+GN

m	Err in %
–	4.34
8	**4.32**
4*	5.29
3*	5.71

CIFAR10 Arch. Comparison

Model	no Quant.	$m = 8$
SimpleNet+GN	4.34	4.32
SimpleNet+BN	4.04	3.83
ResNet-50+GN	5.88	6.81
ResNet-50+BN	**3.91**	**3.67**

MNIST

m	Err in %
4	0.4
2*	0.47

CIFAR100

Quant. m, Model	Err in %
8, SimpleNet	23.68
8, WRN	18.53

Table C.2: **Quantization-Aware Training Accuracies:** Clean Err for $m = 8$ bits or lower using our robust fixed-point quantization. We obtain competitive performance for $m = 8$ and $m = 4$ bits. On CIFAR100, a Wide ResNet (WRN) clearly outperforms our standard SimpleNet model. Batch normalization (BN), improving Err slightly on CIFAR10, is significantly less robust than group normalization (GN). * For $m \leq 4$, we report results with weight clipping, CLIPPING$_{0.1}$.

	CIFAR10: quantization robustness							
	Model (see text)	Err in %	RErr in %, p in % p=0.01					
			0.01	0.05	0.1	0.5	1	1.5
$m = 8$ bit	global	4.63	10.70	86.01	90.36	90.71	90.57	–
	per-layer (= NORMAL)	4.36	4.82	5.51	6.37	24.76	72.65	87.40
	+asymmetric	4.36	5.76	6.47	7.85	40.78	76.72	85.83
	+unsigned	4.42	6.58	6.97	7.4	17.00	54.57	83.18
	+rounded (= RQUANT)	4.32	4.60	5.10	5.54	11.28	32.05	68.65
4 bit	integer conversion	5.81	90.46	90.40	90.39	90.36	90.36	90.39
	proper rounding	5.29	5.49	5.75	5.99	7.71	10.62	15.79

Table C.3: **Impact of Quantization Scheme on Robustness:** We report Err and RErr for various bit error rates p with global, per-layer and asymmetric quantization, $m = 8$ bits. Instead of quantizing into signed integer, using unsigned integers works better for asymmetric quantization. Furthermore, proper rounding instead of integer conversion also improves robustness. Note that influence on clean Err is negligible, i.e., the model can "learn around" these difference in quantization-aware training. Especially for $m = 4$ bit, the latter makes a significant difference in terms of robustness.

C.3 Network Architectures and Expected Bit Errors

Table C.1 summarizes the used SimpleNet [HRFS16] architecture, its number of weights and the expected number of bit errors for various bit error rates p. Blocks of convolutional, group normalization and ReLU layers highlighted with a $*$ are subject to bit errors in their activations in the corresponding experiments.

C.4 Additional Experiments

C.4.1 Architecture Comparison

Table C.2 summarizes accuracies for various architectures on MNIST, CIFAR10 and CIFAR100. We report results for various precisions m and specifically compare the used SimpleNet to ResNets [HZRS16b], considering both batch normalization (BN) [IS15a] and group normalization (GN) [WH18]. As can be seen, a ResNet-50 on CIFAR10 is able to outperform our SimpleNet slightly when using BN, it performs significantly worse with GN.

CIFAR10 (m = 8 bit): clipping robustness for post- and during-training quantization								
	Model	Err in %	RErr in %, p in % p=0.01					
			0.01	0.05	0.1	0.5	1	1.5
Post-Training Asymmetric	NORMAL	4.37	4.95 ±0.11	5.47 ±0.17	6.03 ±0.22	15.42 ±3.4	51.83 ±9.92	81.74 ±5.14
	RQUANT	4.27	4.59 ±0.08	5.10 ±0.13	5.54 ±0.15	10.59 ±1.11	30.58 ±6.05	63.72 ±6.89
	CLIPPING$_{0.25}$	4.96	5.24 ±0.07	5.73 ±0.14	6.16 ±0.21	10.51 ±0.91	26.27 ±5.65	61.49 ±9.03
	CLIPPING$_{0.2}$	5.24	5.48 ±0.05	5.87 ±0.09	6.23 ±0.13	9.47 ±0.7	19.78 ±3.58	43.64 ±8.2
	CLIPPING$_{0.15}$	5.38	5.63 ±0.05	6.03 ±0.09	6.38 ±0.13	8.80 ±0.41	15.74 ±2.24	36.29 ±7.34
	CLIPPING$_{0.1}$	5.32	5.52 ±0.04	5.82 ±0.06	6.05 ±0.07	7.45 ±0.26	9.80 ±0.62	17.56 ±3.08
Symmetric (during training)	NORMAL	4.36	4.82 ±0.07	5.51 ±0.19	6.37 ±0.32	24.76 ±4.71	72.65 ±6.35	87.40 ±2.47
	RQUANT	4.39	4.77 ±0.08	5.43 ±0.21	6.10 ±0.32	17.11 ±3.07	55.35 ±9.4	82.84 ±4.52
	CLIPPING$_{0.25}$	4.63	4.99 ±0.07	5.53 ±0.1	6.06 ±0.16	13.55 ±1.42	41.64 ±7.35	73.39 ±7.15
	CLIPPING$_{0.2}$	4.50	4.79 ±0.06	5.25 ±0.09	5.65 ±0.16	9.64 ±0.99	21.37 ±4.23	45.68 ±7.9
	CLIPPING$_{0.15}$	5.18	5.42 ±0.05	5.76 ±0.08	6.07 ±0.09	8.36 ±0.43	13.80 ±1.45	24.70 ±3.77
	CLIPPING$_{0.1}$	4.86	5.07 ±0.04	5.34 ±0.06	5.59 ±0.1	7.12 ±0.3	9.44 ±0.7	13.14 ±1.79
	CLIPPING$_{0.05}$	5.56	5.70 ±0.03	5.89 ±0.06	6.03 ±0.08	6.68 ±0.14	7.31 ±0.2	8.06 ±0.36
Asymmetric (default) quant. (during training)	NORMAL	4.36	4.82 ±0.07	5.51 ±0.19	6.37 ±0.32	24.76 ±4.71	72.65 ±6.35	87.40 ±2.47
	RQUANT	4.32	4.60 ±0.08	5.10 ±0.13	5.54 ±0.2	11.28 ±1.47	32.05 ±6	68.65 ±9.23
	CLIPPING$_{0.25}$	4.58	4.84 ±0.05	5.29 ±0.12	5.71 ±0.16	10.52 ±1.14	27.95 ±4.16	62.46 ±8.89
	CLIPPING$_{0.2}$	4.63	4.91 ±0.05	5.28 ±0.08	5.62 ±0.11	8.27 ±0.35	18.00 ±2.84	53.74 ±8.89
	CLIPPING$_{0.15}$	4.42	4.66 ±0.05	5.01 ±0.09	5.31 ±0.12	7.81 ±0.6	13.08 ±2.21	23.85 ±5.07
	CLIPPING$_{0.1}$	4.82	5.04 ±0.04	5.33 ±0.07	5.58 ±0.1	6.95 ±0.24	8.93 ±0.46	12.22 ±1.29
	CLIPPING$_{0.05}$	5.44	5.59 ±0.04	5.76 ±0.07	5.90 ±0.07	6.53 ±0.13	7.18 ±0.16	7.92 ±0.25
	CLIPPING$_{0.2}$ +LS	4.48	4.77 ±0.05	5.19 ±0.1	5.55 ±0.12	9.46 ±0.82	32.49 ±5.07	68.60 ±7.33
	CLIPPING$_{0.15}$ +LS	4.67	4.86 ±0.05	5.23 ±0.08	5.83 ±0.12	7.99 ±0.43	29.40 ±6.99	68.99 ±8.48
	CLIPPING$_{0.1}$ +LS	4.82	5.05 ±0.04	5.37 ±0.08	6.10 ±0.11	7.36 ±0.4	10.59 ±1.01	18.31 ±2.84
	CLIPPING$_{0.05}$ +LS	5.30	5.43 ±0.03	5.63 ±0.06	6.43 ±0.07	6.51 ±0.15	7.30 ±0.23	8.06 ±0.38

Table C.4: **Weight Clipping Improves Robustness.** We report Err and RErr for various experiments on the robustness of weight clipping with $w_{\max}$, i.e., CLIPPING$_{w_{\max}}$. First, we show that the robustness benefit of CLIPPING is independent of quantization-aware training, robustness also improves when applying post-training quantization. Then, we show results for both symmetric and asymmetric quantization. For the latter we demonstrate that label smoothing [SVI+16] reduces the obtained robustness. This supports our hypothesis that weight clipping, driven by minimizing cross-entropy loss during training, improves robustness through redundancy.

C.4.2 Robust Quantization

Table C.3 shows results considering additional bit error rates p. Note that, for $m = 8$ bit, changes in the quantization has negligible impact on clean Err. Only the change from global to per-layer quantization makes a difference. However, considering RErr for larger bit error rates, reducing the quantization range, e.g., through per-layer and asymmetric quantization, improves robustness significantly. Other aspects of the quantization scheme also play an important role, especially for low-precision such as $m = 4$ bit: asymmetric quantization into *unsigned* integers and proper rounding.

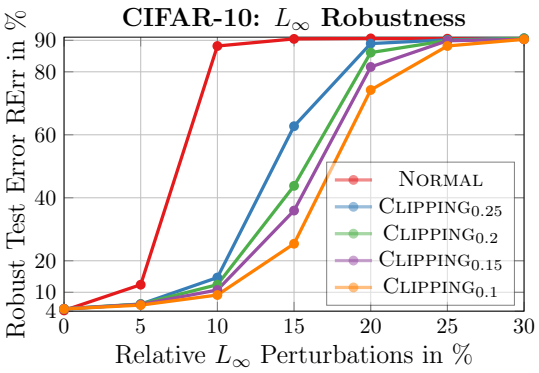


Figure C.2: **Weight Clipping Improves L_∞ Robustness:** On CIFAR10, we plot RErr for *relative* L_∞ perturbations on weights: Random noise with L_∞-norm smaller than or equal to x% of the weight range is applied. CLIPPING clearly improves robustness. Again, the relative magnitude of noise is not affected by weight clipping. Note that L_∞ noise usually affects all weights, while random bit errors affect only a portion of the weights.

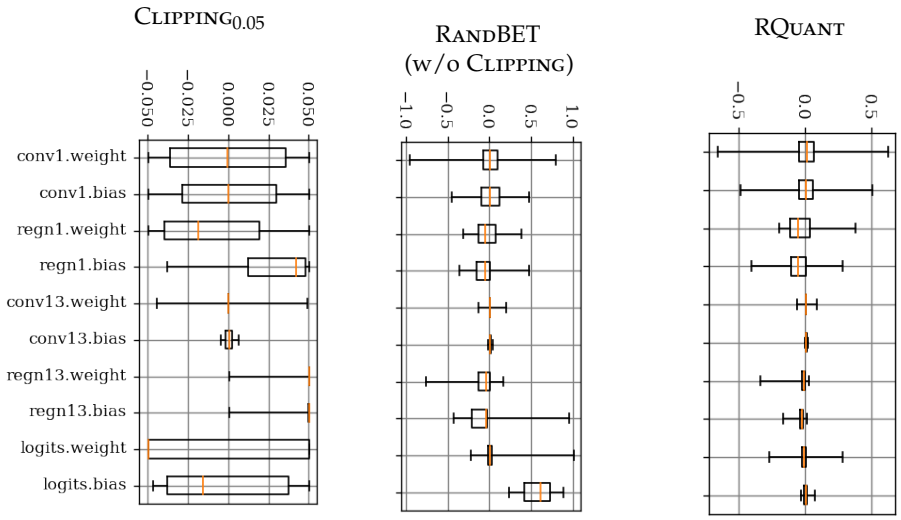


Figure C.3: **Weight Clipping Increases Redundancy:** We show weight distributions of selected layers for CLIPPING$_{0.05}$, RANDBET (without weight clipping) as well as RQUANT. We show weights and biases for the logit layer as well as the first and last (13th) convolutional layer. Scale/Bias parameters of GN are also included.

C.4.3 (Global) Weight Clipping (CLIPPING)

In Table C.4 we present additional robustness results for weight clipping. Note that weight clipping constraints the weights during training to $[-w_{\max}, w_{\max}]$ through projection. We demonstrate that weight clipping can also be used independent of quantization. To this end, we train deep neural networks with weight clipping, but without quantization. We apply post-training quantization and evaluate bit error robustness. While the robustness is reduced slightly compared to quantization-aware training *and* weight clipping, the robustness benefits of weight clipping are clearly visible. For example, clipping at $w_{\max} = 0.1$ improves RErr from 30.58% to 9.8% against $p = 1\%$ bit error rate when performing post-training quantization. With symmetric quantization-aware training, CLIPPING$_{0.1}$ improves slightly to 7.31%. Below (middle), we show results for weight clipping and symmetric quantization. These results are complemented in Table C.5 with RANDBET. Symmetric quantization might be preferable due to reduced computation and energy cost compared to asymmetric quantization. However, this also increases RErr slightly. Nevertheless, CLIPPING consistently improves robustness, independent of the difference in quantization. Finally, on the bottom, we show results confirming the adverse effect of label smoothing [SVI+16] on RErr. Figure C.2 also shows that the obtained

CIFAR10 (m = 8 bit): RandBET with symmetric quantization							
Model	Err in %	RErr in %, p in % p=0.01					
		0.01	0.05	0.1	0.5	1	1.5
NORMAL	4.36	4.82 ±0.07	5.51 ±0.19	6.37 ±0.32	24.76 ±4.71	72.65 ±6.35	87.40 ±2.47
RQUANT	4.39	4.77 ±0.08	5.43 ±0.21	6.10 ±0.32	17.11 ±3.07	55.35 ±9.4	82.84 ±4.52
CLIPPING$_{0.1}$	4.86	5.07 ±0.04	5.34 ±0.06	5.59 ±0.1	7.12 ±0.3	9.44 ±0.7	13.14 ±1.79
RANDBET$_{0.1}$ p=0.01	5.07	5.27 ±0.04	5.54 ±0.07	5.73 ±0.11	7.18 ±0.29	9.63 ±0.9	13.81 ±2.2
RANDBET$_{0.1}$ p=0.1	4.62	4.83 ±0.04	5.09 ±0.08	5.31 ±0.08	6.70 ±0.28	8.89 ±0.59	12.20 ±1.33
RANDBET$_{0.1}$ p=1	5.03	5.22 ±0.04	5.43 ±0.06	5.61 ±0.07	6.56 ±0.13	7.70 ±0.26	8.99 ±0.42
RANDBET$_{0.1}$ p=1.5	5.24	5.37 ±0.03	5.57 ±0.06	5.76 ±0.07	6.66 ±0.14	7.62 ±0.25	8.71 ±0.42
RANDBET$_{0.1}$ p=2	5.82	5.97 ±0.04	6.19 ±0.07	6.37 ±0.09	7.22 ±0.19	8.03 ±0.23	8.96 ±0.38

Table C.5: **RandBET Robustness with *Symmetric* Quantization:** Average RErr and standard deviation for CLIPPING and RANDBET with $w_{\max} = 0.1$ and **s**ymmetric quantization, i.e., larger quantization range than asymmetric quantization. Also c.f. Table C.4. Robustness decreases slightly compared to asymmetric quantization, however, CLIPPING and RANDBET are still effective in reducing RErr against high bit error rates p.

robustness generalizes to other error models such as L_∞ weight perturbations, see caption for details.

As CLIPPING adds a hyperparameter, Table C.4 also illustrates that $w_{\max}$ can easily be tuned based on clean performance. Specifically, lower $w_{\max}$ will eventually increase Err and reduce confidences (alongside increasing cross-entropy loss). This increase in Err is usually not desirable except when optimizing for robust performance, i.e., considering RErr. Also, we found that weight clipping does not (negatively) interact with any other hyperparameters or regularizers. For example, we use weight clipping in combination with AutoAugment/Cutout and weight decay without problems. Furthermore, it was not necessary to adjust our training setup (i.e., optimizer, learning rate, epochs, etc.), even for low $w_{\max}$.

Figure C.3 presents further supporting evidence for our hypothesis: While RANDBET mainly affects the logits layer, CLIPPING clearly increases the weight range used by the deep neural network. Here, the weight range is understood relative to $w_{\max}$ (or the maximum absolute weight value for NORMAL).

C.4.4 Random Bit Error Training (RANDBET)

Table C.5 shows complementary results for RANDBET using *symmetric* quantization. Symmetric quantization generally tends to reduce robustness, i.e., increase RErr, across all bit error rates p. Thus, the positive impact of RANDBET is pronounced, i.e., RANDBET becomes more important to obtain high robustness when less robust fixed-point quantization is used. These experiments also demonstrate the utility of RANDBET independent of the quantization scheme at hand.

As ablation for RANDBET, we also consider two variants of RANDBET motivated by related work [KOY+19]. Specifically, in [KOY+19], the bit error rate seen during training is increased slowly during training. Note that [KOY+19] trains on fixed bit error patterns. Thus, increasing the bit error rate during training is essential to ensure that the deep neural network is robust to any bit error rate $p' < p$ smaller than the target bit error rate. While this is generally the case using our RANDBET, Table C.6 shows that slowly increasing the random bit error rate during training, called "curricular" RANDBET, has no significant benefit over standard RANDBET. In fact, RErr increases slightly. Similarly, we found that RANDBET tends to increase the range of weights: the weights are "spread out", c.f. Figure C.3 (top right). This also increases the

CIFAR10 (m = 8 bit): RandBET variants			
	Err	RErr in %	
	in %	p=0.1	p=1
RandBET p=0.1, $w_{max} = 0.1$	4.93	5.67	8.65
RandBET p=1, $w_{max} = 0.1$	5.06	5.87	7.60
Curriculum RandBET p=1, $w_{max} = 0.1$	4.89	5.78	8.51
Curriculum RandBET p=1, $w_{max} = 0.1$	5.32	6.13	7.98
Alternating RandBET p=1, $w_{max} = 0.1$	5.07	5.91	8.93
Alternating RandBET p=1, $w_{max} = 0.1$	5.24	6.25	8.02

Table C.6: **RandBET Variants:** Err and RErr for RandBET and two variants: curricular RandBET, with p being increased slowly from $p/20$ to p during the first half of training; and "alternating" RandBET where weight updates increasing quantization range, i.e., increasing the maximum absolute weight per layer, are not possible based on gradients from perturbed weights, see Section C.4.4 for details. Both variants decrease robustness slightly. This is in contrast to, e.g., [KOY+19], using curricular training on profiled bit errors with success.

CIFAR10 (m = 8 bit): ResNet architectures			
	Err	RErr in %	
	in %	p=0.5	p=1.5
ResNet-20			
RQuant	4.34	13.89 ±2.45	81.25 ±5.08
Clipping$_{0.1}$	4.83	6.76 ±0.16	11.23 ±0.97
RandBET$_{0.1}$, p=1	5.28	6.72 ±0.19	8.96 ±0.49
ResNet-50			
RQuant	6.81	32.94 ±5.51	90.98 ±0.67
Clipping$_{0.1}$	5.99	9.27 ±0.44	36.39 ±7.03
RandBET$_{0.1}$, p=1	6.04	7.87 ±0.22	11.27 ±0.6

Table C.7: **RandBET with ResNets:** We report RErr for RQuant, Clipping and RandBET using ResNet-20 and ResNet-50. According to Table C.2, Err increases significantly when using group normalization for ResNet-50, explaining the generally higher RErr. However, using ResNets, Clipping and RandBET continue to improve robustness significantly, despite a ResNet-50 having roughly 23.5Mio weights.

quantization range, which has negative impact on robustness. Thus, we experimented with RandBET using two weight updates per iteration: one using clean weights, one on weights with bit errors. This is in contrast to averaging both updates as described in Section 6.3.3. Updates computed from perturbed weights are limited to the current quantization ranges, i.e., the maximum absolute error cannot change. This is ensured through projection. This makes sure that RandBET does not increase the quantization range during training as changes in the quantization range are limited to updates from clean weights. Again, Table C.6 shows this variant to perform slightly worse.

Table C.7 also shows results on CIFAR10 using ResNet-20 and ResNet-50. We note that, in both cases, we use group normalization (GN) instead of batch normalization (BN). ResNet-50, in particular, suffers from using GN due to the significant depth: the clean Err increases from 3.67% to 6.81% in Table C.2. Nevertheless, Clipping and RandBET remain effective against random bit errors, even for higher bit error rates of $p = 1.5\%$. This is striking as ResNet-50 consists of roughly 23.5Mio weights, compared to 5.5Mio of the used SimpleNet.

Following the RandBET algorithm outlined in Section 6.3.3, RandBET needs an additional forward and backward pass during training, increasing training complexity roughly by a factor of two. In practice, however, we found that training time for RandBET (in comparison with Clipping) roughly triples. This is due to our custom implementation of bit error injection, which was not optimized for speed. However, we believe that training time can be reduced significantly using an efficient CUDA implementation of bit error injection. We also note that

CIFAR10 (m = 8 bit): *per-layer* clipping and RandBET robustness							
Model	Err in %	RErr in %, p in % p=0.01					
		0.01	0.05	0.1	0.5	1	1.5
PLClipping$_1$	4.68	4.98 ±0.1	5.52 ±0.16	6.06 ±0.2	13.56 ±2.74	40.72 ±10.6	73.66 ±7.74
PLClipping$_{0.5}$	4.49	4.73 ±0.06	5.07 ±0.11	5.36 ±0.12	7.76 ±0.48	13.04 ±1.61	23.54 ±4.67
PLClipping$_{0.25}$	4.78	4.93 ±0.04	5.15 ±0.07	5.32 ±0.1	6.17 ±0.17	7.16 ±0.33	8.25 ±0.56
PLClipping$_{0.2}$	4.84	4.95 ±0.04	5.08 ±0.05	5.19 ±0.07	5.81 ±0.13	6.48 ±0.2	7.18 ±0.32
PLClipping$_{0.15}$	5.31	5.41 ±0.03	5.54 ±0.05	5.63 ±0.06	6.06 ±0.08	6.53 ±0.14	6.93 ±0.2
PLClipping$_{0.1}$	5.62	5.69 ±0.03	5.78 ±0.05	5.86 ±0.05	6.27 ±0.09	6.66 ±0.13	6.96 ±0.16
PLRandBET$_{0.25}$ p=0.1	4.49	4.62 ±0.04	4.82 ±0.07	4.98 ±0.08	5.80 ±0.16	6.65 ±0.22	7.59 ±0.34
PLRandBET$_{0.25}$ p=1	4.62	4.73 ±0.03	4.90 ±0.06	5.02 ±0.06	5.62 ±0.13	6.36 ±0.2	7.02 ±0.27
PLRandBET$_{0.1}$ p=1	5.66	5.76 ±0.03	5.88 ±0.06	5.96 ±0.05	6.29 ±0.09	6.59 ±0.11	6.87 ±0.12
PLRandBET$_{0.25}$ p=2	4.94	5.06 ±0.04	5.22 ±0.06	5.33 ±0.06	5.92 ±0.13	6.48 ±0.19	7.04 ±0.25
PLRandBET$_{0.1}$ p=2	5.60	5.67 ±0.02	5.77 ±0.05	5.84 ±0.05	6.20 ±0.09	6.49 ±0.09	6.72 ±0.13

Table C.8: **PLClipping and PLRandBET Further Improve Robustness:** Err and RErr for PLClipping and PLRandBET with various configurations of $w_{\max}$ and p used for training. Compared to Table C.5, reporting results for Clipping and RandBET, i.e., without per-layer weight clipping, robustness can be improved significantly, while simultaneously reducing (clean) Err.

inference time remains unchanged. In this respect, bit error mitigation strategies in hardware are clearly less desirable due to increased inference time, space and energy consumption.

C.4.5 Per-Layer Clipping and RandBET

Table C.8 shows results for PLClipping and PLRandBET considering various bit error rates p. In comparison to the results in Table C.5, robustness can be improved considerably, while also improving clean performance. For example, Clipping$_{0.1}$ (i.e., global) obtains 13.14% RErr a bit error rate of $p = 1.5\%$. Using PLClipping, this can be improved to 6.96% for PLClipping$_{0.1}$. Similarly, RandBET benefits from per-layer clipping. However, the difference between PLClipping and PLRandBET, considering RErr, is significantly smaller than before. This illustrates that *per-layer* weight clipping can have tremendous impact on robustness.

C.4.6 Profiled Bit Errors

Table C.9 shows complementary results for Clipping$_{0.05}$, RandBET$_{0.05}$, PLClipping$_{0.15}$ and PLRandBET$_{0.15}$ trained with $p = 1.5\%$ and $p = 2\%$, respectively, on all profiled chips. Note that for particularly extreme cases, such as chip 3, Clipping might perform slightly better than RandBET, indicating a significantly different bit error distribution as previously assumed. Nevertheless, PLClipping as well as PLRandBET are able to cope even with particularly difficult bit error distributions. Overall, PLRandBET generalizes reasonably well, with very good results on chip 1 and chip 2. Note that, following Figure C.1, the bit errors in chip 2 are strongly aligned along columns. Results on chip 3 are slightly worse. However, PLRandBET does not fail catastrophically with only a $\sim 1\%$ increase in RErr compared to chips 1 and 2.

In Table C.10, we consider only stuck-at-0 and stuck-at-1 bit errors (i.e., where $p_{1\to0}$ and $p_{0\to1}$ are 1). Thus, the bit error rates deviate slightly from those reported in Table C.9. Furthermore, We consider only one weight-to-SRAM mapping, i.e., without offset. PattBET is trained and

CIFAR10: Generalization to Profiled Bit Errors							
		RErr in %					
Model (CIFAR10)	Err in %	Chip 1		Chip 2		Chip 3	
		p≈0.86	p≈2.7	p≈0.14	p≈1	p≈0.03	p≈0.5
RQUANT	4.32	23.57	89.84	6.00	74.00	5.47	80.49
CLIPPING$_{0.05}$	5.44	7.17	10.50	5.98	10.02	5.78	11.88
RANDBET$_{0.05}$ p=1.5	5.62	7.04	9.37	6.00	9.00	5.85	12.44
PLCLIPPING$_{0.15}$	5.27	6.52	8.48	5.64	8.97	5.52	16.14
PLRANDBET$_{0.25}$, p=1	4.62	6.22	9.81	5.13	8.86	4.91	11.94
PLRANDBET$_{0.15}$, p=2	4.99	6.14	7.58	5.34	7.34	5.19	8.63

Table C.9: **Generalization to Profiled Bit Errors:** We show RErr on profiled bit errors, chips 1-3, for RANDBET as well as CLIPPING. Note that for chip 3, CLIPPING$_{0.05}$ performs slightly better than RANDBET. However, using per-layer clipping, PLRANDBET performs best.

CIFAR10: Fixed Pattern Training		
Model (CIFAR10)	RErr in %, p in %	
***Profiled* Bit Errors (Chip 1)**	p≈0.39	p≈1.22
PATTBET, p≈1.22	9.52	7.20
PATTBET, p≈0.39	5.77	67.87
PATTBET$_{0.15}$, p≈1.22	7.67	6.52
PATTBET$_{0.15}$, p≈0.39	5.94	30.96

CIFAR10: Fixed Pattern Training		
Model (CIFAR10)	RErr in %, p in %	
***Profiled* Bit Errors (Chip 2)**	p≈0.1	p≈0.63
PATTBET p≈0.63	85.84	10.76
PATTBET, p≈0.1	90.56	5.93
PATTBET$_{0.15}$ p≈0.63	12.02	8.70
PATTBET$_{0.15}$ p≈0.1	90.68	6.51

Table C.10: **Fixed Pattern Bit Error Training:** We report RErr for training on fixed, profiled bit error patterns (PATTBET). Note that for PATTBET on chip 1/2 we used only the stuck-at-errors shown in Figure C.1, which is why the bit error rates deviate from those reported in Table 6.6, also see Figure C.1.

evaluated on the exact same bit error pattern, but potentially with different bit error rates p. Note that the bit errors for $p' < p$ are a subset of those for bit error rate p. Thus, it is surprising that, on both chips 1 and 2, PATTBET trained on higher bit error rates does not even generalize to lower bit error rates (i.e., higher voltage). This is problematic in practice as the accelerator should not perform worse when increasing voltage.

C.4.7 Summary Results

Figure C.4 summarizes our results: In contrast to Figure 6.9, we consider individual CLIPPING and RANDBET models instead of focusing on the best results per bit error rate p. Additionally, we show our complete results for lower precisions, i.e., $m = 4, 3, 2$ on CIFAR10 and MNIST. Moderate CLIPPING, e.g., using $w_{\max} = 0.15$ on CIFAR10 (in red solid), has negligible impact on clean Err (i.e., $p = 0$ on the x-axis) while improving robustness beyond $p = 0.1\%$ bit error rate. Generally, however, higher robustness is obtained at the cost of increased clean Err, e.g., for $w_{\max} = 0.05$ (in blue dotted). Here, it is important to note that in low-voltage operation, only RErr matters – clean Err is only relevant for voltages higher than $V_{\min}$. Per-layer weight clipping, i.e., PLCLIPPING, is able to avoid the increase in Err in many cases, while preserving improved robustness. RANDBET further improves robustness, both on top of CLIPPING and PLCLIPPING, for high bit error rates while continuing to increase (clean) Err slightly. For example, RANDBET with $w_{\max} = 0.05$ and trained with $p = 2\%$ bit errors increases clean Err to 5.42% but is also able to keep RErr below 7% up to $p = 1\%$ bit error rate (in violet dotted). While per-layer clipping, i.e., PLCLIPPING$_{0.15}$ (in yellow), does not improve robustness

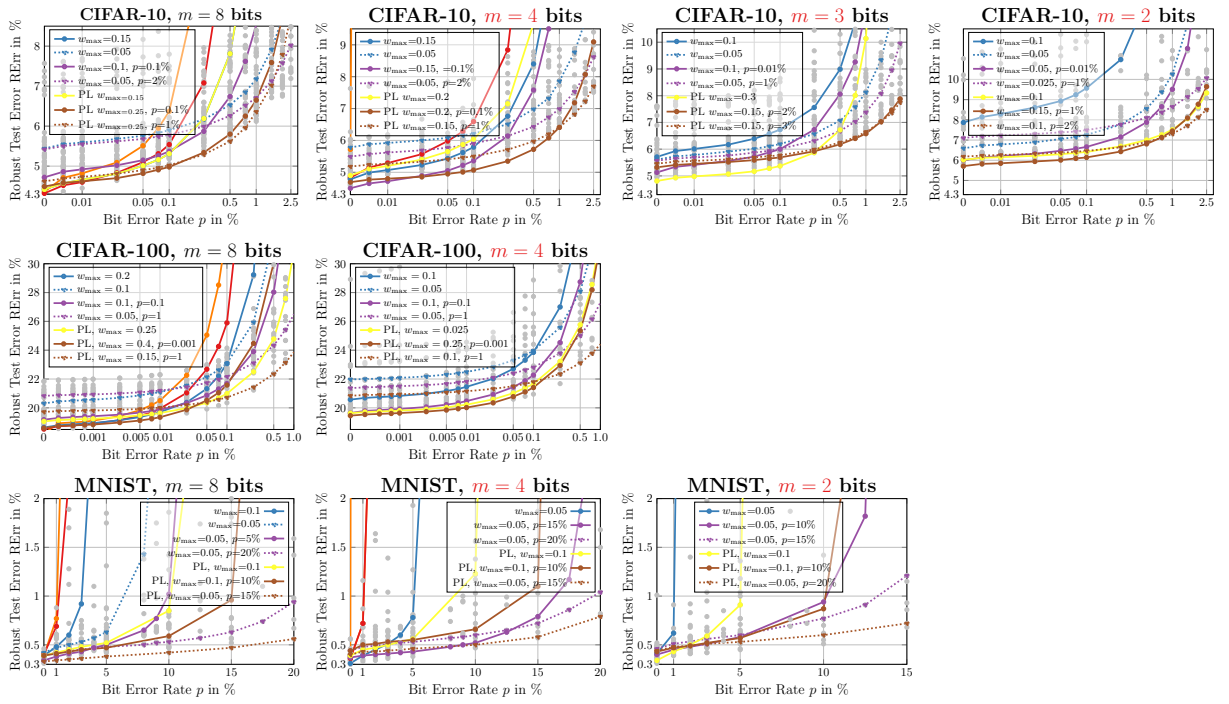


Figure C.4: **Summary Results on CIFAR10, CIFAR100 and MNIST.** We plot RErr against bit error rate, highlighting individual CLIPPING, RANDBET, PLCLIPPING and PLRANDBET models. Note that Figure 6.9, in contrast, presents the best, i.e., lowest RErr, model for each bit error rate p individually. Instead, individual models help to illustrate the involved trade-offs: CLIPPING with small w_{max} or RANDBET with high bit error rate p increases the clean Err, thereby also increasing RErr for very small bit error rates. However, RErr against large bit error rates can be reduced significantly.

compared to CLIPPING$_{0.05}$, clean Err is lowered. More importantly, using PLRANDBET$_{0.15}$ (in brown) clearly outperforms most other approaches, showing that RANDBET is particularly effective on top of PLCLIPPING, in contrast to "just" CLIPPING. Similar observations can be made for MNIST as well as CIFAR100.

The advantage of per-layer clipping, i.e., PLCLIPPING and PLRANDBET, are pronounced when reducing precision. For example, using $m = 2$, PLCLIPPING not only boosts robustness significantly, but also avoids a significant increase in Err. As a result, using PLRANDBET instead of PLCLIPPING is only necessary for high bit error rates, e.g., above $p = 1\%$. Similar trade-offs can be observed on CIFAR100 and MNIST. On CIFAR100, we see that task difficulty also reduces the bit error rate that is tolerable without significant increase it RErr. Here, $p = 0.1\%$ increases RErr by more than 3%, even with RANDBET (and weight clipping). Furthermore, CIFAR100 demonstrates that CLIPPING and RANDBET are applicable to significantly larger architectures such as Wide ResNets without problems. On MNIST, in contrast, bit error rates of up to $p = 20\%$ are easily possible. At such bit error rates, the benefit of RANDBET is extremely significant as even CLIPPING$_{0.025}$ exhibits very high RErr of 32.68% at $p = 20\%$.

CIFAR10: Bit Error Robustness in Inputs (RandBET)			
Model (CIFAR10) bit errors in inputs, p in %	Err in %	RErr in % p=0.1	 p=0.5
RQuant	4.22	11.10	23.70
Clipping$_{0.25}$	4.53	11.30	22.70
Clipping$_{0.1}$	4.81	12.50	25.40
PLClipping$_{0.25}$	4.92	10.80	22.80
PLClipping$_{0.1}$	5.73	12.80	26.50
PLRandBET$_{0.75}$ (weights only), p_w=0.1	4.59	10.20	22.20
PLRandBET$_{0.5}$ (weights only), p_w=0.1	4.48	10.90	21.00
PLRandBET$_{0.25}$ (weights only), p_w=0.1	4.46	11.00	22.80
PLRandBET$_{0.5}$ (weights only), p_w=1	5.04	11.00	24.20
PLRandBET$_{0.25}$ (weights only), p_w=1	4.64	11.30	22.40
PLRandBET$_{0.1}$ (weights only), p_w=1	5.63	13.00	25.80
PLRandBET$_{0.25}$ weights+inputs, p_w=0.1, p_i=0.1	4.94	7.60	11.70
PLRandBET$_{0.25}$ weights+inputs, p_w=0.1, p_i=0.5	4.74	8.00	12.90
PLRandBET$_{0.1}$ weights+inputs, p_w=0.1, p_i=0.1	5.94	9.00	15.10
PLRandBET$_{0.1}$ weights+inputs, p_w=0.1, p_i=0.5	5.72	8.80	17.50
PLRandBET$_{0.25}$ weights+inputs, p_w=1, p_i=0.1	5.57	7.70	9.10
PLRandBET$_{0.25}$ weights+inputs, p_w=1, p_i=0.5	5.39	7.60	8.90
PLRandBET$_{0.25}$ weights+inp.+act., p_w=0.1, p_i=0.1, p_a=0.1	5.16	7.90	12.20
PLRandBET$_{0.25}$ weights+inp.+act., p_w=0.1, p_i=0.1, p_a=0.5	5.02	7.60	12.00
PLRandBET$_{0.25}$ weights+inp.+act., p_w=1, p_i=0.1, p_a=0.1	9.09	11.50	13.80
PLRandBET$_{0.25}$ weights+inp.+act., p_w=1, p_i=0.1, p_a=0.5	8.97	11.10	14.10

Table C.11: **Input Robustness for Clipping and PLRandBET.** Average RErr for bit errors in inputs. Images are quantized using $m = 8$ bit quantization (per channel) in $[0, 1]$, which does *not* introduce errors as the images are already provided in 8-bit (per channel). Note that PLRandBET considers only bit errors in weights during training. We note that Err is reported on the first 1k test images, to be comparable with RErr for bit errors in inputs. Extreme Clipping generally reduces robustness to such input perturbations. Similarly, PLRandBET (bit errors in weights during training) may reduce robustness, indicating that robustness against bit errors in weights is in conflict with robustness against bit errors in the inputs. Nevertheless, PLRandBET (training on bit errors in inputs) can improve robustness significantly.

C.4.8 Bit Errors in Activations and Inputs

Bit Errors in Inputs: Table C.11 presents results against random bit errors in inputs (orange). We report RErr for bit error rates $p = 0.1\%$ and $p = 0.5\%$. We emphasize that in these experiments the weights are *not* subject to random bit errors. Nevertheless, bit errors in the inputs can be devastating, resulting in RErr above 20% for Clipping and PLClipping with $p = 0.5\%$. While 0.5% does not seem like much, it is important to remember that effectively $0.5 \cdot 8 = 4\%$ of pixels are affected. Also, more extreme clipping results in higher RErr, showing that robustness in weights and inputs might be contradictory to some extent. Similarly, RandBET on bit errors in weights does *not* improve robustness. RandBET on bit errors in weights *and* inputs, in contrast, can reduce RErr considerably.

Bit Errors in Activations: Table C.12 shows results for random bit errors in activations (violet). As, by default, we do *not* train with activation quantization, we report Err w/ and w/o activation quantization (in $m = 8$ bits). As can be seen, our simple activation quantization

CIFAR10: Bit Error Robustness in Activations				
Model (CIFAR10) 8-bit act. quant., bit errors in activations, p in %	Err in %	Err in % (quant. act.)	RErr in %	
			p=0.1	p=0.5
RQuant	4.32	4.53	8.93	47.12
Clipping$_{0.25}$	4.58	4.67	7.97	31.98
Clipping$_{0.1}$	4.82	5.13	7.86	24.38
PLClipping$_{0.25}$	4.96	5.16	7.38	21.58
PLClipping$_{0.1}$	5.62	5.84	8.72	27.36
PLRandBET$_{0.75}$ (weights only), p_w=0.1	4.57	4.78	8.74	38.99
PLRandBET$_{0.5}$ (weights only), p_w=0.1	4.48	4.79	7.46	27.67
PLRandBET$_{0.25}$ (weights only), p_w=0.1	4.49	4.71	7.25	24.94
PLRandBET$_{0.5}$ (weights only), p_w=1	5.11	5.33	8.10	26.90
PLRandBET$_{0.25}$ (weights only), p_w=1	4.62	4.83	6.92	19.83
PLRandBET$_{0.1}$ (weights only), p_w=1	5.66	5.92	9.31	35.79
PLRandBET$_{0.25}$ activations only, p_a=0.1	4.73	4.84	6.40	12.10
PLRandBET$_{0.25}$ activations only, p_a=0.5	5.43	5.68	6.74	10.16
PLRandBET$_{0.1}$ activations only, p_a=0.1	5.92	6.09	7.56	12.62
PLRandBET$_{0.1}$ activations only, p_a=0.5	6.52	6.63	7.77	10.81
PLRandBET$_{0.25}$ weights+act., p_w=1, p_a=0.1	7.59	7.80	8.88	11.79
PLRandBET$_{0.25}$ weights+act., p_w=1, p_a=0.5	7.66	7.89	9.09	12.17
PLRandBET$_{0.25}$ weights+act., p_w=1, p_a=1	7.68	7.83	9.05	12.07
PLRandBET$_{0.25}$ weights+inp.+act., p_w=1, p_i=0.1, p_a=0.1	9.16	9.31	10.54	13.51
PLRandBET$_{0.25}$ weights+inp.+act., p_w=1, p_i=0.1, p_a=0.5	9.01	9.32	10.56	13.65

Table C.12: **Activation Robustness for Clipping and PLRandBET.** Average RErr for bit errors in activations; the same $m = 8$ bit quantization is used for weights and activations. PLRandBET considers only bit errors in weights during training. We note that activation quantization without bit errors has negligible impact on Err. However, when training on bit errors in activations, Err may increase similarly as with bit errors in the weights. Regarding robustness against bit errors in activations, Clipping and PLRandBET improve robustness, however, extreme clipping or RandBET with large bit error rates in the weights reduces robustness.

leads to a slight increase of 0.2 to 0.3% in Err. Bit errors in activations turn out to be difficult to handle, even for Clipping and PLClipping. In contrast to bit errors in inputs, weight clipping has a positive effect on robustness against bit errors in activations. However, the benefit is less pronounced than for bit errors in weights, e.g., PLClipping does not improve over Clipping. Similarly, RandBET with bit errors in weights can improve robustness, but does not so consistently, see the variations in RErr for RandBET (weights only) in Table C.12. RandBET with bit errors in activations, in contrast, has the expected effect of reducing RErr considerably, allowing reasonably low RErr for $p = 0.1\%$.

C.4.9 Adversarial Bit Error Attack

Table C.14 presents a comprehensive ablation study regarding our adversarial bit error attack. We report RErr on MNIST and CIFAR10, considering an additional MNIST-like architecture and an architecture with "halved channels" on CIFAR10. Specifically, considering Table C.1, the MNIST-like architecture is the same architecture as used for MNIST, but using a larger first convolutional layer (conv1) due to the larger input dimensionality on CIFAR10. Specifically,

	Err %	RErr in %			
MNIST		ϵ=80	ϵ=160	ϵ=240	ϵ=320
RQuant	0.37	91.08	91.08	91.08	91.08
Clipping$_{0.05}$	0.38	85.09	88.81	90.11	90.26
RandBET$_{0.05}$, p=20	0.39	10.13	69.90	81.16	81.94
AdvBET$_{0.05}$, ϵ=160	0.33	11.63	21.08	31.71	61.01
AdvBET$_{0.05}$, ϵ=240	0.32	66.92	82.83	85.12	87.98
AdvBET$_{0.05}$, ϵ=320	0.27	10.41	85.73	85.57	88.12
AdvBET$_{0.05}$ (mom 0.9), ϵ=160	0.29	10.37	86.12	84.83	87.28
AdvBET$_{0.05}$ (mom 0.9), ϵ=240	0.29	11.58	21.66	41.27	50.73
AdvBET$_{0.05}$ (mom 0.9), ϵ=320	0.30	10.38	21.76	39.92	69.61
AdvBET$_{0.05}$ T, ϵ=160	0.39	10.54	21.41	29.72	49.77
AdvBET$_{0.05}$ T, ϵ=240	0.36	10.10	19.44	31.23	51.01
AdvBET$_{0.05}$ T, ϵ=320	0.36	10.51	19.72	31.28	50.77
CIFAR10		ϵ=160	ϵ=320	ϵ=480	ϵ=640
RQuant	4.89	91.18	91.18	91.18	91.18
Clipping$_{0.05}$	5.34	20.48	60.76	79.12	83.93
RandBET$_{0.05}$, p=2	5.42	14.66	33.86	54.24	80.36
AdvBET$_{0.05}$, ϵ=160	5.54	15.20	26.22	55.06	77.43
AdvBET$_{0.05}$, ϵ=320	5.78	15.27	42.79	70.56	91.13
AdvBET$_{0.05}$, ϵ=480	5.99	15.41	44.66	83.39	91.47
AdvBET$_{0.05}$, ϵ=640	6.49	15.90	33.79	70.61	90.48

Table C.13: **Adversarial Bit Error Robustness on MNIST and CIFAR10.** We provide additional results regarding RErr for RQuant, Clipping RandBET and AdvBET on MNIST and CIFAR10. We evaluate adversarial bit error robustness for various ϵ and also train AdvBET with different ϵ. On MNIST, we also present an ablation when training AdvBET against attacks with or without momentum 0.9 or with targeted attack ("T"). As targeted attacks are generally more effective, this also helps improve robustness using AdvBET. On CIFAR10, we found AdvBET with $\epsilon > 160$ to reduce robustness. We suspect that training gets more difficult and might require additional capacity or more sophisticated training schemes.

the number of weights in conv1 increase from 288 to 864 (factor 3 due to 3 input channels on CIFAR10). Then, we consider the CIFAR10 architecture but with all channel widths halved (similar to MNIST). These two architecture result in roughly 1Mio and 1.3Mio weights. While attacks on all weights (targeted or untargeted) are not very effective on MNIST, compared to attacks on conv1 or logit layer, the models on CIFAR10 are more vulnerable in this regard, especially RQuant. However, even on CIFAR10, the conv1 and logit layers are most susceptible to bit errors. Attacking all *other* layers (i.e., all layers except conv1 and logit), in contrast, is not very fruitful in terms of increasing RErr, especially considering Clipping. Only AdvBET is able to reduce the vulnerability of these layers. Interestingly, Clipping$_{0.05}$ is more robust on CIFAR10, even when using the MNIST-like architecture. Here, for ϵ=160, RErr is 58.09% on MNIST but only 50.4% on CIFAR10. As the only difference between both models is the larger first convolutional layer conv1 on CIFAR10, this further supports our experiments showing that conv1 is particularly vulnerable. In all cases, targeted attacks are more successful.

MNIST		Worst RErr in %										
	Err	ϵ=80	ϵ=160									
$W\approx$1Mio	in %	(all U+T)	(all U+T)	U (all)	T (all)	U (log)	T (log)	U (conv)	T (conv)	U (l+c)	T (l+c)	U (*)	T (*)
RQuant	0.37	91.08	91.08	0.48	0.51	89.86	91.08	85.42	84.27	89.86	91.08	0.48	0.48
Clipping$_{0.05}$	0.38	1.00	85.09	2.30	0.74	10.77	0.72	83.09	85.09	9.90	0.90	0.39	0.41
AdvBET$_{0.05}$, ϵ=160	0.29	0.33	10.37	10.37	0.36	10.03	0.37	0.40	0.47	0.36	0.41	0.29	0.29
CIFAR10		Worst RErr in %										
like MNIST	Err	ϵ=160	ϵ=320									
$W\approx$1Mio	in %	(all U+T)	(all U+T)	U (all)	T (all)	U (log)	T (log)	U (conv)	T (conv)	U (l+c)	T (l+c)	U (*)	T (*)
RQuant	6.63	91.49	91.49	61.34	91.49	90.42	91.49	86.91	86.46	90.42	91.49	37.77	45.56
Clipping$_{0.05}$	7.31	50.40	75.40	59.74	70.16	59.21	75.40	60.16	64.91	58.91	68.43	8.44	10.01
half channels	Err	ϵ=160	ϵ=320									
$W\approx$1.3Mio	in %	(all U+T)	(all U+T)	U (all)	T (all)	U (log)	T (log)	U (conv)	T (conv)	U (l+c)	T (l+c)	U (*)	T (*)
RQuant	6.67	91.58	91.58	46.96	91.58	90.44	91.58	87.46	87.78	90.44	91.58	53.96	52.46
Clipping$_{0.05}$	7.79	37.47	82.80	58.22	82.60	54.61	82.80	64.96	62.26	53.50	82.59	8.32	9.69
Clipping$_{0.05}$ (ResNet-20)	7.30	38.91	64.72	35.19	35.57	43.96	62.37	47.94	55.60	45.53	64.72	8.19	8.94
full channels	Err	ϵ=160	ϵ=320									
$W\approx$5.5Mio	in %	(all U+T)	(all U+T)	U (all)	T (all)	U (log)	T (log)	U (conv)	T (conv)	U (l+c)	T (l+c)	U (*)	T (*)
RQuant	4.89	91.18	91.18	8.54	91.18	90.68	91.18	86.28	89.06	90.68	91.18	6.73	7.46
Clipping$_{0.05}$	5.34	20.48	60.76	24.04	35.20	23.67	35.86	52.57	60.76	25.27	35.86	5.80	5.61
AdvBET$_{0.05}$, ϵ=160	5.54	15.20	26.22	10.01	20.20	25.70	26.22	8.84	12.33	8.91	25.47	5.82	6.18

Table C.14: **Adversarial Bit Error Ablation on MNIST and CIFAR10.** (Worst) RErr for various models on MNIST and CIFAR10 against our adversarial bit error attack with ϵ allowed bit errors. For CIFAR10, we consider an MNIST-like architecture (c.f. Table C.1), a model with halved channels, and the original model. We report worst RErr across all attacks and restarts as well as individually for: attacking all weights, attacking only **log**it layer, attacking only the first **conv**olutional layer, attacking only both logit and conv1 layer ("**l+c**") and attacking all layers except logit and conv1 ("*"). In all cases, we consider targeted or untargeted attacks. Just because the first convolutional layer for the MNIST-like model on CIFAR10 has 3× more weights, Clipping$_{0.05}$ is significantly more robust for ϵ=160. This is emphasized when considering the slightly larger "halved channels" model and can be confirmed using a ResNet-20. Across all datasets, attacking logit or conv1 layer is highly effective. Only AdvBET is able to reduce the vulnerability of these layers.

D Confidence-Calibrated Adversarial Training: Generalizing to Unseen Attacks

D.1 PGD with Momentum and Backtracking

Algorithm 7 summarizes PGD with momentum and backtracking for an individual input example x considering the L_∞ norm. The algorithm is easily adapted for batches of inputs and other L_p norms.

Algorithm 7 Projected Gradient Descent (PGD) with Backtracking. Pseudocode for the used PGD procedure using momentum and backtracking subject to the constraints $\tilde{x}_i = x_i + \delta_i \in [0,1]$ and $\|\delta\|_\infty \leq \epsilon$. In practice, the procedure is applied on batches of inputs. The algorithm is easily adapted to work with arbitrary L_p-norm; only the projections on Line 7 and 7 as well as the normalized gradient in Line 7 need to be adapted.

1: **input:** iterations T, learning rate γ, momentum β, learning rate factor α, initial $\delta^{(0)}$
2: $\tilde{x} := x + \delta^{(0)}$ {best adversarial example obtained}
3: $v := 0$, $g^{(-1)} := 0$ {best objective v and accumulated gradients g}
4: **for** $t = 0, \ldots, T$ **do**
5: clip $\delta_i^{(t)}$ to $[-\epsilon, \epsilon]$, clip $x_i + \delta_i^{(t)}$ to $[0,1]$ {projection onto L_∞ ϵ-ball and $[0,1]$}
6: $v^{(t)} := \mathcal{F}(x + \delta^{(t)}, y)$ {forward and backward pass, see Equation (7.2) or Equation (7.4)}
7: $g^{(t)} := \text{sign}\left(\nabla_{\delta^{(t)}} \mathcal{F}(x + \delta^{(t)}, y)\right)$
8: **if** $v^{(t)} > v$ **then**
9: $\tilde{x} := x + \delta^{(t)}$, $v := v^{(t)}$ {keep track of adversarial example resulting in best objective}
10: **end if**
11: **if** $t = T$ **then**
12: **break** {iteration T is only meant to check whether last update improved objective}
13: **end if**
14: $g^{(t)} := \beta g^{(t-1)} + (1-\beta) g^{(t)}$ {integrate momentum term}
15: $\hat{\delta}^{(t)} := \delta^{(t)} + \gamma g^{(t)}$ {"try" the update step and see if objective increases}
16: clip $\hat{\delta}_i^{(t)}$ to $[-\epsilon, \epsilon]$, clip $x_i + \hat{\delta}_i^{(t)}$ to $[0,1]$
17: $\hat{v}^{(t)} := \mathcal{F}(x + \hat{\delta}^{(t)}, y)$
18: **if** $\hat{v}^{(t)} \geq v^{(t)}$ **then**
19: $\delta^{(t+1)} := \hat{\delta}^{(t)}$ {only keep the update if the objective increased}
20: **else**
21: $\gamma := \gamma / \alpha$ {decrease learning rate otherwise}
22: **end if**
23: **end for**
24: **return** $\tilde{x}$, $\tilde{v}$

D.2 Attack Initialization

Deviating from [MMS+18], we initialize δ uniformly over directions and norm (instead of uniform initialization over the volume of the ϵ-ball):

$$\delta = u\epsilon \frac{\delta'}{\|\delta'\|_\infty}, \quad \delta' \sim \mathcal{N}(0, I), u \sim U(0,1) \tag{D.1}$$

where δ' is sampled from a standard Gaussian and $u \in [0,1]$ from a uniform distribution. We also consider zero initialization, i.e., $\delta = 0$. For random initialization we always consider multiple restarts, 10 for PGD-Conf and 50 for PGD-CE; for zero initialization, we use 1 restart.

D.3 ROC AUC and Computing Confidence Threshold

To compute receiver operating characteristic (ROC) curves, and the area under the curve, i.e., ROC AUC, we define negatives as *successful* adversarial examples (corresponding to correctly classified test examples) and positives as the corresponding *correctly classified* test examples. The ROC AUC as well as the curve itself can easily be calculated using scikit-learn [PVG+11]. Practically, the generated curve could be used to directly estimate a threshold corresponding to a pre-determined true positive rate (TPR). However, this requires interpolation and after trying several interpolation schemes, we concluded that the results are distorted significantly, especially for TPRs close to 100%. Thus, we follow a simpler scheme: on a held out validation set of size 1000 (the last 1000 samples of the test set), we sorted the corresponding confidences, and picked the confidence threshold in order to obtain (at least) the desired TPR, e.g., 99%.

In Section 7.3, instead of reporting ROC AUC, we reported only confidence-thresholded robust test error (RErr), which implicitly subsumes the false positive rate (FPR), at a confidence threshold of 99%TPR. Again, we note that this is an extremely conservative choice, allowing to reject at most 1% correctly classified clean examples. In addition, comparison to other approaches is fair as the corresponding confidence threshold only depends on correctly classified clean examples, not on adversarial examples. As also seen in Figure D.1, ROC AUC

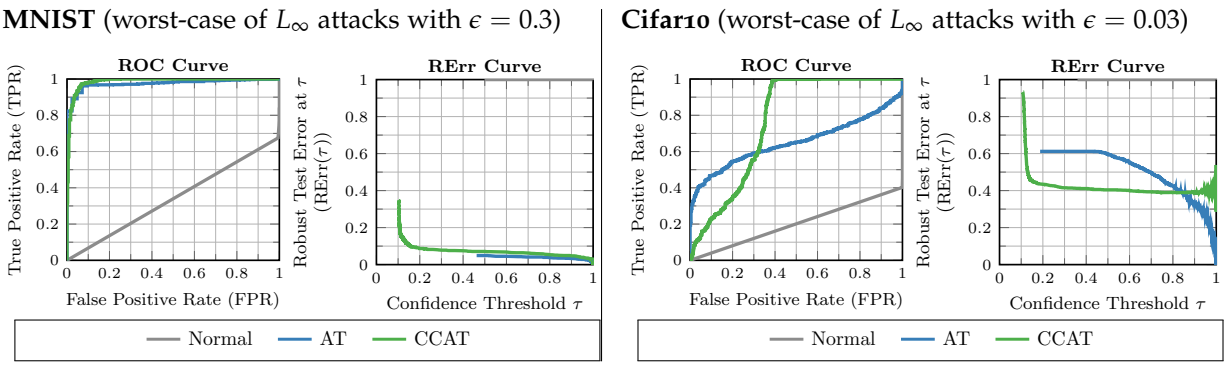


Figure D.1: **ROC and RErr Curves.** ROC curves, i.e. FPR plotted against TPR for all possible confidence thresholds τ, and (confidence-thresholded) RErr curves, i.e., RErr over confidence threshold τ for AT and CCAT, including different ρ parameters. Worst-case adversarial examples across all L_∞ attacks with $\epsilon = 0.3$ (MNIST) and $\epsilon = 0.03$ (Cifar10) were tested. For evaluation, the confidence threshold τ is fixed at 99%TPR, allowing to reject at most 1% correctly classified clean examples.

MNIST: Attack Ablation with confidence-thresholded RErr in % for τ@99%TPR (L_∞ attack with $\epsilon = 0.3$ for training and testing)									
Optimization	momentum+backtrack					mom		–	
Initialization	zero				rand	zero		zero	
Iterations	40	200	1000	2000	2000	60	300	60	300
AT	0.4	0.4	0.4	0.3	0.6	0.4	0.6	0.4	0.4
AT Conf (AT trained with PGD-Conf)	0.8	0.8	0.8	0.8	1.1	1.0	1.1	1.0	1.0
CCAT	0.2	1.4	4.6	4.6	3.7	0.7	0.7	0.2	0.2

SVHN: Attack Ablation with confidence-thresholded RErr in % for τ@99%TPR (L_∞ attack with $\epsilon = 0.03$ for training and testing)									
Optimization	momentum+backtrack					mom		–	
Initialization	zero				rand	zero		zero	
Iterations	40	200	1000	2000	2000	60	300	60	300
AT	38.4	46.2	49.9	50.1	51.8	37.7	38.1	29.9	30.8
AT Conf (AT trained with PGD-Conf)	27.4	40.5	46.9	47.3	48.1	27.1	28.5	21.1	23.8
CCAT	4.0	5.0	22.8	23.3	5.2	2.6	2.6	2.6	2.6

CIFAR10: Attack Ablation with confidence-thresholded RErr in % for τ@99%TPR (L_∞ attack with $\epsilon = 0.03$ for training and testing)									
Optimization	momentum+backtrack					mom		–	
Initialization	zero				rand	zero		zero	
Iterations	40	200	1000	2000	2000	60	300	60	300
AT	60.9	60.8	60.8	60.8	60.9	60.9	60.9	57.4	57.6
AT Conf (AT trained with PGD-Conf)	60.4	60.6	60.5	60.5	60.9	60.4	60.6	56.2	56.6
CCAT	14.8	16.2	40.2	41.3	34.9	7.2	7.2	7.2	7.2

Table D.1: **Detailed Attack Ablation Studies.** We compare our L_∞ PGD-Conf attack with T iterations and different combinations of momentum, backtracking and initialization on all three datasets. We consider AT, AT trained with PGD-Conf (AT Conf), and CCAT and report RErr for confidence threshold τ@99%TPR. As backtracking requires an additional forward pass per iteration, we use $T = 60$ and $T = 300$ for attacks without backtracking to be comparable to attacks with $T = 40$ and $T = 200$ with backtracking. Against CCAT, $T = 1000$ iterations or more are required and backtracking is essential to achieve high RErr. AT, in contrast, is "easier" to attack, requiring less iterations and less sophisticated optimization.

is not a practical metric to evaluate the detection/rejection of adversarial examples. This is because rejecting a significant part of correctly classified clean examples is not acceptable. In this sense, ROC AUC measures how well positives and negatives can be distinguished in general, while we are only interested in the performance for very high TPR, e.g., 99%TPR. In Table D.3 to D.5, we report FPR alongside RErr for different TPRs.

D.3.1 Ablation Study

In the following, we include ablation studies for our attack PGD-Conf, in Table D.1, and for CCAT, in Table D.2.

Attack: Regarding the proposed attack PGD-Conf using momentum and backtracking, Table D.1 shows that backtracking and sufficient iterations are essential to attack CCAT. On SVHN, for AT, the difference in RErr between $T = 200$ and $T = 1000$ iterations is only 3.7%, specifically, 46.2% and 49.9%. For CCAT, in contrast, using $T = 200$ iterations is not sufficient,

SVHN: Training Ablation for Detection (τ@99%TPR) and Standard Settings ($\tau = 0$) (L_∞ attack with $\epsilon = 0.03$ during training and testing)					
	Detection Setting τ@99%TPR			**Standard Setting** $\tau = 0$	
	ROC AUC	Err in %	RErr in %	Err in %	RErr in %
NORMAL	0.17	2.6	99.9	3.6	99.9
AT	0.55	2.5	54.9	3.4	56.9
AT Conf (AT trained with PGD-Conf)	0.61	2.8	52.5	3.7	58.7
CCAT, $\rho = 1$	0.74	2.2	43.0	2.7	82.4
CCAT, $\rho = 2$	0.68	2.1	44.2	2.9	79.6
CCAT, $\rho = 4$	0.68	1.8	35.8	2.7	80.4
CCAT, $\rho = 6$	0.64	1.8	32.8	2.9	72.1
CCAT, $\rho = 8$	0.63	2.2	42.3	2.9	84.6
CCAT, $\rho = 10$	0.67	2.1	38.5	2.9	91.0
CCAT, $\rho = 12$	0.67	1.9	36.3	2.8	81.8

CIFAR10: Training Ablation for Detection (τ@99%TPR) and Standard Settings ($\tau = 0$) (L_∞ attack with $\epsilon = 0.03$ during training and testing)					
	Detection Setting τ@99%TPR			**Standard Setting** $\tau = 0$	
	ROC AUC	Err in %	RErr in %	Err in %	RErr in %
NORMAL	0.20	7.4	100.0	8.3	100.0
AT	0.65	15.1	60.9	16.6	61.3
AT Conf (AT trained with PGD-Conf)	0.63	15.1	61.5	16.1	61.7
CCAT, $\rho = 1$	0.63	8.7	72.4	9.7	95.3
CCAT, $\rho = 2$	0.60	8.4	70.6	9.7	95.1
CCAT, $\rho = 4$	0.61	8.6	66.3	9.8	93.5
CCAT, $\rho = 6$	0.54	8.0	69.8	9.2	94.1
CCAT, $\rho = 8$	0.58	8.5	65.3	9.4	93.2
CCAT, $\rho = 10$	0.60	8.7	63.0	10.1	95.0
CCAT, $\rho = 12$	0.62	9.4	63.0	10.1	96.6

Table D.2: **Training Ablation Studies.** We report unthresholded RErr and Err, i.e., $\tau = 0$ ("Standard Setting"), and τ@99%TPR as well as ROC AUC ("Detection Setting") for CCAT with various values for ρ. The models are tested against our L_∞ PGD-Conf attack with $T = 1000$ iterations and zero as well as random initialization. On Cifar10, $\rho = 10$ works best and performance stagnates for $\rho > 10$. On SVHN, we also use $\rho = 10$, although $\rho = 6$ shows better results.

with merely 5% RErr. However, $T = 1000$ iterations with zero initialization increases RErr to 22.8%. For more iterations, i.e., $T = 2000$, RErr stagnates with 23.3%. When using random initialization (one restart), RErr drops to 5.2%, even when using $T = 2000$ iterations. Similar significant drops are observed without backtracking. These observations generalize to MNIST and Cifar10.

Training: Table D.2 reports results for CCAT with different values for ρ. We note that ρ controls the (speed of the) transition from (correct) one-hot distribution to uniform distribution depending on the distance of adversarial example to the corresponding original training example. Here, higher ρ results in a sharper (i.e., faster) transition from one-hot to uniform

distribution. It is also important to note that the power transition does not preserve a bias towards the true label, i.e., for the maximum possible perturbation ($\|\delta\|_\infty = \epsilon$), the network is forced to predict a purely uniform distribution. As can be seen, both on SVHN and Cifar10, higher ρ usually results in better robustness. Thus, we chose $\rho = 10$. Only on SVHN, $\rho = 6$ of $\rho = 12$ perform slightly better. However, we found that $\rho = 10$ generalizes better to previously unseen attacks.

D.4 All Results

Table D.3 shows FPR corresponding to the results in Table 7.2. Table D.4 reports our main results requiring only 98%TPR; Table D.5 shows results for 95%TPR. This implies, that compared to 99%TPR, up to 1% (or 4%) more correctly classified test examples can be rejected, increasing the confidence threshold and potentially improving robustness. For relatively simple tasks such as MNIST and SVHN, where Err is low, this is a significant "sacrifice". However, as can be seen, robustness in terms of RErr only improves slightly. We found that the same holds for 95%TPR, however, rejecting more than 2% of correctly classified examples seems prohibitive large for the considered datasets.

MNIST: FPR and **RErr** in % for τ@99%TPR												
	L_∞ $\epsilon = 0.3$		L_∞ $\epsilon = 0.4$		L_2 $\epsilon = 3$		L_1 $\epsilon = 18$		L_0 $\epsilon = 15$		adv. frames	
	seen		**unseen**		**unseen**		**unseen**		**unseen**		**unseen**	
	FPR↓	RErr ↓	FPR↓	RErr ↓	FPR↓	RErr ↓	FPR↓	RErr ↓	FPR↓	RErr ↓	FPR↓	RErr ↓
NORMAL	99.3	100.0	99.3	100.0	99.3	100.0	99.3	100.0	91.6	92.3	87.0	87.7
AT-50%	1.0	1.7	99.3	100.0	80.7	81.5	23.8	24.6	23.0	23.9	72.9	73.7
AT-100%	1.0	1.7	99.2	100.0	83.9	84.8	20.2	21.3	12.9	13.9	61.3	62.3
CCAT	6.9	7.4	11.4	11.9	0.0	0.3	1.3	1.8	14.2	14.8	0.0	0.2
* MSD	32.1	34.3	96.7	98.9	57.0	59.2	53.7	55.9	64.2	66.4	6.6	8.8
* TRADES	3.4	4.0	99.3	99.9	43.6	44.3	8.2	9.0	34.6	35.5	0.0	0.2

SVHN: FPR and **RErr** in % for τ@99%TPR												
	L_∞ $\epsilon = 0.03$		L_∞ $\epsilon = 0.06$		L_2 $\epsilon = 2$		L_1 $\epsilon = 24$		L_0 $\epsilon = 10$		adv. frames	
	seen		**unseen**		**unseen**		**unseen**		**unseen**		**unseen**	
	FPR↓	RErr ↓	FPR↓	RErr ↓	FPR↓	RErr ↓	FPR↓	RErr ↓	FPR↓	RErr ↓	FPR↓	RErr ↓
NORMAL	95.8	99.9	95.9	100.0	95.9	100.0	95.9	100.0	79.6	83.7	74.6	78.7
AT-50%	52.3	56.0	84.7	88.4	95.7	99.4	95.8	99.5	70.0	73.6	30.0	33.6
AT-100%	42.1	48.3	80.9	87.1	93.3	99.5	93.6	99.8	83.1	89.4	19.9	26.0
CCAT	35.5	39.1	49.5	53.1	25.4	29.0	28.1	31.7	0.4	3.5	1.0	3.7
* LID	87.1	91	89.2	93.1	96.1	92.2	85.7	90	37.6	41.6	85.1	89.8
* MAHA	68.6	73	75.2	79.5	73.2	78.1	63.2	67.5	36.9	41.5	6.3	9.9

CIFAR10: FPR and **RErr** in % for τ@99%TPR												
	L_∞ $\epsilon = 0.03$		L_∞ $\epsilon = 0.06$		L_2 $\epsilon = 2$		L_1 $\epsilon = 24$		L_0 $\epsilon = 10$		adv. frames	
	seen		**unseen**		**unseen**		**unseen**		**unseen**		**unseen**	
	FPR↓	RErr ↓	FPR↓	RErr ↓	FPR↓	RErr ↓	FPR↓	RErr ↓	FPR↓	RErr ↓	FPR↓	RErr ↓
NORMAL	93.0	100.0	93.0	100.0	93.0	100.0	93.0	100.0	77.7	84.7	89.7	96.7
AT-50%	47.6	62.7	78.6	93.7	83.3	98.4	83.3	98.4	59.3	74.4	63.6	78.7
AT-100%	42.3	59.9	72.7	90.3	80.7	98.3	80.4	98.0	54.7	72.3	62.0	79.6
CCAT	59.9	68.4	83.9	92.4	43.7	52.2	50.3	58.8	14.4	23.0	57.4	66.1
* MSD	35.3	53.2	71.5	89.4	70.6	88.5	50.7	68.6	21.4	39.2	64.7	82.6
* TRADES	28.9	43.5	66.4	81.0	56.3	70.9	82.3	96.9	22.3	36.9	57.5	72.1
* AT-Madry	33.8	45.1	73.2	84.5	87.4	98.7	86.5	97.8	31.0	42.3	62.0	73.3
* LID	92.7	99	92.9	99.2	64	70.6	82.9	89.4	40.6	47	59.9	66.1
* MAHA	87.7	94.1	89	95.3	84.2	90.6	91.3	97.6	43.5	49.8	64.1	70

Table D.3: **Main Results: FPR for** 99%**TPR.** For **99**%TPR, we report confidence-thresholded RErr *and* FPR. We emphasize that only PGD-CE and PGD-Conf were used against LID and MAHA. In general, the observations of Section 7.4 can be confirmed considering FPR. Due to the poor Err of AT, MSD or TRADES on Cifar10, these methods benefit most from considering FPR instead of (confidence-thresholded) RErr. * Pre-trained models with different architectures.

MNIST: FPR and confidence-thresholded **RErr** in % for τ@98%TPR														
	L_∞ $\epsilon = 0.3$		L_∞ $\epsilon = 0.4$		L_2 $\epsilon = 3$		L_1 $\epsilon = 18$		L_0 $\epsilon = 15$		adv. frames		distal	corr. MNIST-C
	seen		**unseen**		**unseen**		**unseen**		**unseen**		**unseen**		**unseen**	**unseen**
	FPR	RErr	FPR	RErr	FPR	RErr	FPR	RErr	FPR	RErr	FPR	RErr	FPR	Err
NORMAL	99.3	100.0	99.3	100.0	99.3	100.0	99.3	100.0	87.3	88.1	79.8	80.5	100.0	31.0
AT-50%	0.5	0.8	99.3	100.0	66.7	67.9	16.3	17.5	16.1	17.2	61.3	62.5	100.0	12.3
AT-100%	0.6	1.3	99.2	100.0	77.3	78.3	16.9	18.0	9.2	10.2	52.6	53.7	100.0	15.4
CCAT	5.2	5.7	8.8	9.3	0.0	0.2	0.6	0.9	7.6	8.1	0.0	0.1	0.0	5.3
* MSD	28.8	31.0	96.6	98.8	53.9	56.2	51.3	53.5	61.5	63.7	4.4	6.6	100.0	5.6
* TRADES	1.2	1.9	99.1	99.7	31.6	32.6	4.3	5.1	28.0	29.7	0.0	0.1	100.0	5.7

SVHN: FPR and confidence-thresholded **RErr** in % for τ@98%TPR													
	L_∞ $\epsilon = 0.03$		L_∞ $\epsilon = 0.06$		L_2 $\epsilon = 2$		L_1 $\epsilon = 24$		L_0 $\epsilon = 10$		adv. frames		distal
	seen		**unseen**		**unseen**		**unseen**		**unseen**		**unseen**		**unseen**
	FPR	RErr	FPR	RErr	FPR	RErr	FPR	RErr	FPR	RErr	FPR	RErr	FPR
NORMAL	95.7	99.8	95.9	100.0	95.9	100.0	95.9	100.0	72.7	76.8	72.4	76.5	87.1
AT-50%	50.0	53.7	83.4	87.1	95.5	99.2	95.7	99.4	54.0	57.9	26.8	30.6	86.3
AT-100%	42.1	48.3	80.9	87.1	93.3	99.5	93.6	99.8	82.2	88.8	19.3	25.1	81.0
CCAT	34.0	37.6	40.5	44.1	20.3	23.9	25.0	28.6	0.2	2.6	0.1	2.2	0.0

CIFAR10: FPR and confidence-thresholded **RErr** in % for τ@98%TPR														
	L_∞ $\epsilon = 0.03$		L_∞ $\epsilon = 0.06$		L_2 $\epsilon = 2$		L_1 $\epsilon = 24$		L_0 $\epsilon = 10$		adv. frames		corr.	corr. CIFAR10-C
	seen		**unseen**		**unseen**		**unseen**		**unseen**		**unseen**		**unseen**	**unseen**
	FPR	RErr	FPR	RErr	FPR	RErr	FPR	RErr	FPR	RErr	FPR	RErr	FPR	Err
NORMAL	93.0	100.0	93.0	100.0	93.0	100.0	93.0	100.0	70.1	77.1	89.6	96.6	83.3	11.4
AT-50%	47.6	62.7	78.6	93.7	83.3	98.4	83.3	98.4	57.2	72.4	63.6	78.7	75.0	15.1
AT-100%	42.1	59.7	72.7	90.3	80.7	98.3	80.4	98.0	52.1	70.0	62.0	79.6	72.5	17.8
CCAT	59.4	67.9	83.5	92.0	43.3	51.8	50.0	58.5	11.7	20.3	56.4	65.1	0.0	8.1
* MSD	35.1	53.0	71.5	89.4	69.9	87.8	50.6	68.5	17.8	35.8	64.7	82.6	76.7	17.1
* TRADES	28.9	43.5	66.4	81.0	56.2	70.8	82.3	96.9	21.9	36.4	57.4	72.0	76.2	14.1
* AT-Madry	33.6	44.9	73.2	84.5	87.4	98.7	86.5	97.8	30.8	42.0	61.9	73.2	78.5	11.4

Table D.4: **Main Results: Generalizable Robustness for 98%TPR.** While reporting results for 99%TPR in Section 7.4, reducing the TPR requirement for confidence-thresholding to 98%TPR generally improves results, but only slightly. We report FPR and confidence-thresholded RErr for 98%TPR. For MNIST-C and Cifar10-C, we report mean Err across all corruptions. L_∞ attacks with ϵ=0.3 on MNIST and $\epsilon = 0.03$ on SVHN/Cifar10 were used for training (**seen**). All other attacks were not used during training (**unseen**). * Pre-trained models with different architectures.

MNIST: FPR and confidence-thresholded **RErr** in % for τ@95%TPR														
	L_∞ $\epsilon = 0.3$		L_∞ $\epsilon = 0.4$		L_2 $\epsilon = 3$		L_1 $\epsilon = 18$		L_0 $\epsilon = 15$		adv. frames		distal	corr. MNIST-C
	seen		**unseen**		**unseen**		**unseen**		**unseen**		**unseen**		**unseen**	**unseen**
	FPR	RErr	FPR	RErr	FPR	RErr	FPR	RErr	FPR	RErr	FPR	RErr	FPR	Err
NORMAL	99.3	100.0	99.3	100.0	99.3	100.0	99.0	99.7	75.6	76.7	65.9	67.0	100.0	27.5
AT-50%	0.2	0.4	99.3	100.0	54.6	56.7	12.3	13.7	11.1	12.3	47.3	49.2	100.0	8.6
AT-100%	0.2	0.9	99.2	100.0	63.9	65.6	10.7	12.4	3.6	4.4	35.0	37.2	100.0	10.5
CCAT	3.1	3.7	5.7	6.3	0.0	0.1	0.0	0.1	2.0	2.2	0.0	0.1	0.0	5.5
* MSD	22.0	24.6	95.0	97.2	44.0	46.8	42.1	44.6	54.5	57.3	2.3	4.4	100.0	4.4
* TRADES	0.6	1.0	98.1	98.8	16.5	18.3	2.1	2.7	21.4	24.0	0.0	0.0	100.0	3.2

SVHN: FPR and confidence-thresholded **RErr** in % for τ@95%TPR													
	L_∞ $\epsilon = 0.03$		L_∞ $\epsilon = 0.06$		L_2 $\epsilon = 2$		L_1 $\epsilon = 24$		L_0 $\epsilon = 10$		adv. frames		distal
	seen		**unseen**		**unseen**		**unseen**		**unseen**		**unseen**		**unseen**
	FPR	RErr	FPR	RErr	FPR	RErr	FPR	RErr	FPR	RErr	FPR	RErr	FPR
NORMAL	95.6	99.7	95.9	100.0	95.9	100.0	95.9	100.0	65.3	69.5	66.8	71.1	87.1
AT-50%	45.5	49.2	79.7	83.4	95.4	99.1	95.5	99.2	34.8	38.9	21.5	25.5	59.3
AT-100%	40.5	46.9	80.9	87.1	93.3	99.5	93.6	99.8	78.5	85.5	15.9	21.7	75.1
CCAT	32.8	36.5	38.6	42.2	16.5	20.3	21.1	24.9	0.0	1.2	0.0	1.2	0.0

CIFAR10: FPR and confidence-thresholded **RErr** in % for τ@95%TPR														
	L_∞ $\epsilon = 0.03$		L_∞ $\epsilon = 0.06$		L_2 $\epsilon = 2$		L_1 $\epsilon = 24$		L_0 $\epsilon = 10$		adv. frames		corr.	corr. CIFAR10-C
	seen		**unseen**		**unseen**		**unseen**		**unseen**		**unseen**		**unseen**	**unseen**
	FPR	RErr	FPR	RErr	FPR	RErr	FPR	RErr	FPR	RErr	FPR	RErr	FPR	Err
NORMAL	93.0	100.0	93.0	100.0	93.0	100.0	93.0	100.0	52.0	59.0	88.8	95.9	83.3	7.7
AT-50%	46.6	61.7	78.6	93.7	83.3	98.4	83.3	98.4	43.9	59.6	62.8	77.9	75.0	12.1
AT-100%	41.3	59.1	72.7	90.3	80.6	98.2	80.4	98.0	47.2	65.3	62.0	79.7	72.5	15.6
CCAT	57.4	66.0	80.8	89.3	39.7	48.2	48.9	57.4	3.6	11.1	50.8	59.7	0.0	6.0
* MSD	32.8	50.9	71.5	89.4	67.7	85.6	49.1	67.3	11.4	28.4	64.2	82.2	76.7	13.7
* TRADES	26.5	41.3	66.1	80.7	53.9	68.5	82.3	96.9	17.6	32.0	56.4	71.1	76.2	10.7
* AT-Madry	32.4	43.9	73.2	84.5	87.4	98.7	86.5	97.8	28.6	40.1	61.5	72.8	78.5	9.1

Table D.5: **Main Results: Generalizable Robustness for 95%TPR.** We report FPR and RErr for **95**%TPR, in comparison with 98% in Table D.4. For MNIST-C and Cifar10-C, we report mean Err across all corruptions. L_∞ attacks with ϵ=0.3 on MNIST and $\epsilon = 0.03$ on SVHN/Cifar10 **seen** during training; all other attacks **unseen** during training. Results improve slightly in comparison with 98%TPR. However, the improvements are rather small and do not justify the significantly increased fraction of "thrown away" (correctly classified) clean examples. * Pre-trained models with different architectures.

Learning Optimal Conformal Classifiers

E

E.1 Experimental Setup

Datasets and Splits: We consider Camelyon2016 [BVvD+17], GermanCredit [DG17], WineQuality [CCA+09], MNIST [LBBH98], EMNIST [CATvS17], Fashion-MNIST [CATvS17] and CIFAR [Kri09] with a fixed split of training, calibration and test examples. Table E.1 summarizes key statistics of the used datasets which we elaborate on in the following. Except Camelyon, all datasets are provided by Tensorflow [AAB+15][1]. For Camelyon, we use the pre-computed features of [WHK20] which are based on open source code from the Camelyon2016 challenge[2]. On Camelyon, we use the original training set, but split test examples into 100 validation and 17 test examples. This is because less than 100 calibration examples are not meaningful for α=0.05. As we evaluate 10 random calibration/test splits, the few test examples are not problematic in practice. On GermanCredit, we manually created training/calibration/test splits, roughly matching 70%/10%/20%, following WineQuality. We take the first 122.8k examples, split as in Table E.1.

Models and Training: We consider linear models, multi-layer perceptrons (MLPs) and ResNets [HZRS16a] as shown in Table E.1. Specifically, we use a linear model on MNIST and GermanCredit, 1- or 2-layer MLPs on Camelyon2016, WineQuality and Fashion-MNIST, and ResNet-34/50 [HZRS16a] on CIFAR10/100. Models and training are implemented in Jax [BFH+18][3] and the ResNets follow the implementation and architecture provided by Haiku

[1] https://www.tensorflow.org/datasets
[2] https://github.com/arjunvekariyagithub/camelyon16-grand-challenge
[3] https://github.com/google/jax

Dataset Statistics							
Dataset	Train	Cal	Test	Dimensions	Classes	Epochs	Model
Camelyon2016* [BVvD+17]	280	100	17	31	2	100	1-layer MLP
GermanCredit [DG17]	700	100	200	24	2	100	Linear
WineQuality [CCA+09]	4500	500	898	11	2	100	2-layer MLP
MNIST [LBBH98]	55k	5k	10k	28×28	10	50	Linear
EMNIST** [CATvS17]	98.8k	5.2k	18.8k	28×28	52	75	2-layer MLP
Fashion-MNIST [XRV17]	55k	5k	10k	28×28	10	150	2-layer MLP
CIFAR10 [Kri09]	45k	5k	10k	$32 \times 32 \times 3$	10	150	ResNet-34
CIFAR100 [Kri09]	45k	5k	10k	$32 \times 32 \times 3$	100	150	ResNet-50

Table E.1: **Used Datasets:** Summary of train/calibration/test splits, epochs and models used on all datasets in our experiments. The calibration set is usually 10% or less of the training set. On most datasets, the test set is roughly two times larger than the calibration set. When computing random calibration/test splits for evaluation, see text, the number of calibration and test examples stays constant. * On Camelyon, we use features provided by [WHK20] instead of the original images. ** For EMNIST, we use a custom subset of the "byClass" split.

ConfTr Hyperparameters (for ThrLP during training and Thr at test time)					
Dataset, Method	Batch Size	Learning rate	Temp. T	Size weight λ	κ in Equation (8.6)
Camelyon, ConfTr	20	0.005	0.1	5	1
Camelyon, ConfTr +$\mathcal{L}_{\text{class}}$	0.01	10	0.01	5	1
GermanCredit, ConfTr	200	0.05	1	5	1
GermanCredit, ConfTr +$\mathcal{L}_{\text{class}}$	400	0.05	0.1	5	1
WineQuality, ConfTr	100	0.005	0.5	0.05	1
WineQuality, ConfTr +$\mathcal{L}_{\text{class}}$	100	0.005	0.1	0.5	1
MNIST, ConfTr	500	0.05	0.5	0.01	1
MNIST, ConfTr +$\mathcal{L}_{\text{class}}$	100	0.01	1	0.5	1
EMNIST, ConfTr	100	0.01	1	0.01	1
EMNIST, ConfTr +$\mathcal{L}_{\text{class}}$	100	0.01	1	5	1
Fashion-MNIST, ConfTr	100	0.01	0.1	0.01	0
Fashion-MNIST, ConfTr +$\mathcal{L}_{\text{class}}$	100	0.01	0.1	0.5	1
CIFAR10, fine-tune ConfTr	500	0.01	1	0.05	0
CIFAR10, fine-tune ConfTr +$\mathcal{L}_{\text{class}}$	500	0.05	0.1	1	1
CIFAR10, "extend" ConfTr	100	0.01	1	0.005	0
CIFAR10, "extend" ConfTr +$\mathcal{L}_{\text{class}}$	500	0.05	0.1	0.1	1
CIFAR100, fine-tune ConfTr	100	0.005	1	0.005	0
CIFAR100, fine-tune ConfTr +$\mathcal{L}_{\text{class}}$	100	0.005	1	0.01	1

Table E.2: **Used ConfTr Hyperparameters** with and without $\mathcal{L}_{\text{class}}$ for ThrLP during training and Thr at test time. The hyperparameters for APS at test time might vary slightly from those reported here. The exact grid search performed to obtained these hyperparameters can be found in the text. Note that, while hyperparameters fluctuate slightly, λ needs to be chosen higher when training with $\mathcal{L}_{\text{class}}$. Additionally, and in contrast to Bel, $\kappa = 1$ in Equation (8.6) performs better, especially combined with $\mathcal{L}_{\text{class}}$. Note that dispersion for smooth sorting is fixed to $\epsilon = 0.1$.

[HCNB20][4]. Our l-layer MLPs comprise l *hidden* layers. We use 32, 256, 128, 64 units per hidden layer on Camelyon, WineQuality, EMNIST and Fashion-MNIST, respectively. These were chosen by grid search over $\{16, 32, 64, 128, 256\}$. The baseline models are trained with cross-entropy loss, while ConfTr follows Algorithm 5 and CoverTr follows Algorithm 6. The epochs are listed in Table E.1 and we follow a multistep learning rate schedule, multiplying the initial learning rate by 0.1 after $2/5$, $3/5$ and $4/5$ of the epochs. We use Haiku's default initializer. On CIFAR, we apply whitening using the per-channel mean and standard deviation computed on the training set. On the non-image datasets (Camelyon, GermanCredit, WineQuality), we whiten each feature individually. On MNIST, EMNIST and Fashion-MNIST, the input pixels are just scaled to $[-1, 1]$.

Fine-Tuning on CIFAR: On CIFAR10 and CIFAR100, we train base ResNet-34/ResNet-50 models which are then fine-tuned using Bel, CoverTr or ConfTr. We specifically use a ResNet-34 with only 4 base channels to obtain an accuracy of 82.6%, using only random flips and crops as data augmentation. The rationale is to focus on the results for CP at test time, without optimizing accuracy of the base model. On CIFAR100, we use 64 base channels for the ResNet-50 and additionally employ AutoAugment [CZM$^+$19] and Cutout [DT17] as data augmentation. This model obtains 73.64% accuracy. These base models are trained on 100% of the training examples (without calibration examples). We also consider "extending" the ResNet by training

[4]https://github.com/deepmind/dm-haiku

MNIST: *test* trials, Cover/Ineff for THR			
Method	Acc	Cover	Ineff
Baseline	92.45	99.09±0.2	2.23±0.15
ConfTr	90.38	99.05±0.2	2.14±0.13
ConfTr +$\mathcal{L}_{\text{class}}$	91.14	99.03±0.19	2.09±0.12

MNIST: *Training* trials, Cover/Ineff for THR			
Method	Acc	Cover	Ineff
Baseline	92.4±0.06	99.09±0.8	2.23±0.01
ConfTr	90.2±0.12	99.03±0.22	2.18±0.025
ConfTr +$\mathcal{L}_{\text{class}}$	91.2±0.05	99.05±0.21	2.11±0.028
ConfTr with APS	87.9±4.81	99.09±0.29	5.79±3.1

Table E.3: **Importance of Random Trials:** We report coverage and inefficiency with the corresponding standard deviation across 10 *test* (left) and 10 *training* trials (right). ConfTr was trained using THRLP if not stated otherwise. For test trials, a fixed model is used. Results for training trials additionally include 10 test trials, but the standard deviation is reported only across the training trials. These results help to disentangle the impact of test and training trials. For example, while ConfTr with APS (during training) works in the best case, the standard deviation of 3.1 across multiple training trials indicates that training is *not* stable.

Fashion-MNIST: Ablation for CoverTr and ConfTr											
Method	Baseline			Bel		CoverTr		ConfTr			
Train				THRL		THR	THRLP	THRLP	THRLP	+$\mathcal{L}_{\text{class}}$	
Test	THRL	THR	APS	THRL	THR	THR	THR	THR	APS	THR	APS
Ineff	2.52	2.05	2.36	1.83	1.9	4.03	2.69	1.69	1.82	1.67	1.73
Acc	89.16	89.16	89.16	84.29	84.61	89.23	87.48	88.86	87.43	89.23	88.69

Table E.4: **Ablation for CoverTr and ConfTr on Fashion-MNIST:** Complementary to Table 8.2, we report inefficiency and accuracy for [Bel21] (Bel), CoverTr and ConfTr considering various CP methods for training and testing on Fashion-MNIST. Besides not being able to train CoverTr with THR or APS, the observations on MNIST can be confirmed, i.e., Bel requires THRL (i.e., on logits) to function properly and training CoverTr or ConfTr with APS is difficult.

a 2-layer MLP with 128 units per hidden layer on top of the features (instead of re-initializing and fine-tuning the logit layer). All reported results either correspond to fine-tuned (i.e., linear model on features) or extended models (i.e., 2-layer MLP on features).

Random Training and Test Trials: For statistically meaningful results, we perform random *test* and *training* trials. For test trials, we throw all calibration and test examples together and sample a new calibration/test split for each trial, preserving the original calibration/test composition which is summarized in Table E.1. Additionally, and in contrast to [Bel21], we consider random training trials: For training trials, after hyperparameters optimization on all training examples, we train 10 models with the final hyperparameters on a new training set obtained by sampling the original one with up to 5 replacements. For example, on MNIST, with 55k training examples, we randomly sample 10 training sets of same size with each, on average, containing only ∼68% unique examples from the original training set. As a consequence, our evaluation protocol accounts for randomness at test time (i.e., regarding the calibration set) and at training time (i.e., regarding the training set, model initialization, etc.).

Hyperparameters: The final hyperparameters selected for ConfTr (for THR at test time) on all datasets are summarized in Table E.2. These were obtained using grid search over the following hyperparameters: batch size in $\{1000, 500, 100\}$ for WineQuality, MNIST, EMNIST, Fashion-MNIST and CIFAR, $\{300, 200, 100, 50\}$ on GermanCredit and $\{80, 40, 20, 10\}$ on Camelyon; learning rate in $\{0.05, 0.01, 0.005\}$; temperature $T \in \{0.01, 0.1, 0.5, 1\}$; size weight $\lambda \in \{0.0001, 0.0005, 0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1, 5, 10\}$ (c.f. Equation (5), right); and $\kappa \in \{0, 1\}$ (c.f. Equation (8.6)). Grid search was done for each dataset individually on 100%

Batch Size and Learning Rate									
Batch Size	1000	1000	1000	**500**	500	500	100	100	100
Learning Rate	0.05	0.01	0.005	**0.05**	0.01	0.005	0.05	0.01	0.005
Ineff	2.27	2.24	2.24	2.18	2.18	**2.17**	8.04	7.32	9.66
Acc	89.05	89.18	89.06	90.23	90.22	**90.27**	11.5	22.46	12.13

Size Weight λ							
λ	0.001	0.005	**0.01**	0.05	0.1	1	10
Ineff	2.18	2.18	2.18	2.19	2.19	2.19	**2.16**
Acc	90.2	20.23	90.23	90.2	90.25	90.23	**90.26**

Temperature T							
T	0.01	0.05	0.1	**0.5**	1	5	10
Ineff	2.39	2.23	2.2	2.19	**2.18**	2.2	2.29
Acc	88.54	89.94	90.02	90.24	**90.28**	90.05	89.63

Confidence Level α (during training)				
α	0.1	0.05	**0.01**	0.005
Ineff	8.07	7.23	2.18	**2.17**
Acc	12.88	39.82	**90.23**	89.47

Table E.5: **Hyperparameter Ablation on MNIST:** For ConfTr without $\mathcal{L}_{\text{class}}$, we report inefficiency and accuracy when varying hyperparameters individually: batch size/learning rate, size weight λ, temperature T and confidence level α. While size weight λ and temperature T have insignificant impact, too small batch size can prevent ConfTr from converging. Furthermore, the chosen hyperparameters do not generalize well to higher $\alpha \in \{0.1, 0.05\}$.

of the training examples (c.f. Table E.1). That is, for hyperparameter optimization we did *not* perform random training trials as described above. The best hyperparameters according to inefficiency after evaluating 3 random calibration/test splits were selected, both for THR and APS at test time, with and without $\mathcal{L}_{\text{class}}$.

Table E.2 allows us to make several observations. First, on the comparably small (and binary) datasets Camelyon and GermanCredit, the size weight $\lambda = 5$ is rather high. For ConfTr without $\mathcal{L}_{\text{class}}$, this just indicates that a higher learning rate could be used. Then using $\mathcal{L}_{\text{class}}$, however, this shows that the size loss is rather important for ConfTr, especially on binary datasets. Second, we found the temperature T to have low impact on results, also see Section E.4. On multiclass datasets, the size weight λ is usually higher when employing $\mathcal{L}_{\text{class}}$. Finally, especially with $\mathcal{L}_{\text{class}}$, using "valid" size loss, i.e., $\kappa = 1$, to not penalize confidence sets of size 1, works better than $\kappa = 0$.

E.2 Importance of Random Trials

In Table E.3 we highlight the importance of random training and test trials for evaluation. On the left, we show the impact of trials at test time, i.e., 10 random calibration/test splits, for a fixed model on MNIST. While the standard deviation of coverage is comparably small, usually $\leq 0.2\%$, standard deviation of inefficiency is higher in relative terms. This makes sense as coverage is guaranteed, while inefficiency depends more strongly on the sampled calibration set. The right table, in contrast, shows that training trials exhibit lower standard deviation in terms of inefficiency. However, training with, e.g., APS will mainly result in high inefficiency, on average, because of large standard deviation. In fact, ConfTr with APS or THR at training time results in worse inefficiency mainly because training is less stable. This supports the importance of running multiple training trials for ConfTr.

CIFAR10:				Fine-tuning						"Extend"	
Method	Baselines			Bel	CoverTr	ConfTr				ConfTr	
Train				ThrL	ThrLP	ThrLP	ThrLP	$+\mathcal{L}_{\text{class}}$		ThrLP	$+\mathcal{L}_{\text{class}}$
Test	ThrL	Thr	APS	Thr	Thr	Thr	APS	Thr	APS	Thr	Thr
Ineff	3.92	2.93	3.3	2.93	2.84	2.88	3.05	2.84	2.93	2.89	2.96
Acc	82.6	82.6	82.6	82.18	82.36	82.32	82.34	82.4	82.4	82.3	82.23

CIFAR10: Ensemble Results							
Method	(Ensemble Models)			Ensemble+MLP			+ConfTr
Train							ThrLP
Test	ThrL	Thr	APS	ThrL	Thr	APS	Thr
Avg. Ineff	4.19	3.1	3.48	3.12	2.4	2.77	2.35
Best Ineff	3.74	2.84	3.17	3.0	2.33	2.71	2.3
Avg. Acc	80.65	80.65	80.65	85.88	85.88	85.88	85.88
Best Acc	82.58	82.58	82.58	86.01	86.01	86.01	86.02

EMNIST									
Method	Baselines			Bel		ConfTr			
Train				ThrL	ThrL	ThrLP	ThrLP	$+\mathcal{L}_{\text{class}}$	
Test	ThrL	Thr	APS	ThrL	Thr	Thr	APS	Thr	APS
Ineff	5.07	2.66	4.23	3.95	3.48	2.66	2.86	2.49	2.87
Ineff, α=0.005	9.23	4.1	6.04	–	–	3.37	–	–	–
Ineff, α=0.001	23.89	15.73	19.33	–	–	13.65	–	–	–
Acc	83.79	83.79	83.79	80.69	80.69	77.1	77.43	77.49	78.09

CIFAR100								
Method	Baselines			Bel	ConfTr			
Train				ThrL	ThrLP	ThrLP	$+\mathcal{L}_{\text{class}}$	
Test	ThrL	Thr	APS	Thr	Thr	APS	Thr	APS
Ineff	19.22	10.63	16.62	10.91	10.78	12.99	10.44	12.73
Acc	73.36	73.36	73.36	72.65	72.02	72.78	73.27	72.99

Table E.6: **Inefficiency and Accuracy on Multiclass Datasets:** Results for CoverTr and training a non-linear 2-layer MLP on the ResNet features on CIFAR10, EMNIST and CIFAR100. We report inefficiency and accuracy in all cases, focusing on ConfTr in comparison to Bel. On EMNIST, we additionally consider $\alpha = 0.005, 0.001$ (for the baseline and ConfTr only). ConfTr consistently improves inefficiency of Thr and APS.

E.3 Coverage/Conformal Training Ablation on Fashion-MNIST

Table E.4 presents additional ablation for CoverTr and ConfTr on Fashion-MNIST, complementary to Table 8.2. In contrast to MNIST, Bel is able to improve inefficiency slightly over the baseline using Thr. This also reduces the improvement of ConfTr over Bel on Fashion-MNIST. Nevertheless, we were unable to train CoverTr with Thr or APS on Fashion-MNIST. Overall, the results in Table E.4 support the observations made on MNIST.

E.4 Impact of Hyperparameters

In Table E.5, we conduct ablation for individual hyperparameters of ConfTr with ThrLP and without $\mathcal{L}_{\text{class}}$ on MNIST. The hyperparameters used in our experiments, c.f. Table E.2, are highlighted in **bold**. As outlined in Section E.1, hyperparameter optimization was conducted

on 100% training examples with only 3 random test trials, while Table E.5 shows results using random training *and* test trials. We found batch size and learning rate to be most impactful. While batch sizes 1000 and 500 both work, batch size 100 prevents ConfTr from converging properly. This might be due to the used $\alpha = 0.01$ which might be too low for batch size 100 where only 50 examples are available for calibration during training. Without $\mathcal{L}_{\text{class}}$, the size weight λ merely scales the learning rate and thus has little to no impact. For ConfTr *with* $\mathcal{L}_{\text{class}}$, we generally found the size weight λ to be more important for balancing classification loss $\mathcal{L}$ and size loss Ω in Equation (8.7). Temperature has no significant impact, although a temperature of 0.5 or 1 works best. Finally, the hyperparameters do generalize to a lower confidence level $\alpha = 0.005$. Significantly lower values, e.g., $\alpha = 0.001$, are, however, not meaningful due to the batch size of 500. However, significantly higher confidence levels, e.g., $\alpha = 0.1$ or $\alpha = 0.05$, require re-optimizing the other hyperparameters.

E.5 All Inefficiency Results

Table E.6 shows complementary results for ConfTr on CIFAR10, EMNIST and CIFAR100. On CIFAR10, we also include ConfTr using a 2-layer MLP on top of ResNet features – instead of the linear model used in Section 8.4.2, referred to as "extending". However, inefficiency increases slightly compared to re-initializing and training just the (linear) logit layer. This shows that the smaller inefficiency improvements on CIFAR are not due to the linear model used, but rather caused by the features themselves. We suspect that this is because the features are trained to optimize cross-entropy loss, leaving ConfTr less flexibility to optimize inefficiency. In Table E.7, we consider three binary datasets, i.e., WineQuality, GermanCredit and Camelyon. On binary datasets, ThrL, Thr and APS perform very similar. This already suggests that there is little room for inefficiency improvements. Indeed, ConfTr is not able to improve inefficiency significantly. However, this is partly due to our thorough evaluation scheme: On Camelyon

WineQuality									
Method	Baselines			Bel	CoverTr	ConfTr			
Train				ThrL	ThrLP	ThrLP	ThrLP	$+\mathcal{L}_{\text{class}}$	
Test	ThrL	Thr	APS	Thr	Thr	Thr	APS	Thr	APS
Ineff, α=0.01	1.76	1.76	1.79	1.77	1.81	1.75	1.82	1.74	1.77
Ineff, α=0.05	1.48	1.49	1.53	1.57	1.50	1.51	–	1.52	–
Acc	82.82	82.82	82.82	71.3	81.5	73.8	74.24	73.91	73.91

GermanCredit						
Method	Baselines			Bel	ConfTr	
Train				ThrL	ThrLP	$+\mathcal{L}_{\text{class}}$
Test	ThrL	Thr	APS	Thr	Thr	Thr
Ineff	1.89	1.86	1.90	1.85	1.88	1.77
Acc	74.4	74.4	74.4	72.35	72.81	69.5

Camelyon* α=0.05						
Method	Baselines			Bel	ConfTr	
Train				ThrL	ThrLP	$+\mathcal{L}_{\text{class}}$
Test	ThrL	Thr	APS	Thr	Thr	Thr
Best Ineff	1.41	1.47	1.59	1.25	1.2	1.25
Best Acc	88	88	88	92	91.5	85

Table E.7: **Inefficiency and Accuracy on Binary Datasets.** Experimental results on the binary datasets WineQuality, GermanCredit and Camelyon. While we include APS on WineQuality, we focus on Thr on GermanCredit and Camelyon due to slightly lower inefficiency. However, ThrL, Thr and APS perform very similarly on all tested binary datasets. Generally, ConfTr does not improve significantly over the baseline. * On Camelyon, we report the best results without training trials as sub-sampling the 280 training examples is prohibitively expensive.

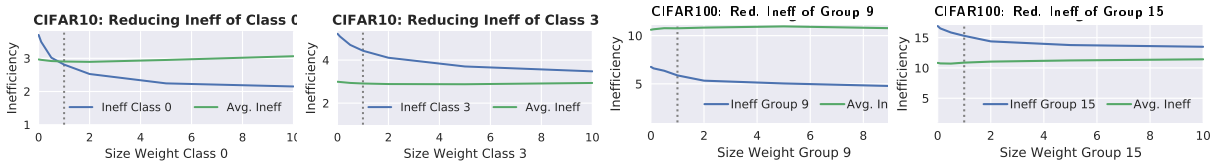


Figure E.1: **Reducing Class- and Group-Conditional Inefficiency on CIFAR.** Results, complementary to Figure 8.3, showing the impact of higher size weights ω in Equation (8.6) for classes 0 and 3 ("airplane" and "cat") on CIFAR10 and coarse classes 9 and 15 ("large man-made outdoor things" and "reptiles") on CIFAR100. ConfTr allows reducing inefficiency (blue) in all cases, irrespective of whether inefficiency is generally above or below average (green).

(α=0.05), we do *not* report averages across all training trials, but the results corresponding to the best model. This is because sub-sampling the training examples is unreasonable given that there are only 280 of them. Thus, Camelyon shows that ConfTr *can* improve inefficiency. On WineQuality or GermanCredit, this is "hidden" in reporting averages across 10 training runs.

E.6 Shaping Class-Conditional Inefficiency on Other Datasets

Figure E.1 and E.2 provide complementary results demonstrating the ability of ConfTr to shape the class- or group-conditional inefficiency distribution. First, Figure E.1 plots inefficiency for individual classes on CIFAR10 and coarse classes on CIFAR100. In both cases, significant inefficiency reductions are possible for high weights ω in Equation (8.6), irrespective or whether the corresponding (coarse) class has above-average inefficiency to begin with. This means that inefficiency reduction is possible for easier and harder classes alike. Second, Figure E.2 plots the relative inefficiency changes, in percentage, possible per-class or group on MNIST, Fashion-MNIST and CIFAR100. For CIFAR100, we show only the first 10 classes for brevity. In all cases, significant inefficiency reductions are possible, at the expense of a slight increases in average inefficiency across all classes. Here, MNIST is considerably easier than Fashion-MNIST: higher inefficiency reductions are possible per class and the cost in terms of average inefficiency increase is smaller. On CIFAR100, inefficiency reductions of 40% or more are possible. This is likely because of the high number of classes, i.e., ConfTr has a lot of flexibility to find suitable trade-offs during training.

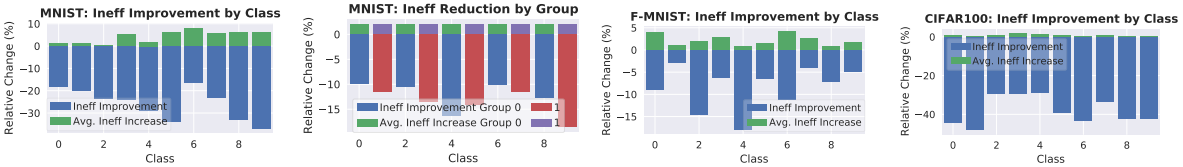


Figure E.2: **Relative Class and Group-Conditional Inefficiency Improvements:** We plot the possible (relative) inefficiency reduction by class or group ("odd" vs "even") on MNIST and Fashion-MNIST. On CIFAR100, we consider the first 10 classes for brevity. In all cases, significant per-class or -group inefficiency reductions are possible.

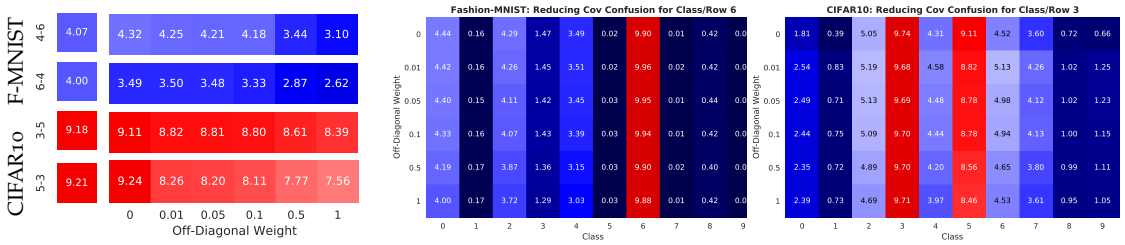


Figure E.3: **Coverage Confusion Changes on Fashion-MNIST and CIFAR10:** *Left:* coverage confusion change when targeting classes 4 and 6 ("coat" and "shirt") on Fashion-MNIST and 3 and 5 ("cat" and "dog") on CIFAR10. The separate cell on the left is the ConfTr baseline which is, up to slight variations, close to $L_{y,k} = 0$. *Middle and right:* coverage confusion for a whole row, i.e., $\Sigma_{y,k}$ with fixed class y and all $k \neq y$. We show row 6 on Fashion-MNIST and 3 on CIFAR10. In both cases, coverage confusion can be reduced significantly.

E.7 Manipulating Coverage Confusion on Other Datasets

Figure E.3 and E.4 provide additional results for reducing coverage confusion using ConfTr. First, supplementary to Figure 8.4, we provide the actual numbers in Figure E.3. In particular, we visualize how the actual coverage confusion entries (left) or rows (right) change depending on the off-diagonal weights $L_{y,k}$. Second, Figure E.4 presents additional results on MNIST and CIFAR10. From these examples it can be seen that reducing coverage confusion is easier on MNIST, reducing linearly with the corresponding penalty $L_{y,k}$. Moreover, the achieved reductions are more significant. On CIFAR10, in contrast, coverage confusion reduces very quickly for small $L_{y,k}$ before stagnating for larger $L_{y,k}$. At the same time, not all targeted class pairs might yield significant coverage confusion reductions.

E.8 Miscoverage Results on Additional Datasets

Table E.8 provides miscoverage results for different settings on MNIST, Fashion-MNIST and CIFAR10. We are able to reduce miscoverage significantly on MNIST and Fashion-MNIST. Only on CIFAR10, considering "vehicles" vs. "animals" as on CIFAR100 in Section 8.4.3, we are unable to obtain significant reductions. While, we are able to reduce $\text{MisCover}_{0\to1}$ slightly from 22.22% to 20%, $\text{MisCover}_{1\to0}$ increases slightly from 16.45% to 16.73% even for high off-diagonal weights used in L. Compared to CIFAR100, this might be due to less flexibility to find suitable trade-offs with only 10 classes. Moreover, miscoverages on CIFAR10 are rather small to begin with, indicating that vehicles and animals do not overlap much by default.

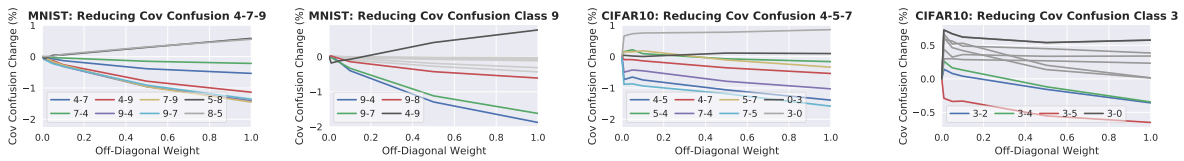


Figure E.4: **Coverage Confusion Reduction on MNIST and CIFAR10:** Controlling coverage confusion for various class pairs. On MNIST, coverage confusion reduction is usually more significant and the reduction scales roughly linear with the weight $L_{y,k}$. On CIFAR10, in contrast, coverage confusion cannot always be reduced for multiple class pairs at the same time (see light gray).

K_0= 2 vs. K_1= Others			
MNIST		MisCover ↓	
Method	Ineff	0→1	1→0
ConfTr	2.11	49.68	14.74
L_{K_0,K_1}=1	2.15	36.63	17.42
L_{K_1,K_0}=1	2.09	51.54	7.62

K_0= Even vs. K_1= Odd			
MNIST		MisCover ↓	
Method	Ineff	0→1	1→0
ConfTr	2.11	38.84	38.69
L_{K_0,K_1}=1	2.16	29.36	49.08
L_{K_1,K_0}=1	2.09	44.3	26.08

K_0= 6 ("shirt") vs. K_1= Others			
F-MNIST		MisCover ↓	
Method	Ineff	0→1	1→0
ConfTr	1.67	80.28	20.93
L_{K_0,K_1}=1	1.70	72.58	25.81
L_{K_1,K_0}=1	1.72	81.18	17.66

K_0= "vehicles" vs. K_1= "animals"			
CIFAR10		MisCover ↓	
Method	Ineff	0→1	1→0
ConfTr	2.84	22.22	16.45
L_{K_0,K_1}=1	2.92	20.00	22.69
L_{K_1,K_0}=1	2.87	24.76	16.73

Table E.8: **Miscoverage on MNIST, Fashion-MNIST and CIFAR10:** We present inefficiency and miscoverage for various cases: On MNIST, we consider 2 vs. other classes as well as even vs. odd classes. In both cases, miscoverage can be reduced significantly. As in Table 8.5, however, reducing $\text{MisCover}_{0\to1}$ usually increases $\text{MisCover}_{1\to0}$ and vice-versa. On Fashion-MNIST, we consider 6 ("shirt") vs. other classes. Only on CIFAR10, considering "vehicles" vs. "animals", miscoverage cannot be reduced significantly. In particular, we were unable to reduce $\text{MisCover}_{1\to0}$.

E.9 Additional Results on Binary Datasets

Figure E.5 shows results complementing Figure 8.5 (right). Specifically, we show that reducing inefficiency for class 1 ("good wine") is unfortunately not possible. This might also be due to the fact that class 1 is the majority class, with ∼63% of examples. However, in addition to improving coverage conditioned on class 0, we are able to reduce coverage confusion $\Sigma_{0,1}$, c.f. Section 8.3.3. We found that these results generalize to GermanCredit, however, being less pronounced, presumably because of significantly fewer training and calibration examples.

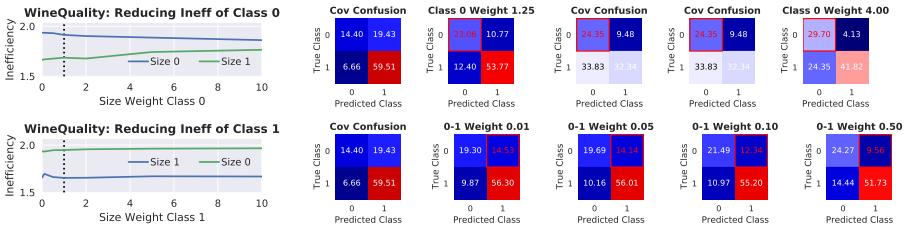


Figure E.5: **Manipulating Inefficiency and Coverage Confusion on WineQuality:** Complementing Figure 8.5 (right), we plot the possible inefficiency reduction for class 1 ("good wine", left) and full coverage confusion matrices for increased $L_{0,0} > 1$ and $L_{1,0} > 0$ (right, top and bottom, respectively). While we can reduce inefficiency for class 0 ("bad wine"), this is not possible for class 1. However, class-conditional coverage for class 0 can be improved significantly and we can reduce coverage confusion $\Sigma_{0,1}$.

E.10 Effect of Conformal Training on Class-Conditional Inefficiency and Coverage Confusion

Figure E.6 shows that standard ConfTr (without $\mathcal{L}_{\text{class}}$) does not have a significant influence on the class-conditional inefficiency distribution compared to the baseline. Similarly, ConfTr with $\mathcal{L}_{\text{class}}$ and identity loss matrix $L = I_K$ does not influence coverage confusion besides reducing overall inefficiency. Specifically, on MNIST, Fashion-MNIST and CIFAR10, we show the class-conditional inefficiency distribution (left) as well as the coverage confusion matrices (middle and right) for the baseline and ConfTr. On the left, we consider ConfTr without $\mathcal{L}_{\text{class}}$, and on the right with $\mathcal{L}_{\text{class}}$. As can be seen, only an overall reduction of inefficiency is visible, the distribution of $\text{Ineff}[y]$, c.f. Equation (8.9), across classes y remains roughly the same. For coverage confusion Σ from Equation (8.10), the same observation can be made, i.e., an overall reduction of inefficiency also reduces confusion, but the spatial pattern remains the same.

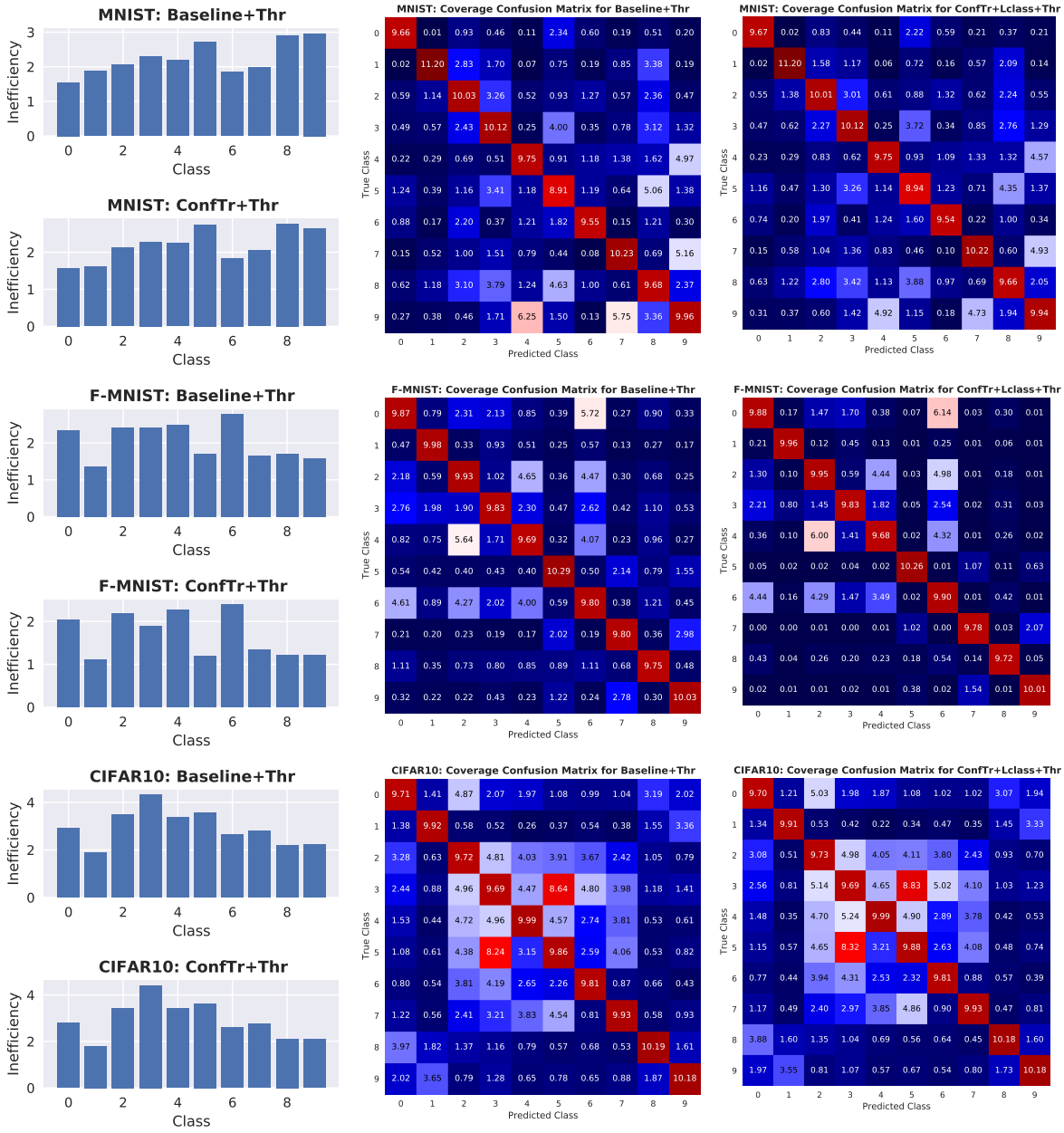


Figure E.6: **Class-Conditional Inefficiency and Coverage Confusion:** Comparison between baseline and ConfTr regarding class-conditional inefficiency and coverage confusion Σ, c.f. Section 8.3.3. For the inefficiency comparison, we consider ConfTr *without* $\mathcal{L}_{\text{class}}$, while for coverage confusion, ConfTr was trained with $\mathcal{L}_{\text{class}}$. As ConfTr reduces overall inefficiency quite significantly on MNIST and Fashion-MNIST, class-conditional inefficiency is also lower, on average. But the distribution across classes remains similar. The same holds for coverage confusion, where lower overall inefficiency reduces confusion across the matrix, but the "pattern" remains roughly the same. On CIFAR10, ConfTr does not improve average inefficiency significantly, such that the confusion matrix remains mostly the same.