GHENT UNIVERSITY

FACULTY OF SCIENCES

Improving the robustness of deep neural networks to adversarial perturbations

Jonathan Peck

Supervisors:
Prof. Dr. Yvan Saeys
Prof. Dr. Ir. Bart Goossens

Academic year 2021-2022

Dissertation submitted in fulfillment of the requirements for the degree of Doctor of Computer Science

Dedicated to the memory of my father,

Joris Peck (1955 - † 2015)*

Permission for use of content

The author gives permission to make this dissertation available for consultation and to copy parts of this dissertation for personal use.

In the case of any other use, the limitations of the copyright have to be respected, in particular with regard to the obligation to state explicitly the source when quoting results from this disseration.

Jonathan Peck

November 2022

Acknowledgements

This has been a long journey, and many people deserve my thanks. I will try my best to list them all, but should anyone feel left out, they can get in touch with me and I will buy them a beer or one of those delicious artisanal lemonades they sell at knol&kool in Ghent.

First and foremost, I wish to thank my promotors, Yvan Saeys and Bart Goossens, for believing in me and helping me all these years. Scientific research is an incredibly uncertain enterprise, where you might work on an idea for months at a time only to have it not bear any worthwhile fruit at all. This happened several times during my Ph.D., and without your encouragement and guidance I'd have given up long ago. Thank you for giving me the freedom to pursue my interests wherever they led me. Many thanks as well to the members of the examination board: prof. Kris Coolsaet, dr. Daniel Peralta Cámara, prof. Tijl De Bie, prof. Aleksandra Pizurica and prof. Sepp Hochreiter.

I would also like to thank the past and current members of the DAMBI group at the VIB. I first joined DAMBI in 2016 when I did my Master's thesis, and I've seen quite a few people come and go. In particular, I thank Sarah Vluymans for being an excellent mentor and valued colleague in the first few years of my Ph.D; we still miss you at the Sterre. Thanks also to Arne Gevaert for your help with the AI courses and your dry wit that kept things entertaining even during the COVID years. Thanks Helena Todorov, for taking over the ML courses when Sarah left; I never could have done it on my own. Thanks Robin Vandaele, Joris Roels, Maxim Lippeveld, Daniel Peralta, Benjamin Rombaut, Ruth Seurinck, Sofie Van Gassen, Robin Browaeys, Artuur Couckuyt, Louise Deconinck, Annelies Emmaneel, David Novak, Lotte Pollaris, Katrien Quintelier, Lauren Theunissen, Yentl Van den Berghe, Robrecht Cannoodt and Wouter Saelens. All great people and great scientists.

Of course, I also thank all of the past and present TWIST people at the Sterre. Thanks Robbert Gurdeep Singh for keeping me company all this time. We were students of Computer Science together, we started our Ph.D. together (sort of, I'm sorry) and we shared the same office for years, but now it seems our roads must diverge. You really brightened up the place, and S9 will never be the same without you. Thanks Dieter Mourisse and Pieter Verschaffelt, fellow teaching assistants for Computergebruik. It would have been impossible to teach that course without you. Thanks also to Toon Baeyens, Rien Maertens, Steven Van Overberghe, Charlotte Van Petegem, Niko Strijbol, Heidi Van den Camp, Felix Van der Jeugt, Tom Lauwaerts, Domien Craens.

Thanks Martine De Cock for having me at the University of Washington Tacoma. Your hospitality and love for your students made all the difference. Thanks Anderson Nascimento, for our interesting discussions and collaboration. Thanks Andrey Kolobov, for showing me around the

Microsoft campus and for the warm welcome in Seattle.

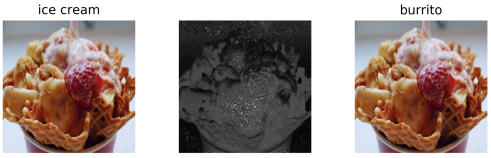
Finally, I wish to thank a few people in my personal life. My parents, first of all, for giving me both the moral and financial support I needed to survive university. Although my father is no longer here to witness it, I know he would be proud. Thanks Harm Delva, Axelle De Bruyne, Jeroen Van Wauwe, Robin Van den Broeck, Michiel De Witte and Sam Adriaensen for being there when I needed friends. Thanks Marieke Lavaert and Hannah Defoer for commiserating with me over academic life. Thanks WiNA and Nemesis for being the best student clubs in all of Ghent.

Special thanks go to Orla Martens, my best and most trusted friend without whom I would never have gotten this far in life. You made me who I am today, and I wouldn't have had it any other way. I'll be in your debt forever. Special thanks also to Jonas Vermeersch, the love of my life who always stayed by my side. It's challenging to have a relationship with a Ph.D. student under the best of circumstances, yet you continued to support me even through the COVID years as I struggled to keep things together. I don't know how you manage it, but I love you, you fabulous fool. 大好き。

Jonathan Peck

September 2022

Summary



Over the past decade, artificial neural networks have ushered in a revolution in science and society. Nowadays, neural networks are applied to various problems such as speech recognition on smartphones, self-driving cars, malware detection and even assisting doctors in making medical diagnoses. Often, neural networks achieve accuracy scores that rival or even surpass human domain experts. It is all the more surprising, then, that these same networks can be misled by minor manipulations that are invisible to the human eye. A neural network trained to identify lung cancer from MRI images, for example, can come to an incorrect diagnosis when a single pixel in the image is deliberately manipulated in a very specific way. Despite the fact that these perturbations are of no consequence to the underlying task and often would go unnoticed by human experts, neural networks tend to be incredibly sensitive to them. Developing defense methods which make our models resilient to such attacks therefore becomes paramount. In this work, I propose four methods that can be employed under different circumstances to protect systems based on artificial intelligence against adversarial attacks.

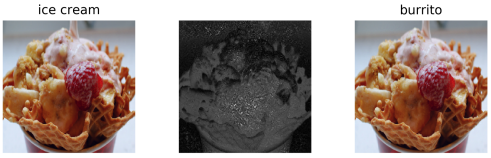
Three out of four methods proposed in this work are based on conformal prediction, a family of statistical methods for developing calibrated machine learning algorithms. The core idea underlying these defenses is the observation that modern deep neural networks tend to be highly over-confident in their predictions, leading to much higher class probabilities than what is justified based on the data. By making use of existing conformal prediction algorithms, we are able to better calibrate the predictions of the models and detect when adversarial manipulation may have taken place. Our first defense, the **binary IVAP defense**, accomplishes this by applying the inductive Venn-ABERS predictor (IVAP) to existing neural networks for binary classification and thresholding the uncertainty of the IVAP. If the uncertainty is too high, we label the sample as suspicious. The binary IVAP defense is easy to tune, having only one scalar hyperparameter (the uncertainty threshold), and very efficient. Our experiments show that it is also highly effective at detecting adversarial manipulation. Subjecting the binary IVAP defense to an adaptive attack shows that, in order to fool it reliably, we must generate perturbations that are much larger and much more visible than prior methods. These results illustrate the enormous potential for conformal prediction methods in the field of adversarial machine learning.

Our second defense, the **MultIVAP**, extends the binary IVAP to the multiclass setting. The algorithm is constructed in such a way that the favorable statistical guarantees of the original IVAP are preserved. The MultIVAP can take any existing classifier and transforms it into a multi-probabilistic predictor, *i.e.*, the algorithm uses the underlying classifier to determine a *set* of possible labels at some user-specified significance level. Like the binary IVAP, there is only one hyperparameter to tune, namely the significance level which determines an upper bound on the long-term error rate. We find that the method is reasonably efficient, incurring an overhead that is roughly linear in the number of classes. It also exhibits high robustness against oblivious as well as adaptive attacks. As an additional feature, the MultIVAP can return empty prediction sets, indicating that the model is too unreliable on the given data to output any label at all.

Our third defense, the **CANN detector**, is based on the general conformal prediction algorithm instead of the specialized IVAP method. It makes use of the PICE algorithm to estimate the principal inertia components of the data set, which can be used to compute a non-conformity measure to instantiate the conformal prediction algorithm. The CANN detector exhibits high robustness against oblivious and adaptive attacks and compares favorably to prior methods both in terms of computational efficiency and detection rates of the adversarial examples. Like the MultIVAP, it can be used to detect adversarial manipulation for arbitrary classifiers. It is more difficult to implement, however, as it requires the user to specify an additional neural network architecture to estimate the principal inertia components. This additional complexity does allow more freedom to optimize the defense, and it only needs to happen once for each data set since the principal inertia components depend only on the data itself.

Finally, our fourth defense, the **adversarial density filter (AdvDF)**, is not based on conformal prediction but rather on the continuity properties of auto-encoders. By pre-processing the data using a neural network structured like a variational auto-encoder, where the latent representations are subjected to Gaussian noise before reconstruction, we can obtain bounds on the robust error of the resulting classifier that increase only sub-linearly in the perturbation budget. This is in contrast to most prior work, where such bounds often increase at least linearly or quadratically with the budget. The error bounds depend on the continuity properties of the encoder module, which must be uniformly continuous with small modulus of continuity. This is also more favorable than prior work, where often the entire auto-encoder or even the entire model had to be constrained to be Lipschitz continuous. Uniform continuity is much less restrictive than Lipschitz continuity, and we only need to constrain the encoder, not the entire architecture. The AdvDF exhibits high robustness against oblivious as well as several adaptive attacks. It also compares very favorably to randomized smoothing, the current state of the art in certified robust defense. The AdvDF often matches or outperforms randomized smoothing in terms of robust accuracy, and it is orders of magnitude faster as it does not require the computationally burdensome sampling step that randomized smoothing needs.

Samenvatting



Artificiële neurale netwerken hebben het afgelopen decennium een revolutie ontketend in de wetenschap en de maatschappij. Tegenwoordig vinden zij toepassingen in uiteenlopende domeinen zoals stemherkenning op een smartphone, zelfrijdende auto's, malwaredetectie en zelfs het bijstaan van artsen in medische diagnoses. Vaak bereiken neurale netwerken accuraatheden die menselijke experts evenaren of overstijgen. Dit maakt het dan ook des te verrassender dat deze zelfde netwerken misleid kunnen worden door kleine manipulaties die meestal onzichtbaar zijn voor het menselijk oog. Een neuraal netwerk dat bijvoorbeeld getraind is om longkanker te identificeren op basis van MRI-beelden, kan een verkeerde diagnose stellen wanneer één enkele pixel in het beeld doelbewust gemanipuleerd wordt. Ondanks het feit dat dergelijke manipulaties van geen enkel belang zijn voor de onderliggende taak en vaak helemaal niet opgemerkt zouden worden door menselijke experts, blijken neurale netwerken hiervoor bijzonder gevoelig. Het ontwikkelen van verdedigingsmethodes die onze modellen resistent maken tegen dergelijke aanvallen dringt zich dan ook op. In dit werk stel ik vier methodes voor die onder verschillende omstandigheden gebruikt kunnen worden om systemen gebaseerd op artificiële intelligentie te beschermen tegen malafide manipulaties.

Drie van de vier methods die ik voorstel in dit werk, zijn gebaseerd op *conformal prediction*. Dit is een familie van statistische methodes waarmee men gecalibreerde machine learning algoritmen kan construeren. Het kernidee dat aan de basis ligt van deze methodes, is de observatie dat moderne neurale netwerken slecht gecalibreerd zijn en vaak probabiliteiten overschatten. Door gebruik te maken van conformal prediction, kunnen we deze modellen beter calibreren en detecteren wanneer de data gemanipuleerd is geweest. Onze eerste methode, de **binaire IVAP**, verwezenlijkt dit door het inductive Venn-ABERS predictor (IVAP) algoritme toe te passen op binaire classificatiemodellen en de onzekerheid te begrenzen. Indien de uitvoer te onzeker is, markeren we de invoer als verdacht. De binaire IVAP is zeer makkelijk af te stellen, aangezien het slechts een enkele scalaire hyperparameter heeft (de onzekerheidsgrens), en is zeer efficiënt. Onze experimenten tonen aan dat het ook heel effectief is in het detecteren van manipulatie. Als we de IVAP onderwerpen aan een adaptieve aanval, zien we tevens dat de methode enkel omzeild kan worden door perturbaties toe te voegen aan de data die veel groter zijn dan wat men typisch beschouwt in de literatuur. Deze resultaten tonen aan dat conformal prediction een

groot potentieel heeft in adversarial machine learning.

Onze tweede methode, de **MultIVAP**, breidt de binaire IVAP uit naar de multiclass setting. Het algoritme wordt zodanig geconstrueerd dat de gunstige statistische garanties van de IVAP bewaard blijven. De MultIVAP kan eender welke bestaande classifier nemen en transformeren naar een multi-probabilistische predictor, zodat het algoritme een *verzameling* van labels geeft die allemaal mogelijk zijn op een bepaald significantieniveau. Zoals de binaire IVAP is er slechts één hyperparameter die we moeten afstellen, namelijk het significantieniveau dat een bovengrens vormt op de foutmarge op lange termijn. We ondervinden dat de methode redelijk efficiënt is, met een overhead die ongeveer lineair is in het aantal klassen. De robuustheid tegen bestaande en adaptieve aanvallen is ook hoog. Als bijkomstig feature kan de MultIVAP tevens voorspellingen weigeren door een lege verzameling terug te geven, in welk geval het model te onbetrouwbaar is op het gegeven sample om eender welke voorspelling te doen.

Onze derde methode, de **CANN detector**, is gebaseerd op het algemene conformal prediction algoritme in plaats van de gespecialiseerde IVAP-methode. Het maakt gebruik van het PICE-algoritme om de principal inertia components van de dataset te schatten, die gebruikt kunnen worden om een maat van non-conformiteit te berekenen waarmee we het conformal prediction algoritme kunnen instantiëren. De CANN detector heeft een hoge robuustheid tegen bestaande en adaptieve aanvallen en is competitief met bestaande methodes in termen van computationele efficiëntie en detectie van perturbaties. Net zoals de MultIVAP kan de CANN detector gebruikt worden voor eender welke classifier. Het is echter moeilijker om te implementeren, omdat men een bijkomstig neuraal netwerk moet ontwikkelen om de principal inertia components te schatten. Deze bijkomende complexiteit laat wel meer vrijheid toe om de methode te optimaliseren, en hoeft slechts eenmalig te gebeuren voor elke dataset aangezien de principal inertia components enkel afhankelijk zijn van de gebruikte data.

Tenslotte is onze vierde methode, de **adversarial density filter** (**AdvDF**), niet gebaseerd op conformal prediction maar op de continuïteitseigenschappen van auto-encoders. Door de data op voorhand te verwerken met een neuraal netwerk dat een gelijkaardige structuur heeft als een variational auto-encoder, waar de latente representaties onderworpen worden aan Gaussiaanse ruis alvorens gereconstrueerd te worden, kunnen we bovengrenzen bewijzen op de fout die het model zal maken onder de invloed van adversarial perturbaties die slechts sub-lineair toenemen in het perturbatiebudget. Dit is in tegenstelling tot de meeste bestaande methodes, waar de bovengrenzen vaak lineair of kwadratisch toenemen met het budget. De bovengrenzen zijn afhankelijk van de continuïteitseigenschappen van de encodermodule, die uniform continu moet zijn met kleine modulus van continuïteit. Dit is ook gunstiger dan bestaand werk, waar vaak de gehele auto-encoder of zelfs het volledige model Lipschitz-continu moest zijn. Uniforme continuïteit is veel minder beperkend dan Lipschitz-continuïteit, en wij hoeven enkel de encodermodule te beperken, niet het volledige netwerk. De AdvDF vertoont hoge robuustheid tegen bestaande en adaptieve aanvallen. We zijn ook in staat om de accuraatheid van randomized smoothing te behalen of vaak zelfs te overtreffen, terwijl onze methode verschillende grootteordes sneller is omdat het niet de computationeel intensieve samplingstap nodig heeft die randomized smoothing vereist.

Contents

1 Introduction **1**

1.1 Deep neural networks . . . 6
1.1.1 Basic structure . . . 6
1.1.2 Activation functions . . . 7
1.1.3 Convolutional neural networks . . . 9
1.1.4 Transformer networks . . . 11
1.1.5 Training algorithms . . . 11
1.1.6 Initialization strategies . . . 13
1.1.7 Further reading . . . 15
1.1.8 The importance of depth and width . . . 17
1.2 Rise to power . . . 18
1.3 Reaching the limits . . . 18
1.4 Social (ir)responsibility . . . 23
1.5 A Potemkin village . . . 25
1.6 Objectives of this thesis . . . 27
1.6.1 Guidelines for practitioners . . . 28
1.6.2 On the relevance of adversarial robustness . . . 30

2 Robustness of deep neural networks **33**

2.1 Shortcut learning . . . 33
2.2 Formal problem statement . . . 37
2.3 Adversarial attacks . . . 40
2.3.1 L-BFGS . . . 42
2.3.2 Fast gradient sign . . . 43
2.3.3 Projected gradient descent . . . 43
2.3.4 Mimicry attack . . . 44
2.3.5 The Carlini-Wagner attacks . . . 45
2.3.6 Randomized gradient-free attack . . . 47
2.3.7 Universal adversarial perturbations . . . 48
2.3.8 Other black-box attacks . . . 50
2.3.9 Real-world attacks . . . 50
2.4 Adversarial defenses . . . 52
2.4.1 Adversarial training . . . 53
2.4.2 Defensive distillation . . . 57

2.4.3 Randomized smoothing . . . 59
2.4.4 Mahalanobis distance-based detector . . . 60
2.4.5 Certifiably robust variational auto-encoders . . . 60
2.4.6 Combination therapies . . . 61
2.4.7 Evaluation and benchmarking . . . 62
2.5 Theoretical results . . . 64
2.5.1 The linear model . . . 65
2.5.2 Boundary tilting and the linearity hypothesis . . . 66
2.5.3 Geometrical perspectives . . . 67
2.5.4 Continuity properties . . . 70
2.5.5 Hardness results . . . 71
2.5.6 Detection vs robust classification . . . 72
2.5.7 Robustness certification . . . 72
2.6 Historical notes . . . 75

3 Conformal prediction 77
3.1 Background on conformal prediction . . . 79
3.2 The general conformal prediction algorithm . . . 81
3.3 Inductive Venn-ABERS predictors . . . 82
3.3.1 Detecting adversarial manipulation using IVAPs . . . 84
3.3.2 Data sets . . . 86
3.3.3 Adaptive adversarial attack . . . 87
3.3.4 Experiments . . . 88
3.3.5 Ablation study . . . 93
3.3.6 Comparison to other methods . . . 96
3.3.7 Conclusions . . . 98
3.4 Case study: DGA detection . . . 99
3.4.1 Data sets . . . 100
3.4.2 Models . . . 101
3.4.3 Results . . . 102
3.4.4 Conclusions . . . 103

4 The MultIVAP conformal predictor 107
4.1 Formal construction . . . 109
4.2 Computational complexity . . . 112
4.3 Choosing the calibration set . . . 112
4.4 Example for the binary case . . . 112
4.5 Adaptive attack . . . 115
4.6 Experiments . . . 116
4.7 Conclusions . . . 124

5 The CANN detector 129
5.1 Correspondence analysis . . . 130
5.2 Conformal CANN detector . . . 131
5.3 Adaptive attack . . . 133
5.4 Experiments . . . 134
5.4.1 Defense-oblivious attack . . . 137
5.4.2 Defense-aware attack . . . 138

5.5 Discussion . . . 140
5.6 Comparison to other methods . . . 141
5.7 Conclusions . . . 144

6 Adversarial density filtering **147**
6.1 Preliminaries . . . 147
6.2 The Adversarial Density Filter (AdvDF) . . . 149
6.2.1 Computing the risk bounds . . . 152
6.2.2 Robustness evaluation . . . 153
6.2.3 Black-box attack . . . 155
6.3 Experiments . . . 156
6.3.1 Data sets and models . . . 156
6.3.2 Results . . . 157
6.3.3 Comparison to randomized smoothing . . . 161
6.4 Theoretical results . . . 162
6.4.1 Proof of theorem 6.1 and corollary 6.2 . . . 162
6.4.2 Relationship to Pinot et al. [2021] . . . 165
6.4.3 Relationship to Barrett et al. [2021] . . . 165
6.4.4 Generalization to the exponential family . . . 167
6.5 Additional figures . . . 168
6.5.1 Examples of reconstructed inputs . . . 169
6.5.2 Robustness curves . . . 170
6.5.3 Calibration curves . . . 175
6.5.4 Performance on CIFAR-10-C . . . 176
6.5.5 Theoretical assumptions . . . 178
6.5.6 Hyperparameters . . . 181
6.5.7 Computational efficiency . . . 182
6.6 Conclusions . . . 182

7 Conclusions **185**
7.1 Summary of contributions . . . 185
7.2 Future research directions . . . 187
7.2.1 Explainable AI . . . 188
7.2.2 Moving beyond the toy problem . . . 188
7.2.3 Benchmarks . . . 194
7.2.4 Lightweight solutions . . . 196
7.2.5 Conformal prediction . . . 197

Bibliography **199**

Epilogue **222**

Chapter 1

Introduction

> "You are wise," he said. "If it is so," I said, "it is only because I have been fool enough for a hundred lifetimes."
>
> *Circe*
> Madeline Miller

This thesis focuses on the fragility of modern artificial intelligence (AI) systems, in particular *deep neural networks* (DNNs). Despite the astounding progress that has been made in AI over the past decade, we find that even our most sophisticated AI systems are actually very brittle: the tiniest, most inconsequential change to the environment might cause the system to go haywire. Consider a few concrete examples:

- We can design AI systems to be very adept at simple computer games, such as Atari Breakout (see figure 1.1). However, regardless of how well these AI models perform on the original game, as soon as we make certain inconsequential changes (such as altering the background color or moving certain UI elements around a little), the AI breaks down completely [Behzadan and Munir, 2017].
- We can develop image recognition systems that can identify objects in real-life images with high accuracy, such as vision systems for autonomous vehicles. However, these systems are easily tricked into misidentifying objects that have been slightly manipulated. For instance, an autonomous driving AI can be fooled into misreading traffic signs due to graffiti or specially crafted roadside advertisements that no human would have any issues with [Eykholt et al., 2018, Kong et al., 2020]. An example is shown in figure 1.2.
- State of the art AI systems for language translation can be highly accurate and realistic, unless specific typographical errors are introduced into some of the words. In that case, the translation can become nonsensical or wrong despite the fact that human translators might not even notice the spelling errors [Ebrahimi et al., 2018]. Some examples are shown in table 1.1.

The general problem here is that we can achieve very high performance on specific tasks using deep neural networks, but as soon as the task specification changes ever so slightly, the networks

Figure 1.1: Atari Breakout. The goal of the game is to use the paddle (the red rectangle at the bottom center of the screen) to guide a projectile (the red pixel left of the center of the screen) towards the rainbow-colored blocks at the top of the screen. The game is won when all blocks have been destroyed by the projectile. The player loses if the paddle does not catch the projectile before it hits the bottom of the screen. Modern reinforcement learning algorithms have become very adept at playing such simple games, but they are very brittle: if any of the colors change (*e.g.*, the background, the color of the projectile or that of the paddle), these algorithms tend to break down completely.

can break down completely. What is especially troubling about this state of affairs is that the changes necessary to sabotage state of the art AI systems do not even have to be relevant to the task at hand: changing the background color of an Atari game, altering a handful of pixels in a high-definition picture, introducing a few typos, etc. None of these changes affect the essence of the task the AI is asked to carry out: the exact same rules apply both in the original setting as well as the manipulated one. We would therefore expect that an AI that has learned to properly solve the given task should not have any difficulty in the face of these so-called *adversarial perturbations*, but this turns out not to be the case at all. It begs the question: what have our AI models *actually* learned?

This is not merely an academic exercise; as our society increasingly relies on DNNs to make important decisions, from approving bank loans to hiring job candidates and even the deployment of police forces [O'Neil, 2016], their trustworthiness becomes crucial. It is important, however, to realize exactly *why* adversarial perturbations are, in fact, a real problem. From the moment adversarial examples gained widespread notoriety thanks to the work of Goodfellow et al. [2014], researchers have consistently framed the problem as one of *cybersecurity*: almost every paper in adversarial ML will begin by stressing how dangerous it would be if, for example, a street sign were vandalized so that an autonomous vehicle could no longer recognize it. As Gilmer et al. [2018] correctly point out, this is not a serious threat: one could much more easily *knock the stop sign over* and be done with it instead of relying on sophisticated optimization algorithms to develop highly specific graffiti. No, the main reason why, in my opinion, adversarial examples are a problem is because they call into question the generalization ability of our models. Indeed, the problem with adversarial examples is much more subtle. The real danger is not when the ML system is under attack, as such attacks *should* be countered by many layers of security and additional fail-safe mechanisms. Rather, the danger is that we will be tempted to trust an ML model under "normal" circumstances, when it is not under attack, despite the fact that it appears to be making the correct decisions for entirely wrong reasons. Can we trust an

Figure 1.2: State of the art computer vision systems for self-driving cars think this stop sign is actually a 45 km/h speed limit sign. This confusion is caused by the black and white stickers applied to the sign. Without them, the stop sign is correctly identified. Image taken from Eykholt et al. [2018].

	German input sentence	English machine translation
Source	`Das ist Dr. Bob Childs - er ist Geigenbauer und Psychotherapeut.`	`This is Dr. Bob Childs - he's a wizard maker and a therapist's therapist.`
Adversarial	`Das ist Dr. Bob Childs - er ist Geigenbauer und Psy6hothearpeiut.`	`This is Dr. Bob Childs - he's a brick maker and a psychopath.`
Source	`In den letzten Jahren hat sie sich zu einer sichtbaren Feministin entwickelt.`	`In the last few years, they've evolved to a safe feminist.`
Adversarial	`In den letzten Jahren hat sie sich zu einer sichtbaren FbeminisMin entwickelt.`	`In the last few years, they've evolved to a safe ruin.`

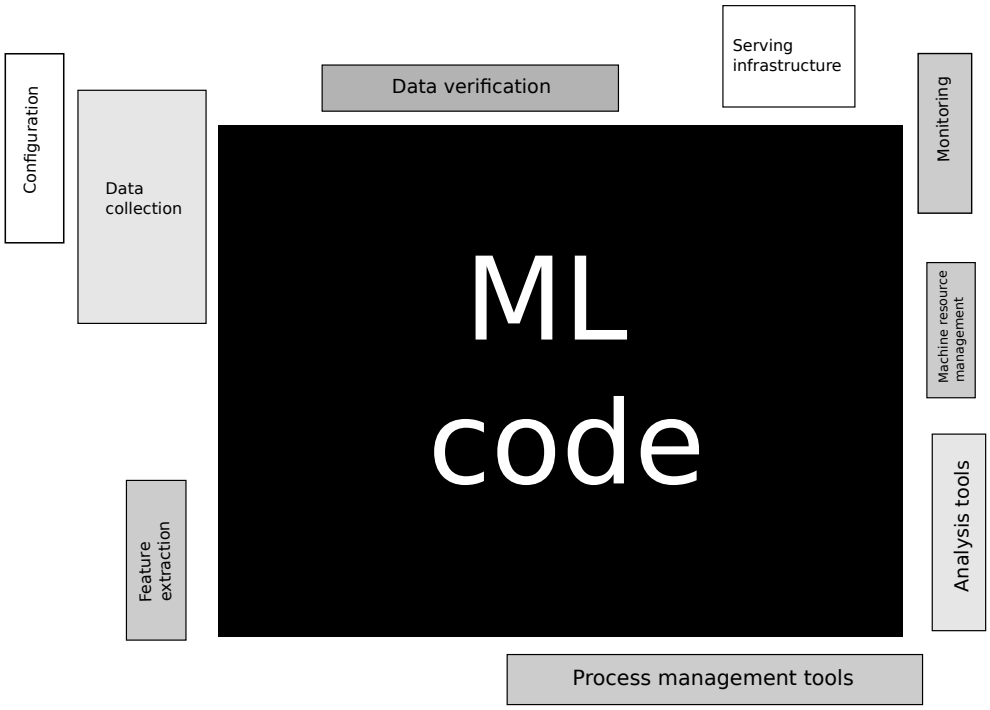
Table 1.1: By introducing specific typographical errors, neural machine translation systems can be tricked into producing erroneous translations. These examples were taken from Ebrahimi et al. [2018].

AI to decide whether someone should be approved for a loan, knowing that this decision may be influenced by the presence or absence of certain letters in their first name? Can we trust AI models to recommend good job candidates if the decision ultimately hinges on the color of a very particular pixel in their resume photo? Can we trust medical diagnoses made by AI systems when the prediction is largely based on the specific make of the imaging device that took the pictures? Clearly, nobody in their right mind would allow any human who operates in this manner to approve loan applications, hire job candidates or make medical diagnoses. Yet this is precisely the state of affairs in modern ML: we routinely develop and deploy all sorts of models to perform these highly sensitive and important tasks that greatly affect human life *despite* the knowledge that these models are so brittle. Although this brittleness *can* of course have security implications in specific cases, one in general has much bigger problems that go well beyond the scope of machine learning if one relies on a mission-critical system that can be dangerously undermined because of the malfunctioning of a single component (in this case, a DNN). It is similar to the difference between hiring a competent person who can sometimes be tricked into making wrong decisions by a malicious adversary (*e.g.*, a social engineer) and hiring someone who is not competent at all and who only got the job because they faked their resume details and bluffed their way through the interview.

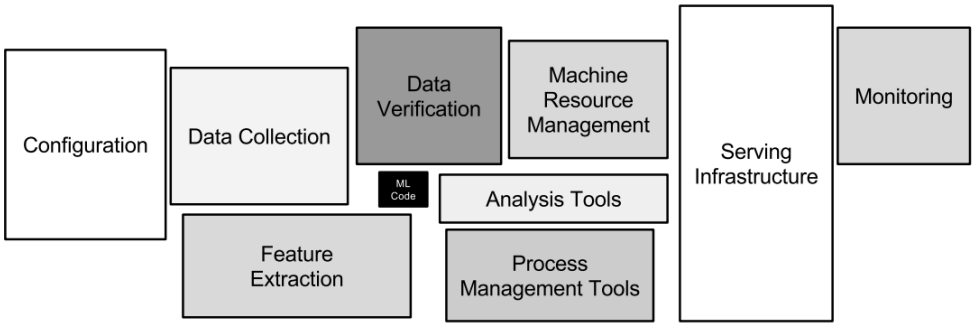
Any well-designed system that utilizes machine learning as part of its architecture should not crucially rely on the ML component alone. The work of Sculley et al. [2015] is particularly relevant in this regard. Figure 1.3b illustrates their main idea: ML code is almost always just a tiny fraction of the overall system, and there are many other components that require much more attention and which *should* provide necessary redundancies in case one of the components fails (such as the ML part). If a failure of just the ML vision system alone causes a self-driving car to become unsafe, then that car is badly designed and it should never be allowed on the road. There should always be redundancies and fail-safes for any mission-critical system so that the failure of a single component does not cause a total breakdown. In that sense, the ML community has developed an inflated sense of self-importance as illustrated in figure 1.3a: underlying many of the adversarial ML papers that cast adversarial examples as a security risk is the tacit assumption that the entire system stands or falls with the integrity of the ML component. This is tantamount to the belief that the ML code is the most important part of the system, and that other components either do not matter or can be safely ignored.

To summarize, contrary to the dominant view within the field, my idea of the adversarial robustness problem is not rooted in cybersecurity at all. Indeed, it is my view that ML should be kept as far away as possible from anything relating to security, as I believe ML to be fundamentally unsafe. Its security should therefore be guaranteed not by using *more* ML, but *less* of it: by encapsulating the ML component in many layers of checks and fail-safes that do not require ML themselves. I instead conceive of the robustness problem as a more general issue of *reliability*: at its core, the fragility of DNNs calls into question the extent to which our ML models have actually learned anything. It suggests that these models may well be inappropriate to use even in the so-called "benign" setting, where the data are not deliberately manipulated. It therefore forms a fundamental obstacle to the adoption of deep learning in many fields where high-stakes decisions must be made. It is one of the major problems we as a field *must* move beyond if we are to make further progress.

In this introductory chapter, I will first give a brief overview of what deep neural networks actually are, why they have become so popular, and what problems we have with them today. I will then zoom in on the specific problem with DNNs tackled in this work, namely their lack of



(a) Expectation: the ML component of the system is the only one that really matters. The other components do not concern us and can be safely brushed aside.



(b) Reality: ML code is usually only a small fraction of the overall system. Engineering an ML system in the real world requires the expertise of many qualified individuals working together. Figure due to Sculley et al. [2015].

Figure 1.3: Expectations of ML practitioners vs the reality of designing ML systems in the real world.

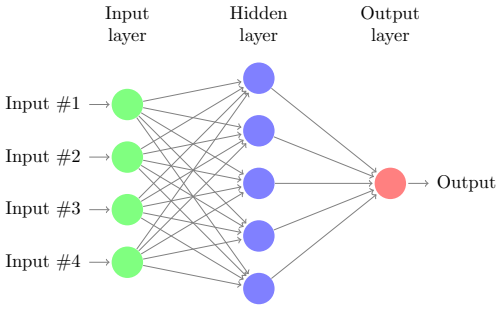


Figure 1.4: Basic schematic of a feed-forward neural network.

robustness. I conclude the chapter with an outline of the thesis structure and a formulation of the concrete research objectives.

1.1 Deep neural networks

Deep neural networks are, at the time of this writing, arguably the single most successful tool ever developed in the field of artificial intelligence (AI). Popularized by the work of Krizhevsky et al. [2012], DNNs have ushered in breakthroughs in a wide range of domains where progress had been relatively stagnant for years. As such, thanks to the development of DNNs, research interest in the field of AI was rekindled after the long "AI winter" of the 1990s, and its popularity has only grown since then. In 2017, Google Brain co-founder Andrew Ng compared the revolutionary potential of AI to that of electricity:[1]

> *Just as electricity transformed almost everything 100 years ago, today I actually have a hard time thinking of an industry that I don't think AI will transform in the next several years.*

The contributions of deep learning to the wider scientific and industrial enterprise have also been recognized by leading academic institutions. In 2018, for example, the Association for Computing Machinery (ACM) presented Yoshua Bengio, Geoffrey Hinton and Yann LeCun with the prestigious A.M. Turing Award, "[f]or conceptual and engineering breakthroughs that have made deep neural networks a critical component of computing."[2]

Despite their undeniable importance, however, the details of how deep neural networks are constructed and trained often remain shrouded in mystery, at least to the general public. As they are also the primary object of study for my research, I devote the next few subsections to the explanation of the basic structure and operation of DNNs.

1.1.1 Basic structure

A simple example of a neural network is shown in figure 1.4. It illustrates the basic structure of any neural network (deep or shallow), namely an input layer followed by one or more "hidden"

[1] `https://www.gsb.stanford.edu/insights/andrew-ng-why-ai-new-electricity`. Accessed 2021-12-02.

[2] `https://amturing.acm.org/award_winners/bengio_3406375.cfm`. Accessed 2021-12-02.

layers, which finally yield the output. This terminology is suggestive: the "hidden" layers are so named because they are the only layers of which the output cannot be directly observed in reality. In a typical (supervised) machine learning problem, we can at least observe pairs of inputs and outputs directly in the data set on which the model is to be trained. However, we do not directly observe what happens in between, *i.e.*, how the input is transformed into the output. For instance, given a picture of a cat, we humans can immediately recognize that it is a picture of a cat and hence `cat` is the appropriate output of this classification problem. However, we cannot directly observe what precise function is computed by our brains in order to turn these visual stimuli into the concept of a cat. The hidden layers of a neural network are meant to capture such transformations. Thus, the actual "magic" happens in the hidden layers, and they constitute the most important part of any neural network. Mathematically, a DNN is typically specified as follows for an input vector $x \in \mathbb{R}^n$:

$$\begin{aligned} h_0(x) &= x, \\ h_1(x) &= g_1(W_1 h_0(x) + b_1), \\ &\vdots \\ h_L(x) &= g_L(W_L h_{L-1}(x) + b_L). \end{aligned} \tag{1.1}$$

The input vector x is successively transformed according to linear functions determined by the matrices $W_1, \ldots, W_L$ and biases $b_1, \ldots, b_L$. The result of each linear transformation is then fed through the appropriate *activation function* $g_1, \ldots, g_L$. These functions are meant to add non-linear behavior to the network; otherwise, the entire thing would simply reduce to one big linear transformation.

1.1.2 Activation functions

Much research has gone into developing good activation functions, since they must balance competing objectives: on the one hand, the activation function must be easy to compute and must have a well-behaved derivative, in order to make gradient-based learning possible; on the other hand, it must be non-linear and complex enough to allow the neural network to learn highly complicated functions.

Indeed, for a long time, a poor choice of activation function was one of the reasons DNNs were hard to train: researchers tended to use the *sigmoid function*,

$$\sigma(x) = \frac{1}{1+e^{-x}},$$

as the activation function for every layer. However, this function leads to a *vanishing gradient problem* when the networks become very deep: when many sigmoid functions are composed together, the gradient of the resulting function can become very unstable [Hochreiter et al., 2001]. In particular, for very large and very small inputs, the sigmoid function saturates and the gradient vanishes. This leads to a stalling of any gradient-based optimization algorithm, since there is no signal with which to update the parameters. A highly desirable property of good activation functions is therefore to maintain useful gradient information regardless of the input and the number of layers. Ideally, the gradient must not be consistently very small (so that the gradient does not vanish after a few layers) nor very large (so that the gradient does not explode). Finding functions which satisfy these desiderata yet also lead to useful DNNs is still an active

area of research, with alternatives to the old sigmoid function being proposed regularly. Some examples of such functions that are used today include:

- The *rectified linear unit* or ReLU [Glorot et al., 2011] and its variants, such as the *leaky ReLU* [Maas et al., 2013] and *parametric ReLU* [He et al., 2015]. These are given by

$$\mathrm{ReLU}(x) = \begin{cases} x & \text{if } x > 0 \\ 0 & \text{otherwise.} \end{cases} \qquad \mathrm{LeakyReLU}(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha x & \text{otherwise.} \end{cases}$$

  Here, α is a non-negative real number. The parametric ReLU has the same functional form as the leaky ReLU, but it includes α as a trainable parameter of the neural network. By contrast, in the case of the leaky ReLU, α is a fixed constant. These rectifier units were key innovations that allowed DNNs to be trained much more successfully, especially when combined with appropriate initialization strategies.

- The *softplus* function, which can be seen as a smooth approximation of the ReLU:

$$\mathrm{softplus}(x) = \log(e^x + 1).$$

- The *exponential linear unit* or ELU [Clevert et al., 2015]:

$$\mathrm{ELU}(x) = \begin{cases} x & \text{if } x \geq 0, \\ \alpha(e^x - 1) & \text{otherwise.} \end{cases}$$

  Here, $\alpha > 0$ is a fixed constant.

- The *scaled exponential linear unit* or SELU [Klambauer et al., 2017]:

$$\mathrm{SELU}(x) = \begin{cases} sx & \text{if } x \geq 0, \\ s\alpha(e^x - 1) & \text{otherwise.} \end{cases}$$

  Here, α and s are constants that are typically set to the following exact values:

$$\alpha = 1.67326324, \qquad s = 1.05070098.$$

  This activation function was proposed by Klambauer et al. in order to create *self-normalizing neural networks* (SNNs). SNNs have the property that their intermediate layers self-normalize, in the sense that the mean activation tends to zero and the variance tends to unity. This eliminates the need for special layers such as batch normalization [Ioffe and Szegedy, 2015] — with their associated pathologies; see Wu and He [2018] — in order to maintain a good gradient signal throughout training.

- The *sigmoid linear unit* (SiLU), also known as the Swish activation function [Elfwing et al., 2018, Hendrycks and Gimpel, 2016, Ramachandran et al., 2017]:

$$\mathrm{SiLU}(x) = x \cdot \mathrm{sigmoid}(x).$$

  Interestingly, this activation function appears to significantly increase the robustness of DNNs when used appropriately [Gowal et al., 2020, Rebuffi et al., 2021].

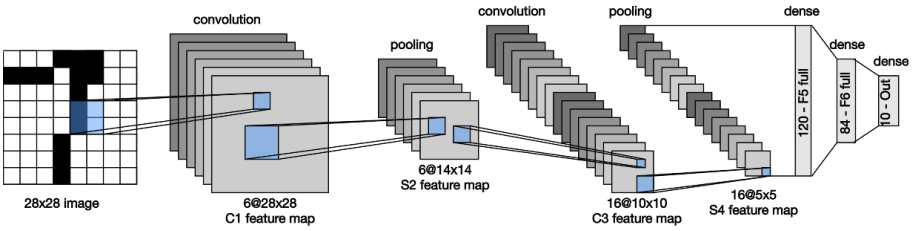


Figure 1.5: Illustration of the LeNet convolutional neural network from LeCun et al. [1989].

- The *softmax* function, which converts a vector of arbitrary real numbers $x \in \mathbb{R}^n$ to a vector of probabilities:

$$\text{softmax}(x) = \frac{\exp(x)}{\sum_{i=1}^{n} \exp(x_i)}.$$

  Here, $\exp(x)$ is computed component-wise:

$$\exp(x) = \begin{bmatrix} e^{x_1} & \dots & e^{x_n} \end{bmatrix}.$$

  Hence, every component of $\text{softmax}(x)$ lies in the interval $[0,1]$ and all the components together sum to unity. Because of these properties, the softmax activation function is typically only used in the final hidden layer in order to produce a categorical random variable.

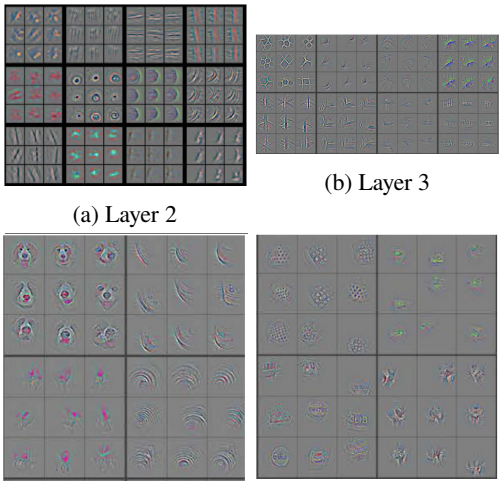
1.1.3 Convolutional neural networks

Aside from the choice of activation function, it is also possible to impose specific constraints on the weight matrices $W_1, \dots, W_L$, which lead to specialized classes of neural networks. The most famous example of this are the *convolutional neural networks* (CNNs), such as AlexNet and Microsoft's residual networks [He et al., 2016, Krizhevsky et al., 2012]. In essence, a CNN is a DNN where one or more of the layers (or even all of them; see Springenberg et al. [2014]) perform *discrete convolutions* instead of general matrix operations. The discrete convolution itself is actually a matrix multiplication, where the weight matrix has the special structure of a *Toeplitz matrix* [Goodfellow et al., 2016]. This is a special type of matrix where each descending diagonal from left to right is constant. That is, A is an $n \times n$ Toeplitz matrix if

$$A_{i,j} = A_{i+1,j+1} = a_{i-j}$$

for constants $a_{-(n-1)}, \dots, a_0, \dots, a_{n-1}$. Such matrices have only $2n-1$ degrees of freedom, as opposed to n^2 for general $n \times n$ matrices. This allows for extremely efficient computation of discrete convolutions on GPUs, which is one of the reasons CNNs became very popular.

Figure 1.5 shows the general structure of a typical CNN. The input, which is often an image encoded as a 2D matrix or 3D tensor, is first processed by a series of convolutional layers. There



(a) Layer 2 (b) Layer 3

(c) Layer 4 (d) Layer 5

Figure 1.6: Visualization of the features learned by a CNN model, provided by the deconvolution method of Zeiler and Fergus [2014].

can also be *pooling* layers, which perform a sort of dimensionality reduction by computing certain summary statistics of groups of pixels. CNNs then typically end in a handful of *dense* layers, which perform general matrix multiplications. It would not be much of an exaggeration to claim that CNNs are almost single-handedly responsible for the current dominance of deep learning methods in machine learning: the performance gains achieved by early CNNs such as AlexNet were so impressive compared to classical methods they essentially caused a sort of paradigm shift in the field. As of 2022, however, CNNs are but one of many possible architectures from which successful deep networks can be created, but they are still very popular, particularly in computer vision.

Despite their remarkable performance, the inner workings of CNNs remained highly mysterious for many years. In an influential early work, Zeiler and Fergus [2014] employed *deconvolutional networks* – an architecture they originally proposed in Zeiler et al. [2011] – to study how the individual layers of a CNN affect the distribution of features. Deconvolutional networks can be trained to essentially perform the inverse operation of any given CNN, *i.e.*, they can map feature representations of a CNN back into the input pixel space. By maximizing the activation of selectively chosen layer outputs, this inverse projection reveals the structures in the input images that are most relevant to a specific layer and feature. Their main result is illustrated in figure 1.6. It suggests that CNNs learn filters which become sensitive to progressively more detailed structures in the input image: the first few layers emphasize certain simple shapes such as lines and circles, whereas deeper layers respond to complex features such as the structure of a dog's face.

Later, Mallat [2016] and more recently Ye [2022] would develop a rigorous mathematical foundation to explain the empirical findings of Zeiler and Fergus in terms of wavelet scattering transforms. It can be proven that such transforms are invariant to translations and diffeomorphisms, which explains their effectiveness in image classification.

1.1.4 Transformer networks

CNNs were very popular for several years in the area of image recognition. In the field of natural language processing (NLP), however, the *transformer* was the dominant class of networks. Originally introduced by Vaswani et al. [2017], transformers quickly established themselves as the new state of the art in NLP and became widely used after 2017. Several landmark results in this vein include the *Bidirectional Encoder Representations from Transformers* or BERT [Devlin et al., 2018] and the *Generative Pre-trained Transformer* (GPT) series of models [Brown et al., 2020, Radford et al., 2018, 2019].

Owing to their massive success in NLP, researchers also tried to apply transformers to image recognition. In a widely celebrated result, Dosovitskiy et al. [2020] showed that transformers can match or even outperform CNNs using fewer computational resources. Although the long-term implications of these findings are not yet clear at the time of this writing, it seems plausible that the field of deep learning will move away from CNNs in the near future in favor of transformer architectures.

1.1.5 Training algorithms

The workhorse of modern ML is the *gradient descent* algorithm [Boyd and Vandenberghe, 2004]. At its core, gradient descent is a very simple and efficient algorithm for optimizing complicated objective functions. Formally, given an objective function $f : \mathbb{R}^n \to \mathbb{R}$ to be minimized over a set of parameters x, gradient descent iteratively computes the following updates:

$$x_{t+1} \leftarrow x_t - \lambda_t \nabla f(x_t). \tag{1.2}$$

Starting from an initial candidate solution x_0, gradient descent produces successive iterates $x_1, x_2, \ldots$ based on the gradient of the objective function f with respect to the parameters as well as non-negative real numbers $\lambda_1, \lambda_2, \ldots$. These are known as the *learning rates*, and must be chosen by the practitioner. How exactly to choose the learning rate at each iteration (also known as the *learning rate schedule*) is a very active area of research, as this choice can be very delicate: gradient descent may not converge at all if the learning rate is chosen poorly.

The motivation behind this algorithm is as follows. Under certain conditions, a function f admits a *Taylor decomposition* [Spivak, 2018]

$$f(x+\boldsymbol{\eta}) = f(x) + \boldsymbol{\eta}^\intercal \nabla f(x) + \mathcal{O}(\|\boldsymbol{\eta}\|^2).$$

Assuming $\boldsymbol{\eta}$ is small in magnitude, we can discard the higher-order terms and obtain the approximation

$$f(x+\boldsymbol{\eta}) \approx f(x) + \boldsymbol{\eta}^\intercal \nabla f(x).$$

If we let $\boldsymbol{\eta} = -\lambda \nabla f(x)$ for some $\lambda > 0$, then

$$f(x+\boldsymbol{\eta}) \approx f(x) - \lambda \|\nabla f(x)\|^2.$$

If this linearization is accurate and the gradient of f at x is non-zero, it holds that $f(x+\boldsymbol{\eta}) < f(x)$. By iterating this procedure, we obtain a series of inequalities

$$f(x_0) > f(x_1) > f(x_2) > \ldots$$

In effect, we are reducing the value of f at each iteration and hence we must reach some (local) minimum of f at some point in time.

Of course, this algorithm depends on a number of assumptions which may not hold in practice, and there are many failure cases where gradient descent does not converge to any useful result at all [Sun, 2019]. As such, many variants of the basic gradient descent algorithm have been proposed for use in DNNs. The most well-known of these are Adam [Kingma and Ba, 2014], Adagrad [Duchi et al., 2011], RMSprop [Hinton et al., 2012], Adadelta [Zeiler, 2012], Adamax [Kingma and Ba, 2014] and Nadam [Dozat, 2016].

To apply these algorithms to a DNN, researchers typically follow the paradigm of *maximum a posteriori estimation* or MAP estimation [Goodfellow et al., 2016, Murphy, 2012]. Specifically, we first collect a data set of samples $S = \{(x_i, y_i) \mid i = 1, \ldots, m\}$ from an unknown data distribution $\mathcal{D}$. For instance, if we want to train a DNN to recognize cats and dogs in an image, we collect a large number of pictures of cats and dogs labeled with their corresponding categories (*i.e.*, `cat` or `dog`). Following Bayes' theorem, the probability that a model θ explains the observed data is given by

$$\Pr[\theta \mid S] = \frac{\Pr[\theta]\Pr[S \mid \theta]}{\Pr[S]}.$$

We aim to find the optimal set of parameters θ^{MAP} that maximizes this probability. Hence, we wish to solve the following optimization problem:

$$\theta^{\text{MAP}} = \max_{\theta} \frac{\Pr[\theta]\Pr[S \mid \theta]}{\Pr[S]}.$$

Since the objective function is independent of the evidence $\Pr[S]$, we can ignore this term:

$$\theta^{\text{MAP}} = \max_{\theta} \Pr[\theta]\Pr[S \mid \theta].$$

Assuming S is a set of independent and identically distributed samples from the data distribution $\mathcal{D}$, we can further decompose the objective as

$$\theta^{\text{MAP}} = \max_{\theta} \Pr[\theta] \prod_{i=1}^{m} \Pr[y_i \mid x_i, \theta].$$

To aid numerical stability, we can take logarithms to obtain

$$\theta^{\text{MAP}} = \max_{\theta} \log \Pr[\theta] + \sum_{i=1}^{m} \log \Pr[y_i \mid x_i, \theta]. \tag{1.3}$$

This way, we avoid multiplying together many (potentially small) numbers, which would lead to numerical underflow [Heath, 2018, Trefethen and Bau III, 1997]. Often, it is considered more natural to formulate (1.3) as a minimization problem instead:

$$\theta^{\text{MAP}} = \min_{\theta} - \underbrace{\log \Pr[\theta]}_{\text{regularization term}} \underbrace{- \sum_{i=1}^{m} \log \Pr[y_i \mid x_i, \theta]}_{\text{log-likelihood loss}}. \tag{1.4}$$

We can decompose this objective function into two distinct components:

1. The *regularization term* $\log \Pr[\theta]$. This term derives from the prior probability distribution on the parameters of the network, $\Pr[\theta]$. It expresses our prior beliefs about which parameters might be "plausible" or "preferable" and which are to be avoided. Typical priors will focus on keeping the magnitude of the weights under control, such as ℓ_p regularization [Murphy, 2012] or orthogonal weight regularization [Brock et al., 2016]. It is also possible to have no regularization at all, which is equivalent to choosing a uniform prior on θ. In that case, the resulting technique is known as *maximum likelihood estimation* or MLE. However, especially for DNNs, this method is prone to *overfitting*, a phenomenon where the DNN memorizes the training data and does not generalize to novel inputs. That said, regularization by itself is not sufficient to keep DNNs from overfitting. In fact, understanding the precise conditions under which DNNs overfit is one of the major open problems of the field [Zhang et al., 2021a].

2. The *log-likelihood loss* $\sum_{i=1}^{m} \log \Pr[y_i \mid x_i, \theta]$. In practice, $\Pr[y_i \mid x_i, \theta]$ is computed by evaluating the neural network on the input x_i using the given parameters θ and then taking its predicted probability for the target class y_i. Minimizing the negative log of this probability amounts to pushing the predicted probability towards 100%, *i.e.*, encouraging the network to associate high probabilities with the correct labels. The hope is that, if we minimize this loss on a finite data set S of sufficiently high quality, then the resulting model will generalize to unseen inputs as well. Understanding the conditions under which ML models generalize from their finite training data sets to unseen inputs is the topic of *learning theory* [Kearns et al., 1994, Shalev-Shwartz and Ben-David, 2014, Vapnik, 1999].

The optimization problem (1.4) can then be solved using any of the aforementioned algorithms by simply plugging in this objective function. That is, we perform gradient descent on the full loss function

$$\mathcal{L}(\theta) = -\log \Pr[\theta] - \sum_{i=1}^{m} \log \Pr[y_i \mid x_i, \theta]$$

with respect to the parameters θ. Almost always, θ will be a set of real-valued vectors and matrices, so the update equation (1.2) is straightforward to apply.

1.1.6 Initialization strategies

An important aspect of neural network training that we have not discussed so far, is the question of how to *initialize* the weights. Perhaps unsurprisingly, as it turns out, this is a delicate operation that requires some care in order for DNNs to properly converge: a bad initialization can prevent the network from learning anything useful at all, even with the most sophisticated optimizers. Intuitively, one would probably expect that a simple random uniform or random normal initialization of the parameters would suffice, but in general this is not the case [Glorot and Bengio, 2010]. The following are some of the most common strategies for initializing the weights that do tend to work well:

- Truncated normal. This strategy samples the weights from a normal distribution, typically with zero mean and a standard deviation of $1/20$. However, the distribution is "truncated:" values that lie more than two standard deviations from the mean (*i.e.*, outside of the interval $[-0.10, 0.10]$) are discarded and re-sampled. I have not been able to find any academic publications that motivate this strategy theoretically or even empirically in any sort of systematic study. Instead, the effectiveness of the truncated normal

distribution appears to be a piece of academic folklore wisdom: many practitioners have observed that trained DNNs tend to have small weights and that small weights avoid saturation of activation functions. The regular normal distribution technically has infinite support and so may result in arbitrarily large values. Truncation is a simple method to prevent this potential issue.

- Glorot normal, also known as Xavier normal [Glorot and Bengio, 2010]. This method draws the initial parameter values from a normal distribution with zero mean and standard deviation
$$\sigma = \sqrt{\frac{2}{N_{\text{in}} + N_{\text{out}}}},$$
where N_{in} and N_{out} refer to the number of inputs and outputs of the neuron.

- Glorot uniform, also known as Xavier uniform [Glorot and Bengio, 2010]. Here, the weights are drawn from a uniform distribution supported on the symmetric interval $[-\ell, \ell]$ where
$$\ell = \sqrt{\frac{6}{N_{\text{in}} + N_{\text{out}}}}.$$

- He normal [He et al., 2015]. This method samples the weights from a normal distribution with zero mean and standard deviation
$$\sigma = \sqrt{\frac{2}{N_{\text{in}}}}.$$

- He uniform [He et al., 2015]. This method samples the weights from a uniform distribution supported on the symmetric interval $[-\ell, \ell]$ where
$$\ell = \sqrt{\frac{6}{N_{\text{in}}}}.$$

- Orthogonal [Saxe et al., 2013]. Here, the weights are initialized to random orthogonal vectors. If the weights form a matrix (as in a standard dense layer), the initialization is done by sampling a random matrix from a normal distribution and then performing a QR decomposition. The parameters are then set to the values of the Q matrix.

The objective of good initialization strategies is to control the magnitude of the values propagated through the network. In most cases, this amounts to controlling the variance of the layer activations. For instance, following the argument by He et al. [2015], we model forward propagation through a neural network with L layers using the representation (1.1) and assume each activation function g_l is a rectifier unit. Then, assuming the weights and inputs are i.i.d. from zero-mean distributions, we have

$$\operatorname{Var}[h_{l+1}(x)] = N_l \operatorname{Var}[W_l h_l(x)] = N_l \operatorname{Var}[W_l] \operatorname{Var}[h_l(x)].$$

With a few additional assumptions, this reduces to

$$\operatorname{Var}[h_{l+1}(x)] = \frac{1}{2} N_l \operatorname{Var}[W_l] \operatorname{Var}[h_l(x)].$$

The final layer then satisfies

$$\mathrm{Var}[h_L(x)] = \mathrm{Var}[h_1(x)] \prod_{l=2}^{L} \frac{1}{2} N_l \,\mathrm{Var}[W_l].$$

The variance of the final layer (and, in fact, all intermediate layers) can then be controlled by guaranteeing

$$\forall l : \frac{1}{2} N_l \,\mathrm{Var}[W_l] = 1,$$

which implies the condition

$$\mathrm{Var}[W_l] = \frac{2}{N_l}.$$

This motivates the He strategy, which samples the weights from a distribution with standard deviation

$$\sqrt{\frac{2}{N_{\text{in}}}}.$$

It is important to realize that, similar to the Glorot strategy, one can in principle use *any* distribution with this standard deviation to initialize the weights according to this line of reasoning. However, practical implementations limit themselves to the uniform and normal distributions, since these are easy to sample from.

1.1.7 Further reading

The particular type of DNN I have discussed here is known as the *feed-forward neural network*, since it consists of a series of successive layers that process the output of the previous layer one after the other. There is another class, known as *recurrent* neural networks, that allow for "loops" in this construction: in a recurrent network, the output of a layer may be fed back into that same layer an arbitrary number of times. Of course, since computation time must be finite on any realistic computer, the number of recursions will be limited to a finite number as well. Therefore, recurrent networks can in principle always be "unrolled" into a standard feed-forward network. I will not be going into detail about recurrent networks here; I refer the interested reader to the excellent introductory textbook of Goodfellow et al. [2016] or the more recent work of Murphy [2022]. These books also cover many other aspects of ML which I cannot discuss here, such as unsupervised learning, auto-encoders, etc.

An interesting and more intuitive perspective of how neural networks work at a high level can be found in Montufar et al. [2014]. The basic ideas are illustrated in figure 1.7. Essentially, the successive layers of a neural network can be interpreted as folding operators on their input spaces, identifying separate neighborhoods in the input with common partitions of the output. This is especially true for ReLU networks, for instance, where each layer identifies a subset of its input with zero (*i.e.*, all inputs which are mapped to negative values by the affine transformation, before the ReLU activation is applied, are clamped to zero). Formally, a map $F : U \to V$ is said to *identify* two neighborhoods $S, T \subseteq U$ if it maps them to a common subset of the output space, *i.e.*, $F(S) = F(T) \subseteq W \subseteq V$. Such a map can be considered as an operator that "folds" its domain such that S and T coincide. Neural networks essentially compose many such folding operators, resulting in a recursive folding of the input space. This means that any function computed on the folded output space of a given layer will apply to all of the collapsed subsets identified by the the

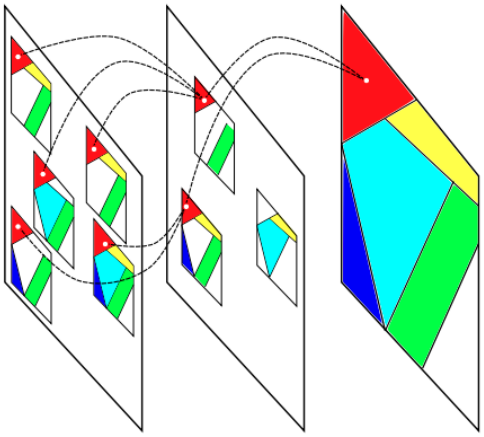


Figure 1.7: Example of the identification of input regions across the layers of a deep model, taken from Montufar et al. [2014]. Depicted are three layers, and data is processed from left to right. The first layer takes five neighborhoods in its input space (shown in red) and identifies them with two neighborhoods in its output space. The next layer, in turn, identifies these two red neighborhoods with a single common subset of its output space. In this way, functions computed on the red neighborhood in the final output space are replicated across a total of five different subsets of the input space. In general, the number of replications is exponential in the depth of the network.

previous layers. As shown by Montufar et al. [2014], neural networks are expressive enough to create an exponential number of folded regions as a function of depth, which can explain their representational power. It can also explain their sensitivity to adversarial perturbations, however: if the dimensionality of the input space remains fixed, then as the depth of the network increases, functions computed on the outputs of layers are replicated across increasingly small regions of the input space. Tiny perturbations may then cause inputs to shift to entirely different regions of the output space, potentially making the network vulnerable to adversarial examples. This is the spirit of the work by Croce et al. [2019], who attempt to defend DNNs from adversarial perturbations by explicitly maximizing the size of such regions.

1.1.8 The importance of depth and width

Long before the rise of DNNs, it was already known that neural networks could approximate essentially any function using only a single hidden layer, as long as that layer had sufficiently many parameters [Hornik et al., 1989]. The number of parameters in a given layer is usually referred to as the *width* of the layer, as opposed to the *depth* of the network which is simply the total number of layers. With the advent of AlexNet and the subsequent deep learning "renaissance" in the early 2010s, it became clear that classical universal approximation results for neural networks were, in a sense, incomplete as they tended to neglect depth. Empirically, deep networks could apparently afford to be less wide in individual layers, resulting in large overall reductions in the number of parameters and computational resources required to fit complicated data sets.

Early theoretical works such as Montufar et al. [2014] hinted at an *exponential* relationship between model capacity and depth, a finding that was reasonably well-supported by empirical advances in neural network architectures. Safran and Shamir [2017] proved an interesting theoretical result in this vein: they exhibited several simple and natural functions where deeper networks require exponentially fewer parameters than shallow networks. That is, there exist deep neural networks that cannot be realized by shallower networks without an exponential increase in width. Lu et al. [2017] strengthened these results by asking the dual question: do there exist wide networks that cannot be realized by narrower ones without significantly increased depth? In this case, it turns out that narrow networks incur only a *polynomial* penalty to their depth in order to match wider networks, so the trade-off is much less severe. It would therefore seem that depth is more important than width for improving the performance of DNNs.

However, depth alone is not the whole story, and later works showed that the width of individual layers was important as well despite the results of Lu et al. [2017]. For example, Nguyen et al. [2018] showed that neural networks must have at least one layer that is wider than the input dimension in order to produce disconnected decision regions. That is, if the network has a pyramidal structure where all layers have width less than or equal to the input dimension, then the resulting decision regions will be connected. This can be considered problematic for several reasons:

- It is generally a very strong assumption to require connectedness of the decision regions for any particular classification problem. Indeed, for most problems the different classes will be fragmented throughout the input space and hence the decision regions will not be connected.
- Connected decision regions can imply vulnerability of the classifier to adversarial pertur-

bations, since the classifier will make incorrect interpolations between samples to connect disparate decision regions.

Although the condition that at least one layer should be wider than the input is relatively weak, the results of Nguyen et al. [2018] demonstrate that width cannot be neglected, even if it can be considered less important than depth.

1.2 Rise to power

The core ideas underpinning modern DNNs are actually quite old: the architectural principles were already known in the 1980s in works such as the *Neocognitron* by Fukushima and Miyake [1982] and knowledge of the *backpropagation* algorithm used to train neural networks goes at least as far back as the 1960s [Kelley, 1960]. Despite this long history, two important factors impeded the widespread adoption of DNN techniques:

1. DNNs generally require a lot of data to train. Before the early 2000s, most data sets simply weren't big enough for DNNs to work well.
2. DNNs were not computationally efficient. A DNN consists of many layers of computational units stacked on top of each other, relying heavily on matrix algebra to calculate their outputs. It was only with the rise of parallel computation on GPUs that efficient algorithms could be developed to train and deploy DNNs.

These factors combined to make DNNs unattractive to ML practitioners compared to other, more traditional algorithms such as the support-vector machine [Cortes and Vapnik, 1995] or random forest [Ho, 1995].

Nevertheless, interest in DNNs never died out completely. Researchers at the Canadian Institute For Advanced Research (CIFAR) in particular remained active in this area of ML. They eventually published the AlexNet paper in 2012 [Krizhevsky et al., 2012], demonstrating that DNNs had now become a viable alternative to existing ML algorithms (at least in the area of computer vision). The performance improvement was shocking: AlexNet had suddenly bested the accuracy of the state of the art by over 10 percentage points on a problem the ML community had already been working on for years. After the publication of AlexNet, things moved very quickly: DNNs became "the next big thing" in ML and by 2015 it seemed the community had almost entirely abandoned all other approaches in favor of DNNs. What had started as a way to improve computer vision algorithms had now grown into an entirely new branch of ML with applications ranging from medical diagnosis to speech recognition and autonomous driving. This growth is nicely illustrated in figure 1.8 which depicts the number of deep learning papers uploaded to arXiv[3]. The graph suggests that, in the past five years alone (2015 - 2020), the number of scientific publications in the field of AI grew sixfold.

1.3 Reaching the limits

Especially in the early years, DNNs proved to be a rather cheap way to dramatically improve over the state of the art (SOTA). The basic procedure is as follows:

[3] `https://www.arxiv.org/`

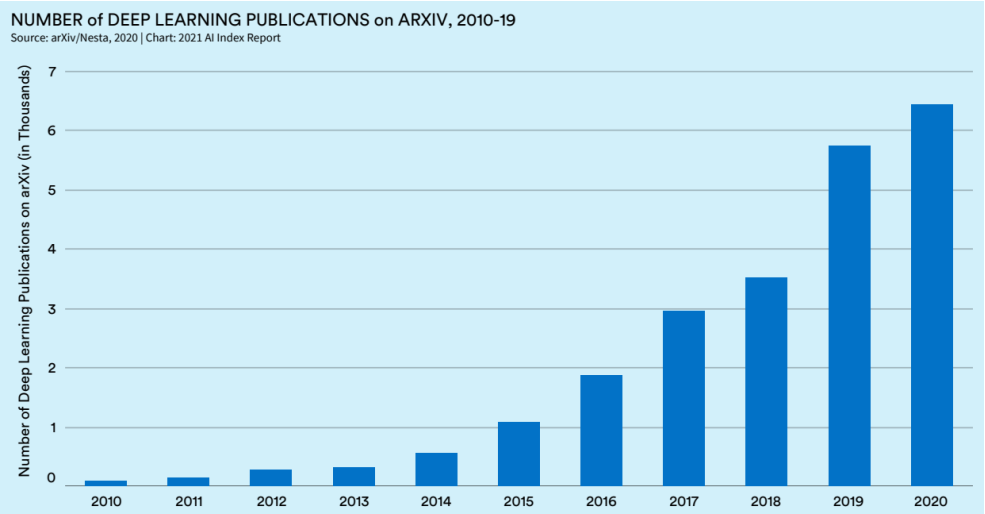


Figure 1.8: The number of deep learning papers uploaded to arXiv from 2010 to 2020. Figure due to Zhang et al. [2021b].

1. Amass lots of data.
2. Create a really big neural network.
3. Train the network on the largest computing cluster you can afford.
4. Publish a paper reporting your improvements on the SOTA.

In general, most publications nowadays tend to improve on the SOTA by either collecting more data, creating a bigger neural network, training the network on a larger cluster, or some combination of these. Although there are certainly exceptions to this — DeepMind's AlphaFold model for protein structure prediction being a notable example where the innovations are of a much more fundamental nature [Jumper et al., 2021] — the vast majority of published work seems to follow this trend. For instance, in 2016, AlexNet was beaten by a residual network architecture developed by Microsoft [He et al., 2016]. AlexNet had 8 layers; Microsoft's network has over 100. Of course, I should add that the increased depth was not the only innovation made by the Microsoft team: at the time, training neural networks with that many layers was a major open problem, because the networks tended to become progressively more unstable and harder to train as their depth increased. The idea of *residual layers*, *i.e.*, layers that compute a residual with respect to the input (see figure 1.9), was the prime innovation that allowed the networks to become that deep in the first place. Nevertheless, the general trend in ML today appears to be to keep making neural networks deeper; most other innovations, from training algorithms to architectural design principles, merely seem to be in service of this one goal. This philosophy was perhaps most famously articulated by Szegedy et al. [2015], who introduced a number of architectural innovations in their GoogLeNet model precisely to accommodate many more layers. Other popular and very well-known techniques such as batch normalization [Ioffe and Szegedy, 2015], weight normalization [Salimans and Kingma, 2016] and group normalization [Wu and He, 2018] aim to achieve the same thing: stabilize and accelerate training of ever deeper networks. The question of whether or not it is useful to keep increasing this depth is

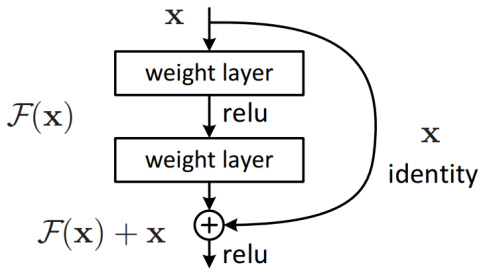


Figure 1.9: Illustration of a residual layer as introduced by He et al. [2016].

rarely considered; the tacit assumption is that bigger always means better.

Figure 1.10 gives a more detailed comparison of model complexity versus accuracy. While we see that the raw number of trainable parameters does not necessarily correlate with accuracy (a finding in line with the OpenAI report by Hernandez and Brown [2020]), the diminishing returns are clear: accuracy appears to be an approximately logarithmic function of model complexity. A nice illustration of this phenomenon was given recently by the Vision Transformer (ViT) model [Dosovitskiy et al., 2020], where the authors show that if they pre-train a huge model on an equally huge data set, they can surpass the SOTA by approximately one percentage point. A systematic investigation into the scaling laws of these sorts of Transformer models was undertaken recently by Kaplan et al. [2020]. They empirically verified that the loss of large neural language models scales according to a power law with model size, data set size and amount of computational resources used for training. Surprisingly, they find that architectural details such as network depth and width have little effect; rather, the raw number of parameters appears to be the decisive factor. This contradicts several previous studies into the generalization behavior of DNNs, where it is commonly claimed that depth and width need to be appropriately balanced [Lu et al., 2017, Nguyen et al., 2018, Safran and Shamir, 2017].[4] Furthermore, Kaplan et al. also make the puzzling observation that larger models appear to be *more* sample-efficient, *i.e.*, they require *fewer* samples to train than smaller models to reach the same performance. This also contradicts conventional wisdom from statistical learning theory that more data is required to fit models with more parameters [Shalev-Shwartz and Ben-David, 2014, Vapnik, 1999].

These results are intriguing, to say the least, as they demonstrate that our theoretical understanding of DNNs is still fundamentally lacking. It also lends credibility to the idea that DL, in its current form, may have reached its peak. Kaplan et al. observed that the loss of DNNs scales according to the following power law:

$$\mathcal{L}(N,D) = \left(\left(\frac{N_c}{N} \right)^{\frac{\alpha_N}{\alpha_D}} + \frac{D_c}{D} \right)^{\alpha_D},$$

where N is the number of parameters of the model, D is the data set size and the constants N_c,

[4] The caveat with these prior works is that they tend to only consider feed-forward, densely connected networks with ReLU activations. Transformers are more complicated than this, so these theoretical results may not apply to them. Nevertheless, it would still be quite peculiar if Transformers managed to escape the theoretical limitations proven for ReLU networks purely by virtue of their special architecture. After all, as universal function approximators, ReLU networks should be able to simulate Transformer models.

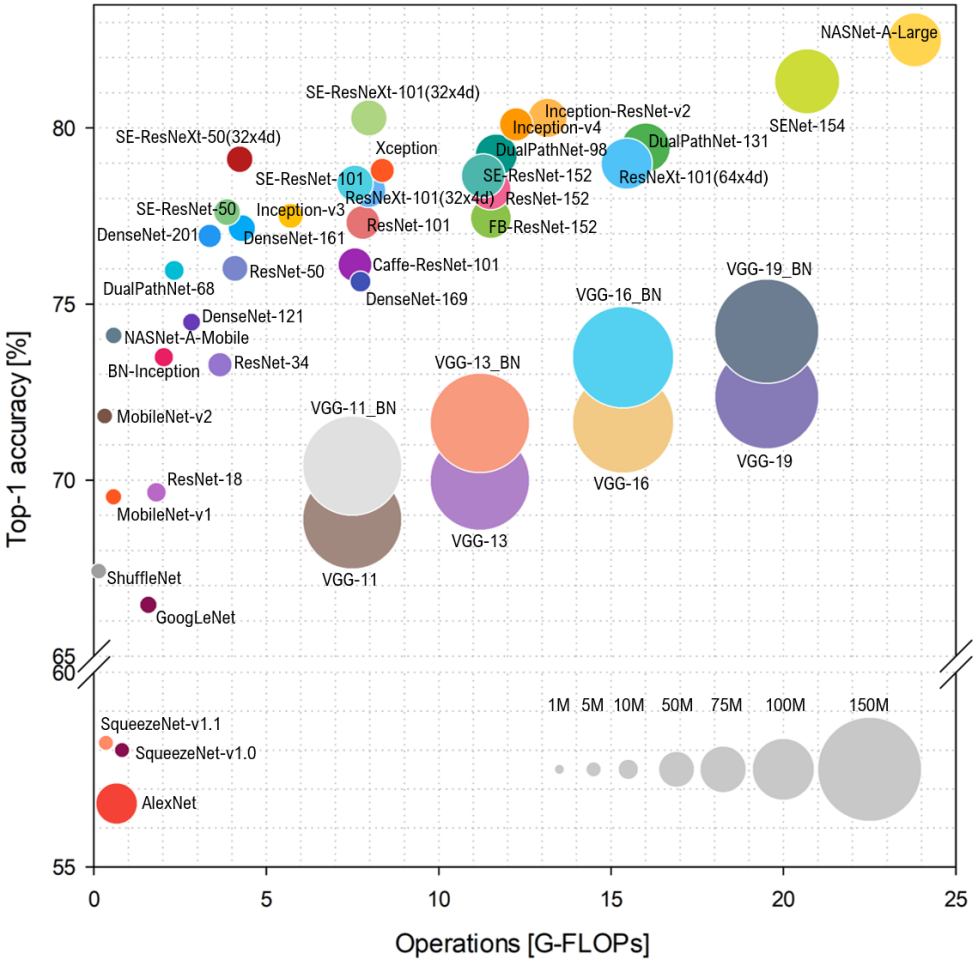


Figure 1.10: Comparison of many DNN models in terms of their accuracy and complexity. Source: `https://github.com/CeLuigi/models-comparison.pytorch`. Accessed 2021-08-19.

D_c, α_N, α_D are given by

$$N_c = 8.8 \times 10^{13}, \qquad \alpha_N = 0.076,$$
$$D_c = 5.4 \times 10^{13}, \qquad \alpha_D = 0.095.$$

If this empirical approximation is accurate, then there are clear diminishing returns to increasing the model size and data set size: increasing the number of parameters from 1 million to 2 million will have a much larger impact than going from 1 billion to 2 billion, and similarly for the number of samples. This state of affairs may lead one to wonder whether we have reached the peak of what DL has to offer, as some scholars have suggested [Marcus, 2018]. In order to make progress, can we continue to rely indefinitely on this tactic of creating ever larger models and data sets trained on increasingly powerful supercomputing clusters?

I believe the field of DL has come to a point where it absolutely needs to reckon with this question. Otherwise, we run the risk of stagnating and becoming irrelevant. We will stagnate, because the diminishing returns in performance imply that we will eventually hit a ceiling beyond which we cannot meaningfully improve anymore by simply scaling up our models, data sets and computing clusters. Moreover, the disastrous consequences of anthropogenic climate change are being taken increasingly seriously by the public as well as governments, and the ML industry is no small contributor to this problem [Bender et al., 2021]. We can no doubt expect the cost of scaling DL models to rise prohibitively as society cracks down on the companies responsible for large CO2 emissions. We also risk becoming irrelevant, because industry and the rest of the scientific community alike do not share our unbridled enthusiasm for large neural networks. In order to be useful, ML models must in many cases be able to cope with limited available memory and computational resources. Often, they must be able to respond to queries within fractions of milliseconds. It is also highly likely that, in order to fit a specific practical use case, the model will need to be retrained or fine-tuned on proprietary data. The vast majority of practitioners therefore have absolutely no use for models so large and so cumbersome that only the biggest technology companies have the resources to deploy them.

There is a clear problematic trend in the field of ML where we appear to become progressively more isolated from the rest of the world as time goes on (both within and without academia). We are increasingly focused on optimizing performance on artificial benchmarks rather than achieving real-world goals. To illustrate this isolation, let us turn for a moment to Noam Chomsky, one of the leading linguists of our time. Chomsky famously does not believe that neural networks are capable of accurately modeling human language and cognition,[5] yet his concerns seem to fall on deaf ears within the natural language processing (NLP) community, which just keeps carrying on as usual. This is the ML equivalent of physicists not taking criticisms of their work by Stephen Hawking seriously. In medicine, as well, we find an overabundance of studies claiming to have developed ML systems that are highly accurate for diagnosing certain diseases, but which end up not having any clinical utility at all due to severe methodological flaws [Roberts et al., 2021]. One would think that ML experts would follow the recommendations of actual clinicians when designing DL systems for medical applications, but as with NLP, it seems the ML community simply does not listen to experts outside of their own field. To move forward, we must take seriously the criticisms and recommendations of other experts, *especially* those outside of ML, because ML is rarely pursued for its own sake: often, a machine learning system is developed in service of a particular practical need for automation in some other domain. It is

[5] `https://chomsky.info/1967____/`. Accessed 2021-08-23.

therefore absolutely crucial that ML practitioners communicate with the relevant experts in the domains where their ML systems will be deployed.

The fundamental promise of AI is to "solve" intelligence, *i.e.*, to create artificial systems which exhibit intelligent behavior in the sense that they can *learn* from experience and *adapt* to novel situations and problems. However, as long as the field remains opaque to outsiders, this promise can never be fulfilled. It cannot be disputed that "solving" intelligence will require an interdisciplinary approach that combines insights from the *full* range of sciences, including psychology and sociology, as well as fields that are not generally characterized as "scientific," such as various branches of philosophy. This interdisciplinary character is sorely lacking in contemporary AI. Prime examples of this are the naive and over-stated claims about artificial general intelligence (AGI) that are regularly made by high-profile organizations and research groups, such as DeepMind who reduce the entire concept of intelligence to merely one of optimizing the correct mathematical functions [Silver et al., 2021]. This position appears to be extremely popular within contemporary ML yet is roundly rejected by almost every knowledgeable person outside of the field [Lent, 2021, Marcus and Davis, 2019, Roitblat, 2020]. It is an old-fashioned ideology that can be traced back to 17th century philosophers such as René Descartes, who believed in a thoroughly mechanistic worldview where living beings were nothing but elaborate machines [Descartes, 1989]. One can see why this idea is so tempting to ML researchers, because it implies that they will indeed solve intelligence by simply carrying on as they always have; no need for any new ideas or paradigm shifts. We merely have to keep adding layers to our neural networks, like rungs on a ladder, and be confident that we will one day climb all the way to the moon.

1.4 Social (ir)responsibility

There are other considerations as well which the ML community has largely neglected. Although it has become fashionable to boast about how much fewer computational resources are required to train SOTA models, the fact remains that DL has a terrible carbon footprint: training a single DL model can emit as much carbon as five cars would in their *entire lifetime* [Bender et al., 2021, Hao, 2019]. Given the climate catastrophe currently unfolding worldwide, mainly due to the emission of CO2, we must ask ourselves whether this cost is worth it. In my opinion, it is certainly not: the ML community *cannot* keep avoiding all of its societal responsibilities and *must* find ways of making DL more sustainable. Preferably, there should be a regulatory framework imposed that demands accountability of ML practitioners for the carbon emissions they cause. This is already the case in many other industries; I see no reason why ML should be treated differently.

Similarly, the ML community needs to reckon with the societal impact of its data collection practices and the surveillance apparatus it has helped to build [Amnesty International, 2019]. One of the most popular applications of ML is *facial recognition*, where a DL system is constructed to identify human faces (and sometimes even complete identities) from live video footage. This obsession the ML community has with facial recognition is in itself sinister and borderline voyeuristic,[6] but it is compounded by the fact that facial recognition systems require an enormous number of images of real human faces in order to be trained successfully. This has led

[6] Are there not **many** other problems that are vastly more interesting and useful than this? One struggles to imagine ethical reasons why facial recognition could be such a hot topic for ML researchers.

to the creation of famous data sets such as ImageNet [Fei-Fei et al., 2009], WebFace [Yi et al., 2014] and Labeled Faces in the Wild [Huang et al., 2008], all of which contain images scraped off the internet using automated scripts with absolutely no regard for individual privacy or consent of the subjects whose actual faces are included in the data. In the case of ImageNet, public backlash about the unauthorized use of individuals' photos has forced its creators to censor or outright delete significant portions of the data set [Knight, 2021]. While certainly a step in the right direction, unaltered archived copies of the data set no doubt exist and are in active circulation among researchers reluctant to update. Moreover, although it is the most popular, the ImageNet data set is still only one data set among many.

The recent work by Birhane and Prabhu [2021] provides sobering and shocking revelations about the extent of this problem. They look at several highly popular large-scale image data sets and find that many of the samples contained in these collections fall into verifiably pornographic categories: shot in non-consensual settings, beach voyeurism and exposed private parts. Non-consensual pictures of children are sometimes also included,[7] meaning that the ML community has built much of its recent breakthroughs and successes (unwittingly, of course) on the exploitation of child pornography.

Recent legislation such as the European General Data Protection Regulation (GDPR),[8] the California Consumer Privacy Act (CCPA)[9] and Japan's Act on the Protection of Personal Information[10] provides the people with much more control over their personal data and more stringent protections of their privacy. These legal developments are certainly steps in the right direction, but it is clear that effective enforcement of these laws is still lacking. If anything, companies keep collecting ever *more* personal data, not less, seemingly unhindered by regulation such as the GDPR [Amnesty International, 2019]. Countermeasures against indiscriminate surveillance must therefore be *multi-faceted*. We must rely not only on strong laws to curtail the unauthorized exploitation of personal data; we must provide people with tools they can use to defend themselves as well and to render such exploitation impractical in the first place. The *data leverage* framework of Vincent et al. [2021] as well as the idea of *unlearnable samples* introduced by Huang et al. [2021] work toward such tools, but we will need more than that.

It is clear that the field will *not* hold itself accountable and will *not* adequately self-regulate: recent events such as the wrongful firing and subsequent organized harassment of renowned researchers Timnit Gebru[11] and Margaret Mitchell[12] are dire indications. Google essentially fired two of its best AI ethics researchers for writing a paper [Bender et al., 2021] that critiqued the large language models developed by Google. They then became targets of an organized harassment campaign spurred on by high-profile Google insiders and well-known researchers within ML, such as Jeff Dean and Pedro Domingos.[13] The message is clear: these companies will tolerate no dissent within their own ranks and they will spurn those critical of their methods.

[7] `https://bit.ly/2y1sC7i`. Accessed 2021-09-09.

[8] `http://ec.europa.eu/justice/data-protection/reform/files/regulation_oj_en.pdf`. Accessed 2021-05-22.

[9] `https://www.oag.ca.gov/privacy/ccpa`. Accessed 2021-05-22.

[10] `https://www.ifrc.org/docs/idrl/928EN.pdf`. Accessed 2021-05-22.

[11] `https://www.wired.com/story/google-timnit-gebru-ai-what-really-happened/`. Accessed 2021-08-23.

[12] `https://www.theguardian.com/technology/2021/feb/19/google-fires-margaret-mitchell-ai-ethics-team`. Accessed 2021-08-23.

[13] `https://www.theverge.com/22309962/timnit-gebru-google-harassment-campaign-jeff-dean`. Accessed 2021-08-23.

The only conclusion we can draw is that pressure to regulate the industry must therefore come from without.

1.5 A Potemkin village

According to legends [David-Fox, 2013], when the Russian empress Catherine II traveled to Novorossiya[14] along with eminent European diplomats and dignitaries, Grigory Potemkin, then governor of the region, erected a series of hollow façades of villages in order to impress them. These villages were dismantled as soon as the visitors departed, only to be re-built further along their route of travel. The population consisted entirely of actors hired by Potemkin and disguised as peasants, to give the impression that the fake villages were actually inhabited. The goal of this deception was to demonstrate the success of Russian civilization in colonizing the new imperial lands.

Although historians dispute the credibility of these stories, the concept of a *Potemkin village* has endured. It refers to any construction, literal or figurative, meant to serve as a façade to keep up an appearance of prosperity when one is in fact not faring very well at all. The analogy between DNNs and Potemkin villages was first suggested by Goodfellow et al. [2014] in one of the early papers on adversarial examples for DNNs:

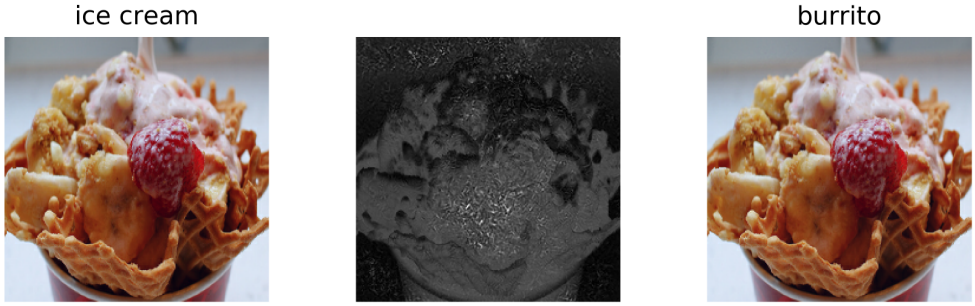
> *These results suggest that classifiers based on modern machine learning techniques, even those that obtain excellent performance on the test set, are not learning the true underlying concepts that determine the correct output label. Instead, these algorithms have built a Potemkin village that works well on naturally occurring data, but is exposed as a fake when one visits points in space that do not have high probability in the data distribution.*

Figure 1.11 shows several examples of such adversarial perturbations in various settings. The most common domain of application is computer vision, where we can generate perturbations to fool object detectors into recognizing items that are not present in the image at all, or trick segmentation algorithms into drawing pictures of cartoon characters instead of properly segmenting the input. Aside from computer vision, however, there has also been some research into textual adversarial examples that can deceive language models. The errors introduced by these perturbations can cause, *e.g.*, incorrect translations or erroneous sentiment recognition.

At this point, it is worth mentioning that the idea of adversarial examples is much older than the works of Goodfellow et al. [2014], Szegedy et al. [2013]. The vast majority of works published in the adversarial ML literature tend to start with the claim that adversarial examples were first discovered by Szegedy et al. [2013]. Indeed, Goodfellow himself has gone so far as to publicly assert that he himself *coined* the term "adversarial examples."[15] Although it is entirely possible

[14] "New Russia," now part of Ukraine.

[15] He specifically makes this assertion in his keynote speech at the 2018 IEEE Symposium on Security & Privacy, titled *Defense Against the Dark Arts: An Overview of Adversarial Example Research and Future Research Directions*. Apparently, Christian Szegedy was going to name them "hard examples," despite the fact that a brief literature study would have revealed that this phenomenon already had a name and had been well-studied in classical ML. The tendency of contemporary ML researchers to neglect their literature studies and subsequently re-invent concepts that already exist is another problem one could write about at length. A more recent example of this trend is the *mixup* technique by Zhang et al. [2017], which is essentially identical to SMOTE [Chawla et al., 2002] published 15 years earlier. The pattern appears to be this: if a technique was not previously introduced explicitly in the context of deep learning, then the community will ignore it and believe they have discovered something entirely novel merely because they applied it



(a) An adversarial example in the image recognition setting. Here, a ResNet-50 architecture correctly recognizes a sample from the ImageNet validation set as being a picture of an ice cream. When the perturbation in the middle is added pixel-wise to this image (brighter pixels correspond to larger perturbations), we obtain the image on the right. Although visually identical to the original, this image is erroneously classified as a burrito instead.

(b) The *Houdini* algorithm developed by Facebook AI can be used to manipulate automatic image segmentation models [Cisse et al., 2017b]. The original sample on the left is correctly segmented into three parts: the road (highlighted in purple), the cars (highlighted in blue) and the scenery (highlighted in green). When Houdini adds a special perturbation to this sample, however, the segmentation model draws a picture of a minion instead.

	Sentence (SEC)	Prediction
Original Sentence	The essence of this film falls on judgments by police officers who fortunately ethical and moral men act on situations within situations in a city with a super abundance of violence and killing Good compound interacting story lines and above average characterizations <eos>	Positive
Adversarial Example	The essence from THIS film falls on judgments by police officers who fortunately ethical and moral men act on situations within situations in a city with a super abundance of violence and killing Good compound interacting story lines and above average characterizations <eos>	Negative
Original Sentence	There is really but one thing to say about this sorry movie It should never have been made The first one one of my favourites An American Werewolf in London is a great movie with a good plot good actors and good FX But this one It stinks to heaven with a cry of helplessness <eos>	Negative
Adversarial Example	There is really but one thing to say about that sorry movie It should never have been made The first one one of my favourites An American Werewolf in London is a great movie with a good plot good actors and good FX But this one It stinks to heaven with a cry of helplessness <eos>	Positive

(c) Adversarial examples in natural language processing. Sato et al. [2018] show that ML models for sentiment analysis can be easily fooled by replacing certain words in the text, even though these changes should be inconsequential.

Figure 1.11: Examples of the vulnerability of machine learning models to various types of adversarial attack.

for researchers to re-discover the same phenomenon independently, adversarial examples were already known under this and various other names at least as early as 2004 [Dalvi et al., 2004]. For a nice historical overview of this field, I refer the reader to Biggio and Roli [2018].

1.6 Objectives of this thesis

The majority of the work in this thesis will be dedicated to *defenses* against adversarial perturbations, *i.e.*, attempts to make deep neural networks more robust to adversarial manipulation.

I begin with a contemporary overview of the field of adversarial machine learning in chapter 2. There, I formally describe the problem of adversarial robustness and present some related work. This related work encompasses both theoretical as well as empirical contributions. I must note, however, that adversarial machine learning has become an incredibly popular field of study within AI, with several hundred new papers appearing on the subject every month. As such, the related work I discuss in this manuscript is by no means exhaustive, and I will no doubt have either omitted or entirely missed some other important works. For this I apologize in advance.

Chapter 3 introduces the field of *conformal prediction* (CP), which underpins the majority of contributions in this thesis. Despite being a relatively niche subject, CP seems to me ideally situated for adversarial defense. It is a sub-discipline of frequentist statistics[16] focused on developing learning algorithms that can explicitly take into account "anomalous" data, *i.e.*, data that deviates significantly from what the algorithm has seen in the past. This chapter explains the basic principles of the field and details an early adversarial defense based on CP, a specialization of the so-called *inductive Venn-ABERS predictor* (IVAP) to defend *binary* classifiers against adversarial perturbations, with an application to a real-world cybersecurity problem. The subsequent chapters 4 and 5 further build on these ideas to produce two more sophisticated defenses. Chapter 4 extends the analysis of chapter 3 to multi-class classification problems and develops a generalization which I call the *MultIVAP*. This algorithm can improve the adversarial robustness of *any* classifier, including but not limited to DNNs. Finally, in chapter 5, I go another step further and construct an adversarial defense based on the general conformal prediction algorithm (of which the IVAP is a special case) and a custom non-conformity measure derived from the theory of *correspondence analysis* (CA). These three algorithms all fall into the category of *detector methods*, where the primary goal is to *detect* whether a given input is adversarial or benign and "reject" or "flag" it when we suspect the former. Detectors can therefore offer protection against misclassification resulting from adversarial manipulation, but as of yet they cannot (efficiently) be used to construct classifiers that are intrinsically robust and which need no "reject" option.

In chapter 6, I present a method of defense that aims to make classifiers intrinsically robust rather than "merely" able to detect adversarial manipulation. It is a so-called *purification* method, as the defense essentially consists of a "purifier" or "filter" which removes any harmful distortions that may have been added to the input. The resulting algorithm, which I have called the *adversarial density filter* (AdvDF), is very computationally efficient at inference time, as it only requires a pre-processing step which can be handled by another (relatively small) neural network. The AdvDF also has the additional advantage of being *certified*, *i.e.*, there are explicit

to deep networks for the first time.

[16] Making it all the more niche for the modern ML community, which seems to be exclusively interested in *Bayesian* statistics.

theoretical bounds on the maximum error that can be introduced by *any* adversary restricted to a certain perturbation budget. These bounds are asymptotically more favorable than other bounds previously proposed in the literature, and they apply to a much more general class of models, allowing great flexibility in the design of the AdvDF module. Furthermore, the techniques used to prove these bounds can no doubt be refined in future work to derive even better methods.

Finally, chapter 7 concludes the work. I reflect on the work that has been done in the field of adversarial ML (both my own contributions as well as those by other researchers) and the field of ML more broadly. I also offer my view of how AI could make progress on the major issues it is facing today.

1.6.1 Guidelines for practitioners

To facilitate the practical application of the methods developed in this work, I end this chapter with a brief overview of the different algorithms and their salient properties.

The binary IVAP defense. This method, detailed in algorithm 3.3, is a lightweight defense that can be used for binary classification problems. It acts as a wrapper around an existing classifier and can help to determine whether the predictions of this model are unreliable. Aside from the pre-trained model and some training data, there is only a single hyperparameter to be tuned: the rejection threshold β. This is a real number in the interval $[0, 1]$ which can either be set manually or easily tuned automatically according to various metrics of interest. The binary IVAP defense can be readily applied to any binary classifier without the need for re-training; the underlying model can be treated as a total black-box. We proposed this defense in Peck et al. [2020] and subsequently applied it to the detection of malicious domain names with positive results [Grumer et al., 2019, Peck et al., 2019b].

The MultIVAP defense. The MultIVAP defense algorithm 4.1 is a multi-class generalization of the binary IVAP defense algorithm 3.3 that aims to maintain the favorable statistical guarantees enjoyed by the IVAP. As such, it is similar to the IVAP defense in that it is a detector method that wraps around an existing classifier. It can then calibrate the predictions of the underlying model using some additional training data and hyperparameter tuning. Despite being based on a one-vs-all extension of the IVAP, the MultIVAP retains the simplicity of the IVAP defense when it comes to hyperparameter tuning, as it has only a single hyperparameter: a confidence threshold $\varepsilon \in [0, 1]$ that determines how conservative or liberal the MultIVAP will be in its calibration. However, contrary to most commonly used supervised classification algorithms, the MultIVAP is an inherently multi-label predictor: for every input sample, the MultIVAP outputs a set of labels that can range from the empty set to the full set of all possible classes. In case the MultIVAP returns the empty set or a set containing too many labels, the prediction can be interpreted as a rejection, indicating that the underlying model is unreliable on the given sample. The MultIVAP always guarantees that every label in its predicted set can be assigned to the input sample with probability at least $1 - \varepsilon$, so the tunable hyperparameter ε has an immediate and intuitive interpretation. Moreover, similarly to the IVAP defense, the MultIVAP can be applied to any classifier without needing to re-train it; the underlying model is essentially a black-box. We proposed this defense in Peck et al. [2019a] and demonstrated its effectiveness on common benchmarks compared to existing methods.

The conformal CANN detector. The IVAP-based defenses algorithms 3.3 and 4.1 are effi-

cient, lightweight and provide favorable statistical guarantees, but they crucially rely on a useful underlying model. In our own experiments [Grumer et al., 2019] we have found that the IVAP can struggle to properly calibrate underperforming models, leading to overconfidence. For this reason, the CANN detector sacrifices the simplicity of the IVAP in favor of more fine-grained control over the defense itself. More specifically, the CANN detector is a direct instantiation of the general conformal prediction algorithm 3.1, where the non-conformity measure is supplied by a variant of the Principal Inertia Component Estimator (PICE) proposed by Hsu et al. [2019]. This is essentially a neural network trained using a very specific loss function. As such, practitioners can create adversarial defenses by constructing domain-specific neural networks with which to instantiate the PICE. When plugged into the conformal prediction algorithm, these can give rise to more powerful defenses than what can be accomplished by directly applying the IVAP or MultIVAP to an existing model. The disadvantage here, of course, is the fact that one needs to construct and train an additional model. However, once this model is trained, it can be re-used to defend any other classifier trained on the same data, as the downstream model is once again a complete black-box.

The adversarial density filter. The adversarial density filter (AdvDF) is a purification defense rather than a detector like the previous methods. It essentially has the structure of a variational auto-encoder, but it utilizes a specific loss function which confers certain favorable statistical guarantees on downstream models. Specifically, we show that the classification error of a downstream model using the AdvDF as a pre-processing step to purify the inputs provably grows sublinearly with the perturbation budget of the adversary. This result holds regardless of the specific attack algorithm and depends crucially on the continuity properties of the AdvDF encoder module. Like the other methods, the AdvDF can treat the defended models as complete black-boxes. Similarly to the CANN detector, however, the AdvDF itself consists of an entirely separate neural network that must be trained accordingly, but this need happen only once for each data set: a trained AdvDF network can be readily applied to any existing model, although some fine-tuning may be necessary to maintain performance on clean data. Unlike our other defenses, the AdvDF is unsupervised and its application is not limited to classifiers; it can in principle also be used to defend generative models, clustering algorithms or other unsupervised models.

To summarize, we proposed four defense algorithms against adversarial perturbations, three of which are detectors and one of which is a purification defense. The detector defenses are all limited to supervised classification tasks and are based on conformal prediction. Therefore, they tend to output multiple labels (or none at all) for any given input. To successfully deploy these methods, one must be able to handle such set-valued predictions. In particular, as with all detectors, a mechanism must be in place to handle rejections, *i.e.*, when the detector signals that the underlying model is unreliable, such as by outputting too many labels or no labels at all. This may be done by flagging rejected samples for manual review or letting them be processed by more complicated and computationally demanding models. The purification defense is more broadly applicable, not being limited to supervised problems but able to handle unsupervised tasks as well. It requires the construction of a separate neural network, however, but this network can be re-used across different models.

1.6.2 On the relevance of adversarial robustness

Before concluding this introductory chapter, I would like to take a moment to discuss the relevance of adversarial robustness to the broader field of machine learning in general. When reading the literature in this domain, one can get the impression that the main reason to study adversarial robustness is because of its implications to security, as with the example of the adversarial traffic sign [Eykholt et al., 2018]. However, as pointed out by Gilmer et al. [2018], often the security implications of adversarial examples are not very severe: in the case of self-driving cars, for instance, the system already has to cope with "natural" problems such as traffic signs that have been knocked over, vandalized at random, damaged or obstructed by weather or temporarily removed as part of routine road maintenance. As such, mission-critical systems generally do not rely exclusively on machine learning but make use of additional data, safety checks and redundancies which would make irrelevant many of the threat models commonly studied in our field. Self-driving cars can use LIDAR and GPS, for example, to cope with missing or incorrectly identified traffic signs.

That said, there do of course exist specific cases where adversarial examples as they are commonly defined in the literature absolutely make sense, such as spam filtering or content moderation, but these are highly specific applications where security happens to be paramount. In many applications of machine learning, security of the ML system itself is not really that important. The main question, then, is the following:

> *Is adversarial robustness relevant for ML applications where security and trustworthiness are not real concerns?*

I will argue here, in accordance with plenty of literature on this subject, that the answer is yes, adversarial robustness can certainly be relevant even in applications where security is not a factor and the model doesn't have to be very trustworthy.

First and foremost, adversarial examples are highly interesting in their own right, even if there is no true "adversary" to speak of. The fact that we can change a single pixel in a high-resolution input image to a machine learning model and have said model completely change its output points to some deep problems in our current methodologies. It calls into question how well our models have truly "learned" to solve their respective problems. Understanding why adversarial examples exist and how to eliminate them (to the extent that this is possible at all) will undoubtedly have profound consequences for the way we practice machine learning. This is perhaps the most obvious reason why adversarial robustness is broadly relevant to the field.

Less obvious but equally important reasons are given by more recent research that compares robust and non-robust models according to various benchmarks. It had been argued for quite some time that adversarially robust models learn better feature representations, based on several pieces of evidence. Tsipras et al. [2019], for example, find that robust models tend to have more interpretable gradients that align better with salient visual features of the input. Santurkar et al. [2019] took this another step further and showed that many of the fundamental ML tasks we perform today can be accomplished much more easily by adversarially robust classifiers, even if the task has no immediate relationship with classification (*e.g.*, image inpainting, style transfer and super-resolution). It was also shown by Salman et al. [2020a], Utrera et al. [2020] that adversarially robust models perform better in transfer learning tasks than their non-robust counterparts. Together, these findings strongly suggest that adversarial robustness is useful for essentially all machine learning tasks, as it fundamentally improves the quality of the learned representations.

The intuition that adversarial robustness should help generalization has also been experimentally verified: the AdvProp algorithm by Xie et al. [2020] uses adversarial examples to improve the generalization performance of convolutional neural networks, FreeLB by Zhu et al. [2019] does the same for language models and version 4 of the YOLO object detection algorithm also used adversarial robustness to improve performance [Bochkovskiy et al., 2020].

It should be clear from the above citations that adversarial robustness can certainly be useful for many, if not all, applications of machine learning. Of course, this was probably not originally foreseen by Szegedy et al. [2013] when they first discovered this phenomenon in deep neural networks. In their original paper, Szegedy et al. saw this as an "intriguing property" that was simply highly unexpected and therefore interesting. They conjectured that adversarial examples may be related to generalization performance, but at the time concrete evidence for this (albeit intuitively plausible) hypothesis was lacking. We have come a long way since 2013, and adversarial robustness is nowadays widely recognized not merely as an academic exercise but a valuable practical tool for all machine learning practitioners and researchers.

Chapter 2

Robustness of deep neural networks

> Le Poète est semblable au prince des nuées
> Qui hante la tempête et se rit de l'archer;
> Exilé sur le sol au milieu des huées,
> Ses ailes de géant l'empêchent de marcher.
>
> *Les Fleurs du Mal*
> Charles Baudelaire

In this chapter, I will provide a contemporary overview of advances in the field of adversarial robustness. I begin by introducing the problem informally and working my way to a formal definition that I believe captures the essence of adversarial robustness quite well. I then proceed to cover notable adversarial attacks and defenses as well as interesting theoretical results that have been described in the literature. As I have mentioned before, the field of adversarial ML has attracted an enormous amount of research interest, and so it is impossible to cover all relevant works in a single chapter, or even a single book (though some have tried; see, *e.g.*, Vorobeychik and Kantarcioglu [2018]). The selection of works I have decided to present here is therefore necessarily limited, but I do believe it covers the most important areas of the field.

2.1 Shortcut learning

I recall an anecdote by one of my colleagues at the University of Ghent about an interesting experience she had during final examinations a few years ago. At the time, she worked at the Faculty of Medicine, where exams are commonly taken in the form of multiple choice questionnaires that get graded automatically by computer. The system simply compares the dots the students have colored in to the answer key and assigns grades accordingly. As it turns out, the professor in charge of the course had been re-using the exact same exam for at least eight consecutive years. The students were aware of this, naturally, and as a result they had developed a shortcut to passing this particular exam: they simply memorized the question numbers and the corresponding answers, typically a single letter like A, B, C or D. So instead of studying the

actual course material, many students were able to pass with flying colors by merely associating numbers with letters. When the professor finally decided to make slight changes to the exam, almost 100% of the student body suddenly failed the course, despite the new exam being highly similar to the old one.

This behavior is known in psychology as *surface learning* or *unintended cue learning* [Scouller, 1998], where students are merely optimizing their performance on a given test rather than trying to actually understand the course material. If the test is badly designed (like the exam from our anecdote), this can lead to bad students who do not understand the material at all outperforming good students who made an honest effort by a very large margin. Interestingly, DNNs are vulnerable to essentially the same type of behavior: they will achieve exceedingly high accuracy on standardized benchmarks, often even outperforming human experts on various tasks [Buetti-Dinh et al., 2019, Ibarz et al., 2018, Silver et al., 2016], yet they break down completely when slight changes are made to the problem. For instance, Geirhos et al. [2018] found that DNNs can be more accurate than humans at identifying objects in images, but this superiority quickly vanishes when the images are slightly manipulated (*e.g.*, rotated, distorted with small amounts of noise, etc).

In the context of DNNs, this behavior is commonly referred to as *shortcut learning* [Geirhos et al., 2020]. The idea is that DNNs tend to learn relationships between the inputs and outputs that are overly specialized to the idiosyncrasies of the particular data set on which they are trained. This phenomenon is highly related to, but distinct from, the notion of *overfitting*. When a DNN overfits, it is essentially "memorizing" the training data, much like the students who simply memorized pairs of questions and answers rather than understanding the reasons behind them. However, shortcut learning is not entirely the same as overfitting, since the DNN is not memorizing the data set *per se*. Indeed, the common method for identifying overfitting relies on plotting two curves during optimization (see figure 2.1): the *training* loss (the loss of the model on the training data set) and the *validation* loss (the loss of the model on a held-out portion of the data, a portion that is not used for training). The tell-tale sign of overfitting is when the training loss continues to decrease while the validation loss has begun to increase. With shortcut learning, on the other hand, it is entirely possible that we see both curves continuing to decrease, because the model is not memorizing the training data; rather, it is learning rules that allow it to generalize to unseen samples, but only from the very specific distribution of images from which our data are sampled. As soon as this distribution shifts even slightly, the model breaks down.

A nice example of this was given by the authors of CycleGAN [Zhu et al., 2017], which is a DNN that allows one to make very specific semantic modifications to given images. For instance, one could take a picture of a landscape during the summer season, and ask CycleGAN to turn it into an equivalent picture taken during a hypothetical winter season instead. Figure 2.2 shows a few examples from the original CycleGAN publication that illustrates many of its use-cases. At a first glance, CycleGAN appears remarkably capable at accomplishing its task, turning out highly realistic artistic renditions of the original images in their alternate settings. Upon closer inspection, however, we can notice subtle details that betray certain shortcuts CycleGAN has learned from its data. Take the center pair of images in figure 2.2, where a picture of a horse is transformed into an equivalent picture of a zebra. Notice how the original image was taken on a green grass field with a lush green forest in the background. Notice how these shades of green have all turned brown in the transformed image, making it appear as though the environment has suffered a harsh summer with great drought. The explanation for this phenomenon is due to a certain bias in the data set: the only pictures of zebras in the data set on which CycleGAN

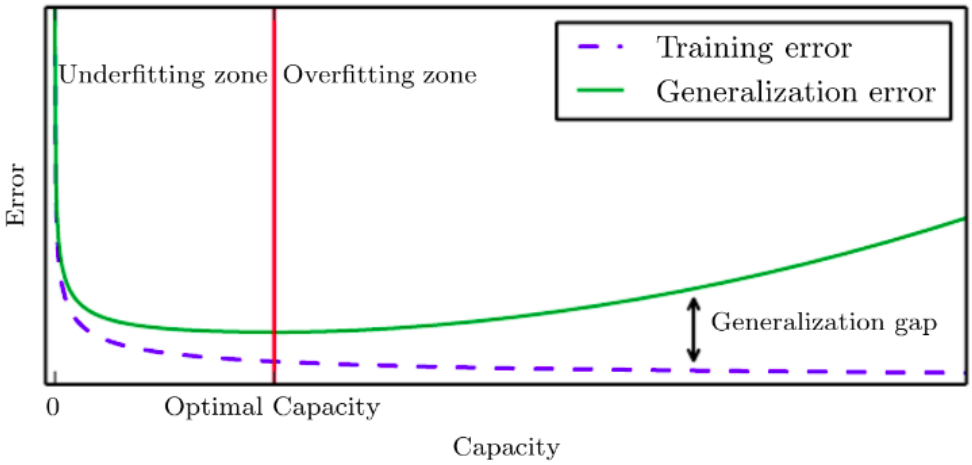


Figure 2.1: Illustration of the phenomenon of *overfitting*. The dashed blue line is the training loss, *i.e.*, the loss of the model on the training set. The green line is the validation loss, which is the loss of the model on unseen data not used for training. The vertical red line marks the transition between the underfitting regime — where the model underperforms on both training and validation data — and the overfitting regime, where the model continues to improve on the training set but deteriorates on the validation data. Image due to Goodfellow et al. [2016].

was trained, were taken in savanna landscapes. CycleGAN has therefore learned to associate zebras with savannas, and transforms the environment accordingly. Another comparable failure case is shown in figure 2.3, where CycleGAN was again asked to transform a horse to a zebra. Because the network was never trained on horses with riders on them, it transforms the human rider as well.

Similar issues have been discussed, for example, by Shankar et al. [2017], who point out that the most popular machine learning data sets also tend to have a heavy eurocentric and amerocentric bias: 60% of samples in the Open Images data set[1] comes from North America or Europe. China and India, on the other hand, represent less than 4% of the data despite being the most populous countries in the world. As a concrete example of this bias, Shankar et al. show that classifiers trained on this data are very accurate when it comes to classifying bridegrooms in photos of weddings taking place in Europe or North America. Attempting to classify Pakistani or Ethiopian bridegrooms, however, often leads to nonsensical classifications such as "chain mail."

It should be clear now that this problem is distinct from overfitting. Indeed, CycleGAN performs very well on images that resemble those present in its data set, which is the most we can expect from any ML algorithm. Rather, the core problem here is insufficient diversity of the training set which encouraged the model to make assumptions that are easily violated in practice. In essence, if a DNN can make *any* simplifying or reductionist assumptions that allow it to generalize to the *exact* data distribution on which it is trained, then it will almost certainly do so. To counteract this issue, the practitioner who builds the ML system should carefully investigate the data set

[1] `https://storage.googleapis.com/openimages/web/index.html`. Accessed 2022-01-07.

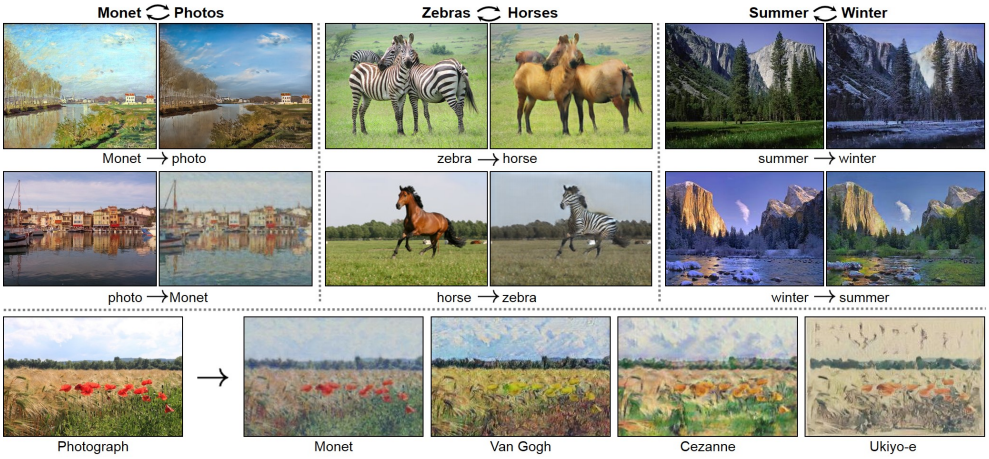


Figure 2.2: Examples of CycleGAN [Zhu et al., 2017] applied to different inputs.

Figure 2.3: A failure case of CycleGAN [Zhu et al., 2017]. The network was not trained on horses with riders and so cannot distinguish between the body of the horse and the body of the human riding on it.

for potential biases or patterns that do not generalize to the real world and adjust the training data accordingly. The reader should hopefully start to appreciate now how daunting the problem of shortcut learning actually is: contemporary ML training data sets typically have millions of samples and there are infinitely many undesirable patterns that could leak into the data, so a systematic manual investigation of this kind is simply not feasible.

Therefore, to address this issue, automated mechanisms will have to be developed that can protect ML models from making these types of mistakes. To do this in a principled manner, of course, we will first have to formalize the problem mathematically. Section 2.2 provides a typical formalization popular in the literature that we will be using throughout this work. With the mathematical framework established, section 2.3 presents algorithms for the generation of adversarial examples (so-called *adversarial attacks*); section 2.4 presents known *adversarial defenses* that have been previously proposed in the literature. Section 2.5 presents some interesting theoretical results. Finally, section 2.6 ends with historical notes on the field of adversarial ML.

2.2 Formal problem statement

The basic idea behind adversarial examples as they were conceived by Szegedy et al. [2013] is the following. Starting from a classifier $f : \mathbb{R}^n \to \{1, \ldots, k\}$, a "benign" sample $x \in \mathbb{R}^n$ (*i.e.*, just an ordinary sample from the data distribution, such as a random sample from the training set), find the minimum set of modifications necessary such that the classification output by f is altered, *i.e.*, $f(x) \neq f(\tilde{x})$ where $\tilde{x}$ is the modified "adversarial" sample.

Note that this definition crucially relies on what is considered a "minimal" modification. Indeed, the precise choice of measure is the subject of some debate and controversy within the adversarial ML community. It has become standard practice to use L_p norms for this purpose, *i.e.*, to measure the "size" of the modification as

$$\|\tilde{x} - x\|_p = \left(\sum_{i=1}^{n} |\tilde{x}_i - x_i|^p \right)^{1/p},$$

where p is typically set to 2 or ∞. One then sets a specific threshold for a given data set and attempts to create models with the guarantee that their predictions will not change when inputs are perturbed within that radius. This particular formalization is known as *the L_p threat model*. Table 2.1 gives an overview of commonly used bounds for popular data sets. Although these bounds are obviously somewhat arbitrary, the idea behind them is to capture the threshold of "perceptibility." That is, for each data set, these bounds are supposed to guarantee that any additive perturbation of a sample within that radius does not visibly alter it.

Although many publications in the adversarial ML literature will insist that this particular threat model is of paramount importance for deep reasons related to cybersecurity, it is in fact merely a historical accident that the ML community became so obsessed with it. Goodfellow himself has stated that the original idea behind their work [Szegedy et al., 2013] was to study how "smooth" the internal representations of neural networks actually were.[2] To that end, the objective of

[2] See his IEEE S&P 2018 talk, *Defense against the dark arts*. Incidentally, this was only the second of two major findings of their original paper [Szegedy et al., 2013], the first one being related to how neural networks encode semantic information in their latent spaces. Despite being the primary result of the paper, this finding is almost never discussed anymore, and people seem to have forgotten that the paper talked about anything other than adversarial examples.

Data set	L_2 threshold	L_∞ threshold
MNIST	1.5	0.3
Fashion-MNIST	1.5	0.1
CIFAR-10	128/255	8/255
CIFAR-100	128/255	8/255
ImageNet	0.05	4/255

Table 2.1: Some common L_p thresholds for adversarial robustness on popular data sets. These thresholds assume that the pixel values have been scaled to the range $[0, 1]$.

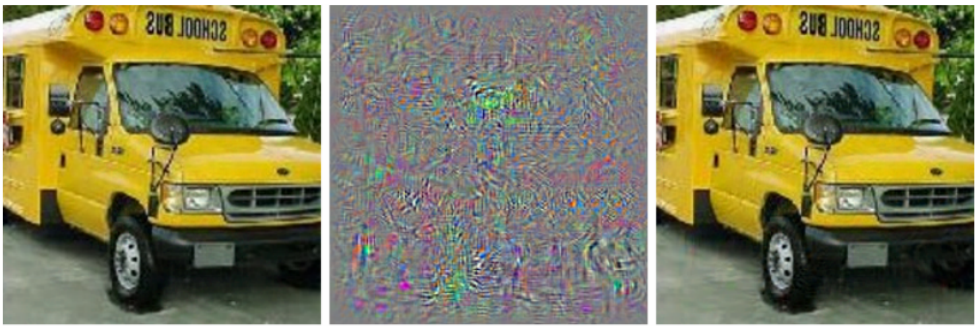


Figure 2.4: The school bus ostrich due to Szegedy et al. [2013]. This image is permanently burned into the retinas of anyone active in the field of adversarial ML. The image on the left is correctly classified as a school bus by AlexNet [Krizhevsky et al., 2012], whereas the image on the right is classified by the same network as an ostrich. The only difference between the two is the perturbation in the middle, which is almost invisible to the naked eye.

Szegedy et al. was to take an existing image and have it be interpolated between different classes by maximizing the loss of the model under an additive perturbation. If one started with an image of, say, an ostrich, and maximized the loss of the classifier under an additive perturbation of this ostrich, it may eventually be smoothly morphed into a different class, such as a school bus. At one point during the optimization, one would therefore expect to have an image of something resembling an eldritch abomination where the ostrich is merged with the school bus. Szegedy et al. discovered purely by accident that this is not what happens. Instead, the optimization terminates very quickly, and one obtains a very small perturbation that does not make any visible changes to the image, yet the output of the classifier is changed. Figure 2.4 illustrates this "school bus ostrich" example.

It is important to note that Szegedy et al. did not introduce this phenomenon from the perspective of cybersecurity; this is only a post-hoc motivation that researchers in this field added on later. Instead, Szegedy et al. approached adversarial examples from a generalization point of view:

> *The existence of the adversarial negatives appears to be in contradiction with the network's ability to achieve high generalization performance. Indeed, if the network can generalize well, how can it be confused by these adversarial negatives, which are indistinguishable from the regular examples?*

The field has since exhibited a remarkable form of "overfitting" on this original work, taking the exact same toy optimization problem posed by Szegedy et al. and making it the central object of study of *thousands* of papers[3] despite this clearly not being the point of the work at all. Rather, in my opinion, the implication of Szegedy et al.'s work is much broader: they have shown that DNNs appear to rely heavily on irrelevant features of the input despite achieving high accuracy on unseen data. The L_p threat model is merely the simplest possible mathematically tractable formalization of this problem.

As such, to stay true to the spirit of the original work by Szegedy et al. [2013], I propose the following more general definition of an *adversarial example*. Given a function $f : \mathcal{X} \to \mathcal{Y}$, constants[4] $\varepsilon_{\mathcal{X}} > 0$ and $\varepsilon_{\mathcal{Y}} > 0$ and an element $x \in \mathcal{X}$. An element $\tilde{x} \in \mathcal{X}$ is said to be $(\varepsilon_{\mathcal{X}}, \varepsilon_{\mathcal{Y}})$-*adversarial* to f and x if the following properties are satisfied:

1. $d_{\mathcal{X}}(x, \tilde{x}) \leq \varepsilon_{\mathcal{X}}$,
2. $d_{\mathcal{Y}}(f(x), f(\tilde{x})) \geq \varepsilon_{\mathcal{Y}}$.

Here, $d_{\mathcal{X}} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}_+$ and $d_{\mathcal{Y}} : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_+$ are similarity measures in the input space $\mathcal{X}$ and output space $\mathcal{Y}$ respectively. These functions do *not* need to satisfy any properties of a metric; they merely need to map pairs of inputs and outputs to non-negative real numbers as a means to quantify their "similarity," however that concept may be defined for the particular task at hand. Mathematically speaking, of course, it would be more convenient to have $d_{\mathcal{X}}$ and $d_{\mathcal{Y}}$ satisfy all properties of metrics, as this would facilitate proofs of lower and upper robustness bounds. In practice, however, we may simply not have such a luxury. For example, in the image domain, we may wish to quantify the distance $d_{\mathcal{X}}(x, \tilde{x})$ using some approximation of visual similarity according to the human visual system, which may not yield a valid metric. In the domain of natural language processing, we may be looking at vectorized word embeddings and quantifying their distance according to cosine similarity, which is known not to be a metric.

The above definition easily specializes to the typical L_p threat model used in the vast majority of the literature: we simply take $d_{\mathcal{X}}(x, x') = \|x - x'\|_p$, $d_{\mathcal{Y}}(y, y') = 1[y \neq y']$ and $\varepsilon_{\mathcal{Y}} = 1$. However, it can also capture much more general classes of adversarial threat models. For instance, if $\mathcal{X}$ is the image manifold corresponding to the data distribution and $d_{\mathcal{X}}$ is the geodesic distance between two points on this manifold, then an adversarial sample could be a much more complicated transformation of the original sample: we would not be limited to small additive perturbations but could also use rotations, reflections, shearing transforms and other distortions unique to images. As a tractable proxy for this geodesic distance (which may be very hard to estimate), we could use the structural similarity index metric (SSIM) proposed by Wang et al. [2004]. It has been known for many years now that L_p norms are a poor proxy for human perception [Wang and Bovik, 2009], so it is highly desirable to look at the adversarial robustness problem from this more general lens. There have been some papers that attempted to achieve this, such as Hendrycks et al. [2020] who proposed *DeepAugment*. This method perturbs the intermediate representations of a generative image-to-image model in order to create adversarial examples that go beyond the L_p threat model (see figure 2.6).

[3] See Carlini's list of papers on adversarial examples at `https://nicholas.carlini.com/writing/2019/all-adversarial-example-papers.html`. Accessed 2021-09-20.

[4] We may consider an alternative definition where $\varepsilon_{\mathcal{X}}$ and $\varepsilon_{\mathcal{Y}}$ are not constant but may depend on x and f. Although I believe this would be an interesting generalization, almost nobody in the adversarial ML literature actually looks at the problem in this way.

That said, the vast majority of papers still make use of this threat model, and I will be no different. There is something to be said about the L_p threat model, of course: if we cannot even solve the problem of robustness in this limited toy setting, we have absolutely no hopes of solving it for more complicated classes of perturbations. Regarding the specific choices for p, the most common are the following:

- $p = 0$. Although this does not induce a true metric, the L_0 "distance" between two images measures the number of pixels that differ between them. The size of the perturbation is therefore not constrained on a per-pixel level; individual pixels may be modified arbitrarily. It is only the *number* of pixels modified that matters.
- $p = 2$. The L_2 threat model measures the perturbation size as the Euclidean distance between images. This corresponds to the mean-squared error metric which is common in image processing.
- $p = \infty$. For L_∞, it is only the *maximum* per-pixel modification that matters. This threat model does not take into account the number of pixels modified, but it does constrain the maximum amount by which any pixel may deviate from the original value.

There is also another threat model, known as *patch attacks*, which is becoming increasingly popular [Levine and Feizi, 2020]. In a patch attack, an adversary is constrained to rectangular patches of fixed dimension. There is no constraint on the per-pixel deviation, so individual pixels may be modified arbitrarily, but the modifications must all be done within a rectangular patch of fixed size that may appear anywhere in the image. Although similar to L_0, it is argued that this threat model is more realistic than any L_p norm. Figure 2.5 shows an example of a successful patch attack proposed by Thys et al. [2019] against an older version of the YOLO detector [Redmon and Farhadi, 2017].

2.3 Adversarial attacks

In this section, I survey some of the notable adversarial attacks on DNNs that have been proposed since the work of Szegedy et al. [2013]. In general, adversarial attacks may be roughly classified according to the amount of information about the target they require. The taxonomy I will use in this work is as follows:

I. White-box, gradient-based. The attack requires complete knowledge of the model to be attacked, including its exact architecture and weights as well as the ability to evaluate the gradients of the model on arbitrary inputs.

II. White-box, gradient-free. The attack requires knowledge of the architecture and weights of the model used by the victim, but does not need its gradients.

III. Black-box, surrogate-based. The attack does not need the original model, but requires access to a *surrogate*, *i.e.*, a model trained on similar data as the target.

IV. Black-box, score-based. The attack does not need knowledge of the model, but requires the ability to obtain the probability scores assigned by the model on arbitrary inputs.

V. Black-box, decision-based. The attack does not need knowledge of the model, but requires the ability to obtain the predicted class of arbitrary inputs.

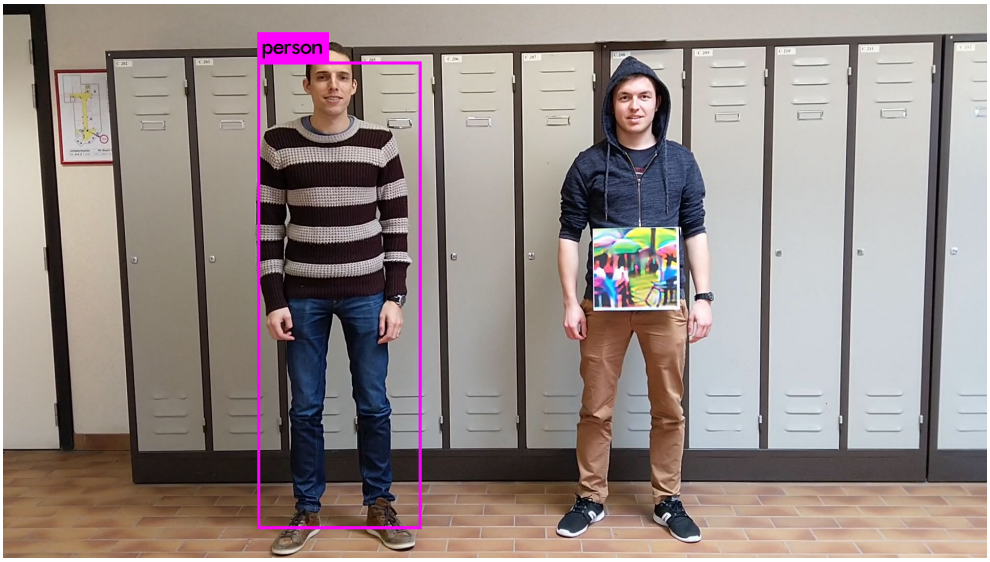


Figure 2.5: A patch attack that can successfully hide people from the YOLOv2 detector [Redmon and Farhadi, 2017]. Left: The person without a patch is successfully detected. Right: The person holding the patch is ignored. Image taken from Thys et al. [2019].

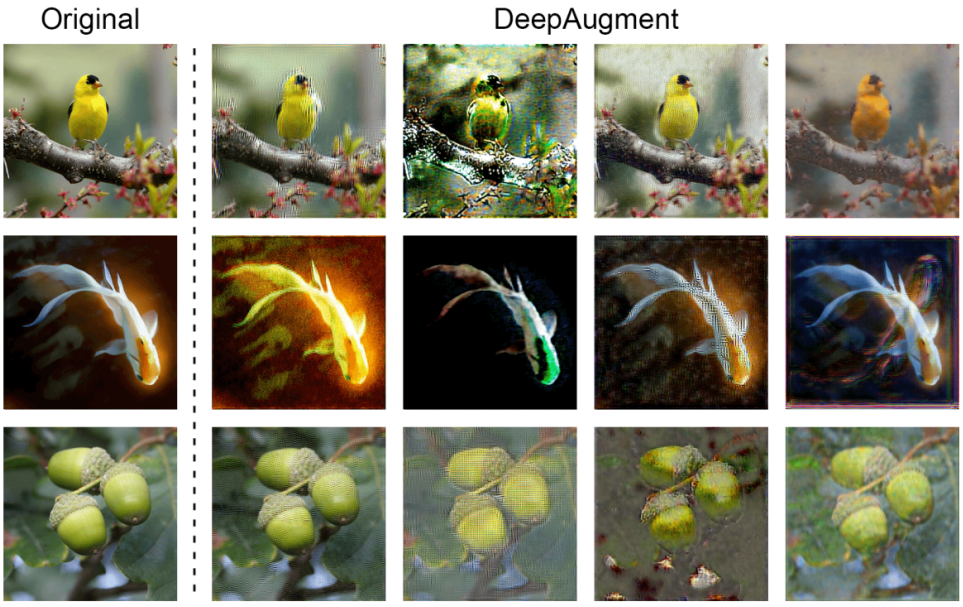


Figure 2.6: Examples of the DeepAugment technique proposed by Hendrycks et al. [2020].

The idea behind these categories is to rank the adversarial attacks according to the effort they require of the adversary in terms of gathering intelligence on their target. Category I attacks require full access to the entire model and are therefore clearly at the top of this hierarchy. Category II attacks do not require gradient information, but they may require other architectural details, so they rank slightly below category I. Category III attacks are based on surrogate models, so the attacker does not necessarily have to know anything about the target model except for the general type of data on which it was trained. They allow the adversary to train their own models for this purpose, and try to guarantee that the adversarial examples generated for the surrogate will also work on the real target. However, since category III attacks still require the adversary to gather sufficient data comparable to the training data of the target, I have ranked them above score-based and decision-based black-box attacks, since these do not require the adversary to train an entire model. Although this decision may be subject to some debate, I believe it is generally harder for an adversary to train their own models than it is to obtain predictions from their target. Score-based attacks rank above decision-based attacks for the obvious reason that obtaining real scores from a model gives much more information than just the label itself.

Aside from the categories above which quantify the level of information required for the attack to operate, we can further characterize adversarial attacks according to the following properties:

- **Targeted.** The attack requires the user to specify a target class y_t beforehand. The adversarial perturbation will then be optimized to push the sample towards this specific class. The attack is successful only if $f(\tilde{x}) = y_t$.
- **Untargeted.** The attack does not require the user to specify a target class. The adversarial perturbation is merely optimized to induce a difference in classifications, *i.e.*, $f(\tilde{x}) \neq f(x)$. The precise output of the model on the adversarial sample is not relevant.

There is no real preference ranking for targeted and untargeted attacks as there might be for the other categories; whether a user prefers a targeted attack or an untargeted one entirely depends on the use case. For example, if a user wishes to impersonate a specific employee to fool a biometric security system, a targeted attack is required since a specific classification result must be reached. On the other hand, if the user wants to cause a computer vision system to not recognize a specific object, an untargeted attack suffices since the exact result is irrelevant as long as it differs from the ground truth. Furthermore, any targeted attack can (in principle) be converted to an untargeted one by simply running the attack for every class except the original and returning one of the successful results. However, attacks that were designed to be untargeted will generally be much more efficient than this.

The earliest adversarial attacks developed against DNNs were category I: white-box and gradient-based. As time went on, of course, researchers crafted increasingly sophisticated algorithms that were able to attack DNNs in ever more restricted settings. I discuss a few of these attacks below.

2.3.1 L-BFGS

One of the first attacks to be created for DNNs is known as the *L-BFGS attack*. It was proposed by Szegedy et al. [2013] and so-named because it is merely an application of the limited-memory Broyden–Fletcher–Goldfarb–Shanno (BFGS) optimization algorithm [Liu and Nocedal, 1989] to the following problem:

$$\min_{\boldsymbol{\delta}} \; c\|\boldsymbol{\delta}\| + \mathcal{L}(x+\boldsymbol{\delta}, y_t, f) \text{ subject to } x+\boldsymbol{\delta} \in [0,1]^n. \tag{2.1}$$

Here, y_t is the target class which must be specified beforehand. The parameter $c > 0$ is tuned via a line search so that the smallest value is used for which the minimizer $\boldsymbol{\delta}$ of (2.1) satisfies $f(x+\boldsymbol{\delta}) = y_t$. It is this algorithm that created the infamous "school bus ostrich" shown in figure 2.4. It is a **targeted white-box gradient-based** attack, as we must specify a target class y_t and L-BFGS requires access to the gradients of the objective function. In this case, the objective function includes the loss of the model on arbitrary samples, and so the attack requires full access to the target model including its gradients.

2.3.2 Fast gradient sign

The *fast gradient sign* (FGS) attack was proposed by Goodfellow et al. [2014] in order to make the generation of adversarial examples much more efficient than the L-BFGS method. It is an **untargeted white-box gradient-based** attack. To design it, Goodfellow et al. start from the following optimization problem:

$$\max_{\boldsymbol{\delta}} \ \mathcal{L}(x+\boldsymbol{\delta}, y, f) \text{ subject to } \|\boldsymbol{\delta}\|_\infty \leq \varepsilon_\mathcal{X}. \tag{2.2}$$

Note that (2.2) is specialized to the L_∞ norm. They then consider a first-order Taylor approximation of the loss term:

$$\mathcal{L}(x+\boldsymbol{\delta}, y, f) \approx \mathcal{L}(x, y, f) + \boldsymbol{\delta}^\mathsf{T} \nabla_x \mathcal{L}(x, y, f).$$

Assuming this linear approximation is accurate, we can choose the perturbation as follows:

$$\boldsymbol{\delta} = \varepsilon_\mathcal{X} \operatorname{sgn} \nabla_x \mathcal{L}(x, y, f). \tag{2.3}$$

We then clearly have $\|\boldsymbol{\delta}\|_\infty = \varepsilon_\mathcal{X}$, satisfying our norm constraint. Furthermore, if we plug this value into the Taylor approximation, we obtain

$$\mathcal{L}(x+\boldsymbol{\delta}, y, f) \approx \mathcal{L}(x, y, f) + \varepsilon_\mathcal{X} \|\nabla_x \mathcal{L}(x, y, f)\|_1.$$

If the original sample (x, y) is not a stationary point of the loss, then the 1-norm of the gradient will likely be proportional to the dimensionality of the data. Therefore, in high dimensions, merely following the sign of the gradient vector can lead to large increases in loss even though the input perturbation is very small. Indeed, experiments using the FGS method were highly successful, achieving very high error rates with relatively small values of $\varepsilon_\mathcal{X}$. It was also very fast, since the computation of the perturbation $\boldsymbol{\delta}$ in (2.3) requires only a single backward pass through the network per sample. These factors combined made FGS one of the most popular adversarial attacks for many years.

2.3.3 Projected gradient descent

Similar to L-BFGS, the *projected gradient descent* (PGD) attack is named after an existing general-purpose optimization algorithm that has simply been specialized to the crafting of adversarial perturbations. Specifically, PGD finds an adversarial example by iterating the following update:

$$\tilde{x}_{t+1} \leftarrow \mathcal{P}_{x+\mathcal{S}}(\tilde{x}_t + \alpha \operatorname{sgn} \nabla_x \mathcal{L}(x, y, \theta)). \tag{2.4}$$

Here, $\alpha > 0$ is a user-specified constant, $\mathcal{S}$ is an appropriate L_p norm ball centered at the origin and $\mathcal{P}_U(x)$ denotes the projection of x onto U. Depending on the choice of norm, the projection

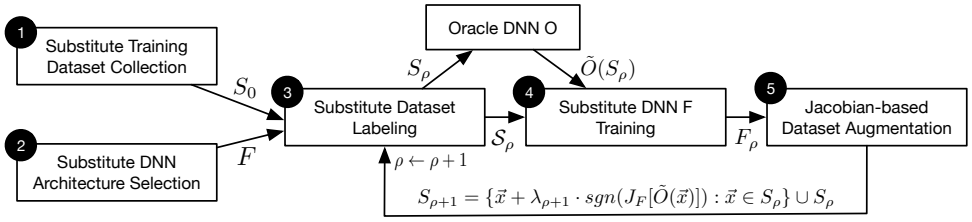


Figure 2.7: The methodology of Papernot et al. [2017] for attacking a surrogate model.

operator $\mathcal{P}_{x+\mathcal{S}}$ can take very different forms. In general, it is the solution to an optimization problem that finds a point $u \in U$ closest to x according to the particular norm:

$$\mathcal{P}_U(x) = \underset{u \in U}{\operatorname{argmin}} \ \|x-u\|_p.$$

In the case of L_∞, the projection can be accomplished merely by clipping the components of the perturbed sample within the admissible range; for L_2, an orthogonal projection onto the ball $x+\mathcal{S}$ must be carried out.

The PGD attack was made famous by Madry et al. [2017], who performed an extensive analysis of its theoretical and empirical properties. They argued that PGD is essentially an optimal first-order adversary, in the sense that no other *efficient* attack algorithm that uses only gradients of the loss to construct perturbations could outperform it in terms of success rate. Stronger adversaries would therefore either have to use higher-order information, which is notoriously expensive to compute for DNNs, or they would need to perform more expensive computations using the first-order gradient information. Either way, they would be considerably less efficient than PGD. Based on this insight, they used PGD to construct robust models for MNIST and CIFAR-10. These models remained remarkably robust for some time, although the CIFAR-10 model was eventually broken.

2.3.4 Mimicry attack

Concurrently to Szegedy et al. [2013], Biggio et al. [2013] were also working on adversarial examples in the context of DNNs and support vector machines. They developed a general adversarial attack algorithm that tries to find the smallest additive perturbation to an input such that the classification of the model changes, but the sample also remains within the support of the original data distribution. Formally, they consider the following problem:

$$\underset{\tilde{x} \in \mathcal{X}}{\operatorname{argmin}} \ \hat{f}(\tilde{x}) - \frac{\lambda}{m} \sum_{i:y_i=-1}^{m} \kappa\left(\frac{\tilde{x}-x_i}{h}\right) \text{ subject to } d_{\mathcal{X}}(x,\tilde{x}) \leq \varepsilon_{\mathcal{X}}. \tag{2.5}$$

Here, $\hat{f}$ is a surrogate model trained by the adversary, $\lambda > 0$ is a regularization parameter, m is the number of benign samples available to the adversary, κ is a kernel and h is its bandwidth. It is a **targeted black-box surrogate-based** attack.

Biggio et al. already considered the possibility that the adversary may not have full access to the model f used by the victim nor the full data set on which f was trained. They therefore account for the fact that we may be using a surrogate model that we trained ourselves, with the hopes that

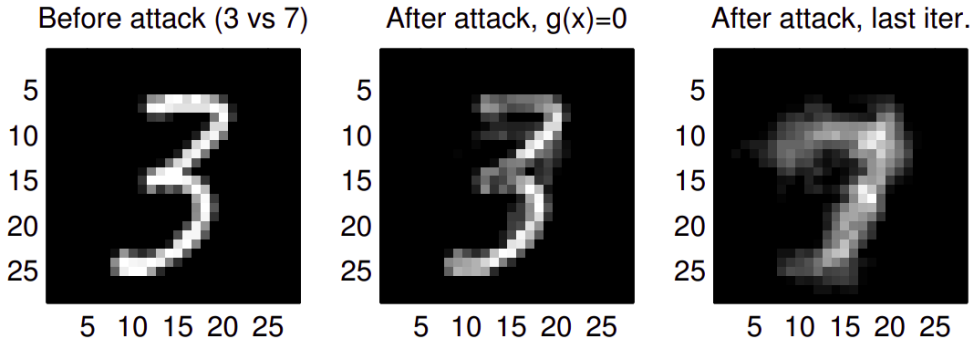


Figure 2.8: The mimicry attack by Biggio et al. [2013]. If this attack is run for sufficiently many iterations, the adversarial sample eventually morphs into the target class.

adversarials generated for the surrogate will also work on the real target. This methodology was further developed by Papernot et al. [2017] and is illustrated in figure 2.7. Essentially, to create a suitable surrogate, one first collects a sufficiently large number of unlabeled data samples and queries the target on this data. The predictions returned by the target are then used to create a labeled data set on which we can train our own model. This model should then resemble the target well enough that adversarials generated for the surrogate are likely to also work on the target.

Biggio et al. also account for the fact that the norm constraint by itself may not be sufficient to guarantee that the adversarial sample will lie close to the original data manifold. Since this might cause the adversarials to be detected by the victim, a regularization term is added to the objective which penalizes the distance of the generated sample $\tilde{x}$ to the data samples $x_1, \ldots, x_m$ according to a kernel density estimate. They refer to this technique as "mimicry."

Figure 2.8 shows an example of this mimicry attack on the MNIST data set. Although the attack can already cause misclassification with almost imperceptible modifications, allowing the algorithm to optimize to complete convergence produces samples that visibly morph into their target classes. This is in stark contrast to other contemporaneous attacks such as projected gradient descent and fast gradient sign, which will never produce samples that resemble their target classes at all. This suggests that adversarial attacks which do not incorporate an explicit mechanism for keeping their samples on the data manifold will indeed produce adversarials that are qualitatively different from benign data. Such samples may be detected using statistical tests or uncertainty measures, as suggested in prior work [Gao et al., 2020, Grosse et al., 2017, Roth et al., 2019]. As a response to this, other researchers have proposed novel attacks that explicitly attempt to constrain the adversarials to lie on the original data manifold, such as *ManiFool* [Kanbak et al., 2018].

2.3.5 The Carlini-Wagner attacks

Carlini and Wagner [2017c] originally proposed three different attacks specifically to target *defensive distillation* [Papernot et al., 2016b], which was one of the only promising adversarial defenses at the time. They succeeded in reducing the robust accuracy of defensive distillation to 0%, a result that gained immediate notoriety as it was the first complete "break" of an established

adversarial defense. The three attacks were each designed for a specific threat model, in this case L_0, L_2 and L_∞ respectively. This set of attacks has become known in the literature as *the* Carlini-Wagner (C&W) attack, despite the fact that there are actually three of them and they are certainly not identical.

The basic insight that led to the C&W attacks is that one must very carefully craft an appropriate objective function in order to create effective adversarial samples. Specifically, Carlini and Wagner propose a general framework where one starts from the following optimization problem:

$$\min_{\boldsymbol{\delta}} \; d_X(x, x+\boldsymbol{\delta}) + \lambda g(x+\boldsymbol{\delta}). \tag{2.6}$$

Here, $\lambda > 0$ is a tunable parameter and g is a function with the property that

$$f(x+\boldsymbol{\delta}) = y_t \iff g(x+\boldsymbol{\delta}) \leq 0. \tag{2.7}$$

If g satisfies (2.7), then minimizing g is a consistent proxy for optimizing $\boldsymbol{\delta}$ such that the target classifier f outputs the desired class y_t. The basic C&W attacks are all therefore **targeted white-box gradient-based** types, although Carlini and Wagner proposed untargeted variants as well.

Clearly, the main design consideration for the C&W attacks is the specific choice of g in (2.6). Carlini and Wagner experiment with many different options, but the most effective one turned out to be

$$g(x') = \max\left\{\max_{i \neq t} Z_i(x') - Z_t(x'), -\kappa\right\}.$$

Here, $Z(x)$ represents the vector of logits of the model on the given sample x and $\kappa \geq 0$ is a constant that controls the confidence of the resulting adversarial. Essentially, the C&W attacks optimize the perturbation $\boldsymbol{\delta}$ such that the resulting adversarial $x+\boldsymbol{\delta}$ has a higher logit value for the target class than any of the other classes, leading to a targeted misclassification. Increasing κ causes the attack to generate adversarial samples with higher confidence in the target class, which can help them transfer to other models.

We can interpret (2.6) as the Lagrangian relaxation of the constrained optimization problem

$$\min_{\boldsymbol{\delta}} d_X(x, x+\boldsymbol{\delta}) \text{ subject to } g(x+\boldsymbol{\delta}) \leq 0. \tag{2.8}$$

Due to (2.7), problem (2.8) is itself a relaxation of

$$\min_{\boldsymbol{\delta}} d_X(x, x+\boldsymbol{\delta}) \text{ subject to } f(x+\boldsymbol{\delta}) = y_t, \tag{2.9}$$

which is the quintessential optimization problem for adversarial attacks. Now, if $\boldsymbol{\delta}$ is any feasible solution to (2.9), it holds that

$$d_X(x, x+\boldsymbol{\delta}) + \lambda g(x+\boldsymbol{\delta}) \leq d_X(x, x+\boldsymbol{\delta}).$$

That is, for any feasible solution $\boldsymbol{\delta}$ (*i.e.*, such that $f(x+\boldsymbol{\delta}) = y_t$), the objective value of (2.6) is a lower bound on the objective value of (2.9). To obtain the best possible approximation of an optimal solution to (2.9) via the relaxation (2.6), this bound needs to become as tight as possible. This means that the parameter λ must be minimized, because $g(x+\boldsymbol{\delta}) \leq 0$. Indeed,

Carlini and Wagner confirm this empirically, and therefore extend their attack algorithm with a modified binary search procedure to choose the minimal constant λ for which the optimizer can still find feasible solutions.

The Carlini-Wagner attacks are historically significant in this field for two reasons. They were the first attacks that broke an established defense that seemed highly promising: at the time, defensive distillation was able to reduce attack success rates from 95% (*i.e.*, 5% robust accuracy) to 0.5% (99.5% robust accuracy). They are relatively slow, however, requiring many iterations to reach a good solution. Moreover, since they optimize over the logit space, high-confidence adversarials produced by the C&W attacks may lead to "over-optimized" perturbations that can be easily identified via IQR-thresholding of the logit values, since such values are atypical of benign samples [Ozbulak et al., 2019].

2.3.6 Randomized gradient-free attack

The randomized gradient-free attack [Croce et al., 2020] is notable for being perhaps the only existing **white-box gradient-free** attack: it requires access to the entire model specification yet does not use gradient information. The algorithm exploits the fact that DNNs with RELU activation functions divide their input spaces into *linear regions*, *i.e.*, connected subsets where the DNN reduces to a linear function. This property was demonstrated, *e.g.*, in Arora et al. [2016]. Croce et al. make use of an explicit construction of these linear regions in order to find minimal perturbations that cause a given sample to cross a decision boundary. The resulting optimization problem is a quadratic program that requires only the parameters of the network, but not its gradients with respect to arbitrary inputs.

One might wonder why it is useful to make the distinction between category I and category II if the latter category is so sparsely populated. The reason is that the vast majority of category I attacks work by optimizing the additive perturbation $\boldsymbol{\delta}$ using gradient descent over some function of the loss of the model. Thus, they crucially rely on the gradients of the loss with respect to the model parameters or input. However, as shown by Athalye et al. [2018a], category I attacks can severely overestimate the true robustness of their targets due to a phenomenon known as *gradient masking*. This occurs when a model performs certain operations that are not differentiable, such as JPEG compression or randomization. These distort the gradient signal used by category I attacks and can easily cause them to fail to converge to any satisfactory solution. Gradient masking can also arise when adversarial defenses consist of explicitly penalizing the magnitude of the gradient itself. One of the earliest adversarial defenses proposed for DNNs was the *deep contractive network* by Gu and Rigazio [2014] which worked in this manner: the defense explicitly penalized the norm of the gradient during training and was only evaluated against the L-BFGS attack by Szegedy et al. [2013]. Therefore, it is highly likely that the increased robustness observed by Gu and Rigazio was due to gradient masking.

Essentially, gradient masking defenses cause the models to have near-zero loss on training data samples, such that category I attacks starting from such samples will not converge due to a lack of useful gradient information. This is especially obvious in the case of the FGS attack, which will not perturb the original sample *at all* if the gradient of the loss at that sample is zero. As such, one could defend against FGS by encouraging the model to have vanishing gradients near the data samples. However, it is a mistake to assume that gradient masking actually improves robustness. This belief is mistaken, because as shown by Athalye et al. [2018a], the robustness supposedly gained from gradient masking defenses quickly vanishes when the attacks are

modified to accommodate this problem. Athalye et al. themselves suggested a technique known as *backward pass differentiable approximation* (BPDA), which circumvents gradient masking by approximating non-differentiable components of the target network with differentiable operations. For most defenses, the application of BPDA reduces to a simple substitution where the non-differentiable components are approximated by the identity function. BPDA is thus a method for adapting category I attacks such that they can also work even when the model is masking gradients. The motivation behind category II attacks such as the randomized gradient-free attack by Croce et al. [2020] is that it would be better if the attack was never vulnerable to gradient masking to begin with, so that these types of "band aids" become unnecessary. Indeed, although techniques such as BPDA allow us to still craft adversarial examples using category I attacks against gradient masking defenses, there is no telling to what extent the masked gradients may still be causing us to overestimate robustness. Hence the need for designing white-box attacks that do not need to use any gradients of the model.

2.3.7 Universal adversarial perturbations

Bringing the phenomenon of adversarial examples to its logical extreme, Moosavi-Dezfooli et al. [2017] introduced *universal adversarial perturbations* (UAPs). A UAP is a *single* perturbation $\boldsymbol{\delta}$ with the property that $x+\boldsymbol{\delta}$ is adversarial for *any* model $f : \mathcal{X} \to \mathcal{Y}$ and *any* sample $x \in \mathcal{X}$. Figure 2.9 shows an example taken from Moosavi-Dezfooli et al. [2017].

We give the pseudo-code for the UAP algorithm in algorithm 2.1. The function $\mathcal{P}$ denotes the projection operator onto the L_p ball of desired radius. Essentially, all this algorithm does is create adversarial perturbations for each of the individual data samples and aggregate them all together into a single one. Note that any adversarial attack can in principle be used on line 4, and so the specific category to which this attack belongs can be subject to some debate. Based on the original implementation by Moosavi-Dezfooli et al. [2017], one could classify it as **untargeted black-box surrogate-based**.

Algorithm 2.1: Universal adversarial perturbations

Data: data set S, classifier f
Result: universal adversarial perturbation $\boldsymbol{\delta}$

1 **while** *fooling rate is too low* **do**
2 **for** *sample* $x \in S$ **do**
3 **if** $f(x+\boldsymbol{\delta}) = f(x)$ **then**
4 Generate an adversarial perturbation $\boldsymbol{\eta}$ for $x+\boldsymbol{\delta}$ and f.
5 Update the universal perturbation:

$$\boldsymbol{\delta} \leftarrow \mathcal{P}(\boldsymbol{\delta}+\boldsymbol{\eta}).$$

6 **end**
7 **end**
8 **end**
9 **return** $\boldsymbol{\delta}$

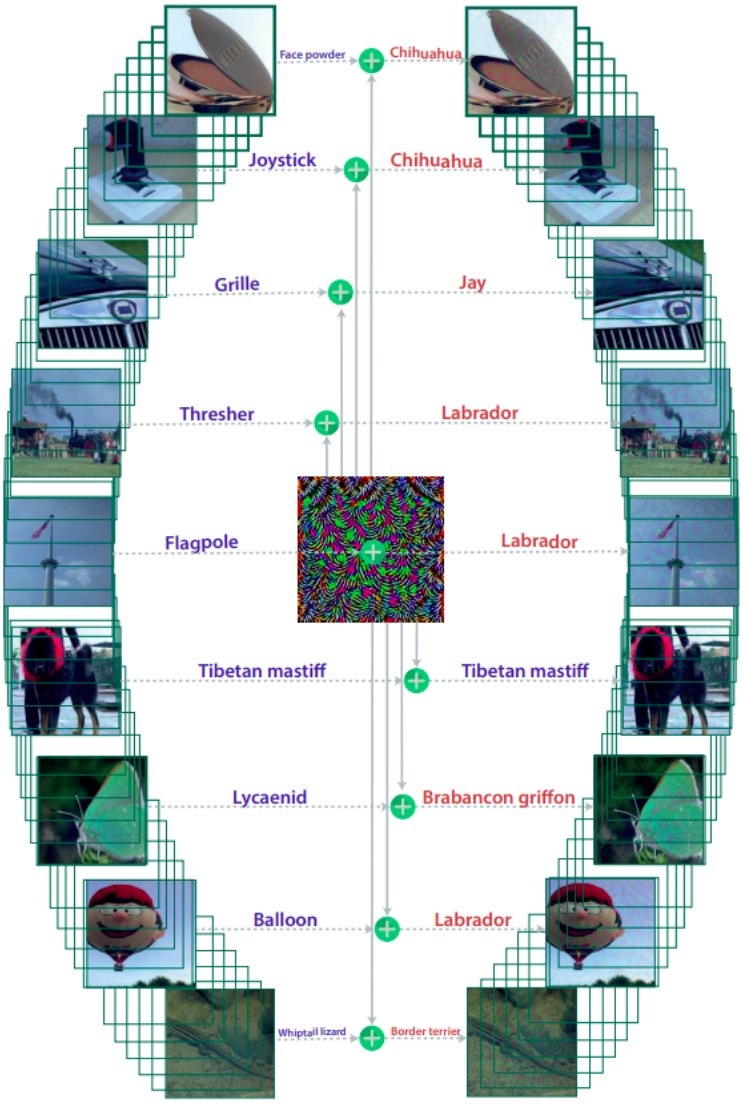


Figure 2.9: Example of a universal adversarial perturbation created by Moosavi-Dezfooli et al. [2017]. The *single* perturbation shown in the middle can cause a model to confuse most of the samples on the left with their erroneous classifications on the right.

2.3.8 Other black-box attacks

Due to their practical usefulness, many black-box attacks have been proposed over the years, and it is not feasible to survey them all here. I will therefore conclude my discussion of black-box attacks with a few "honorable mentions," to which I will not dedicate an entire subsection.

The *Simple Black-box Attack* (SimBA) is a **black-box score-based** attack proposed by Guo et al. [2019]. As its name implies, it is an exceedingly simple attack to implement and carry out: in order to perturb a given sample, SimBA randomly samples a vector from a pre-defined orthonormal basis and either adds or subtracts it to the input. If the user chooses the standard basis in $\mathbb{R}^n$, SimBA will perturb only a single randomly chosen pixel at a time. A more interesting choice of basis is the discrete cosine transform (DCT) basis, which makes the algorithm very effective and query-efficient. This strategy calls to mind the *Fourier perspective* on model robustness argued by Yin et al. [2019b], which holds that adversarial examples arise from reduced robustness of models to perturbations in the low frequency domain.

Chen et al. [2017] proposed the *Zeroth Order Optimization* (ZOO) attack, which is **black-box score-based** (both targeted and untargeted). ZOO can be viewed as an attempt to "lift" the problem of attacking a black-box model to the problem of attacking a white-box one. It accomplishes this by using the symmetric difference quotient to numerically estimate the gradient of the target model and optimizing a loss function similar to the Carlini-Wagner attack.

The *square attack* by Andriushchenko et al. [2020] is notable for being one of the most efficient and strong **untargeted black-box score-based** attacks proposed to date. It has been included in the AutoAttack framework, a comprehensive benchmark for evaluating robustness of models proposed by Croce and Hein [2020b] at ICML 2020. This framework is the foundation of the RobustBench leaderboard,[5] which has become the *de facto* standard reference for recording state of the art robustness results.

Brendel et al. [2017] proposed the *boundary attack*, a **black-box decision-based** type (targeted and untargeted). Together with *GeoDA* [Rahmati et al., 2020], these attacks are notable for being the only category V types that appear to have been described in the literature to the best of my knowledge.

2.3.9 Real-world attacks

The attacks discussed in the previous sections are all focused on a "lab setting." That is, they all assume we can perfectly manipulate a given input and this input will be provided to the model *exactly* as we crafted it. For many applications of machine learning, this is unrealistic: think, for example, of an autonomous vehicle that uses computer vision to recognize traffic signs. The inputs to this system are images of the road, but these images can vary from moment to moment due to all sorts of factors. There might be people or other obstacles that obscure parts of the signs, or the signs may have been physically damaged somehow. Weather conditions, seasons and time of day all also have an impact on how traffic signs will be perceived by any vision system (be it human or machine).

To address these issues, the adversarial ML community has developed *real-world attacks* which are designed to cope with typical distortions to which the inputs might be subjected. Seminal work in this direction was done by Athalye et al. [2018b], who proposed a technique called

[5] `https://robustbench.github.io`. Accessed 2021-09-28.

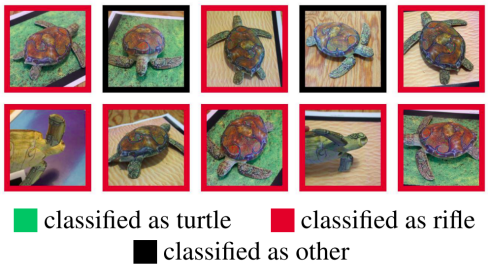


Figure 2.10: The "adversarial turtle" by Athalye et al. [2018b]. This 3D printed turtle figurine has a special texture applied to it constructed using the EOT technique. It causes computer vision systems to misclassify the turtle as a rifle under real-world conditions.

expectation over transformation (EOT). This technique has since become widely used not only to create real-world adversarial examples but also to increase robustness of models in general (even in the lab setting). The idea behind EOT is to model the distortions to which the input might be subjected as part of the optimization procedure. Formally, Athalye et al. define a distribution T of possible transformations. A transformation is simply a $\mathcal{X} \to \mathcal{X}$ function, so T is a distribution on *functions* of the input. Before being processed by the vision system, an input $x \in \mathcal{X}$ will be affected by some random transformation $t \sim T$. Hence, the system will actually observe $t(x)$ instead of x itself. The objective of EOT is therefore to solve a slightly different optimization problem:

$$\max_{\tilde{x} \in \mathcal{X}} \mathop{\mathbb{E}}_{t \sim T} [\log \Pr[y' \mid t(\tilde{x})]] \text{ subject to } \mathop{\mathbb{E}}_{t \sim T} [d_{\mathcal{X}}(t(\tilde{x}), t(x))] \leq \varepsilon. \tag{2.10}$$

Here, y' is a chosen target class and $\Pr[y \mid x]$ refers to the probability estimated by the target model for class y given input x. In practice, (2.10) is solved using a Lagrangian relaxation:

$$\max_{\tilde{x} \in \mathcal{X}} \mathop{\mathbb{E}}_{t \sim T} [\log \Pr[y' \mid t(\tilde{x})]] - \lambda \mathop{\mathbb{E}}_{t \sim T} [d_{\mathcal{X}}(t(\tilde{x}), t(x))] \leq \varepsilon, \tag{2.11}$$

where $\lambda > 0$ is a hyperparameter. Athalye et al. used projected gradient descent to solve (2.11), so their original attack is a **targeted white-box gradient-based** type.

The EOT algorithm allows for some truly impressive adversarial attacks. For instance, the main contribution of Athalye et al. [2018b] is to specialize EOT for affine transformations, allowing them to craft adversarial textures that can be applied to real-world (3D printed) objects, such as the famous "adversarial turtle" shown in figure 2.10. These textures cause the targeted vision system to misclassify the object even under various common real-world distortions, such as changes in pose, surrounding environment, lighting and camera angles.

Concurrently to the work of Athalye et al. [2018b], Eykholt et al. [2018] proposed real-world adversarial attacks on autonomous vehicle systems using a very similar framework as EOT which they called *robust physical perturbations* (RP_2). A typical example of this attack is shown in figure 2.11. The RP_2 method allows the creation of adversarial examples using simple black and white stickers that are easily applied to physical objects and which consistently fool computer vision systems under real-world conditions. Like EOT, it is a **targeted white-box gradient-based** attack.

Figure 2.11: The adversarial stop sign created by the RP_2 method of Eykholt et al. [2018]. By placing a few small black and white rectangular stickers in specific locations, the stop sign is misidentified by a computer vision system as a speed limit sign.

2.4 Adversarial defenses

Having discussed many of the attacks proposed against DNNs, we can look at a few of the defenses that have been described in the literature. Before we begin, however, the reader must be aware that exceptionally many defenses have been published. Indeed, new proposals appear on arXiv at a frequency of multiple papers per week. It is therefore not possible to discuss every single one, and I shall not attempt to do so. Instead, I have made a selection of adversarial defenses to present here that I personally believe are interesting or representative of the field.

Like adversarial attacks, the defenses described in the literature can be roughly classified into a few distinct groups. The taxonomy I propose here is based on the general strategy used to defend a model:

- **Purification**. The defense attempts to remove the adversarial perturbation from the input and restore the sample to its original form, which can then be processed by the underlying model as usual. This is a highly popular form of defense and is typically implemented via specialized auto-encoder architectures. Note, however, that these sorts of defenses assume that the adversary works in a particular way, namely by first sampling a natural sample from the original data distribution and subsequently perturbing it to cause misclassification. This is not necessarily the case in practice: as shown by Hein et al. [2019], Nguyen et al. [2015], for example, DNNs can also give high-confidence erroneous predictions for input samples that have no recognizable semantics and certainly do not come from the original data distribution. In case there is no perceptibility constraint on the adversarial sample (a realistic scenario as per Gilmer et al. [2018]), it is not clear what robustness could be achieved by purification defenses.

- **Detection**. These defenses focus on detecting adversarially manipulated inputs before

they are processed by the model. They do not attempt to correct the prediction of the model in any way; instead, they extend the set of model outputs to include a special "reject" signal which implies that the model is likely not reliable on the given sample. How one handles such rejection depends on the application: the sample could be flagged for manual review by human moderators or it could be passed on to more computationally expensive models. It could also trigger the system to resort to some "safe" default option, such as denying the upload of a file (in the case of content filtering mechanisms) or denying access to a facility (in the case of biometric security systems). Adversarial detectors often use techniques inspired from *out-of-distribution* (OOD) detection, where the goal is to ascertain whether a given sample came from a known distribution or some other (unknown) distribution where the model cannot be trusted [Lee et al., 2018].

- **Hardening**. Hardening defenses do not attempt to remove the adversarial noise nor to detect when manipulation might have occurred. Instead, these defenses seek to make the model *inherently invulnerable* to adversarial perturbations.

Apart from these broad categories, adversarial defenses can also be **randomized** or **deterministic**. These distinctions are relevant, since randomized defenses need to be evaluated differently from deterministic ones [Croce and Hein, 2020b]. In the literature, the distinction is also often made between defenses that are **certified** versus those that are not. A certified adversarial defense is one for which a *certificate of robustness* can be produced. This is a mathematical proof that shows that no perturbation up to a given radius ε_X could possibly change the outcome of the classifier. This also must be taken into account during evaluation, because there is no point in trying to attack a certified defense using perturbations that lie within its certified radius (unless the certificate of robustness is wrong).

There are exceptions to this, however. In practice, many certificates of robustness are *probabilistic*, in the following sense. They assume that the adversary creates adversarial examples by first sampling a natural sample x from the data distribution on which the model is trained and then finds a perturbation $\boldsymbol{\delta}$ to cause misclassification. Probabilistic certificates of robustness then usually give the guarantee that, under this mode of operation, there is only a small probability (taken over the data distribution) that there exists such a perturbation $\boldsymbol{\delta}$. Although probabilistic certificates are clearly better than no certificates at all, they do not completely rule out the existence of adversarials within their radius of certification. Furthermore, these certificates typically suffer from the same problem as purification-based defenses: their guarantees become meaningless when the adversary does not respect their chosen mode of operation, as with rubbish examples [Nguyen et al., 2015].

2.4.1 Adversarial training

The most successful defense that has been described in the literature as of this writing is *adversarial training* (AT). The basic idea couldn't be more straightforward: simply include adversarial examples in the training pipeline as a form of data augmentation. In practice, of course, the devil is in the details, and the way we implement AT has changed throughout the years as the field has gained more insights into the factors and design decisions that can make or break the method.

The first variant of AT was proposed by Goodfellow et al. [2014], in the same paper where they introduced the FGS attack. Since FGS is so efficient, their AT scheme took the form of a regularized loss function that essentially performs the FGS attack against all samples in the

mini-batch and then updates the parameters of the model according to a weighted average of the loss on the original and adversarial samples:

$$\mathcal{L}(x,y,f) = \alpha J(x,y,f) + (1-\alpha)J(x+\varepsilon\,\mathrm{sgn}\,\nabla_x J(x,y,f),y,f).$$

Here, $\alpha \in [0,1]$ and J is the cross-entropy loss. Due to its extremely high efficiency and seemingly high effectiveness, FGS AT was the preferred method for increasing robustness for some time. A few years later, however, Kurakin et al. [2016] showed that FGS AT suffers from a "label leaking" effect: because the FGS adversarials are computed using the gradient of the loss on the *true* label, statistical artefacts are introduced that encode the true label in the adversarial image in a form that can be easily detected by DNNs. Thus, using FGS AT, adversarially trained models often have much higher accuracy on FGS adversarials than on the original, unaltered samples. Later, Carlini and Wagner [2017c] also found that FGS is actually a very weak attack, as FGS AT provides no real protection against stronger attacks such as PGD or even an iterated version of FGS. These findings caused FGS to fall out of favor both as an attack as well as the basis for any defense.

The next major innovation in the design of AT schemes came from Madry et al. [2017], who proposed a more principled way to implement it. They characterized the problem as one of *robust optimization* (RO) [Ben-Tal and Nemirovski, 2002]:

$$\theta^\star = \underset{\theta}{\operatorname{argmin}}\ \mathbb{E}\left[\max_{\delta \in B_\varepsilon} \mathcal{L}(X+\boldsymbol{\delta}, Y, \theta)\right], \tag{2.12}$$

where B_ε is the ε ball around the origin in the norm of choice. That is, instead of minimizing the expected loss on a data set of i.i.d. samples, we try to minimize the loss under *worst-case* norm-bounded additive perturbations of the input. The RO view therefore casts AT as a bi-level optimization problem:

1. **Inner maximization**. In this step, the current mini-batch of samples is subjected to an adversarial attack with respect to the current model parameters.
2. **Outer minimization**. After the adversarial examples have been constructed, the original mini-batch is replaced with their adversarial counterparts. These are then used in a subsequent optimizer to minimize the loss.

Madry et al. originally proposed PGD for the inner maximization, using various arguments to support the idea that PGD is an optimal first-order adversary. Using PGD AT, they constructed robust models for MNIST and CIFAR-10 that maintained state of the art robust accuracy in the L_∞ threat model for some time. They publicly launched the MNIST challenge[6] and the CIFAR-10 challenge[7] where researchers were invited to submit adversarial examples against their models, essentially crowd-sourcing their robustness assessments. The CIFAR-10 model was considered broken by December 2017 using a variant of one of the C&W attacks, which reduced robust accuracy at $L_\infty = 8/255$ to 47.76%. As of this writing, however, the MNIST model remains highly robust: even against the best known attack, it still has a robust accuracy of 88% at $L_\infty = 0.3$.

Thanks to the work by Madry et al. [2017], by the end of 2017 the field had realized that AT could be an incredibly powerful method of defense. The only real issue was *scale*: training

[6] `https://github.com/MadryLab/mnist_challenge`. Accessed 2021-10-11.
[7] `https://github.com/MadryLab/cifar10_challenge`. Accessed 2021-10-11.

a robust model on a large data set such as ImageNet using the method proposed by Madry et al. could easily take *weeks*. This caused researchers to experiment with more efficient methods, such as generating the adversarials using pre-trained generative models[Baluja and Fischer, 2017, Poursaeed et al., 2018, Xiao et al., 2018], but these approaches tend to be unstable on complex large-scale data sets. Other work has focused on making the base AT algorithm itself more efficient, with some remarkable recent results [Shafahi et al., 2019, Wong et al., 2020].

The precise classification of AT according to the taxonomy I proposed here depends on the exact implementation. The FGS AT and PGD AT variants, for example, could be classified as **non-certified deterministic hardening** defenses, since they cannot provide robustness certificates, do not introduce any additional randomness as part of the defense and aim to make the model inherently more robust rather than focusing on detecting or purifying adversarial noise. Other works utilize more advanced ideas from optimization theory in order to implement a training regime that *can* provide robustness certificates, such as Sinha et al. [2017]. Typically, such work goes beyond the original formulation (2.12) by Madry et al. [2017] and instead considers a *distributionally* robust optimization (DRO) problem:

$$\theta^\star = \underset{\theta}{\operatorname{argmin}} \sup_{Q \in \mathcal{Q}} \mathop{\mathbb{E}}_{Q} [\mathcal{L}(X, Y, \theta)]. \tag{2.13}$$

Here, $\mathcal{Q}$ is a class of distributions centered around the original data distribution Q_0. The most common form of DRO takes a *Wasserstein ball* around Q_0,

$$\mathcal{Q} = \{Q \mid W_c(Q_0, Q) \leq \varepsilon\},$$

where W_c is the Wasserstein metric

$$W_c(P, Q) = \inf_{M \in \Pi(P,Q)} \mathop{\mathbb{E}}_{M}[c(Z, Z')].$$

The Wasserstein metric $W_c(P, Q)$ between two distributions P and Q is parameterized by a *cost function* c, which takes samples from both distributions P and Q and outputs a non-negative real number. The set $\Pi(P, Q)$ is the collection of all *couplings* of P and Q, *i.e.*, the set of all joint distributions where the respective marginals are P and Q. The Wasserstein metric therefore computes the smallest expected cost $c(z, z')$ when (z, z') is sampled over all possible joint distributions with marginals P and Q. The DRO problem (2.13) then consists of minimizing the worst-case expected loss when the data X can be sampled from *any* distribution Q in a Wasserstein ball around Q_0. This essentially formalizes robustness against small distributional shifts, from which the method derives its name. DRO methods are typically **certified deterministic hardening** defenses.

Another important example of a **certified** AT defense is TRADES [Zhang et al., 2019b]. Like FGS AT, the concrete implementation of this method takes the form of a regularization added to the loss function. The theoretically principled nature of the regularizer allows TRADES to trade-off accuracy and robustness, and allows for the efficient computation of robustness certificates for adversarially trained models. TRADES gained notoriety for reaching first place in the Adversarial Vision Challenge at NeurIPS 2018.[8] Due to its computational efficiency, certified nature and impressive robustness results, TRADES has become a popular defense method in recent years.

[8] `https://www.crowdai.org/challenges/nips-2018-adversarial-vision-challenge`. Accessed 2021-10-12.

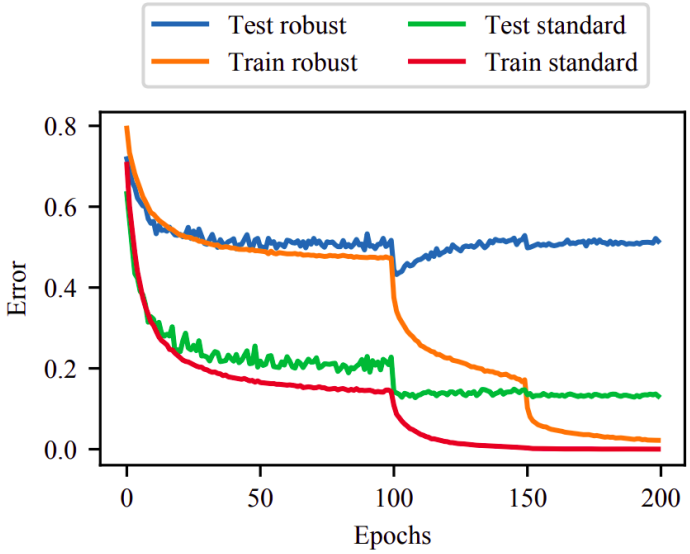


Figure 2.12: Illustration of robust overfitting of the AT method by Madry et al. [2017] on CIFAR-10. The model is more robust in the middle of training than at the end of it. Image due to Rice et al. [2020].

Adversarial training is not without its issues, however: aside from the often much increased computational complexity, there is the problem of *robust overfitting* described by Rice et al. [2020]. This phenomenon is illustrated in figure 2.12. There appears to be a consistent and significant gap between the best robust error obtained *during* training and the robust error achieved at the very *end* of training. Moreover, robust overfitting seems to be a general property of all AT methods, as Rice et al. observe it not only for the original formulation by Madry et al. but also in other settings, such as the improved FGS-AT scheme by Wong et al. [2020] and even TRADES [Zhang et al., 2019b]. After extensive experiments, Rice et al. find that the only effective remedy appears to be early stopping with a validation set. It is therefore important when performing model selection on adversarially trained networks to make use of early stopping to mitigate robust overfitting.

Although Rice et al. do not give an explanation for the phenomenon of robust overfitting, in hindsight one can give a fairly obvious intuitive justification. Looking back at the robust optimization formulation (2.12), at each iteration, one solves an optimization problem that maximizes the error of the current model within the ε-norm ball. It stands to reason that as training progresses and the model loss becomes small, the adversarial perturbations generated by the inner maximization procedure will themselves "overfit" to the target model: the perturbations will be tailored more and more towards the highly specific idiosyncrasies of the model, especially when many iterations are used. This can be seen quite clearly from figure 2.12: the robust *training* error keeps decreasing whereas the robust *test* error starts to increase after epoch 100. This also explains why only early stopping was found to be effective but not other techniques such as regularization or data augmentation: it is not the *model* that is overfitting, but the *adversar-*

ial perturbations themselves. Using early stopping, one can restore the best model parameters over the entire course of training and thus mitigate this phenomenon. However, regularizing the model or incorporating additional data clearly will not do the trick, because this will not prevent the inner maximization procedure from producing overly-optimized perturbations. Aside from early stopping, this conjecture also provides another possible avenue for mitigating robust overfitting: by using adversarial attacks that generate diverse sets of perturbations that do not overfit to any specific model. This may be accomplished by incorporating a *transferability* constraint, where the generated perturbations have to be effective not only against the target model but against other unrelated models as well. There has been much research into transferable adversarial examples [Liu et al., 2016b, Tramèr et al., 2017, Zhou et al., 2018], with some works explicitly focusing on adversarial training. Zheng et al. [2020], for example, propose to re-use adversarial examples from previous epochs, based on the observation that adversarials generated at one epoch can remain effective for many subsequent epochs as well. This method avoids the overhead of training multiple models by considering transferability across epochs instead of models proper, but it simultaneously reduces the potential for overfitting by not always using the most up-to-date model parameters. To my knowledge, Rice et al. [2020] did not experiment with such methods in their paper, and hence their potential to mitigate robust overfitting remains unexplored.

2.4.2 Defensive distillation

Defensive distillation [Papernot et al., 2016b] is notable for being one of the first serious attempts at an effective adversarial defense. Belonging to the category of **uncertified deterministic hardening** defenses, it seemed highly promising when it was first proposed, reducing the success rate of the FGS and the Jacobian saliency map attack (JSMA) [Papernot et al., 2016a] to less than 1%. The defense combines two existing methods for training more well-calibrated neural networks: *temperature scaling* and *distillation*.

Temperature scaling (TS) adapts the final softmax layer of a classifier, rescaling the logits z as follows:

$$f_i(x) = \frac{\exp(z_i(x)/\tau)}{\sum_j \exp(z_j(x)/\tau)}.$$

Here, $\tau > 0$ is the so-called *temperature*. For $\tau = 1$, TS has no effect and reduces to a regular softmax transformation. For $\tau < 1$, the logits are inflated and hence the resulting probability estimates will be more tightly concentrated. For $\tau > 1$, logits are reduced and the probability estimates will tend more towards a uniform distribution. It has been shown that this method can be a very simple and effective way to drastically improve the calibration of a classifier [Guo et al., 2017], so its application to adversarial defense makes sense.

Distillation [Ba and Caruana, 2013, Hinton et al., 2015] is a technique originally proposed for reducing the computational burden of DNNs, by training a smaller "distilled" model on the outputs of a larger one. Specifically, given an original training data set $S = \{(x_i, y_i) \mid i = 1, \ldots, m\}$, distillation will first train a large DNN on S, obtaining a classifier f that outputs probability scores for each class. Then, a smaller network is trained on the modified data set

$$S_f = \{(x_i, f(x_i)) \mid i = 1, \ldots, m\}.$$

Hence, instead of being trained on the true labels (which are typically one-hot encoded vectors), the distilled model is trained on the probability values given by the large network f. The idea is

that such a training procedure yields smoother probability values, which are more informative and will temper the network's tendency towards over-confidence [Guo et al., 2017].

Defensive distillation now proceeds as follows:

1. Train a DNN f on the original training data set $S = \{(x_i, y_i) \mid i = 1, \ldots, m\}$ using TS with $\tau > 1$. This lowers the confidence of the classifier.
2. Create a modified training set
$$S_f = \{(x_i, f(x_i)) \mid i = 1, \ldots, m\}.$$
3. Train a distilled model g on S_f using TS with the same temperature τ as was used for training f.

To deploy the final distilled model g, the temperature is set to one, effectively disabling TS and reverting back to a regular softmax transformation. This will increase the confidence of g in its predicted class.

Aside from being one of the first serious defenses, defensive distillation is also notable for establishing what would later become a recurring pattern in the field of adversarial ML. Shortly after the defense was published, Carlini and Wagner [2016] showed that it was in fact not effective at all: by slightly modifying the JSMA method, they were able to reduce the robust accuracy of defensive distillation from over 99% to 3.6%. They attributed the failure of the original JSMA to two causes:

- Ignoring the softmax layer. JSMA originally worked by computing the gradients of the input with respect to the *penultimate* (pre-softmax) layer. However, in doing so, they neglect the effect the softmax has on the final output, which can be considerable. Defensive distillation exploits this weakness of JSMA by causing the softmax to have a much larger effect on the output than it otherwise would have, essentially drowning out the JSMA perturbations. It is actually only by accident that this form of JSMA even works for *standard* networks, since even for such models there is a risk that the softmax causes the effect of JSMA perturbations to vanish. Carlini and Wagner therefore modified JSMA to take gradients with respect to the final output of the model.
- Vanishing gradients. Even if one modifies the JSMA method to take gradients with respect to the final output, the TS modification still causes a gradient vanishing problem. To prevent this, Carlini and Wagner rescale the logits before passing them through the softmax.

After these modifications, the JSMA method was able to generate adversarials against defensively distilled models without any issues.

This result can be considered the start of a pattern that is all too familiar to researchers working in adversarial ML, as Carlini and Wagner would later go on to break many more defenses shortly after (and sometimes even *before*) they were formally published [Athalye and Carlini, 2018, Carlini, 2019a,b, Carlini and Wagner, 2017a,b]. The seminal work that essentially "woke up" the community was their ICML 2018 contribution [Athalye et al., 2018a], where they identified the same vanishing gradients problem as the root cause of the overestimation of model robustness in most proposed defenses. In later publications, Carlini et al. would focus more on providing systematic analyses of common methodological flaws in the design and evaluation of adversarial

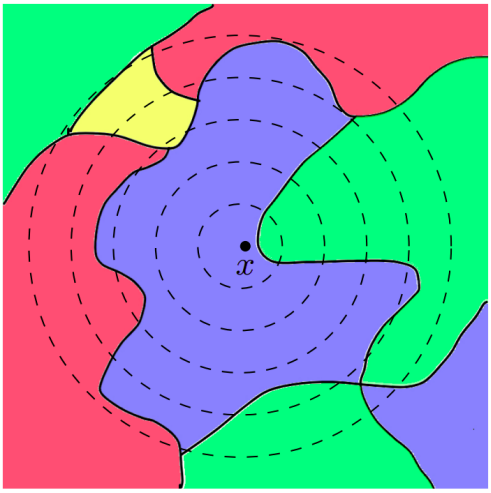


Figure 2.13: Illustration of the idea behind randomized smoothing [Cohen et al., 2019]. The decision regions of the base classifier f are shown in different colors. The dashed circles represent the level sets of the distribution $\mathcal{N}(x, \sigma^2 I)$. Given a sample x, RS samples around x according to $\mathcal{N}(x, \sigma^2 I)$ to empirically estimate the class probabilities when x is subjected to Gaussian corruptions. These probabilities directly lead to robustness certificates in the L_2 threat model.

defenses and providing concrete guidelines and recommendations for the community [Carlini et al., 2019, Pintor et al., 2021, Tramer et al., 2020].

2.4.3 Randomized smoothing

Given the relative ease with which researchers were able to break most proposed defenses very quickly after they became known, the community naturally started to focus more on *certified* defenses where one can mathematically prove that no adversarials exist within a certain radius. Perhaps the most well-known such defense is *randomized smoothing* (RS) [Cohen et al., 2019].

RS is both remarkably simple and remarkably effective. To implement it, one takes an existing pre-trained model f and computes the *smoothed model*

$$\hat{f}(x) = \underset{y \in \mathcal{Y}}{\operatorname{argmax}} \underset{\boldsymbol{\eta} \sim \mathcal{N}(0, \sigma^2 I)}{\Pr} [f(x + \boldsymbol{\eta}) = y]. \tag{2.14}$$

Figure 2.13 illustrates the idea behind the smoothing procedure (2.14). By sampling around a given point x, the stability of the classifier f under small additive perturbations can be assessed. Despite the fact that RS is randomized, Cohen et al. show that one can derive a *deterministic* robustness certificate from it. Specifically, if $\hat{f}$ is defined as in (2.14), then $\hat{f}(x + \boldsymbol{\delta}) = \hat{f}(x)$ for all perturbations $\boldsymbol{\delta}$ satisfying

$$\|\boldsymbol{\delta}\| \leq \frac{\sigma}{2} \left(\Phi^{-1}(p_1) - \Phi^{-1}(p_2) \right) \tag{2.15}$$

where Φ is the standard Gaussian cumulative density function (CDF) and p_1, p_2 are probabilities

that obey the following inequalities:

$$\Pr[f(x+\boldsymbol{\eta}) = \hat{f}(x)] \geq p_1 \geq p_2 \geq \max_{y' \neq \hat{f}(x)} \Pr[f(x+\boldsymbol{\eta}) = y'].$$

In other words, p_1 and p_2 correspond to a lower and upper bound respectively on the top-2 class probabilities assigned by f to the given sample x under Gaussian perturbations. RS therefore belongs to the class of **certified randomized hardening** defenses.

The robustness bound (2.15) provided by RS is very intuitive to interpret and very simple to optimize: in order to maximize the robustness of the smoothed model $\hat{f}$, one can either increase the variance σ^2 or increase the margin $p_1 - p_2$ between the top two predictions. The former option risks lowering classifier accuracy, as the noise may eventually drown out any meaningful signal, so it should be used sparingly. The latter option, however, will be familiar to ML practitioners as maximization of probability margins is known to have all sorts of beneficial effects [Anthony and Bartlett, 1999, Shalev-Shwartz and Ben-David, 2014, Vapnik, 1999]. It would therefore stand to reason that the combination of RS with other techniques that promote large margins, such as the large-margin softmax loss [Liu et al., 2016a] or temperature scaling [Guo et al., 2017], would further aid robustness.

2.4.4 Mahalanobis distance-based detector

Lee et al. [2018] proposed an **uncertified deterministic detector** technique for simultaneously identifying out-of-distribution (OOD) samples as well as adversarial examples. They assume that the class-conditional distribution of the logits produced by the classifier f approximately follows a multivariate Gaussian,

$$\Pr[f(X) \mid Y = y] = \mathcal{N}(\boldsymbol{\mu}_y, \boldsymbol{\Sigma}).$$

Here, $\boldsymbol{\mu}_y$ is the mean logit vector of f when the input belongs to class y and $\boldsymbol{\Sigma}$ is the covariance matrix. Note that Lee et al. do not let $\boldsymbol{\Sigma}$ depend on the class, essentially tying the covariances of all classes together. They then define the Mahalanobis distance-based confidence score:

$$M(x) = \max_{y \in \mathcal{Y}} -(f(x) - \boldsymbol{\mu}_y)^\intercal \boldsymbol{\Sigma}^{-1} (f(x) - \boldsymbol{\mu}_y). \tag{2.16}$$

To compute (2.16) in practice, the mean vectors $\boldsymbol{\mu}_y$ and covariance matrix $\boldsymbol{\Sigma}$ are empirically estimated from data. Detection of OOD samples as well as adversarial examples can then be done simply by thresholding the confidence score $M(x)$.

The Mahalanobis distance-based detector is extremely easy to implement, very computationally efficient and yields relatively high detection rates. It still holds up well today compared to newer techniques [Stutz et al., 2020].

2.4.5 Certifiably robust variational auto-encoders

Barrett et al. [2021] recently introduced a framework for certifying the robustness of variational auto-encoders (VAEs) against adversarial manipulation. Their method yields a **certified randomized purification** defense, as it makes use of the randomness in VAEs to yield probabilistic certificates of robustness of the auto-encoder in the L_2 norm.

They begin by introducing a notion of robustness for VAEs, which they term r-robustness. An image-to-image model $f : \mathcal{X} \to \mathcal{X}$ is called r-robust against a perturbation $\boldsymbol{\delta}$ if

$$\Pr[\|f(x+\boldsymbol{\delta}) - f(x)\|_2 \leq r] > \frac{1}{2}. \tag{2.17}$$

The model f is then said to have an r-robustness margin ε if it is r-robust against *all* perturbations $\boldsymbol{\delta}$ with $\|\boldsymbol{\delta}\|_2 \leq \varepsilon$. It can then be shown [Barrett et al., 2021, theorem 2] that the r-robustness margin of a VAE can be controlled if both the encoder and decoder networks are Lipschitz continuous. Specifically, the r-robustness margin depends on r, the Lipschitz constants of the encoder and decoder and the norm of the encoder standard deviation.

The construction provided by Barrett et al. has the major advantages of yielding efficiently computable robustness certificates as well as allowing the models to be efficiently trained. The constraint that the VAEs have to be Lipschitz, however, is the main limitation of the method. To implement Lipschitz VAEs in practice, the authors have to make use of special activation functions such as GroupSort [Anil et al., 2019] and the model weight matrices have to be orthonormalized at each update (similar to the Parseval networks of Cisse et al. [2017a]). These restrictions severely limit the expressivity of the models. Indeed, to the best of my knowledge such Lipschitz continuous networks have not yet been scaled beyond toy data sets.

2.4.6 Combination therapies

More recent efforts towards improving adversarial robustness have focused on "combination therapies," *i.e.*, the combination of many relatively small changes in model architecture, data curation and training regimes that appear to add up to significantly increased robustness. Gowal et al. [2020], for example, showed that one can greatly boost adversarial robustness by combining a number of adjustments:

1. Carefully choosing an appropriate loss function. For adversarial training, Gowal et al. find that combining TRADES [Zhang et al., 2019b] with early stopping yields the best results.

2. Increasing model capacity. Larger models such as Wide ResNets (WRNs) [Zagoruyko and Komodakis, 2016] achieve higher robustness than smaller popular models like standard ResNets [He et al., 2016].

3. Using additional data. Taking samples from sets such as ImageNet [Fei-Fei et al., 2009] or 80M-TI [Torralba et al., 2008] and *pseudo-labeling* them with pre-trained classifiers also appears to increase robustness.[9] There is a trade-off, however, between the quantity and quality of the data that needs to be taken into account. Furthermore, carefully tuning the weight of these pseudo-labeled samples in the training loss can be beneficial as well.

4. Training the models using weight averaging (WA) [Izmailov et al., 2018]. WA modifies the weight updates performed by gradient descent to keep a running average of the parameters. Gowal et al. find that WA consistently results in higher robustness.

5. Carefully choosing the activation functions. While ReLU remains a good choice, improvements can sometimes be obtained using the SiLU activation.

[9] As of June 2020, the 80M-TI data set has been formally retracted due to the presence of offensive content [Birhane and Prabhu, 2021]. The retraction notice can be found at `http://groups.csail.mit.edu/vision/TinyImages/`. It is therefore no longer to be used by ML researchers.

In a similar vein, Rebuffi et al. [2021] show that WA combined with carefully chosen data augmentation strategies can also significantly boost robustness. In particular, Rebuffi et al. recommend augmenting with samples from state-of-the-art generative models such as Denoising Diffusion Probabilistic Models (DDPMs) [Ho et al., 2020] along with CutMix [Yun et al., 2019]. At the time of this writing, the models trained by Rebuffi et al. score the highest on the RobustBench leaderboard for CIFAR-10 in the L_∞ model ($\varepsilon = 8/255$) and the L_2 model ($\varepsilon = 0.5$).

In all, these works seem to give rise to a new consensus within the adversarial ML field as of 2021:

> *There is no "silver bullet" for adversarial robustness.*

That is, the fragility of our DNNs to adversarial perturbations is likely not caused by any one component of the ML pipeline in isolation. Rather, it is a consequence of the interplay of various elements, from the data curation process to the neural network architecture and specifics of the training algorithm. This can be seen in the results of the works cited above, where considerable improvements in robustness are obtained through the combination of many small tweaks rather than through the construction of an elaborate dedicated defensive component.

2.4.7 Evaluation and benchmarking

Before we conclude the section on adversarial defenses, it is important to reflect on current and past practices adopted by the field for the evaluation and benchmarking of proposed methods. This is particularly relevant for adversarial ML, as experience has shown that great care must be taken in order to properly evaluate defense algorithms. I have hinted at these difficulties at certain points throughout this section, as in the example of the failure of defensive distillation, but a dedicated discussion seems appropriate. Historically, most of the proposed defenses against adversarial examples have failed because of common mistakes in the evaluation methodology. Carlini et al. [2019] list a few typical errors:

Failure to specify a precise threat model. This problem was especially common in early papers on the subject, where the threat model was often largely left implicit. However, by failing to explicitly state the exact threat model for which a defense is designed, it becomes very hard or even impossible to judge when it is appropriate to deploy the defense. It also makes it hard to spot subtle errors in the evaluation, such as when certain tacitly assumed limitations of the adversary are easily violated in practice.

Failure to evaluate against an *adaptive* adversary. Many proposed adversarial defenses turned out to work only because they were specifically designed with certain existing attacks in mind, such as gradient masking defenses that assumed all potential adversarial attacks would crucially rely on accurate information about the gradient of the loss. In reality, however, adversaries will often take the time to carefully analyze a security system before they attack it, and their attack will be tailored to the defense in question. Proper defense evaluations must therefore account for this possibility and adapt their attacks accordingly.

Reporting robust accuracy only for a fixed attack budget. Many papers will only report the robust accuracy of their methods at a single fixed perturbation budget, such as 1/2 for CIFAR-10 L_2 or 4/255 for ImageNet L_∞. For detector defenses, often only a single point on the ROC

curve will be reported. In both cases, however, it is important to report the full curves. For hardened classifiers, it is desirable to have the robust accuracy as a function of the full range of plausible attack budgets; for detectors, the full ROC curve is preferred. We obtain much more information about the defense this way. For one, we can see how quickly its protection degrades under stronger attacks and use this for comparison to other defenses, to aid in selecting the best defense for any given situation. We can also see how the defense performs at different budgets, which may be relevant when the default budgets commonly studied by academics prove unrealistic in practice.

Neglecting basic sanity checks. Although adversarial defense evaluations can become very complicated very quickly, there are a few basic sanity checks that can and should always be carried out. At the very least, one should verify that an *unbounded* attack, *i.e.*, an attack that has no limit on the perturbation budget, reaches 100% success rate. After all, with unbounded perturbations it is possible to transform a given input into any other input and so any target output could be reached. If unbounded attacks fail, it is highly likely that the defense is implemented incorrectly and robustness is over-estimated.

Failing to tune hyperparameters. Most adversarial attacks and defenses have at least some hyperparameters that should be tuned for the specific task. In any evaluation, failing to tune the hyperparameters of the attacks can underpower them and hence easily cause over-estimation of robustness. Similarly, when comparing a new defense to prior work, it is in general unfair to leave hyperparameters of existing defenses at their default values. This is a common and subtle way in which defense evaluations can become overly optimistic.

These are some of the most common pitfalls observed by Carlini et al. [2019] that should caution researchers developing new defenses. As discussed by Tramer et al. [2020], the field has mostly taken these suggestions to heart, and defense evaluations have been getting significantly better in recent years. The introduction of the AutoAttack suite by Croce and Hein [2020b] and the RobustBench leaderboard[10] serves as another important milestone in this regard, as to date this is the most successful attempt at systematically benchmarking existing adversarial defense algorithms.

The RobustBench benchmark works by simply applying the AutoAttack suite to a given defense for several different data sets and threat models. At the time of this writing, the threat models considered are CIFAR-10 under $L_2 = 1/2$, $L_\infty = 8/255$ and common corruptions, CIFAR-100 under $L_\infty = 8/255$ and common corruptions and ImageNet under $L_\infty = 4/255$ and common corruptions. For each defense, the leaderboard reports the original publication that proposed it, their standard accuracy, robust accuracy according to the AutoAttack suite, a flag indicating whether the defense uses additional data and a flag indicating whether the evaluation may be unreliable.

While RobustBench is definitely a step in the right direction and provides a much-needed reference which simply did not exist less than three years ago, it also has important shortcomings that researchers need to be aware of:

- RobustBench relies on AutoAttack. The AutoAttack suite is in fact a diverse ensemble of recently proposed strong adversarial attacks with optimized hyperparameters. While this

[10] `https://robustbench.github.io`. Accessed 2022-08-30.

is certainly better than how things used to be done (*i.e.*, evaluate against FGS and PGD and be done with it), AutoAttack is *not* an adaptive adversary: it is a collection of static adversaries that are deemed to be generally strong against most defenses. It is entirely possible for a defense to overfit against AutoAttack and fail against adaptive adversaries for the same reasons defenses used to fail before.

- RobustBench supports only a handful of rather artificial threat models. In particular, the benchmark is limited to three data sets: CIFAR-10, CIFAR-100 and ImageNet. The first two are highly similar to each other, both being derived from the same parent data set (80 million tiny images). Furthermore, these data sets only cover image classification tasks. Natural language processing, for example, is completely absent from this benchmark, as are any other ML tasks that do not involve classification. Patch attacks, which are considered more realistic alternatives to L_p attacks [Levine and Feizi, 2020], are also not supported.

- RobustBench uses fixed perturbation budgets. As argued by Carlini et al. [2019], a proper evaluation should report robust accuracy for the full range of plausible perturbation budgets.

- RobustBench does not support detector methods. To evaluate a detector method, one would have to study the ROC curve, but RobustBench does not give the option of reporting true positive rates and false positive rates, let alone entire ROC curves.

- RobustBench ignores many additional parameters besides accuracy that are also relevant when considering whether to deploy any particular defense in practice. There is no column for memory or runtime overhead of the defense, for example. While RobustBench does provide a column to indicate whether additional data is necessary, it does not specify the required sample size, making this column rather uninformative.

To re-iterate, RobustBench has certainly improved how adversarial defenses are evaluated and compared, but it is by no means perfect. I would therefore urge researchers developing new defenses to definitely make use of the RobustBench library and the accompanying AutoAttack suite, but to not limit themselves exclusively to this benchmark. Aside from RobustBench, researchers should at the very least also consider adaptive attacks in their evaluation and compare their results to adaptive evaluations of prior work if these are available. Unfortunately, while many of the shortcomings I mentioned above are relatively easy to fix, it will likely never be possible to develop adaptive attacks to novel adversarial defenses automatically; this is something researchers will always have to do manually and with great care. The results of any specific automated benchmarks for adversarial robustness will necessarily always be upper bounds which may change at any moment when new attacks are developed.

2.5 Theoretical results

In this section, I survey some of the theoretical results that have been described in the adversarial ML literature. I begin by considering linear classifiers and the theory that has been developed around adversarials in this restricted setting. Then, I proceed to results for more complex models, such as deep RELU networks (*i.e.*, DNNs that use the RELU as their only non-linearity). Of particular interest to us here are theorems regarding conditions for the (non-)existence of

adversarial examples, bounds on the robustness achievable by a given model and computational hardness results for adversarial defense.

2.5.1 The linear model

Before tackling DNNs, it is instructive to study adversarial perturbations in the case where f is a linear classifier:

$$f(x) = \operatorname{sgn}(w^\intercal x + b).$$

In the L_2 threat model, finding an adversarial example then boils down to the following optimization problem:

$$\min \|\boldsymbol{\delta}\|_2^2 \text{ subject to } w^\intercal(x+\boldsymbol{\delta}) + b = 0. \tag{2.18}$$

The Lagrangian function associated with (2.18) is given by

$$L(\boldsymbol{\delta}, \lambda) = \|\boldsymbol{\delta}\|_2^2 + \lambda(w^\intercal(x+\boldsymbol{\delta}) + b).$$

Taking derivatives with respect to $\boldsymbol{\delta}$,

$$\frac{\partial}{\partial \delta_i} L(\boldsymbol{\delta}, \lambda) = 2\delta_i + \lambda.$$

To reach a stationary point, it is therefore necessary that

$$\delta_i = -\frac{1}{2}\lambda$$

for all i. Hence, taking into account the constraint $w^\intercal(x+\boldsymbol{\delta}) + b = 0$, this leads to

$$\lambda = 2\frac{w^\intercal x + b}{\sum_{i=1}^n w_i}.$$

Finally, we have the explicit solution to (2.18), given by

$$\delta_i = -\frac{w^\intercal x + b}{\sum_{i=1}^n w_i} \tag{2.19}$$

for all i. The L_2 norm of this perturbation is

$$\|\boldsymbol{\delta}\|_2 = \sqrt{n}\left|\frac{w^\intercal x + b}{\sum_{i=1}^n w_i}\right|. \tag{2.20}$$

From (2.20) we can see that a linear model cannot be robust in the L_2 threat model unless $\varepsilon_X = O(\sqrt{n})$, *i.e.*, the robustness radius can grow at most proportionally to the square root of the data dimensionality.

In essence, what we have computed in (2.19) is the orthogonal projection of the sample x onto the decision boundary of the classifier. Under the L_2 metric, this is the shortest possible projection and hence it corresponds to the smallest possible adversarial. This shows that, fundamentally, the problem of robustness boils down to maximizing the distance between the data and the decision boundaries, a property that is reminiscent of the support vector machine or SVM [Cortes and Vapnik, 1995].

Naturally, the robustness of SVMs against adversarial examples has been investigated in the literature. Langenberg et al. [2019], for example, reached the surprising conclusion that the robustness of the SVM is only weakly affected by the parameters determining the margin between classes. Instead, the choice of kernel and the resulting non-linearity appears to have a much greater effect, with polynomial kernels of appropriate degree as well as the radial basis function (RBF) kernels with well-tuned variance being more robust than a linear one. These findings are in line with earlier work by Biggio et al. [2013], Šrndic and Laskov [2013] who also studied SVMs in the adversarial setting. These works lend credibility to the *linearity hypothesis*: the idea that adversarial examples are due to excessive linearity of the model, and hence robustness could be achieved by making the models more non-linear. We turn to this hypothesis in the next subsection.

2.5.2 Boundary tilting and the linearity hypothesis

One of the first hypotheses put forward to explain the existence of adversarial examples in DNNs was the *linearity hypothesis* by Goodfellow et al. [2014]. They reasoned that adversarial examples were due to the excessive *linear* behavior of neural networks, despite the fact that DNNs are typically composed of many non-linear functions. Taking the inner product as the leading example,

$$y(x) = w^{\mathsf{T}} x,$$

it can be seen that the deviation between $y(x)$ and $y(x+\boldsymbol{\delta})$ grows linearly with the magnitude of $\boldsymbol{\delta}$. In particular, if we set $\boldsymbol{\delta} = \varepsilon \operatorname{sgn} w$, then

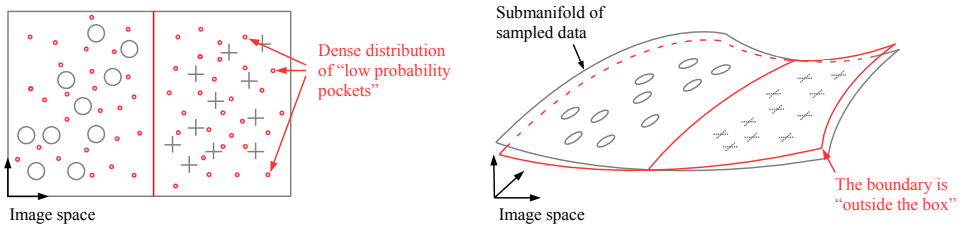
$$y(x+\boldsymbol{\delta}) = y(x) + \varepsilon \|w\|_1.$$

Adversarial examples could then theoretically arise due to the high-dimensional nature of the data: for large n, even a small ε can cause $\varepsilon\|w\|_1$ to be very large.

This linearity hypothesis was popular for some time, until Tanay and Griffin [2016] proposed *boundary tilting* as an alternative. They first argued that the linearity hypothesis is incomplete, using two pieces of evidence:

1. The data dimensionality has no substantial effect on adversarial examples for linear classifiers on MNIST. When the MNIST data set is scaled up from 28×28 resolution to 200×200, the vulnerability of logistic regression models trained on this data is not meaningfully impacted. Yet, according to the linearity hypothesis, models trained on the higher resolution data should be much more vulnerable.

2. There are linear classification problems for which adversarial examples provably do not exist. The concrete example given by Tanay and Griffin is a toy binary classification task on images of 100×100 resolution, where the two classes are defined as follows. The positive class consists of samples where the left half of the image is noisy (random pixel values in the range $[0,1]$) and the right half is completely black (pixel value 0); the negative class consists of images where the left half of the image is noisy (like the first class) and the right half is completely white (pixel value 1). A linear SVM can be trained to perfectly separate these classes, and interpolating between classes across the decision boundary does not yield meaningful adversarials, since the samples will look ambiguous.



(a) The "pocket view" suggested by Szegedy et al. [2013]. Here, adversarial examples are due to pockets of low probability in the input space, which are undersampled and thus not well-modeled by the classifier.

(b) The boundary tilting perspective of Tanay and Griffin [2016]. Here, the decision boundary learned by the classifier is slightly rotated with respect to the ground truth. This leads to samples that lie close to the data manifold but which are classified incorrectly.

Figure 2.14: Comparison of the "pocket view" of Szegedy et al. [2013] and the boundary tilting perspective of Tanay and Griffin [2016] on adversarial examples.

Rejecting the linearity hypothesis, Tanay and Griffin [2016] proposed *boundary tilting* as a new perspective. Under the boundary tilting hypothesis, adversarial examples are due to a misalignment between the true data manifold and the manifold learned by the model. This idea is illustrated in figure 2.14b, where the learned manifold is slightly rotated with respect to the data manifold. Tanay and Griffin contrast this with the "pocket view" put forward by Szegedy et al. [2013], which holds that adversarial examples lie in "pockets" of vanishingly low probability in the input space (see figure 2.14a). In that sense, adversarial examples are "dense" in the input space like the rational numbers on the real line: they certainly exist (there are even infinitely many of them), but they have zero measure.

This line of reasoning can also be found in Biggio and Roli [2018], Melis et al. [2017], who consider *class-enclosing defenses*. The idea is illustrated in figure 2.15 using multiclass SVMs with RBF kernels. A regular SVM essentially partitions the input space into regions associated with specific classes. However, because these partitions must cover the entire input space, the SVM inevitably makes inappropriate associations, since areas that lie completely outside of the support of the natural data manifold will still be marked with a class. Class-enclosing defenses aim to ameliorate this by equipping the model with a "reject" option. Instead of partitioning the entire input space, such defenses consider only high-probability subsets. A sample is only associated with a class if it lies within the appropriate subset; if it does not, the sample is "rejected," indicating that it cannot be reliably classified using this model. This reflects the intuition that natural samples lie in high-probability "pockets," and adversarials necessarily lie in regions of the input space where support is very low.

2.5.3 Geometrical perspectives

There is an extensive body of work aiming to understand the inner workings of DNNs from a geometrical perspective. Research in this area typically employs tools from Riemannian geometry to shed light on how the layers of a neural network sequentially deform the manifold of the input samples to some intermediate, hidden representation. A seminal contribution in this vein is Poole et al. [2016], who were the first to rigorously formalize and prove the long-conjectured

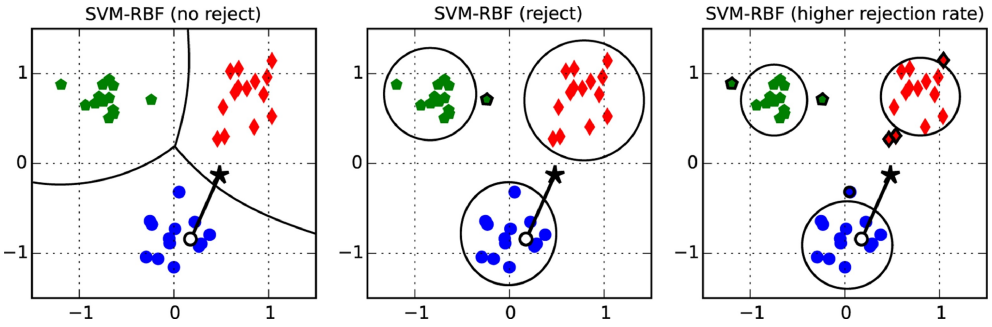
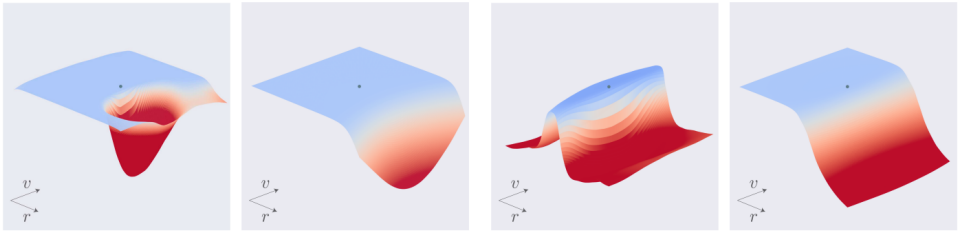


Figure 2.15: Illustration of class-enclosing defenses, taken from Biggio and Roli [2018]. The classes learned by the model no longer completely partition the input space. Instead, each class is viewed as an enclosed subset, and samples lying outside of these sets are rejected.



(a) Original (CIFAR-10) (b) Fine-tuned (CIFAR-10) (c) Original (SVHN) (d) Fine-tuned (SVHN)

Figure 2.16: Illustration provided by Moosavi-Dezfooli et al. [2019] of the relationship between adversarial vulnerability and curvature of the loss surface on two data sets (CIFAR-10 and SVHN), with and without fine-tuning on adversarial examples. Depicted are the negative log-likelihood loss values for inputs sampled from a plane spanned by a direction r orthogonal to the decision boundary of the model and a random direction v. The original sample is shown as a blue dot. Blue regions correspond to low loss and red regions correspond to high loss.

idea that DNNs work by "disentangling" factors of variation. This idea was popularized by Bengio et al. [2013], but a formal definition remained elusive. Poole et al. formalized this idea using differential geometry, viewing the layers of a deep network as deformations of the input manifold, and characterizing the expressivity of a DNN using the extrinsic curvature of the Riemannian manifolds defined by the decision boundaries.

In the context of adversarial robustness, these ideas were taken up by Fawzi et al. [2017] and Moosavi-Dezfooli et al. [2019], who related the curvature of the decision boundaries learned by a DNN to its adversarial robustness. They show that higher curvature of the boundaries implies higher sensitivity of the network to adversarial attack, and propose a regularization scheme to reduce this curvature. These ideas are illustrated in figure 2.16. When the curvature of the loss surface is high, data samples will tend to lie very close to "valleys" where the loss rapidly increases with small perturbations. However, these valleys tend to be rather isolated, so they are easily "stepped over" when following random directions. Hence neural networks can be very robust to random noise but very sensitive to adversarial perturbations.

Interestingly, as can be seen from figure 2.16, adversarial training works to reduce the curvature of the decision boundaries. However, this has a paradoxical side-effect: although adversarially trained models are more robust (in the sense of having fewer adversarial examples within the specified perturbation budget), they are easier to fool. That is, the adversarial examples that still exist for adversarially trained models can be found with simpler attacks, such as PGD with a low number of iterations. This appears to be a direct consequence of the reduced curvature of the boundaries, which allows gradient-based attacks to converge much faster. For models that have high curvature, gradient information is less useful and hence more iterations of optimization are required. In some sense, this is to be expected, since a lower curvature intuitively implies a "simpler" decision boundary, whereas non-robust models (despite having many adversarial examples in the vicinity of each data point) may have much more complicated boundaries that are more difficult to optimize over. This is related to the phenomenon of gradient masking [Athalye et al., 2018a], where some adversarial attacks can severely over-estimate the robustness of models that have highly complicated decision boundaries.

The work of Moosavi-Dezfooli et al. also has an interesting connection to another branch of geometrical approaches to understanding adversarial robustness. Specifically, Moosavi-Dezfooli et al. define the *curvature profile* of a neural network as the spectrum of the Hessian matrix

$$H = \left(\frac{\partial^2 \ell}{\partial x_i \partial x_j} \right) \in \mathbb{R}^{d \times d},$$

where ℓ is the log-likelihood loss function. Although Moosavi-Dezfooli et al. did not seem to consider this connection, under certain regularity conditions, the above quantity essentially corresponds to the Fisher information matrix of the input variables. Other works have considered the robustness of neural networks from the perspective of *information geometry* [Amari, 2016], where one views neural networks as points on a statistical manifold. Zhao et al. [2019], for instance, formulate the One-Step Spectral Attack (OSSA) based on these insights. They consider the following optimization problem as the basis for adversarial attacks:

$$\max_{\boldsymbol{\eta}} \mathrm{D}_{\mathrm{KL}}(p(y \mid x) \| p(y \mid x + \boldsymbol{\eta})) \text{ subject to } \|\boldsymbol{\eta}\| \leq \varepsilon. \tag{2.21}$$

That is, (2.21) maximizes the KL divergence between the output distribution $p(y \mid x)$ of the model on the original input sample x and the distribution $p(y \mid x + \boldsymbol{\eta})$ of the perturbed sample $x + \boldsymbol{\eta}$, where $\boldsymbol{\eta}$ is subject to a norm constraint. Using a Taylor decomposition of the KL divergence yields, for sufficiently small $\boldsymbol{\eta}$,

$$\mathrm{D}_{\mathrm{KL}}(p(y \mid x) \| p(y \mid x + \boldsymbol{\eta})) \approx \frac{1}{2} \boldsymbol{\eta}^{\mathsf{T}} F(x) \boldsymbol{\eta},$$

where $F(x)$ is the Fisher information matrix of x. This reduces (2.21) to a quadratic program the solutions of which are given by the eigenspaces of $F(x)$. In this way, Zhao et al. directly relate the eigenvalues of the Fisher information matrix to the adversarial robustness of the model, a finding that agrees with Moosavi-Dezfooli et al. [2019].

It is instructive to apply these ideas to a very simple model. Consider the binary classifier

$$p(x) = \sigma(\boldsymbol{\theta}^{\mathsf{T}} x), \tag{2.22}$$

where $\sigma(\cdot)$ is the logistic sigmoid function and $\boldsymbol{\theta}$ is the weight vector of the classifier, its only parameter. The Fisher information matrix then has a simple form:

$$F(x) = p(x)(1 - p(x)) \boldsymbol{\theta} \boldsymbol{\theta}^{\mathsf{T}}.$$

The eigenpairs (λ, u) of F satisfy

$$F(x)u = \lambda u.$$

With some re-arranging of terms, this yields

$$u = \frac{p(x)(1-p(x))\boldsymbol{\theta}^\mathsf{T} u}{\lambda}\boldsymbol{\theta}.$$

Adding the constraint $\|u\| = 1$ and applying the Cauchy-Schwarz inequality, we find

$$|\lambda| \leq p(x)(1-p(x))\|\boldsymbol{\theta}\|^2. \tag{2.23}$$

This establishes a simple upper bound on the spectrum of the Fisher information matrix, which is directly related to the robustness of the model (2.22). In the case of this simple classifier, we find that robustness can be increased in two ways:

1. Regularization of $\boldsymbol{\theta}$.
2. High confidence of the predictions. That is, $p(x)$ should be close to 0 or 1, so that $p(x)(1-p(x))$ is minimized.

Although the first finding agrees with most prior work [Hein and Andriushchenko, 2017, Peck et al., 2017, Tanay and Griffin, 2016], the second may be considered strange: after all, overconfidence has been linked to vulnerability to adversarial examples [Hein et al., 2019] as well as poor calibration [Guo et al., 2017]. This suggests that adversarial robustness could be at odds not just with accuracy (as shown by Tsipras et al. [2019]) but with calibration as well. For well-calibrated models, (2.23) can be taken to show that some adversarial vulnerability is inevitable, especially for ambiguous samples where $p(x)$ is close to 50%.

2.5.4 Continuity properties

The function f is *Lipschitz continuous* in $d_\mathcal{X}$ and $d_\mathcal{Y}$ if it satisfies the following property:

$$\exists K > 0 : \forall x, x' \in \mathcal{X} : d_\mathcal{Y}(f(x), f(x')) \leq K d_\mathcal{X}(x, x'). \tag{2.24}$$

That is, the distance between the images $f(x)$ and $f(x')$ is at most a constant factor of the distance between the original samples x and x', for some fixed universal constant K which is referred to as the *(global) Lipschitz constant*.

Lipschitz continuity of f in $d_\mathcal{X}$ and $d_\mathcal{Y}$ is an interesting property to study for adversarial robustness, for the following reason. Let $\tilde{x}$ be $(\varepsilon_\mathcal{X}, \varepsilon_\mathcal{Y})$-adversarial for x and f. Together with (2.24), this implies the chain of inequalities

$$\varepsilon_\mathcal{Y} \leq d_\mathcal{Y}(f(x), f(\tilde{x})) \leq K d_\mathcal{X}(x, \tilde{x}) \leq K\varepsilon_\mathcal{X}.$$

It immediately follows that

$$\varepsilon_\mathcal{X} \geq \frac{1}{K}\varepsilon_\mathcal{Y}. \tag{2.25}$$

That is, the perturbation budget $\varepsilon_\mathcal{X}$ in the input space is bounded from below by the required deviation $\varepsilon_\mathcal{Y}$ in the output, scaled by the inverse of the Lipschitz constant. This means that *decreasing* the Lipschitz constant K of the network will *increase* the minimum required perturbation $\varepsilon_\mathcal{X}$. This simple fact has caused much of the adversarial ML literature to focus on

ways of maintaining Lipschitz continuity of neural networks during training and to certify upper bounds on the resulting Lipschitz constant [Fazlyab et al., 2019, Hein and Andriushchenko, 2017, Jordan and Dimakis, 2020, Scaman and Virmaux, 2018].

The bound (2.25) is rather crude, since the global Lipschitz constant does not take into account the specific sample x that is perturbed. As a result, it may be very large, yielding a very small lower bound despite the fact that f could be relatively robust on typical inputs. It is therefore of interest to study *local Lipschitz continuity*, where the constant K is allowed to depend on x:

$$\forall x \in \mathcal{X} : \exists K_x > 0 : \forall x' \in \mathcal{X} : d_{\mathcal{Y}}(f(x), f(x')) \leq K_x d_{\mathcal{X}}(x, x'). \tag{2.26}$$

This would yield a lower bound on $\varepsilon_{\mathcal{X}}$ that depends on the sample being perturbed, which can be much tighter than the global bound shown in (2.25).

Going further, one could try to work with the minimal assumption that f is continuous in $d_{\mathcal{X}}$ and $d_{\mathcal{Y}}$. Formally, this means

$$\forall x \in \mathcal{X} : \forall \varepsilon > 0 : \exists \delta > 0 : \forall x' \in \mathcal{X} : d_{\mathcal{X}}(x, x') \leq \delta \implies d_{\mathcal{Y}}(f(x), f(x')) \leq \varepsilon. \tag{2.27}$$

In particular, (2.27) implies

$$d_{\mathcal{Y}}(f(x), f(\tilde{x})) > \varepsilon_{\mathcal{Y}} \implies d_{\mathcal{X}}(x, \tilde{x}) > \delta(x, \varepsilon_{\mathcal{Y}}).$$

Hence, if a sample $\tilde{x}$ is adversarial for f and x, it must hold that $d_{\mathcal{X}}(x, \tilde{x}) > \delta(x, \varepsilon_{\mathcal{Y}})$. We therefore have the lower bound

$$\varepsilon_{\mathcal{X}} \geq \delta(x, \varepsilon_{\mathcal{Y}}). \tag{2.28}$$

For locally Lipschitz continuous functions, (2.28) specializes via

$$\delta(x, \varepsilon_{\mathcal{Y}}) = \frac{1}{K_x} \varepsilon_{\mathcal{Y}}.$$

In general, of course, $\delta(x, \varepsilon_{\mathcal{Y}})$ may be a much more complicated function.

Clearly, studying classes of functions for which the continuity properties can be kept under control is of great interest for adversarial robustness. However, depending on the choice of similarity measures $d_{\mathcal{X}}$ and $d_{\mathcal{Y}}$, continuity may not be desirable or even feasible. Furthermore, even if continuity can be guaranteed, it is an open problem how to efficiently control the δ in (2.27) during the optimization of the model parameters. *Parseval networks* [Cisse et al., 2017a] are perhaps the most well-known class of neural networks where this can be achieved efficiently to some extent. Parseval networks restrict their weight matrices to the *Stiefel manifold* [Absil et al., 2009], maintaining approximate orthonormality throughout optimization and hence a near-unit Lipschitz constant. Such methods have not gained much popularity, though, because they are considerably more complicated to implement and computationally burdensome when compared to other effective adversarial defenses (*e.g.*, adversarial training).

2.5.5 Hardness results

Much work has also gone into the study of the *hardness* of robust learning. Specifically, determining the conditions under which robust learning is possible, and to what extent, is one of the major problems in the field. In this regard, Gourdeau et al. [2019] have made several important

contributions. They showed that robust learning is impossible in the distribution-free setting even when the adversary is restricted to perturbing just a single *bit* of the input. Therefore, in order to have any hope of creating robust classifiers, certain distributional assumptions *must* be made; no adversarial defense can claim general robustness on arbitrary distributions. However, there are certain concrete distributions that can be robustly learned under milder conditions: the class of monotone conjunctions is robustly learnable if the adversary is limited to perturbing $O(\log n)$ input bits.

In a similar vein, Yin et al. [2019a] prove that the Rademacher complexity of robust learning will unavoidably be larger than its natural counterpart under realistic conditions. The Rademacher complexity is a fundamental quantity in statistical learning theory, similar to the VC dimension [Vapnik, 1999], that measures the complexity of infinite hypothesis classes in a way that can be directly related to the worst-case performance of any finite-sample learner for this class. A higher complexity of the hypothesis class therefore immediately implies worse lower bounds on the error of any learning algorithm. Similarly, Montasser et al. [2019] prove that hypothesis classes with finite VC dimension can be robustly learned, but only *improperly*. In the case of neural networks, this can imply that additional model capacity is necessary to obtain robustness. Diochnos et al. [2019] study the problem of robust learning in the probably approximately correct (PAC) framework [Valiant, 1984]. They demonstrate conditions under which robust PAC learning requires either *exponentially* more samples or *polynomially* more samples than standard PAC learning, and give examples where robust PAC learning is impossible altogether.

2.5.6 Detection vs robust classification

Recently, Tramèr [2021] proved an interesting connection between *detection* of adversarials (*i.e.*, detector-based defenses) and *robust classification* (*i.e.*, purification and hardening defenses). Specifically, if one can *detect* adversarial examples up to a radius of ε_X, then one can also *classify* them robustly up to a radius of $\varepsilon_X/2$ (although the construction provided by Tramèr is not efficient). The converse also holds: if one can robustly classify up to $\varepsilon_X/2$, then one can also detect adversarial perturbations up to ε_X.

Aside from being an interesting theoretical contribution in its own right, this finding provides a sort of "sanity check" for adversarial detector methods. Given the enormous research effort that has gone into constructing robust classifiers, it stands to reason that a novel detector method should not be able to provide robustness guarantees that would (according to the theorem by Tramèr) translate into (hypothetical) classifiers with a robust accuracy far in excess of the state of the art. Indeed, Tramèr studies 13 published defenses and finds that for 11 of them, their robust detection results would imply such unrealistically robust classifiers. While this result by itself is of course not sufficient to prove that the proposed detectors don't work, it is at least cause for some concern and should motivate the authors to perform more rigorous evaluations.

2.5.7 Robustness certification

In order to assess the robustness of a given DNN in practice, the most straightforward method is to simply attack the network with a strong algorithm and compute its empirical robustness on the generated samples. However, this approach may not be efficient if the attack is slow, since a large number of samples may be required to obtain statistically reliable estimates of robust accuracy. Moreover, in general, there is no guarantee that high robust accuracy against a given

attack implies any robustness at all against other attacks. For this reason, there has also been much interest in developing methods that can efficiently and reliably certify robustness of any given model without the need for extensive experiments with suites of existing attacks. The randomized smoothing defense by Cohen et al. [2019] which we described earlier essentially obtains certification abilities as a side-effect, but other methods exist that focus solely on certification and do not provide any defense themselves. Li et al. [2019] is perhaps most similar in spirit to randomized smoothing, as they also rely crucially on additive Gaussian noise to certify robustness. Their method also allows for the derivation of a training algorithm that improves robustness.

The main disadvantage of dedicated certification techniques is that they tend to be highly specific to certain architectures, as the problem of robustness certification for general neural networks is known to be NP-hard even under very simple threat models [Weng et al., 2018a, theorem 3.1]. For example, for tree-based models such as decision trees, random forests and gradient-boosted trees, Chen et al. [2019] proposed an iterative algorithm that gives tight lower bounds. The CLEVER score [Weng et al., 2018b] has appeared in many publications as a robustness estimate for general neural networks. It is based on estimation of the local Lipschitz constant, and its results seem to align well with empirical robustness tests. In a similar vein, Zhang et al. [2018a] develop the CROWN score, which has also been widely used.

Perhaps the most well-studied scenario is the RELU network, *i.e.*, when the neural network is restricted to using only the RELU or linear activation functions between layers. Weng et al. [2018a] introduce Fast-Lin and Fast-Lip to efficiently certify robustness of such networks. Fast-Lin relies on linear approximations of the individual layers whereas Fast-Lip uses Lipschitz estimation techniques similar to Weng et al. [2018b]. In some cases, researchers cast the robustness certification problem for RELU networks as a semi-definite program (SDP), which is a rich area in the field of optimization theory [Vandenberghe and Boyd, 1996]. Raghunathan et al. [2018] provide one such example, and the work of Wong and Kolter [2018] is similar.

In Peck et al. [2017] we developed theoretical robustness lower bounds which can be applied to essentially any existing neural network. In particular, we start from the formulation (1.1) which characterizes any DNN based on a set of layer weights $W_1, \ldots, W_L$, biases $b_1, \ldots, b_L$ and activation functions $g_1, \ldots, g_L$. Let F denote the full neural network and let $\|\cdot\|$ be any norm.[11] The idea behind our approach is to work backwards from the output layer, since it is generally very easy to quantify the perturbation needed on the output to change the label: given a vector of probabilities $z = h_L(x)$, the smallest perturbation that must be applied to change the label is simply the margin between the highest and second-highest values of z. Then, we assume inductively for any $l = 1, \ldots, L-1$ that layer $l+1$ requires a perturbation of at least κ_{l+1} to change the output label and consider the smallest perturbation r such that

$$\|z_l(u+r)\| = \|z_l(u)\| + \kappa_{l+1}.$$

Here, $z_l(u) = g_l(W_l u + b_l)$ is the function computed by layer l. The most straightforward way to obtain such bounds is via a Taylor decomposition:

$$z_l(u+r) = z_l(u) + J(u)r + \boldsymbol{\varepsilon},$$

[11] In our original work, we specialized to the Euclidean norm, but the analysis readily generalizes to other norms as well.

where $J(u)$ is the Jacobian matrix of z_l at u and $\boldsymbol{\varepsilon}$ is an error term that can be estimated in various ways [Spivak, 2018]. In this way, we derive several specialized bounds for different types of layers:

Softmax. For a softmax layer, which is typically the final layer of a classifier, we find the bound

$$\|r\| \geq \min_{c \neq c'} \frac{|(w_{c'} - w_c)^\intercal x + b_{c'} - b_c|}{\|w_{c'} - w_c\|},$$

where W is the weight matrix of the layer and b is its bias. We also show that this bound is tight, in the sense that it is exact for certain linear classifiers.

Fully-connected layers. For general fully-connected layers, we obtain

$$\|r\| \geq \frac{\sqrt{\|J(x)\|^2 + 2M\sqrt{n}\kappa} - \|J(x)\|}{M\sqrt{n}},$$

where κ is a robustness lower bound for the subsequent layer, n denotes the number of outputs of the layer and M bounds the second-order derivatives of the layer function h_l. The assumption of second-order differentiability is usually satisfied in practice under mild assumptions, and the upper bound M can be computed efficiently.

Convolutional layers. For convolutional layers with ReLU activation functions we find

$$\|r\| \geq \frac{\kappa}{\|W\|_F},$$

where κ is a robustness lower bound of the next layer, W is the convolutional kernel and $\|\cdot\|_F$ denotes the Frobenius norm.

Pooling layers. For L_p pooling layers, we have the bound

$$\|r\| \geq \frac{\kappa}{tq^{2/p}}$$

where κ is the robustness of the next layer, t is the dimensionality of the output and q is the size of the receptive field. For max pooling, we can let $p \to \infty$ to obtain

$$\|r\| \geq \frac{\kappa}{t}.$$

The general conclusion we can draw from these lower bounds, is that robustness can be improved using certain regularization techniques which penalize the norms of the weight matrices. The bound on fully-connected layers implies that activation functions with bounded second-order derivatives are also desirable. Moreover, the norm of the Jacobian $\|J(x)\|$ plays an important role: all else being equal, the robustness of an affine layer is maximized when $\|J(x)\|$ is minimized. We note the basic fact in mathematical analysis that a function is Lipschitz continuous if and only if its Jacobian has bounded spectral norm, so our result supports the theory that Lipschitz continuous layers improve robustness in DNNs.

2.6 Historical notes

Reading the literature on adversarial machine learning, one would be tempted to conclude that this is a relatively new discovery that was first described by Szegedy et al. [2013] and concurrently by Biggio et al. [2013]. However, adversarial examples already had a long history before their discovery in the context of DNNs. The issue of "adversarial classification" was already discussed by Dalvi et al. [2004] in the context of classical ML. In one of the seminal papers of the field, Lowd and Meek [2005] introduced the *adversarial classifier reverse engineering* (ACRE) learning problem, in fact initiating the formal study of the feasibility of constructing adversarial examples.

The ACRE learning problem attempts to quantify the hardness of learning sufficient information about a classifier to construct adversarial attacks when given only query access to it. In this way, Lowd and Meek bring an interesting nuance to the modern discourse on adversarial threat models: while Carlini et al. [2019] rightly emphasize the importance of *adaptive* robustness evaluations (where an adversary has full knowledge of the classifier, including any and all defensive measures), the difficulty of actually obtaining this knowledge in practice cannot be summarily dismissed. Indeed, providing robustness in the worst-case scenario advocated by Carlini et al. would be theoretically optimal, but it may also be unnecessarily pessimistic if no attacker could feasibly obtain white-box access to the deployed model.[12]

The ACRE learning problem assumes a binary classification problem where an attacker has query access to the target model for arbitrary inputs and has access to an adversarial cost function $a(x)$ which maps samples to non-negative real numbers (similarly to a norm in the common L_p threat model). The adversary also knows at least one positive instance x^+ and one negative instance x^-. The *minimal adversarial cost* (MAC) of a classifier f and cost function a is the minimum cost $a(x)$ over all instances x classified negatively by f:

$$\mathrm{MAC}(f,a) = \min_{x \in \mathcal{X}} \{a(x) \mid f(x) = 0\}.$$

The *instances of minimal adversarial cost* (IMAC) is the set of all instances x classified negatively by f and which achieve the minimal cost:

$$\mathrm{IMAC}(f,a) = \{x \in \mathcal{X} \mid f(x) = 0, a(x) = \mathrm{MAC}(f,a)\}.$$

As a relaxation of this definition, Lowd and Meek also define

$$k{-}\mathrm{IMAC}(f,a) = \{x \in \mathcal{X} \mid f(x) = 0, a(x) \leq k\mathrm{MAC}(f,a)\}.$$

Formally, the ACRE learning problem is then defined as the problem of finding instances $x \in \mathrm{IMAC}(f,a)$. A set of classifiers $\mathcal{F}$ and cost functions $\mathcal{A}$ is *ACRE learnable* if there exists an algorithm that, for any $f \in \mathcal{F}$ and $a \in \mathcal{A}$, finds some $x \in \mathrm{IMAC}(f,a)$ using only polynomially many queries in the number of parameters of f and the dimensionality of the data. *ACRE k-learnable* is defined similarly for $x \in k{-}\mathrm{IMAC}(f,a)$.

[12] Of course, one can never exclude the possibility of an "inside job" where an adversary has infiltrated the target organization and illicitly obtained complete access to their systems. In that case, however, it may be safe to say the organization has much bigger problems than just the threat of white-box adversarial attacks. It is also not clear why, if an adversary has obtained such a high level of systems access, they would not simply install some ransomware to encrypt sensitive data instead of bothering with adversarial attacks.

Under this formalism, Lowd and Meek [2005] prove that linear classifiers are ACRE $(1+\varepsilon)$-learnable under linear cost functions for any $\varepsilon > 0$. That is, we can obtain arbitrarily accurate approximations of instances of minimal adversarial cost for any linear classifier using only polynomially many queries in the number of parameters of the model and the dimensionality of the data. The only requirement is that we possess at least one positive example and one negative example, which is feasible in many scenarios. This shows that the white-box threat model is realistic for linear classifiers, since an attacker does not need many queries to learn enough information about the classifier to fool it with adversarial examples.

To the best of our knowledge, the ACRE learning problem has not yet been revisited by contemporary ML research. As such, it opens a few interesting lines of inquiry that remain underexplored:

1. Prove ACRE k-learnability for deep neural networks, or restricted classes of DNNs such as deep ReLU networks.
2. Theorize efficient and effective ML algorithms which are provably not ACRE learnable.
3. Theorize efficient and effective ML algorithms which are provably not ACRE k-learnable for reasonable values of k.

These questions are clearly of great importance to the adversarial ML community, but it appears they have not yet been seriously considered. This is particularly striking given the surge of research interest in black-box attacks, which are essentially variations of the original ACRE algorithm proposed by Lowd and Meek [2005]. It is a shame this line of research appears to have been abandoned.

Chapter 3

Conformal prediction

> A certain type of perfection can only be realized
> through a limitless accumulation of the imperfect.
>
> *Kafka on the Shore*
> Haruki Murakami

This chapter serves as an introduction to the principles and practice of the field of *conformal prediction*, which is relevant to the later chapters 4 and 5 where I develop adversarial defenses based on these ideas. I will first explain the theory of conformal prediction and derive a relatively simple preliminary defense method, detailed in algorithm 3.3. This algorithm is applied in a case study on the detection of malicious domain names to demonstrate its potential. Its various shortcomings are discussed and addressed in the following two chapters.

Deep neural networks in the supervised classification setting are typically constructed based on *scoring functions*. These are mappings $g : \mathcal{X} \to \mathbb{R}^k$ from the input domain $\mathcal{X}$ to a real vector space $\mathbb{R}^k$ where the dimensionality k is equal to the number of possible output labels $|\mathcal{Y}|$. In other words, the scoring function g takes any sample $x \in \mathcal{X}$ and assigns a real-valued score to every class $y \in \mathcal{Y}$. These vectors of scores $g(x) = [g_1(x), \ldots, g_k(x)]$ must subsequently be transformed into individual probabilities $p_1(x), \ldots, p_k(x)$ such that

$$p_y(x) \approx \Pr[Y = y \mid X = x]$$

for every $y \in \mathcal{Y}$. That is, the quantities $p_y(x)$ must be accurate estimates of the true conditional probability that the label is y given that the input is x. The process of transforming these raw scores to probabilities is referred to as *calibration* [Guo et al., 2017]. The most common method used in the literature is known as *Platt's scaling* [Platt et al., 1999]. It calibrates the scores $g(x)$ as follows:

$$p_y(x) = \frac{\exp(w_y^\intercal g(x) + b_y)}{\sum_{y' \in \mathcal{Y}} \exp(w_{y'}^\intercal g(x) + b_{y'})}.$$

Here, $w_1, \ldots, w_k \in \mathbb{R}^k$ are learned weight vectors and $b_1, \ldots, b_k \in \mathbb{R}$ are learned biases. It is easily seen that the resulting values $p_1(x), \ldots, p_k(x)$ do indeed satisfy the basic axioms of

probability, *i.e.*, the values all lie within the interval $[0,1]$ and they sum to unity. The final label is then determined by looking at the most likely estimated outcome:

$$\hat{y}(x) = \operatorname*{argmax}_{y \in \mathcal{Y}} \; p_y(x).$$

However, as discussed by Guo et al. [2017] and Gal [2016], Platt's scaling can be unreliable: it can be prone to over-estimating the probabilities $p_y(x)$. This is especially true for modern neural networks trained using the typical categorical cross-entropy loss, which optimizes the parameters of the model so as to minimize the negative log-likelihood (NLL) of the ground-truth labels over the training data set:

$$\theta^\star = \min_\theta \; -\sum_{i=1}^{m} \log \; p_{y_i}(x_i;\theta).$$

Since $p_y(x;\theta) \in [0,1]$, the NLL takes values in the interval $[0,+\infty)$. The global minimum is reached when the neural network outputs a probability of 100% for the ground-truth labels on every training sample (and 0% probability for all other classes). This method of training, which is typical for neural network classifiers, therefore *encourages* overconfidence. This phenomenon was confirmed empirically by Guo et al. [2017], but it seems to be merely a logical consequence of optimization based on the NLL over the training data.

One obvious way to correct this flaw would be to not use probabilities of 100%, but to use more accurate estimates. Of course, such estimates are almost never available and would be highly debatable and subjective for many real-world tasks. To address this issue, Szegedy et al. [2016] proposed *label smoothing*, a technique that aims to "soften" the outputs computed by neural networks even in the absence of exact ground-truth probabilities. This is done via a simple linear interpolation scheme added on top of an existing calibration method:

$$p'_y(x;\theta) = (1-\varepsilon)p_y(x;\theta) + \varepsilon u(y).$$

The most straightforward choice for u is the discrete uniform distribution on $\mathcal{Y}$, which leads to

$$p'_y(x;\theta) = (1-\varepsilon)p_y(x;\theta) + \frac{\varepsilon}{k}.$$

If the network is overconfident and places 100% probability on a certain output, *i.e.*, $p_y(x;\theta) = 1$ for some y and $p_{y'}(x;\theta) = 0$ for all $y' \neq y$, then the smoothed probability for y will be $p'_y(x;\theta) = 1 - \varepsilon\left(1 + \frac{1}{k}\right)$. In other words, the final probability for y is interpolated between 100% and $1/k$ (*i.e.*, random guessing), and the degree of interpolation is controlled by ε: higher values bring the final prediction closer to $1/k$. Label smoothing has been applied in adversarial settings [Kurakin et al., 2016], but the method lacks theoretical guarantees.

Naturally, the observation that this method of training can lead to overconfident predictors is not new, and is the very reason machine learning practitioners have historically made use of all manner of regularization schemes to prevent overfitting [Vapnik, 1999]. As Guo et al. [2017] point out, however, regularization seems to have diminishing returns as the neural networks become larger: the degree of overconfidence and miscalibration becomes more pronounced as the number of parameters of the network grows.

One plausible avenue of research, therefore, is to consider adversarial examples through the lens of *calibration*. That is, we consider that adversarial examples are a consequence of failing to

properly calibrate our models, leading to a general over-estimation of the conditional probabilities which can be exploited in certain cases to yield high-confidence predictions of erroneous labels. This idea, which we call the *calibration hypothesis*, has been explored in other work such as Qin et al. [2021] and Lee et al. [2018]. Although difficult to prove with certainty, there is evidence in the literature that calibration plays an important role in adversarial robustness. For example, in the works by Hein et al. [2019] and Meinke and Hein [2019], it is shown that ReLU networks in particular[1] can severely over-estimate class membership probabilities for specific samples. The randomized gradient-free adversarial attack by Croce et al. [2020] in fact builds on these ideas to generate adversarial examples for ReLU networks, showing that overconfidence (and hence miscalibration) can be exploited in practice to fool classifiers.

With the calibration hypothesis in mind, it is natural to consider alternatives to Platt scaling. Specifically, we are interested in machine learning algorithms that provide *provable* guarantees on their calibration properties. It turns out that such algorithms do exist: they are known as *conformal predictors* [Gammerman and Vovk, 2007, Shafer and Vovk, 2008, Vovk et al., 2005, 2015].

The contents of this chapter are based on Peck et al. [2020].

3.1 Background on conformal prediction

Conformal prediction (CP) is a sub-field of statistics and machine learning dedicated to the study of learning algorithms with provable guarantees on their calibration properties. The field has a long and respectable history but appears to be mostly unknown within the modern machine learning and deep learning communities. Although I cannot be sure, I suspect the *frequentist* nature of CP is one reason why it remains mostly unknown in the field of deep learning, as there has been a very clear trend towards more *Bayesian* techniques within DL, likely to the detriment of frequentist methods. Indeed, Bayesian neural networks are highly attractive both for adversarial robustness and calibration properties, because they have an inherent ability to provide uncertainties for their predictions [Gal, 2016]. Mathematically, the prediction of a Bayesian neural network (BNN) is computed via marginalization over all possible model parameters:

$$\Pr[Y = y \mid X = x] = \int_{\Theta} \Pr[\theta] \Pr[Y = y \mid X = x, \theta] \mathrm{d}\theta.$$

Here, $\Pr[\theta]$ is a prior distribution over the model parameters that must be fixed by the practitioner (as is typical of Bayesian methods). In practice, once the prior is fixed, this integral can be approximated using sampling, Monte Carlo Dropout, variational inference or stochastic weight averaging (SWAG) [Gal and Ghahramani, 2016, Jordan et al., 1999, Maddox et al., 2019]. At first glance, this would seem very promising: if the calibration hypothesis is true and if BNNs are capable of providing accurate estimates of uncertainty for their own predictions, then they should also exhibit higher robustness to adversarial attacks. To date, however, no adversarial defense seems to have been published using BNNs that withstands sophisticated adaptive attacks. One reason for this may lie in the choice of prior distribution $\Pr[\theta]$. Conventional wisdom within the Bayesian community dictates that one may choose *uninformative priors* such as the Jeffreys prior:

$$\Pr[\theta] \propto \sqrt{\det I(\theta)}.$$

[1] A ReLU network is any deep neural network that uses only linear or ReLU activation functions as non-linearities in their architectures.

Here, $I(\theta)$ is the Fisher information matrix of the parameter θ. This matrix depends on the likelihood function defined by the model, which is immediately known once the network is defined. However, Gelada and Buckman [2020] argue the exact opposite in the case of overparameterized models: if the model has many more parameters than training samples, one should in fact use so-called *generalization-sensitive* priors. These are priors which assign high probability to models that generalize well. Otherwise, the posterior distribution $\Pr[Y \mid X]$ defined by the BNN will not "concentrate," in the sense that it will not be able to distinguish well-performing models from those that are simply overfitting. In such cases, any Bayesian measures of uncertainty computed for our BNN predictions run the risk of being utterly meaningless. To make Bayesian methods work in adversarial settings, it therefore seems that much more care must be taken in the formulation of the prior distribution.

By contrast, CP is a frequentist procedure that requires no priors. Instead, it relies on the computation of p-values based on some measure of "conformity." Typically, conformal predictors must be tuned on a subset of training data, but they require no details about the underlying model: they only need the scores produced by the model on a sufficiently large sample. In particular, it is entirely possible to apply CP methods to BNNs, essentially combining frequentist and Bayesian inference in one model. As discussed by Wasserman [2011], this combination can be useful in certain cases and can yield Bayesian methods that enjoy frequentist validity guarantees. This is sometimes referred to as *Frasian inference*, a portmanteau of frequentist and Bayesian as well as a tribute to Don Fraser, who originated the idea. The question whether this marriage of frequentist and Bayesian inference may lead to more robust models is certainly intriguing, but we leave this as potential future work.

At a high level, CP is concerned with quantifying the degree to which a given sample "conforms" to data that has already been seen. Following Vovk et al. [2005], a *conformal predictor* is a function Γ which takes as input a confidence level ε, a bag[2] of instances $B = \{\!\{(x_1, y_1), \ldots, (x_m, y_m)\}\!\}$ and an input $x \in \mathcal{X}$. It then outputs a *set* of labels $\Gamma^\varepsilon(B, x) \subseteq \mathcal{Y}$ which satisfies a property called *exact validity*. Intuitively, exact validity means that the true label of x is in the set $\Gamma^\varepsilon(B, x)$ with probability at least $1 - \varepsilon$. Formally, consider an infinite sequence of samples $\omega = (x_1, y_1), (x_2, y_2), \ldots$ from the data distribution. Assume that the data distribution is *exchangeable*.[3] Define the *error event* at significance ε after n samples as

$$\mathrm{err}_n^\varepsilon(\Gamma, \omega) = \begin{cases} 1 & \text{if } y_n \notin \Gamma^\varepsilon(\{\!\{(x_1, y_1), \ldots, (x_{n-1}, y_{n-1})\}\!\}, x_n), \\ 0 & \text{otherwise.} \end{cases}$$

Exact validity then means that the random variables $\mathrm{err}_1^\varepsilon(\Gamma, \omega), \mathrm{err}_2^\varepsilon(\Gamma, \omega), \ldots$ are all independent and Bernoulli distributed with parameter ε. Interestingly, this property *cannot* be satisfied by deterministic algorithms [Vovk et al., 2005]; any algorithm that satisfies exact validity *must* be randomized. To accommodate deterministic predictors, a weaker property is used known as *conservative validity*. For conservative validity, it is only necessary that the error events $\mathrm{err}_n^\varepsilon(\Gamma, \omega)$ be *dominated in distribution* by independent and identically distributed Bernoulli random variables with parameter ε. Formally, this means that there must exist two sequences of random variables $\xi_1, \xi_2, \ldots$ and $\eta_1, \eta_2, \ldots$ such that

[2] A *bag* or *multiset* is a collection of items where the order is unimportant (like a set) and duplicates are allowed (like a list).

[3] A probability distribution is said to be exchangeable if the order of observations does not matter; every permutation of a sequence is equally likely. In particular, a product distribution (which results in i.i.d. samples) is exchangeable.

1. each ξ_i is independent and Bernoulli distributed with parameter ε;
2. $\mathrm{err}_i^{\varepsilon}(\Gamma, \omega) \leq \eta_i$ almost surely for all i;
3. the joint distribution of $\eta_1, \ldots, \eta_n$ equals the joint distribution of $\xi_1, \ldots, \xi_n$ for all n.

It is easy to see that exact validity implies conservative validity: we can simply let $\mathrm{err}_i^{\varepsilon}(\Gamma, \omega) = \xi_i = \eta_i$ for all i. As shown by Vovk et al. [2005], conservative validity implies that the predictor is *asymptotically conservative*, which corresponds to the following bound:

$$\limsup_{n\to\infty} \Pr\left[\frac{1}{n}\sum_{i=1}^{n} \mathrm{err}_i^{\varepsilon}(\Gamma, \omega) \leq \varepsilon\right] = 1. \tag{3.1}$$

That is, a predictor is asymptotically conservative if the limit superior of its long-term error rate is bounded by ε almost surely. The predictor is *asymptotically exact* if

$$\lim_{n\to\infty} \Pr\left[\frac{1}{n}\sum_{i=1}^{n} \mathrm{err}_i^{\varepsilon}(\Gamma, \omega) \leq \varepsilon\right] = 1. \tag{3.2}$$

Naturally, if the above limit exists, then (3.1) is equivalent to (3.2) and so asymptotically exact implies asymptotically conservative. It can also be shown [Vovk et al., 2005] that exact validity implies asymptotical exactness.

3.2 The general conformal prediction algorithm

Having established the necessary theoretical background on conformal prediction, the question is raised whether there exist any practical algorithms that achieve exact or conservative validity. For our purposes, we shall limit ourselves to algorithms that are *conservatively* valid, as these can often be made exactly valid with a simple randomization trick [Vovk et al., 2005].

Algorithm 3.1: General conformal prediction

Data: Non-conformity measure Δ, confidence level $\varepsilon \in [0,1]$, bag of examples $\{z_1, \ldots, z_m\} \subseteq \mathcal{X} \times \mathcal{Y}$, object $x \in \mathcal{X}$
Result: prediction region $\Gamma^{\varepsilon}(\{z_1, \ldots, z_m\}, x) \subseteq \mathcal{Y}$

```
foreach y ∈ 𝒴 do
    z_{m+1} ← (x, y)
    for i = 1, ..., m+1 do
        α_i ← Δ({z_1, ..., z_m} \ {z_i}, z_i)
    end
    p_y ← 1/(m+1) #{i = 1, ..., m+1 | α_i ≥ α_{m+1}}
end
return {y ∈ 𝒴 | p_y > ε}
```

Algorithm 3.1 shows the general conformal prediction algorithm [Shafer and Vovk, 2008]. This algorithm must be instantiated with a *non-conformity measure* Δ, which can be *any* function that outputs a non-negative real number given a bag of labeled examples $\{z_1, \ldots, z_m\} \subseteq \mathcal{X} \times \mathcal{Y}$ and a new instance $(x, y) \in \mathcal{X} \times \mathcal{Y}$. Intuitively, non-conformity measures must somehow quantify

how much the instance (x,y) deviates from (or does not "conform" to) previously seen labeled instances. Of course, what constitutes a meaningful notion of non-conformity is heavily domain-specific. The interesting property of algorithm 3.1, however, is that it is *always* conservatively valid regardless of the precise definition of Δ. Indeed, Δ may even be a constant and the algorithm would *still* satisfy conservative validity. Naturally, not all non-conformity measures are equally useful: while having conservative validity is nice, it does not guarantee that the algorithm provides useful predictions. For example, the conformal predictor could always output the entire set of labels: $\Gamma^{\varepsilon}(B,x) = \mathcal{Y}$ for every bag B and object $x \in \mathcal{X}$. This algorithm would be conservatively valid, since its error rate is precisely zero, but it yields no useful information about the true labels. A good non-conformity measure must therefore strive to minimize the size of the prediction region.

Given a non-conformity measure Δ, a confidence level ε and a bag of labeled examples,

$$\{(x_1,y_1),\ldots,(x_m,y_m)\},$$

algorithm 3.1 finds a set of candidate labels for a new object $x \in \mathcal{X}$ by iterating over all $y \in \mathcal{Y}$ and computing non-conformity scores $\alpha_1,\ldots,\alpha_{m+1}$ based on the "virtual" instance (x,y). That is, the conformal prediction algorithm provisionally labels x with every possible class y and, for each class, quantifies the non-conformity of (x,y) with respect to the labeled data. It then constructs the prediction region by including all labels y for which the corresponding p-value exceeds the threshold ε. These p-values are determined by

$$p_y \leftarrow \frac{1}{m+1}\#\{i = 1,\ldots,m+1 \mid \alpha_i \geq \alpha_{m+1}\}.$$

In other words, p_y is the fraction of non-conformity scores α_i that are at least as large as α_{m+1}, the non-conformity score of the virtual sample (x,y). If $p_y > \varepsilon$, the algorithm includes y in the prediction region. It is clear that algorithm 3.1 therefore also satisfies a so-called *nestedness* property:

$$\varepsilon_1 \leq \varepsilon_2 \implies \Gamma^{\varepsilon_1}(B,x) \supseteq \Gamma^{\varepsilon_2}(B,x).$$

That is, the prediction regions become larger as ε becomes smaller. This is in fact a desideratum for all conformal predictors.

3.3 Inductive Venn-ABERS predictors

One particular conformal predictor that is of interest to us here, is the *inductive Venn-ABERS predictor* (IVAP) proposed by Vovk et al. [2015]. The IVAP is designed to take advantage of some other inductive learning rule, such as a deep neural network, to improve its predictive efficiency. It can be viewed as a specialization of the general conformal prediction algorithm, where the non-conformity measure Δ is crafted based on the scores produced by an existing model. It was originally proposed for binary classification and, as such, the IVAP does not produce a discrete set of possible labels; instead, on input a sample $x \in \mathcal{X}$, it yields two probabilities (p_0,p_1) satisfying

$$p_0 \leq \Pr[Y = 1 \mid X = x] \leq p_1.$$

Other works have extended the IVAP to the multi-class case [Manokhin, 2017], but we shall first focus on binary classification. The pseudo-code for the IVAP is shown in algorithm 3.2. The

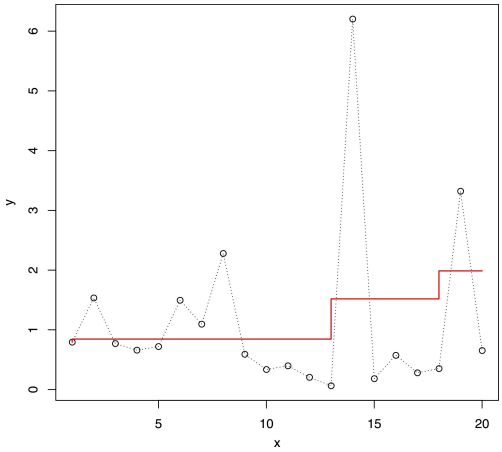


Figure 3.1: Example of isotonic regression on a 2D data set. The isotonic fit is shown as a red line. This image was produced with the `isoreg` function from the R language [R Core Team, 2015].

IVAP first partitions the full data set B into two disjoint sets: a training set B_1 and a calibration set B_2. The training set B_1 is used to fit a different model (such as a DNN) that supplies us with a scoring function $g : \mathcal{X} \to \mathbb{R}$. The scores are then computed on the held-out calibration subset B_2. With these scores, two isotonic regressions are performed based on the calibration set B_2 as well as the two virtual samples $(g(x),0)$ and $(g(x),1)$. Isotonic (or monotonic) regression aims to fit a non-decreasing line to a sequence of observations such that the line lies as close to these observations as possible. Figure 3.1 shows an example of isotonic regression applied to a simple 2D data set.

Algorithm 3.2 performs the isotonic regression as follows. Let the computed scores on the calibration set be $s_1,\ldots,s_m$. First, the scores are sorted in ascending order and duplicates are removed, yielding a subset $s'_1 \leq \cdots \leq s'_n$. The *multiplicity* of a score s'_j is defined as the number of times that score value appears in the data set:

$$w_j = \#\{i \mid s_i = s'_j\}.$$

The "average label" for s'_j is

$$y'_j = \frac{1}{w_j} \sum_{i:s_i=s'_j} y_i.$$

For binary classification with $\mathcal{Y} = \{0,1\}$, the average labels will be real numbers between 0 and 1. The *cumulative sum diagram* (CSD) is the set of points $P_1,\ldots,P_n$ defined by

$$P_i = \left(\sum_{j=1}^{i} w_j, \sum_{j=1}^{i} w_j y'_j \right) = (W_i, Y_i).$$

For these points, the *greatest convex minorant* (GCM) is computed [Tuy, 1998]. Formally, the GCM of a function $f : U \to \mathbb{R}$ is the maximal convex function $h : I \to \mathbb{R}$ defined on a closed interval I containing U such that $h(u) \leq f(u)$ for all $u \in U$. Intuitively, the GCM is the "lowest part" of the convex hull of f. An example is given in figure 3.2.

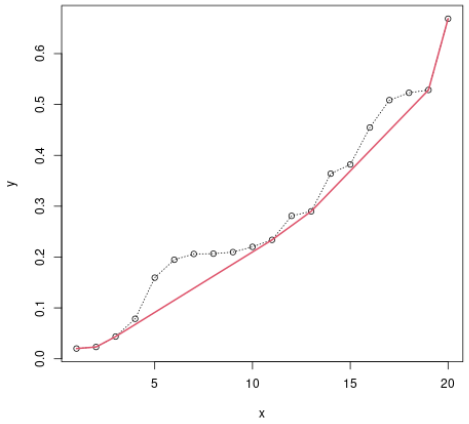


Figure 3.2: Example of the greatest convex minorant computed for a set of points sorted in ascending order. This image was produced with the `fdrtool` package [Klaus and Strimmer, 2015].

The value of the isotonic fit f computed by algorithm 3.2 at s'_j is now defined as the slope of the GCM between W_{j-1} and W_j. That is,

$$f(s'_j) = \frac{h(W_j) - h(W_{j-1})}{w_j},$$

where h is the GCM of the CSD $P_1, \ldots, P_n$. We leave the functions undefined at other points, as we will never need those values.

3.3.1 Detecting adversarial manipulation using IVAPs

According to the calibration hypothesis, adversarial examples may be a consequence of badly calibrated models where the predicted probability of a label can be a poor approximation of the true probability, leading to overconfidence that can be exploited to fool the model with minor modifications to the inputs. In this regard, IVAPs are extremely interesting because they enjoy a very particular type of validity known as *perfect calibration*. Formally, a random variable P is *perfectly calibrated* with respect to another variable Y if the following equality holds:

$$\mathbb{E}[Y \mid P] = P. \tag{3.3}$$

In the case of the IVAP, Vovk et al. [2015] show that it satisfies perfect calibration in the following sense. Let P_0 and P_1 be the random variables corresponding to the output probabilities of the IVAP on a random sample (X,Y) from the data distribution. Then there exists a random variable S supported on $\{0,1\}$ such that P_S is perfectly calibrated with respect to Y. Formally, for the IVAP algorithm, the guarantee expressed by (3.3) reduces to

$$\Pr[Y = 1 \mid P_S] = P_S. \tag{3.4}$$

Algorithm 3.2: Inductive Venn-ABERS predictor

Data: bag of samples $B = \lbag z_1, \ldots, z_m \rbag \subseteq \mathcal{X} \times \mathcal{Y}$, object $x \in \mathcal{X}$, learning algorithm A
Result: pair of probabilities (p_0, p_1) with $p_0 \leq p_1$

Divide the full bag of samples $B = \{z_1, \ldots, z_m\}$ into two disjoint bags $B_1 = \{z_1, \ldots, z_n\}$ and $B_2 = \{z_{n+1}, \ldots, z_m\}$.
Run the learning algorithm A on B_1 to obtain a scoring function g.
for $i = n+1, \ldots, m$ **do**
    $s_i \leftarrow g(x_i)$
end
$s \leftarrow g(x)$
Fit isotonic regression to $\{(s_{n+1}, y_{n+1}), \ldots, (s_m, y_m), (s, 0)\}$, obtaining a function f_0.
Fit isotonic regression to $\{(s_{n+1}, y_{n+1}), \ldots, (s_m, y_m), (s, 1)\}$, obtaining a function f_1.
$p_0 \leftarrow f_0(s)$
$p_1 \leftarrow f_1(s)$
return (p_0, p_1)

Algorithm 3.3: IVAP-based defense against adversarial examples

Data: bag of samples $B = \lbag z_1, \ldots, z_m \rbag \subseteq \mathcal{X} \times \mathcal{Y}$, object $x \in \mathcal{X}$, learning algorithm A, precision $\beta \in [0, 1]$
Result: Label $y \in \mathcal{Y} \cup \{\bot\}$

Run algorithm 3.2 to obtain probabilities p_0 and p_1.
if $p_1 - p_0 \leq \beta$ **then**
    Set $p \leftarrow \frac{p_1}{1 - p_0 + p_1}$.
    if $p > 1/2$ **then**
        return *1*
    else
        return *0*
    end
else
    return $\bot$
end

In other words, for every input sample, at least one of the probabilities p_0 or p_1 will be almost surely equal to the true probability that the label is 1.

The difference $p_1 - p_0$ is therefore a natural measure of confidence in the prediction: if the difference is too large, the model cannot be trusted. This is the core idea of our defense against adversarial examples: we use the IVAP to calibrate an existing model and flag all predictions where the difference $p_1 - p_0$ exceeds a certain threshold β. Flagging is done by extending the label space $\mathcal{Y}$ with a special "reject" option, which we write as $\perp$. For predictions that are not flagged, we determine the final probability using the formula proposed by Vovk et al. [2015]:

$$p = \frac{p_1}{1 - p_0 + p_1}.$$

If $p > 1/2$, we return the label 1; otherwise, we return 0. The pseudo-code for this defense is shown in algorithm 3.3. This method of quantifying uncertainty and rejecting unreliable predictions was in fact already mentioned by Vovk et al. [2015], but to our knowledge nobody has yet attempted to use it for protection against adversarial examples.

We note that the threshold for p can of course be set to a different value than 50%, but we did not explore such tuning here as the approach already worked well with this default. The precision threshold β is tuned by maximizing Youden's index [Youden, 1950] on a held-out validation set of clean and adversarial examples.

Calibration vs uncertainty. Having formally defined the calibration property of the IVAP algorithm, we can briefly turn to a more philosophical discussion about our definitions of *uncertainty* and *calibration*, as these concepts are sometimes confused. It should be clear from (3.3) that perfect calibration does not imply the absence of uncertainty. Indeed, perfect calibration of P with respect to Y only guarantees that the expectation of Y can be derived from P; it does not mean that we know precisely what value Y has taken on at any given moment, even after observing P. The stochastic regularization techniques proposed by Gal [2016], Gal and Ghahramani [2016] are ways of quantifying this uncertainty, typically by using the variance of the model output under various stochastic perturbations to the parameters. Calibration merely means that the predicted probabilities supplied by the model match the empirical frequency of the data. However, even perfectly calibrated models can still yield uncertain predictions. As a simple example, consider a model that predicts the outcome of a fair dice roll. A perfectly calibrated model would always predict 1/6 probability for each outcome, but this makes us no less uncertain about the result of the next roll. We are concerned here only with post-hoc calibration, *i.e.*, attempting to correct predicted probabilities of given models so that they more closely match the ground-truth distribution. That is not to say conformal prediction cannot also deal with uncertainty, however: the notion of *credibility* [Shafer and Vovk, 2008] comes very close, and it would be an interesting avenue for future work to study whether credibility can be incorporated together with calibration to produce even better adversarial defenses.

3.3.2 Data sets

Algorithm 3.3 only works for *binary* classification tasks, so our experimental evaluation has to be appropriately limited. We make use of the following data sets to assess the performance of our defense:

Zeroes vs ones Our first task is a binary classification problem based on the MNIST data set [LeCun et al., 1998]. We take the original MNIST data set and filter out only the images displaying either a zero or a one. The pixel values are normalized to lie in the interval $[0,1]$. We then run our experiments as described above, using a convolutional neural network (CNN) as the scoring classifier for algorithm 3.3.

Cats vs dogs The second task we consider is the classification of cats and dogs in the Asirra data set [Elson et al., 2007]. This data set consists of 25,000 JPEG color images where half contain dogs and half contain cats. Again we train a CNN as the scoring classifier for algorithm 3.3. We resized all images to 64x64 pixels and normalized the pixel values to $[0,1]$ to facilitate processing by our machine learning pipeline.

T-shirts vs trousers The third task is classifying whether a 28x28 grayscale image contains a picture of a T-shirt or a pair of trousers. Similarly to the MNIST zeroes vs ones task, we take the Fashion-MNIST data set [Xiao et al., 2017] and filter out the pictures of T-shirts and trousers. Again, the pixel values are normalized to $[0,1]$ and a CNN is used for this classification problem.

Airplanes vs automobiles Our fourth task is based on the CIFAR-10 data set [Krizhevsky and Hinton, 2009], which consists of 60,000 RGB images 32x32 pixels in size. We filter out the images of airplanes and automobiles and train a CNN to distinguish these two classes.

3.3.3 Adaptive adversarial attack

As pointed out by Carlini and Wagner [2017c], it is not sufficient to demonstrate that a new defense is robust against existing attacks. To make any serious claims of adversarial robustness, we must also develop and test a white-box attack that was designed to specifically target models protected with our defense. Let $x \in \mathcal{X}$ be any input that is not rejected and classified as y by the detector. Our attack must then find $\tilde{x}$ such that

1. $\|x - \tilde{x}\|_p$ is as small as possible;
2. $\tilde{x}$ is not rejected by the detector;
3. $\tilde{x}$ is classified as $1 - y$.

Here, $\|u\|_p$ is the general L_p norm of the vector u. Common choices for p in the literature are $p \in \{1, 2, \infty\}$; we will focus on L_2 and L_∞ norms here.

Following Carlini and Wagner [2017c], we design a differentiable function that is minimized when $\tilde{x}$ lies close to x, is not rejected by the detector and is classified as $1 - y$. To do this efficiently, note that whenever a new sample has the same score as an old sample in the calibration set, it will have no effect on the isotonic regression and the result of applying algorithm 3.3 to it will be the same as applying the algorithm to the old sample. Hence, we search for a sample (s_i, y_i) in the calibration set such that

1. $y_i = 1 - y$, so the sample satisfies the target classification;
2. $f_1(s_i) - f_0(s_i) \leq \beta$, so the sample is not rejected;

3. the confidence value $p = \frac{f_1(s_i)}{1-f_0(s_i)+f_1(s_i)}$ satisfies $p > 1/2$ if $y_i = 1$ and $p \leq 1/2$ otherwise.

From among all samples that satisfy these conditions,[4] we choose the one which minimizes the following expression:

$$\|x - x_i\|_2^2 + c(s(x) - s_i)^2.$$

Here, $s(x)$ is the score assigned to x by the classifier and c is a constant which trades off the relative importance of reducing the size of the perturbation vs fooling the detector. We choose these samples in this way so as to "warm-start" our attack: we seek a sample that lies as close as possible to the original x but also has a score s_i which lies as close as possible to $s(x)$. This way, we hope to minimize the amount of effort required to optimize our adversarial example.

Having fixed such a sample (s_i, y_i), we solve the following optimization problem:

$$\min_{\delta \in [-1,1]^d} \|\delta\|_2^2 + c(s(x+\delta) - s_i)^2 \text{ subject to } x + \delta \in [0,1]^d. \tag{3.5}$$

The constant c can be determined via binary search, as in the Carlini & Wagner attack [Carlini and Wagner, 2017c]. In our case, we are applying the attack to a standard neural network without any non-differentiable components, meaning the score function s is differentiable. The resulting problem (3.5) can therefore be solved using standard gradient descent methods since the function to be minimized is differentiable as well. In particular, we use the Adam optimizer [Kingma and Ba, 2014] to minimize our objective function. The constraint can easily be enforced by clipping the values of $x + \delta$ back into the $[0,1]^d$ hypercube after each iteration of gradient descent.

Note that our custom white-box attack will not suffer from any gradient masking introduced by our defense. Indeed, we only rely on gradients computed from the underlying machine learning model. As long as the unprotected classifier does not mask gradients (which it should not, since the unprotected classifiers will be vanilla neural networks trained in the standard way), this information will be useful.

As described, the above method of selecting a calibration sample and solving (3.5) only considers adversarials that are minimal in L_2 distance. However, L_∞-bounded perturbations are also of interest, so in our experiments we evaluate both the original L_2 formulation as well as an L_∞ variant.

3.3.4 Experiments

We performed experiments with algorithm 3.3 in Peck et al. [2020] on the above mentioned data sets. For each data set, we consider several different settings:

- *Clean*. Here, the metrics are computed on the original proper test set, without any modifications.
- *Adversarial*. The metrics are computed on the adversarial test set. This data set is constructed by running existing adversarial attacks against the underlying neural network

[4] Note that, strictly speaking, we are not guaranteed to find a sample matching all of these requirements. However, if such a sample does not exist, then every sample in the calibration set is either classified as y, rejected or has its predicted label altered by the IVAP. Although this is theoretically possible, it likely points to problematic data or a very poor underlying model.

on the proper test set. They do not take the IVAP into account. The attacks we employed were projected gradient descent with random restarts [Madry et al., 2017], DeepFool [Moosavi-Dezfooli et al., 2016], local search [Narodytska and Kasiviswanathan, 2016], the single pixel attack [Su et al., 2019], NewtonFool [Jang et al., 2017], fast gradient sign [Goodfellow et al., 2014] and the momentum iterative method proposed by Dong et al. [2018].

- *Adapted*. This is similar to the Adversarial scenario but the attacks are modified to take the IVAP into account. That is, we run the same set of existing adversarial attacks, but we target the full defended model instead of the original. For the gradient-based attacks, we use the natural evolution strategies approach to estimate the gradient, thereby avoiding any possible gradient masking issues [Ilyas et al., 2018, Wierstra et al., 2014].
- *Custom L_p*. We compute the metrics for adversarial examples generated using our custom L_p white-box attack on the proper test set. We report results for $p = 2$ and $p = \infty$.

For all scenarios, we report the accuracy, true positive rate (TPR), true negative rate (TNR), false positive rate (FPR) and false negative rate (FNR) of the detectors. These are defined as follows:

$$\text{Accuracy} = \frac{\text{TA} + \text{TR}}{N}, \qquad \text{TPR} = \frac{\text{TA}}{\text{TA} + \text{FR}},$$
$$\text{TNR} = \frac{\text{TR}}{\text{TR} + \text{FA}}, \qquad \text{FPR} = \frac{\text{FA}}{\text{FA} + \text{TR}},$$
$$\text{FNR} = \frac{\text{FR}}{\text{FR} + \text{TA}}.$$

Here, N is the total number of samples, TA is the number of correct predictions the detector accepted, TR is the number of incorrect predictions the detector rejected, FA is the number of incorrect predictions the detector accepted and FR is the number of correct predictions the detector rejected. The hyperparameter β was tuned to maximize Youden's index, which is defined as

$$J = \text{TPR} - \text{FPR}.$$

Youden's J is defined for every point on the ROC curve. Graphically, it corresponds to the distance between the ROC curve and the random chance line.

The implementations of the adversarial attacks, including the gradient estimator using the natural evolution strategies, were provided by the Foolbox library [Rauber et al., 2020]. The implementation of the Venn-ABERS predictor was provided by Toccaceli [Toccaceli, 2017]. The different data splits we perform for these experiments are illustrated schematically in figure 3.3. Almost all neural networks were trained for 50 epochs using the Adam optimizer [Kingma and Ba, 2014] with default parameters in the TensorFlow Keras framework [Chollet et al., 2015]. The exception is the CNN for the cats vs dogs task, which was trained for 100 epochs. No regularization or data augmentation schemes were used; the only preprocessing we perform is a normalization of the input pixels to the $[0, 1]$ range as well as resizing the images in one of the tasks.

Table 3.1 reports the accuracy scores of the unprotected baseline models and table 3.2 summarizes the metrics of the IVAPs for each of the tasks. From these results, we can see that the clean accuracy of the protected model invariably suffers from the addition of the IVAP. However, in

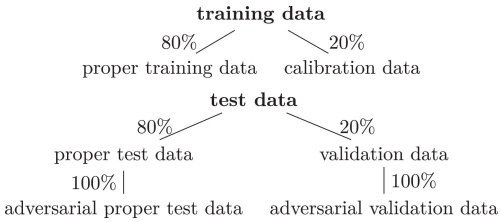


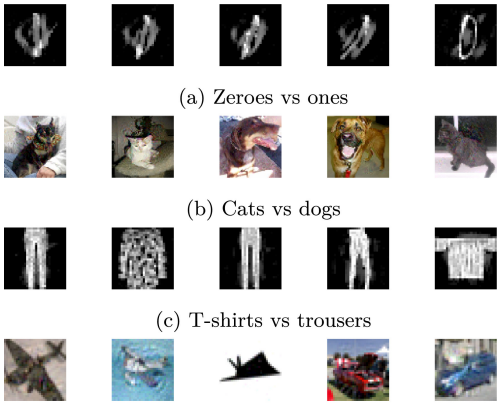
Figure 3.3: Illustration of the different data splits. The training data is split into proper training data on which the scoring classifier is trained and calibration data on which the IVAP fits the isotonic regression. The test data is split into proper test data, which is used to determine the overall accuracy of the resulting algorithm, and validation data. Both of these subsets of test data are used to generate adversarial proper test data and adversarial validation data. The adversarial proper test data is used for evaluating the robustness of the resulting algorithm, whereas the adversarial validation data is used together with the validation data to tune β. All splits are 80% for the left branch and 20% for the right branch.

Data set	Clean accuracy	Adversarial accuracy
Zeroes vs ones	100%	2.31%
Cats vs dogs	89.87%	1.80%
T-shirts vs trousers	99.75%	3.43%
Airplanes vs automobiles	96.69%	3.75%

Table 3.1: Accuracy scores of the unprotected models

Data set	Setting	Accuracy	TPR	TNR	FPR	FNR
Zeroes vs ones	Clean	98.52%	98.51%	100.00%	0.00%	1.49%
	Adversarial	46.07%	0.00%	100.00%	0.00%	100.00%
	Adapted	97.41%	0.00%	99.06%	0.94%	100.00%
	Custom L_2	0.00%	0.00%	0.00%	100.00%	100.00%
	Custom L_∞	0.00%	0.00%	0.00%	100.00%	100.00%
Cats vs dogs	Clean	72.94%	75.90%	48.85%	51.15%	24.10%
	Adversarial	52.56%	71.25%	38.86%	61.14%	28.75%
	Adapted	59.27%	100.00%	59.18%	40.82%	0.00%
	Custom L_2	1.64%	100.00%	0.00%	100.00%	0.00%
	Custom L_∞	1.69%	100.00%	0.00%	100.00%	0.00%
T-shirts vs trousers	Clean	96.88%	96.86%	97.44%	2.56%	3.14%
	Adversarial	50.99%	0.00%	99.97%	0.03%	100.00%
	Adapted	77.53%	0.00%	77.99%	22.01%	100.00%
	Custom L_2	0.00%	0.00%	0.00%	100.00%	100.00%
	Custom L_∞	0.00%	0.00%	0.00%	100.00%	100.00%
Airplanes vs automobiles	Clean	74.38%	73.21%	96.3%	3.7%	26.79%
	Adversarial	58.14%	0.00%	100.00%	0.00%	100.00%
	Adapted	93.99%	0.00%	94.82%	5.18%	100.00%
	Custom L_2	0.00%	0.00%	0.00%	100.00%	100.00%
	Custom L_∞	0.00%	0.00%	0.00%	100.00%	100.00%

Table 3.2: Results of the IVAP defense



(a) Zeroes vs ones

(b) Cats vs dogs

(c) T-shirts vs trousers

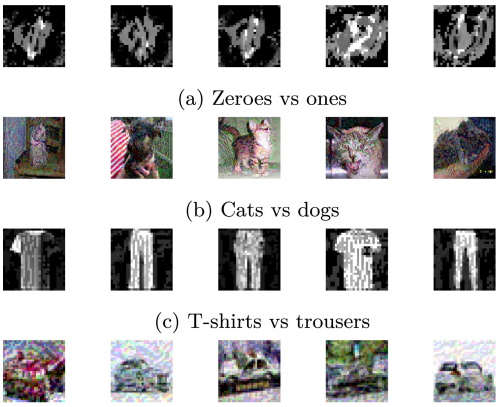
(d) Airplanes vs automobiles

Figure 3.4: Selections of adversarial examples which can fool our detectors, generated by the custom L_2 white-box attack.

almost all tasks, the false positive and false negative rates are relatively low for the Clean setting. Our detectors are therefore capable of accepting correct predictions and rejecting mistaken ones from the underlying model on clean data. The exception is the cats vs dogs task, where the false positive rate is unusually high at 51.15%. However, we believe these results might be improved by using a better underlying CNN. In our initial experiments, we used the same CNN for this task as we did for airplanes vs automobiles and trained it for only 50 epochs. Using a more complex CNN and training it longer significantly improved the results, so we are optimistic that the results we obtained can be improved even further by adapting the underlying CNN and training regime.

We also evaluate our IVAP defense against a suite of adversarial attacks. The results are shown for the Adversarial and Adapted scenarios in table 3.2. When we transfer adversarials generated for the underlying model to the IVAP, the accuracy always degrades significantly. However, it is important to note that all unprotected CNNs have very low accuracy on these samples. As such, in almost all cases, the IVAP has a very high true negative rate as most predictions are in error. The exception is the cats vs dogs task, where the true negative rate is rather low on the transferred adversarials. The accuracy is still much higher than the unprotected model, though (52.56% vs 1.80%), so the IVAP did significantly improve the robustness against this transfer attack. The adversarials generated using existing attacks adapted to the presence of the IVAP also degrade the accuracy, but not as severely as the transfer attack. True negative rates remain high across all tasks and the level of accuracy is, at the very least, much better than without IVAP protection.

To more thoroughly evaluate the strength of our defense, it is necessary to also test it against an adaptive adversary utilizing an attack that is specifically designed to circumvent our defense. For this reason, table 3.2 also shows two rows of test results for each task, where we run our custom white-box attack for both the L_2 and L_∞ norms. At first glance, it appears as though our attack is completely successful and fully breaks our defense, yielding 0.00% accuracy. To investigate this further, figures 3.4 and 3.5 show random selections of five adversarial examples that managed to fool the different detectors, generated by our white-box attack. Empirical cu-



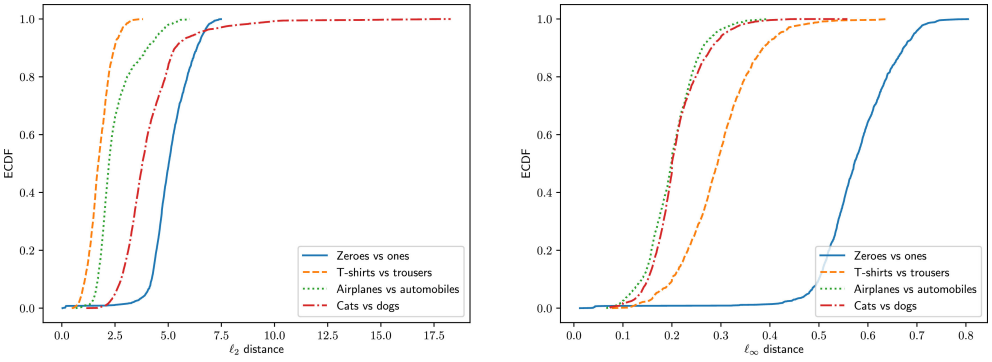
(a) Zeroes vs ones

(b) Cats vs dogs

(c) T-shirts vs trousers

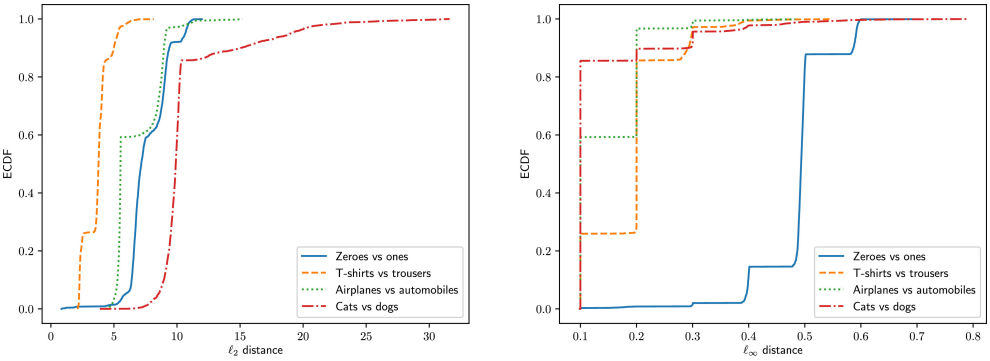
(d) Airplanes vs automobiles

Figure 3.5: Selections of adversarial examples which can fool our detectors, generated by the custom L_∞ white-box attack.



(a) ℓ_2 (b) ℓ_∞

Figure 3.6: Empirical cumulative distributions of the adversarial distortion produced by our L_2 white-box attack. The distance metrics are L_2 (left) and L_∞ (right).



(a) ℓ_2 (b) ℓ_∞

Figure 3.7: Empirical cumulative distributions of the adversarial distortion produced by our L_∞ white-box attack. The distance metrics are L_2 (left) and L_∞ (right).

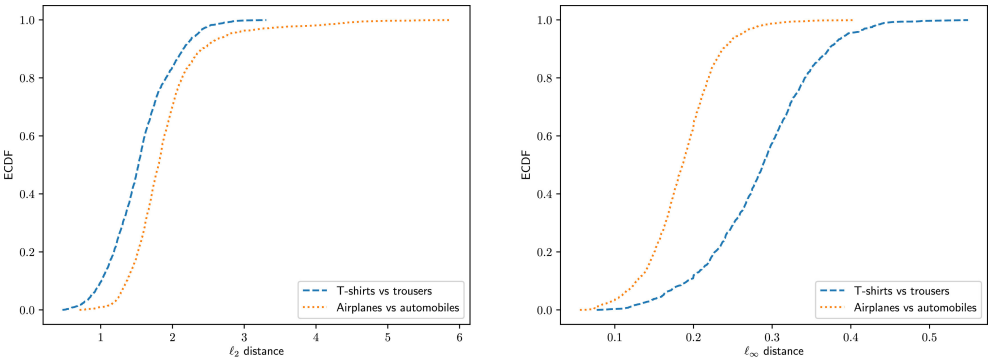
mulative distributions of the distortion levels introduced by our white-box attack are shown in figures 3.6 and 3.7. Visually inspecting the generated adversarials and looking at the distributions of the distortion levels reveals cause for optimism: although some perturbations are still visually imperceptible, there is a significant increase in the overall distortion required to fool the detectors. This is especially the case for the L_∞ adversarials, where almost all perturbations are obvious. Even the L_2 adversarials are generally very unsubtle except for the ones generated on the cats vs dogs task. We believe this is due to the fact that the Asirra images are larger than the CIFAR-10 images (64x64 vs 32x32), so that the L_2 perturbations can be more "spread out" across the pixels. Regardless, if one compares the adversarial images to the originals, the perturbations are still obvious because the adversarials are unusually grainy and blurred.

3.3.5 Ablation study

For the experiments carried out above, the IVAP was only exposed to adversarial attacks which it had already seen: the rejection threshold β was tuned on samples perturbed using the same adversarial attacks as it was then tested against. An important question that arises now is how well the IVAP fares if we test it against attacks that differ from the ones used to tune its hyperparameters, since in practice it is doubtful that we will be able to anticipate what attacks an adversary may use. Moreover, even if we can anticipate this, the set of possible attacks may be so large as to make it computationally infeasible to use them all. Therefore, we also test the performance of the IVAP when β is tuned on adversarials generated only by the DeepFool attack instead of the full suite of attacks. Our choice for DeepFool was motivated by the observations that it is efficient and rather effective, but it is by no means the strongest attack in our collection. It is less efficient than fast gradient sign, but this method was never meant to be a serious attack; it was only meant to demonstrate excessive linearity of DNNs. It is weaker than NewtonFool, for example, since this method utilizes second-order information of the model whereas DeepFool is limited to first-order approximations. It is also weaker than some first-order attacks such as the momentum iterative method, which was at the top of the NIPS 2017 adversarial attack competition [Dong et al., 2018, Kurakin et al., 2018]. We therefore feel that DeepFool is a good choice for evaluating how robust our defense is to future attacks.

Data set	Setting	Accuracy	TPR	TNR	FPR	FNR
T-shirts vs trousers	Clean	97.81%	97.89%	94.87%	5.13%	2.11%
	Adversarial	49.31%	3.81%	93.02%	6.98%	96.19%
	Adapted	56.8%	0.00%	56.93%	43.07%	100.00%
	Custom L_2	0.00%	0.00%	0.00%	100.00%	100.00%
	Custom L_∞	0.00%	0.00%	0.00%	100.00%	100.00%
Airplanes vs automobiles	Clean	93.62%	95.39%	60.49%	39.51%	4.61%
	Adversarial	52.63%	0.00%	90.52%	9.48%	100.00%
	Adapted	69.18%	0.00%	69.73%	30.27%	100.00%
	Custom L_2	0.00%	0.00%	0.00%	100.00%	100.00%
	Custom L_∞	0.00%	0.00%	0.00%	100.00%	100.00%

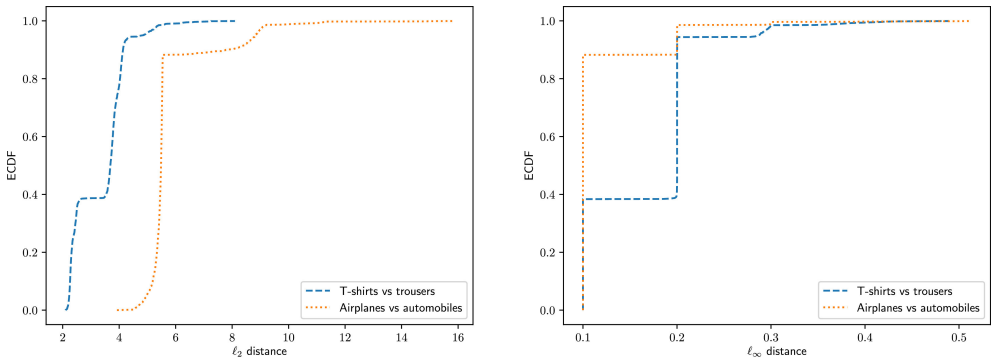
Table 3.3: Results of the ablated IVAP detectors.



(a) ℓ_2 (b) ℓ_∞

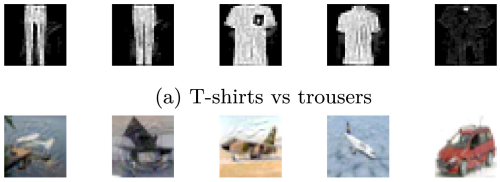
Figure 3.8: Empirical cumulative distributions of the adversarial distortion produced by our L_2 white-box attack on the ablated detectors. The distance metrics are L_2 (left) and L_∞ (right).

We obtained identical values of β for the Zeroes vs ones and Cats vs dogs tasks, so there would be no difference with respect to table 3.2. The results for the other tasks where β was different are shown in table 3.3. The conclusions are similar: the accuracy degrades both on clean and adversarial data, but clean accuracy is still high and true negative rates are high on adversarial data. The custom white-box attack again appears highly successful. Figures 3.8 and 3.9 plot the empirical cumulative distributions of the adversarial distortion levels for the ablated detectors. Figures 3.10 and 3.11 show selections of custom white-box adversarials which fooled the ablated detectors. These results show that robustness to our L_2 white-box attack is significantly lower compared to the non-ablated detectors. The L_∞-bounded perturbations are still very noticeable, however. As an illustrative example, figure 3.12 shows histograms of the differences between the upper and lower probabilities for clean and DeepFool adversarial examples on the zeroes vs ones task. Although there is a small amount of overlap, the vast majority of adversarial examples have a much higher difference than clean samples: clean data has a difference close to 0%, whereas adversarials are almost always close to 50%. The tuned model threshold is placed so that the majority of clean samples are still allowed through but virtually all adversarials are rejected.



(a) ℓ_2 (b) ℓ_∞

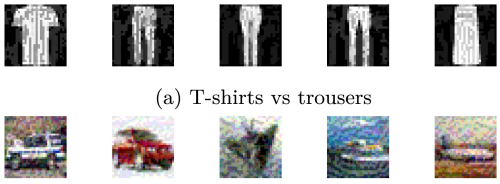
Figure 3.9: Empirical cumulative distributions of the adversarial distortion produced by our L_∞ white-box attack on the ablated detectors. The distance metrics are L_2 (left) and L_∞ (right).



(a) T-shirts vs trousers

(b) Airplanes vs automobiles

Figure 3.10: Selections of adversarial examples which can fool our ablated detectors, generated by the custom L_2 white-box attack.



(a) T-shirts vs trousers

(b) Airplanes vs automobiles

Figure 3.11: Selections of adversarial examples which can fool our ablated detectors, generated by the custom L_∞ white-box attack.

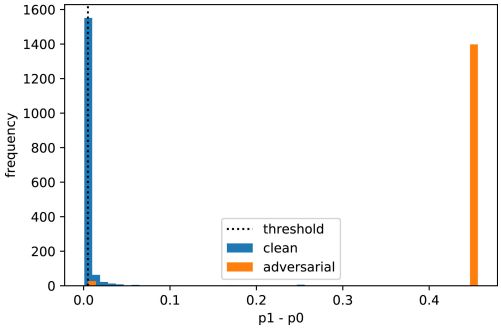


Figure 3.12: Histograms of the differences $p_1 - p_0$ for clean and adversarial data along with the tuned model threshold β. We used the DeepFool attack here to generate adversarials for the zeroes vs ones model.

Data set	Clean accuracy	Robust accuracy	ε	Steps
Zeroes vs ones	99.88%	97.10%	0.30	40
Cats vs dogs	54.94%	3.50%	0.30	10
T-shirts vs trousers	97.12%	85%	0.30	40
Airplanes vs automobiles	82.06%	29.19%	0.30	10

Table 3.4: Summary of parameters and performance indicators for the Madry defense on the machine learning tasks we considered. Step sizes were always set to 0.01 and random start was used each time.

3.3.6 Comparison to other methods

In this section, we compare our IVAP method to the Madry defense [Madry et al., 2017] which, to the best of our knowledge, is the strongest state-of-the-art defense at the time of this writing. Specifically, we apply the Madry defense to each of the CNNs we used in our experiments and compare the accuracies of the different methods. We also perform transfer attacks where we try to fool our IVAP using adversarials generated for the Madry defense and vice versa.

The Madry defense is a form of adversarial training where, at each iteration of the learning algorithm, the L_∞ projected gradient descent attack is used to perturb all samples in the current minibatch. The model is then updated based on this perturbed batch instead of the original. To facilitate fair comparison, we do not use the pre-trained models and weights published by Madry et al. for the MNIST and CIFAR-10 data sets, since these are meant for the full 10-class problems whereas our defense only considers a simpler binary subproblem. Rather, we modified the network architectures for these tasks so they correspond to our own CNNs and trained them according to the procedure outlined by Madry et al. [2017].

Table 3.4 shows the results we obtained with the Madry defense as well as the parameter settings we used for the PGD attack. The defense performs very well on the zeroes vs ones and T-shirts vs trousers tasks. However, on airplanes vs automobiles the adversarial accuracy is low and on the cats vs dogs task both the clean and adversarial accuracies are low. In fact, for the latter task, the adversarial accuracy is almost the same as if no defense was applied at all. The Madry

Data set	L_2 accuracy	L_∞ accuracy
Zeroes vs ones	46.28%	66.78%
Cats vs dogs	55.26%	55.48%
T-shirts vs trousers	94.94%	93.88%
Airplanes vs automobiles	79.94%	78.38%

Table 3.5: Results of the IVAP-to-Madry transfer attack. Here, adversarial examples generated for the IVAP defense by our custom white-box attack are transferred to the Madry defense. We test adversarials generated by both the L_2 and L_∞ variants of our attack.

Data set	Accuracy	TPR	TNR	FPR	FNR
Zeroes vs ones	36.05%	30.91%	100.00%	0.00%	69.09%
Cats vs dogs	61.64%	65.08%	52.3%	47.70%	34.92%
T-shirts vs trousers	72.75%	71.02%	84.47%	15.53%	28.98%
Airplanes vs automobiles	45.81%	34.53%	95.92%	4.08%	65.47%

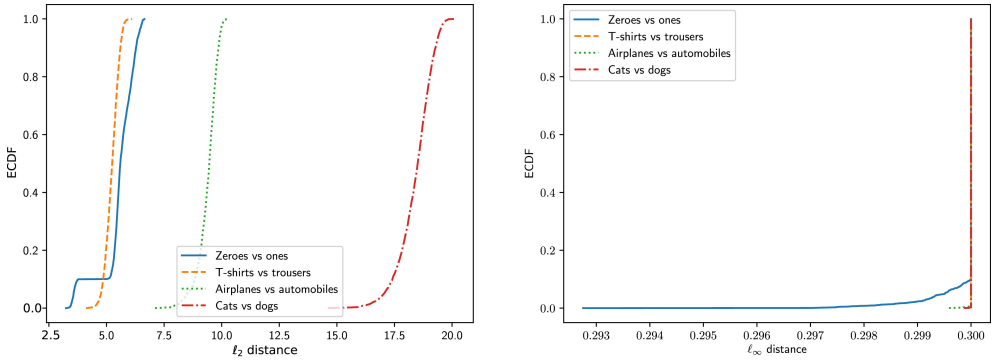
Table 3.6: Results of the Madry-to-IVAP transfer attack, where adversarial examples generated for the Madry defense were tested against our IVAPs.

defense outperforms our method in terms of robust accuracy on two out of four data sets, but it is important to note that these are the simplest data sets in our evaluation: Zeroes vs ones was derived from MNIST and T-shirts vs trousers was derived from Fashion-MNIST. Both of these data sets consist of grayscale images 28x28 pixels in size. Our method has higher robust accuracy on Cats vs dogs and Airplanes vs automobiles, which consist of RGB images 64x64 and 32x32 pixels in size respectively. This suggests that the IVAP defense scales better than Madry at higher dimensionality.

In table 3.5 we transfer the white-box adversarials for our defense to the Madry defense. The Madry defense appears quite robust to our IVAP adversarials for T-shirts vs trousers and airplanes vs automobiles, but much less so on zeroes vs ones. The accuracy on the adversarials for the cats vs dogs task is close to the clean accuracy, so these appear to have little effect on the Madry defense.

Table 3.6 shows what happens when we transfer adversarials for the Madry defense to our IVAP detectors. We observe that for the zeroes vs ones and airplanes vs automobiles, the accuracy drops below 50%. However, false positive rates on these adversarials are very low, so the detectors successfully reject many incorrect predictions. On the other tasks, accuracy remains relatively high. The false positive rate is rather high for the cats vs dogs task, but low for T-shirts vs trousers.

Figure 3.13 plots the empirical distributions of the adversarial distortion levels of the Madry adversarials, both for the L_2 and L_∞ distances. Comparing this figure with Figs. 10 and 11, we see that our custom white-box adversarials for the IVAP defense generally require higher L_2 and L_∞ levels of distortion than the Madry adversarials. Figure 3.14 plots selections of adversarials generated for the Madry defenses by the L_∞ PGD attack. The visual levels of distortion appear comparable to our defense (for the L_∞ variant of our attack), except in the case of the cats vs



(a) ℓ_2 (b) ℓ_∞

Figure 3.13: Empirical cumulative distribution curves for the adversarial distortion of the Madry adversarials.

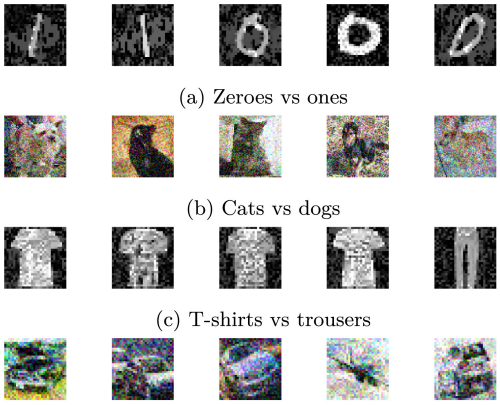
dogs task where the perturbations for the Madry defense seem much larger. It is important to note, however, that the performance of the Madry defense on this task is close to random guessing, whereas the IVAP still obtains over 70% clean accuracy.

We are lead to the conclusion that our defense in fact achieves higher robustness on these tasks than the Madry defense. Moreover, our defense appears to scale much better: adversarially training a deep CNN with L_∞ projected gradient descent quickly becomes computationally intractable as the data increase in dimensionality and the CNN increases in complexity. On the other hand, our IVAP defense can take an existing CNN (trained using efficient, standard methods) and quickly augment it with uncertainty estimates based on a calibration set and a data set of adversarials for the underlying model. The resulting algorithm appears to require higher levels of distortion than the Madry defense in order to be fooled effectively and it seems to achieve higher clean and adversarial accuracy on more realistic tasks. Finally, we note that our method introduces only a single extra scalar parameter, the threshold β, which can be easily manually tuned if desired. It is also possible to tune β based on other objectives than Youden's index which was used here. On the other hand, the Madry defense requires a potentially lengthy adversarial training procedure whenever the defense's parameters are modified.

3.3.7 Conclusions

We have proposed using inductive Venn-ABERS predictors (IVAPs) to protect machine learning models against adversarial manipulation of input data. Our defense uses the width of the uncertainty interval produced by the IVAP as a measure of confidence in the prediction of the underlying model, where the prediction is rejected in case this interval is too wide. The acceptable width is a hyperparameter of the algorithm which can be estimated using a validation set. The resulting algorithm is no longer vulnerable to the original adversarial examples that fooled the unprotected model and attacks which take the detector into account have very limited success. We do not believe this success to be due to gradient masking, as the defense remains robust even when subjected to attacks that do not suffer from this phenomenon.

Of course, the mere fact that our own white-box attack fails against the IVAP defense does



(a) Zeroes vs ones

(b) Cats vs dogs

(c) T-shirts vs trousers

(d) Airplanes vs automobiles

Figure 3.14: Selections of adversarial examples which can fool the Madry defense, generated by the L_∞ PGD attack.

not constitute hard proof that our method is successful. We therefore invite the community to scrutinize our defense and to attempt to develop stronger attacks against it. The performance of the IVAP defense is still not ideal at this stage since clean accuracy is noticeably reduced. However, we believe these findings represent a significant step forward. As such, we suggest that the community further look into methods from the field of conformal prediction in order to achieve adversarial robustness at scale. To our knowledge, we are the first to apply these methods to this problem, although the idea has been mentioned elsewhere already [Vorobeychik and Kantarcioglu, 2018].

3.4 Case study: DGA detection

As a practical application of the IVAP-based defense in a real-world setting, we considered the problem of detecting domain generation algorithms (DGAs) in Grumer et al. [2019]. DGAs are commonly used by malware authors to establish connections with victim computers, to exfiltrate data from these systems or send certain commands to the malware [Plohmann et al., 2016]. Specifically, a DGA is any algorithm which, given a random seed, generates a potentially infinite deterministic list of valid internet domain names that have not previously been registered. Both the malware and its author have a copy of the DGA, and the seed is generally taken to be some piece of data from a publicly available source, *e.g.*, the local time in a certain country, the data of the latest tweet posted by some celebrity, etc. In this way, the attacker can establish a fairly robust communication channel with their malware: the attacker simply registers some of the domains generated by the DGA. The malware finds these domains by iterating over the same list of names (since it knows both the DGA and the seed used by the author) and establishes connections with them to await instructions. If any of the domains are ever seized by law enforcement, the attacker and the malware can simply move down the list until another available domain is found. This ability to dynamically generate new domain names prevents ordinary blacklisting from effectively blocking communication between the botmaster and infected machines. As such, there is a need to develop classification systems which can identify domains generated by DGAs

in real time [Saxe and Berlin, 2017, Schüppen et al., 2018, Tran et al., 2018, Woodbridge et al., 2016]. For such classifiers, a low false positive rate is very important, because a false positive might entail blocking a legitimate domain and disrupting their business.

To give a practical example of a real-world DGA, we consider an early version of the CryptoLocker malware.[5] This ransomware program initially used the following algorithm for generating domain names:

```
def generate_domain(year: int, month: int, day: int) -> str:
    """Generate a domain name for the given date."""
    domain = ""

    for i in range(16):
        year = ((year ^ 8 * year) >> 11) ^ ((year & 0xFFFFFFF0) << 17)
        month = ((month ^ 4 * month) >> 25) ^ 16 * (month & 0xFFFFFFF8)
        day = ((day ^ (day << 13)) >> 19) ^ ((day & 0xFFFFFFFE) << 12)
        domain += chr(((year ^ month ^ day) % 25) + 97)

    return domain + ".com"
```

This DGA generates a 16-character domain name based on the current date. For example, on 25 April 2022, it would generate the domain name `nerrjcctdufmplay.com`; the next day, it would generate `foxiiyctduf mplay.com`. Note that these domain names are fully deterministic and change only once every 24 hours. In later iterations of CryptoLocker, this algorithm was changed and made more sophisticated, since the domain names generated by this particular version are too predictable: for the year 2022, for instance, there are only 16 unique 8-character suffixes used by the algorithm that are easily recognizable and would likely never appear in legitimate domain names:

```
bqidxxcl cajwaquf dufmplay fadifehn
fchttjcl hearwrfb hjedlway jlcywphn
kjxomkmh lsyiodfb ossttqrj ouwfivmh
serklcrj tcpbbypd xlkgifuf xqorwkpd
```

Adding these suffixes to a blocklist would be sufficient to shutdown CryptoLocker for an entire year. This illustrates the subtlety of designing good DGAs: they must be sufficiently complex so that blocklists are not a feasible countermeasure. At the same time, however, DGAs cannot simply produce random strings, as this would allow for detection based purely on information-theoretic analysis of the domain name itself (such as via the Shannon entropy). They must be at least somewhat plausible as domain names, while ensuring that *most* of the generated names have not been registered before.

3.4.1 Data sets

We assess the effectiveness of algorithm 3.3 in DGA detection using three data sets, detailed below.

[5] `https://arstechnica.com/information-technology/2013/10/youre-infected-if-you-want-to-see-your-data-again-pay-us-300-in-bitcoins/`. Accessed 2022-04-25.

Alexa Offers a list of the top one million domains based on their popularity in terms of number of page views and number of unique visitors. It only retains the websites' second level domain names (SLDs), aggregating across any subdomains. For example, according to Alexa, the three highest ranked domain names in terms of popularity on 2022-04-21 are `google.com`, `youtube.com`, and `facebook.com`. This Alexa top 1 million list serves as a relatively reliable source for benign, non-DGA generated domains but is not necessarily an accurate representation of benign web traffic as whole. The Alexa data set is freely available online.[6]

Qname Contains domain names originating from a realtime stream of passive DNS data that consists of roughly 10-12 billion DNS queries per day collected from subscribers including ISPs (Internet Service Providers), schools, and businesses. We retained 1 million domain names that match three criteria: (1) the domain is at least 30 days old, (2) the domain has been resolved at least twice, and (3) queries to the domain have never resulted in an NXDomain response (indicating a non-existent domain). This data set is intended to serve as a source of benign ground truth that is more reflective of real web traffic than Alexa.

Bambenek Offers a daily feed of domains generated by reverse engineering known families of malware. One million different DGA domains were collected over the course of three days to construct a data set of malicious URLs. This data set is available online, but requires a license to use as of July 2019.[7]

The Alexa and Qname data sets serve as *negative* (*i.e.*, benign) examples, whereas Bambenek supplies the *positive* (*i.e.*, malicious) domains. As such, we construct two data sets on which we train and evaluate our models: AlexaBamb, the concatenation of Alexa and Bambenek, and QnameBamb, the concatenation of Qname and Bambenek. In both cases, the positive and negative classes are perfectly balanced, containing 1 million names each. The data sets were split into four parts: 64% training, 4% validation, 16% calibration, and 16% testing.

3.4.2 Models

Algorithm 3.3 is a sort of "wrapper" method, requiring an existing model to calibrate. We utilize three neural network architectures that have been well-reviewed in past DGA classification literature:

LSTM.MI A unidirectional LSTM recurrent neural network architecture original proposed by Tran et al. [2018] that has been shown to be highly effective at DGA detection [Peck et al., 2019b, Sivaguru et al., 2018].

Invincea A neural network architecture created by Saxe and Berlin [2017] that features parallel convolution layers.

MIT A hybrid CNN/RNN neural network architecture introduced by Vosoughi et al. [2016] and shown to be effective for inline DGA detection [Yu et al., 2018]. It features stacked CNN layers followed by a unidirectional LSTM layer.

[6] `https://www.alexa.com/topsites`
[7] `https://osint.bambenekconsulting.com/feeds/`

Data set	Model	TPR	FPR	ACC	FRR	TRR	REJ
QnameBamb	LSTM.MI	0.917	0.001	0.993	–	–	–
	LSTM.MI + IVAP	0.993	0.001	0.996	0.064	0.922	0.100
	Invincea	0.856	0.001	0.991	–	–	–
	Invincea + IVAP	0.986	0.001	0.993	0.126	0.916	0.183
	MIT	0.920	0.001	0.993	–	–	–
	MIT + IVAP	–	–	–	–	–	1.000
AlexaBamb	LSTM.MI	0.964	0.001	0.992	–	–	–
	LSTM.MI + IVAP	0.999	0.001	0.999	0.066	0.950	0.082
	Invincea	0.966	0.001	0.991	–	–	–
	Invincea + IVAP	0.999	0.001	0.999	0.108	0.969	0.123
	MIT	0.909	0.001	0.992	–	–	–
	MIT + IVAP	0.973	0.001	0.988	0.286	0.493	0.290

Table 3.7: Comparisons of model performance with and without IVAP.

3.4.3 Results

The results are shown in table 3.7. For both the QnameBamb and AlexaBamb data sets, we report results for each of the models (LSTM.MI, Invincea, and MIT) with and without the IVAP. The metrics of interest for the baseline models are the true positive rate (TPR), false positive rate (FPR) and accuracy (ACC). To enable a fair comparison with the IVAP, we compute these metrics only for the subset of samples that were not rejected by the IVAP-augmented model. For the IVAP, additional metrics include the false rejection rate (FRR), true rejection rate (TRR) and overall rejection rate (REJ). Here, a rejection is true when the underlying model prediction was indeed wrong and false otherwise. The rejection rate is simply the fraction of samples for which predictions were rejected.

The uncertainty threshold β was tuned on the separate validation set in order to maximize the difference TRR-FRR similarly to Youden's index [Youden, 1950]. Note that this is just one of many possible ways one could tune β. In some applications, one might prefer the lowest possible FPR, for example. Moreover, since β is just a single scalar hyperparameter, it is also possible to specify it manually without resorting to potentially expensive hyperparameter optimization methods on held-out data.

The use of algorithm 3.3 resulted in consistently better predictive scores at the cost of a small rejection rate and the need for additional data sets. Most notably, the TPR always increases across data sets and models after the IVAP process is applied while the false positive rate remains stable. This suggests that the IVAP process can be used to improve the performance of existing DGA classifiers. One exception to this is the MIT model on the QnameBamb data set: here, all predictions were rejected because the probability interval was always larger than the tuned threshold would allow. We speculate that this is due to the QnameBamb data set being more difficult to fit properly, as evidenced by the fact that the classifiers perform consistently better on AlexaBamb than QnameBamb. This means that models trained on QnameBamb might require a larger calibration set to obtain good results with the IVAP.

3.4.4 Conclusions

We applied the IVAP defense to the practical use-case of DGA detection and showed that it can significantly improve the predictive scores of several state-of-the-art classification models. However, we expect that real-time detection of DGA domains based on nothing more than the domain name itself may become exceedingly difficult in the future as DGAs become more sophisticated. We demonstrated this with the CharBot algorithm[8] in Peck et al. [2019b]. The pseudo-code of this algorithm is shown in algorithm 3.4.

Algorithm 3.4: CharBot

Data: a list of SLDs D, a list of TLDs T, a seed s
Result: a DGA domain
1 Initialize the pseudorandom generator with the seed s.
2 Randomly select a SLD $d \in D$.
3 Randomly select two indices i and j so that $1 \leq i, j \leq |d|$.
4 Randomly select two replacement characters c_1 and c_2 from the set of DNS-valid characters.
5 Set $d_i \leftarrow c_1$ and $d_j \leftarrow c_2$.
6 Randomly select a TLD $t \in T$.
7 **return** $d.t$

CharBot is an extremely simple DGA: it randomly selects an *existing* domain name from a list of second-level domains (SLDs) — such as the Alex top 1 million domain names — and introduces two typographical errors. For instance, if CharBot selected `google` as the SLD, it might generate `go0gli` as the DGA SLD. It then concatenates the generated SLD to a random top-level domain (TLD), such as `com`. The end result could then be `go0gli.com`, which does not exist at the time of this writing. Indeed, we found that the vast majority of CharBot domains do not exist, despite the fact that they are generated by randomly replacing two characters from an existing domain. They also easily bypass three popular detectors which were state-of-the-art at the time: FANCI [Schüppen et al., 2018], LSTM.MI [Tran et al., 2018] and the B-RF model from Sivaguru et al. [2018]. When these detectors are re-trained to incorporate CharBot domains explicitly in their training data, they tend to generate far too many false positives on benign data to be useful in practice. Adversarial training therefore does not seem like a viable option in this case.

We suspect that CharBot is so effective for two reasons:

1. Human beings will tend to register domain names that are distinctly different from existing well-known domains, otherwise their domain might not stand out. This means CharBot domains will almost always be unregistered.

2. The distribution of CharBot domains is very close to the natural distribution of legitimate domain names. Hence it is very difficult for any machine learning model to distinguish legitimate domains from CharBot ones without generating too many false positives.

This suggests that effective DGA detectors may have to incorporate additional information beyond just the domain name. We explored this in Sivaguru et al. [2020], where we attempted to

[8] The name CharBot is derived from the first name of one of our co-authors, Charles Grumer, who initially proposed it. It also references the fact that the algorithm is character-based.

Model	Features	AUC	TPR	CharBot
B-RF	DNS	53.23%	16.21%	1.70% ± 0.24%
	Lexical	89.78%	97.44%	3.80% ± 0.00%
	DNS + Lexical	98.19%	99.42%	20.06% ± 0.56%
LSTM.MI	Domain name string	94.47%	98.80%	8.00% ± 0.00%
LSTM.MI + B-RF	Domain name string + DNS	96.51%	99.89%	14.98% ± 0.63%
	Domain name string + DNS + Lexical	99.17%	99.91%	15.76% ± 0.54%

Table 3.8: Performance metrics of DGA classifiers. AUC and TPR metrics are reported at a FPR of 0.1%.

harden the LSTM.MI and B-RF models against CharBot using such side information. In particular, we found significant improvements in detection rates when the following DNS information was incorporated into the detector:

- `rrlength`: The length of the RData field. This field is extracted from the DNS response resource record. The RData contains a list of resolved IP addresses, the time-to-live value of the query and the type of resource record.
- `country`: A potentially multi-valued feature listing the geographic locations to which the IP addresses resolve that are associated with the domain.
- `ttl`: The time-to-live of the DNS query, which is the time interval that the resource record can be cached by the DNS resolver.
- `n_ip`: The number of IP addresses associated with the domain.
- `qtype`: The DNS query type that can be extracted from the question section of the DNS query.
- `rtype`: The resource record type extracted from the RData field.
- `n_asn`: The number of autonomous system numbers (ASNs) the IP addresses of the domain map to.
- `subnet`: A binary feature indicating whether all IP addresses belong to the same subnet.
- `n_countries`: The number of countries the domain maps to.

Table 3.8 reports the results of our experiments for the B-RF and LSTM.MI model as well as a hybrid LSTM.MI + B-RF model, which combines both LSTM.MI and B-RF architectures by extending the B-RF classifier with the confidence score obtained from the LSTM.MI model for the given domain name. Using DNS features in conjunction with standard lexical features extracted from the domain name string yields much higher CharBot detection rates without compromising AUC or TPR at the desired FPR of 0.1%. However, the highest detection rate we could achieve was only about 20%, meaning approximately 80% of CharBot domains still went undetected. Incorporating side information which the adversary cannot modify therefore shows definite promise, but the problem is still far from solved. Moreover, the availability of useful side information strongly depends on the particular task: for DGA detection, we can efficiently extract such data from the DNS system, but other problems (such as image recognition) have

no viable source of side information. This strategy may therefore only work for certain specific problems but is not a general-purpose solution to the adversarial robustness problem.

Chapter 4

The MultIVAP conformal predictor

> You had to draw lines, and that choice was in itself dangerous; all boundaries had a double edge, were like swords that could always be turned against you in the end. But you still had to choose.
>
> *Trouble and Her Friends*
> Melissa Scott

In chapter 3, we introduced the theory of conformal prediction and the IVAP algorithm by Vovk et al. [2015]. Using the IVAP, we created a simple adversarial defense suitable for *binary* classification problems. We demonstrated the effectiveness of this method on the problem of detecting domain generation algorithms (DGAs). These ideas were published in Peck et al. [2020] and Grumer et al. [2019]. The obvious next step is to extend the method to the multi-class setting, so that we can handle an arbitrary number of classes instead of only two. In this chapter, we develop the MultIVAP algorithm, our multi-class extension of the IVAP defense which preserves the favorable statistical guarantees of the conformal predictor at minimal computational cost. These ideas have been published in Peck et al. [2019a].

There are two general ways of transforming a given binary classification algorithm to multi-class: *one-vs-one* (OvO) and *one-vs-all* (OvA), also known as *one-vs-rest* [Murphy, 2012].

One-vs-all The OvA strategy trains k classifiers, one for each of the k classes. Each classifier f_i ($i = 1, \ldots, k$) is a $\mathcal{X} \to [0,1]$ function that is designed to estimate the probability that the input sample belongs to class i (as opposed to *any* other class). The final classification decision is achieved by choosing the highest output probability across the k models:

$$\hat{y}(x) = \underset{i=1,\ldots,k}{\operatorname{argmax}} \; f_i(x).$$

One-vs-one In this setup, one classifier is trained for *each pair* of distinct classes. That is, given a pair of classes i and j with $i \neq j$, we train a classifier f_{ij} that is designed to classify samples into either class i or class j. The predictions of the different models are subsequently ensembled via a simple voting scheme:

$$s_i(x) = \sum_{j \neq i} f_{ij}(x).$$

Here, $s_i(x)$ is the score assigned to class i given the input sample x. It is merely the sum of all scores assigned by the paired classifiers $\{f_{ij} \mid j \neq i\}$ to class i. The final classification decision is then reached by choosing the label associated with the highest score:

$$\hat{y}(x) = \underset{i=1,\ldots,k}{\operatorname{argmax}}\ s_i(x).$$

All else being equal, it is clear that the OvO strategy has a considerably higher computational complexity than the OvA scheme: OvO requires training $k(k-1)/2$ classifiers, whereas OvA requires only k. Inference using the OvO ensembling method is also slightly more complicated than the OvA case. Moreover, the OvO strategy reduces the amount of training data available for the individual classifiers, since f_{ij} can be trained only on samples from classes i and j. On the other hand, for the OvA case, the training data will generally be severely imbalanced. Although OvA may be computationally more efficient, one must take care to resolve its inherent class imbalance, either through over- or undersampling techniques or by using learning algorithms that are not very sensitive to this problem.

At the time of writing, to the best of our knowledge, only Manokhin [2017] had considered the problem of extending IVAPs to multi-class problems. They implemented the OvO scheme where a separate IVAP is calibrated for each pair of distinct classes i and j. The objective is to estimate the pairwise class probabilities

$$r_{ij}(x) \approx \Pr[Y = i \mid Y \in \{i, j\}, X = x].$$

Let p_0^{ij}, p_1^{ij} denote the two output probabilities of the IVAP calibrated for classes i and j. Manokhin uses the minimax method introduced by Vovk et al. [2015] to estimate the class probabilities:[1]

$$r_{ij}(x) = \frac{p_1^{ij}(x)}{1 - p_0^{ij}(x) + p_1^{ij}(x)}.$$

These estimates are subsequently merged into individual probabilities $p_1, \ldots, p_k$ using the voting method proposed by Price et al. [1994]:

$$p_i(x) = \left(\sum_{j \neq i} \frac{1}{r_{ij}(x)} - (k-2) \right)^{-1}. \tag{4.1}$$

Manokhin [2017] experiment with classification algorithms where the class probabilities p_i are calibrated according to (4.1), using IVAPs in a one-vs-one setup to estimate r_{ij}. They find that this method provides more well-calibrated estimates than existing popular alternatives such as Platt's scaling. Since IVAPs are used as the basic components, the method proposed by

[1] We also made use of this method in our binary IVAP defense.

Manokhin inherits the advantage of being compatible with essentially any existing scoring classifier (such as a DNN).

The algorithm we propose here is similar in spirit to Manokhin: we also wish to extend the IVAP to the multi-class case while maintaining the advantages of computational and predictive efficiency as well as compatibility with existing well-performing models. In the interest of efficiency, however, we opt instead for the one-vs-rest strategy. That is, we calibrate a single IVAP for each of the k classes. For any given input x, this yields k pairs of probabilities $p_0^i(x)$ and $p_1^i(x)$ for $i = 1, \ldots, k$. We wish to aggregate these probabilities into concrete label predictions in such a way that perfect calibration as defined by (3.4) is achieved, since the calibration hypothesis leads us to believe that this would constitute a potentially effective defense against adversarial examples. However, we know from the work of Vovk et al. [2005] that this cannot be done unless the resulting classifier is *multi-probabilistic*, *i.e.*, it must output a *set* of labels instead of a single label. In particular, we cannot adopt the algorithm proposed by Manokhin [2017], since this method — when applied in a "plug-and-play" fashion to existing DNNs as a replacement for the softmax layer — would not be multi-probabilistic and hence cannot be perfectly calibrated. It is therefore necessary for us to develop our own way of aggregating the IVAP outputs into sets of predictions $V \subseteq \mathcal{Y}$ in a way which preserves the calibration properties of the original algorithm 3.2.

The contents of this chapter are based on Peck et al. [2019a].

4.1 Formal construction

Let L denote the random variable corresponding to the set of labels associated with an input sample X. The outputs produced by the IVAP bound the probability that a given label should be included in this set:

$$p_0^i(x) \leq \Pr[i \in L \mid X = x] \leq p_1^i(x). \tag{4.2}$$

Our objective is to find the largest set of candidate labels V such that the probability that each $i \in V$ also belongs to L is maximized. In other words, we want to find a set $V \subseteq \mathcal{Y}$ which maximizes $\Pr[V \subseteq L \mid X]$. To achieve this, we will prove both a lower and upper bound on this probability based on (4.2). First, we note that

$$\begin{aligned}
\Pr[V \subseteq L \mid X = x] &= \Pr\left[\bigwedge_{i \in V} (i \in L) \,\middle|\, X = x\right] \\
&\geq \sum_{i \in V} \Pr[i \in L \mid X = x] - (|V| - 1) \\
&\geq \sum_{i \in V} p_0^i(x) - (|V| - 1).
\end{aligned}$$

The first inequality follows from the intersection bound and the second inequality follows directly from (4.2). This establishes a lower bound on $\Pr[V \subseteq L \mid X]$. For the upper bound, we use the elementary property that $\Pr[A \wedge B] \leq \min\{\Pr[A], \Pr[B]\}$ for any events A and B. Hence,

$$\Pr[V \subseteq L \mid X = x] \leq \min_{i \in V} \Pr[i \in L \mid X = x] \leq \min_{i \in V} p_1^i(x).$$

To summarize, we have the inequalities

$$\sum_{i \in V} p_0^i(x) - (|V| - 1) \leq \Pr[V \subseteq L \mid X = x] \leq \min_{i \in V} p_1^i(x). \tag{4.3}$$

Let $\varepsilon \in [0,1]$ be the confidence parameter. In order for $\Pr[V \subseteq L \mid X = x] \geq 1 - \varepsilon$ to hold, by (4.3) it is sufficient to have

$$\sum_{i \in V} p_0^i(x) - (|V| - 1) \geq 1 - \varepsilon.$$

Define indicator variables

$$\alpha_i = \begin{cases} 1 & \text{if } i \in V, \\ 0 & \text{otherwise.} \end{cases}$$

Then the above is equivalent to

$$\sum_{i=1}^{k} \alpha_i \left(p_0^i(x) - 1 \right) \geq -\varepsilon. \tag{4.4}$$

Maintaining the constraint (4.4) guarantees an error rate of at most $1 - \varepsilon$. However, there may be several different subsets of labels which achieve the same error rate guarantee but which may not all be equally likely. To pick the most likely subset of labels, we also maximize the upper bound on $\Pr[V \subseteq L \mid X = x]$. Using the indicator variables α_i, this yields

$$\Pr[V \subseteq L \mid X = x] \leq \min_{i \in \mathcal{Y}} 1 + \alpha_i \left(p_1^i(x) - 1 \right). \tag{4.5}$$

Our objective is to find the largest subset $V \subseteq \mathcal{Y}$ which satisfies the constraint (4.4) and which maximizes the upper bound (4.5). We can formulate this as a mixed integer linear program (MILP) [Boyd and Vandenberghe, 2004]:

$$\begin{aligned} \underset{\alpha_1,\ldots,\alpha_k,q}{\text{maximize}} \quad & \alpha_1 + \cdots + \alpha_k + q \\ \text{subject to} \quad & \alpha_1, \ldots, \alpha_k \in \{0,1\}, \\ & q \in [0,1], \\ & q + \alpha_i \left(1 - p_1^i(x) \right) \leq 1, \qquad i = 1, \ldots, k, \\ & \sum_{i=1}^{k} \alpha_i \left(p_0^i(x) - 1 \right) \geq -\varepsilon. \end{aligned} \tag{4.6}$$

The solution of (4.6) is an assignment to the set of binary variables $\alpha_1, \ldots, \alpha_k$ where each α_i indicates whether label i is included in V or not. The variable q is a dummy designed to maximize the upper bound (4.5). Specifically, we have

$$q + \alpha_i \left(1 - p_1^i(x) \right) \leq 1 \iff q \leq 1 + \alpha_i \left(p_1^i(x) - 1 \right).$$

By maximizing q and applying the above constraint for all $i = 1, \ldots, k$, we effectively maximize the upper bound (4.5). The pseudo-code of the full MultIVAP algorithm is summarized in algorithm 4.1.

Note that (4.6) always has at least one feasible solution: $\alpha_1 = \cdots = \alpha_k = 0$ (which corresponds to choosing $V = \emptyset$) and $q = 1$. This solution trivially satisfies all guarantees, since $\Pr[\emptyset \subseteq L \mid X = x] = 1$. This is one reason why we must choose the *largest* possible set of labels, since the optimal choice for the smallest set would always be empty. Nevertheless, we do not see this as a weakness of the method; rather, the guaranteed existence of the trivial solution $V = \emptyset$ implies the possibility of *abstaining* from prediction. Specifically, if no solution exists other than the

empty set, we can interpret this as *abstention*, *i.e.*, all possible predictions are rejected. This indicates that the model is too unreliable on the given input to output any labels at all, at least at the ε confidence level. If it is absolutely necessary to obtain *some* output, the confidence parameter has to be lowered. This may lead to insufficient confidence for certain applications, however, at which point one either has to switch to more sophisticated (and computationally expensive) models or involve a human expert.

The ability of a model to abstain from predictions may seem strange to many ML engineers and practitioners, but it is in fact quite common in the CP literature. Indeed, there is the notion of *credibility* of a prediction, which seems to be entirely unique to the field of CP. It is defined as the largest ε for which the prediction region is empty [Saunders et al., 1999, Vovk et al., 2005]. This quantity is usually reported in order to prevent overconfidence when the input is unusual [Shafer and Vovk, 2008]. Even within the field of adversarial ML, especially in the context of detector defenses, having the ability to abstain (*e.g.*, when the model detects an adversarial or otherwise suspicious input) is becoming increasingly common [Carlini and Wagner, 2017a, Grosse et al., 2017]. This is simply a matter of necessity, as for detector defenses there is no guarantee that the output of the model on an adversarial example will make any sense at all, and therefore one might even be better off rolling a die rather than relying on the classifier. In such cases, the only reasonable action might be to alert a human expert to review the situation and halt processing of the suspicious input until further notice. This is practical in many settings: for instance, in content filtering systems, a user who uploads a suspicious file might be notified that their submission needs further review and will not be made publicly available on the server until it is cleared manually by a moderator.

Algorithm 4.1: MultIVAP

Data: bag of examples $\lceil z_1, \ldots, z_m \rfloor \subseteq \mathcal{X} \times \mathcal{Y}$, object $x \in \mathcal{X}$, learning algorithm A and confidence level ε

Result: subset of labels $V \subseteq \mathcal{Y}$

Divide the bag of examples into a proper training set $\{z_1, \ldots, z_n\}$ and a calibration set $\{z_{n+1}, \ldots, z_m\}$.

Run the learning algorithm A on the proper training set to obtain a scoring classifier F.

for *i from 1 to k* **do**

  Let F_i be the score assigned by F to class i.

  Create a new calibration set $S_i = \{(x_{n+1}, y'_{n+1}), \ldots, (x_m, y'_m)\}$ where

$$y'_j = \begin{cases} 1 & \text{if } y_j = i, \\ 0 & \text{otherwise.} \end{cases}$$

  Use an IVAP to obtain probabilities p_0^i, p_1^i for x using S_i and F_i for calibration.

end

Solve (4.6) to obtain an optimal solution $V \subseteq \mathcal{Y}$.

4.2 Computational complexity

We now consider the computational complexity of algorithm 4.1. Note that the procedure can be split into two distinct stages:

1. *Calibration*. When the MultIVAP is first initialized, it is instantiated with a learning algorithm and a training data set. During calibration, it runs the learning algorithm and then runs the IVAP algorithm k times, once for each class. This step needs to be performed only once.
2. *Inference*. Once the MultIVAP is fully initialized, new samples can be processed using the precomputed isotonic regressions to obtain the probabilistic bounds. Then, the final prediction is computed by solving a MILP.

The overhead incurred in the calibration phase is $O(kc \log c)$, where c is the size of the calibration set. This follows from the fact that we calibrate k IVAPs on data sets of size c and the complexity of calibrating an IVAP is dominated by the sorting step in the isotonic regressions [Vovk et al., 2015]. The inference phase is harder to analyze, since it requires the solution of a MILP. MILPs are NP-hard in general [Arora and Barak, 2009], but our timing experiments performed in the next section indicate that the overhead is roughly linear in the number of classes.

4.3 Choosing the calibration set

Another important aspect of the MultIVAP (and the IVAP as well) is the question of how to choose the calibration set. Formally, the only requirement is that this set must be an independent sample from the data distribution similar to the test or validation set [Vovk et al., 2015]. However, it would be interesting to have finite-sample guarantees on the performance of the predictor depending on the size c of the calibration set. To the best of our knowledge, such guarantees do not exist yet. Vovk et al. give the only real discussion we have been able to find in the literature on this issue. They state that the lower and upper bounds $p_0^i(x)$, $p_1^i(x)$ used by the MultIVAP will lie very close together for large data sets, but this is only an asymptotic statement that says little about the actual rate of convergence when the data set is finite.

In our own work [Grumer et al., 2019], we have found that the IVAP can be sensitive to the particular choice of calibration set size to the point where the predictions can sometimes become unusable. Unsurprisingly, this appears to depend on the specifics of the underlying model; that is, some models are harder for the IVAP to calibrate than others. In the absence of theoretical finite-sample guarantees, the size of the calibration set may be treated as another hyperparameter which can be tuned on a separate validation set. The optimal size can then be chosen depending on how well the accuracy of the resulting predictor is balanced against other considerations such as computational overhead.

4.4 Example for the binary case

To gain a better understanding of how the MultIVAP works, it is instructive to study the special case of binary classification. While we originally proposed algorithm 3.3 for this purpose, it is important to note that the MultIVAP does not specialize to this algorithm in the binary case. To see how the MultIVAP works for $k = 2$, we first assume for the sake of simplicity that only

a single IVAP is used in this case, and that the labels are mutually exclusive (as they are in the single-label setting commonly assumed for supervised classification). The IVAP yields the bounds

$$p_0^j(x) \leq \Pr[j \in L] \leq p_1^j(x)$$

for any label $j \in \{1,2\}$ and data sample x. In particular, assuming mutually exclusive labels, we have $\Pr[1 \in L] = 1 - \Pr[2 \in L]$ and so the above yields the pair of bounds

$$\begin{aligned} p_0^1(x) \leq \Pr[1 \in L] &\leq p_1^1(x), \\ p_0^2(x) = 1 - p_1^1(x) \leq \Pr[2 \in L] &\leq 1 - p_0^1(x) = p_1^2(x). \end{aligned}$$

The optimization problem (4.6) then reduces to

$$\begin{aligned} \underset{\alpha_1,\alpha_2,q}{\text{maximize}} \quad & \alpha_1 + \alpha_2 + q \\ \text{subject to} \quad & \alpha_1, \alpha_2 \in \{0,1\}, \\ & q \in [0,1], \\ & q + \alpha_1 p_0^2(x) \leq 1, \\ & q + \alpha_2 p_0^1(x) \leq 1, \\ & \alpha_1 \left(1 - p_0^1(x)\right) + \alpha_2 \left(1 - p_0^2(x)\right) \leq \varepsilon. \end{aligned}$$

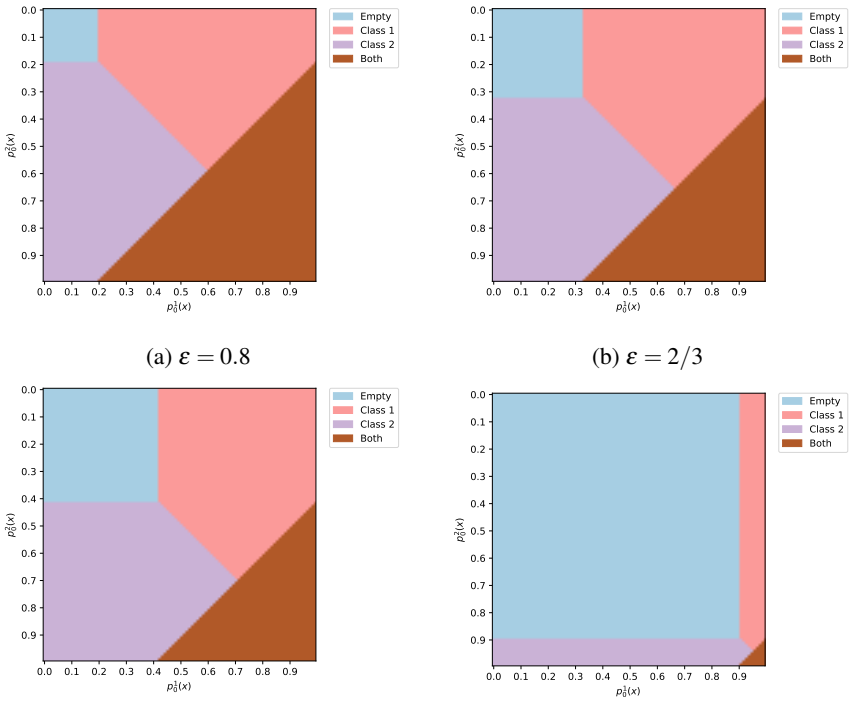
The only probabilities featured here are $p_0^1(x)$ and $p_0^2(x)$, the predicted lower bounds of both classes. This allows us to visualize the decision boundaries of the MultIVAP as a 2D image. Figure 4.1 gives a few examples of these boundaries for different values of ε, as a function of $p_0^1(x)$ and $p_0^2(x)$. We can see that the area corresponding to an empty prediction region (and hence rejection) forms a square the sides of which measure precisely $1-\varepsilon$. In the lower-right corner, there is an equilateral triangle where the MultIVAP outputs both labels (a useless prediction). The sides of this triangle measure precisely ε. The remaining area is divided equally into two convex sets, each corresponding to a single label. Each of these sets consists of a rectangle measuring $\varepsilon \times (1-\varepsilon)$ concatenated with a triangle with base ε and height $\varepsilon/2$. Assuming a uniform distribution of the probability lower bounds within the $[0,1]$ interval, the probability masses of the four possible prediction regions are then given by

$$\begin{aligned} \text{Empty} &= (1-\varepsilon)^2, & \text{Both} &= \frac{\varepsilon^2}{2}, \\ \text{Class 1} &= \varepsilon(1-\varepsilon) + \frac{\varepsilon^2}{4}, & \text{Class 2} &= \varepsilon(1-\varepsilon) + \frac{\varepsilon^2}{4}. \end{aligned}$$

The masses of Class 1 and Class 2 are maximized for $\varepsilon = 2/3$, in which case we have

$$\begin{aligned} \text{Empty} &= \frac{1}{9}, & \text{Both} &= \frac{2}{9}, \\ \text{Class 1} &= \frac{1}{3}, & \text{Class 2} &= \frac{1}{3}. \end{aligned}$$

Note that $\varepsilon = 2/3$ also minimizes the sum of the probability masses of Empty and Both. Interestingly, in this case, the MultIVAP has a higher probability of returning both labels than it has of rejecting the prediction.



(a) $\varepsilon = 0.8$

(b) $\varepsilon = 2/3$

(c) $\varepsilon = \frac{1}{1+1/\sqrt{2}}$

(d) $\varepsilon = 0.1$

Figure 4.1: Decision boundaries of the MultIVAP in the binary case as a function of the probability lower bounds $p_0^1(x)$ and $p_0^2(x)$, for different values of ε. Two values of note are $\varepsilon = 2/3$, in which case the probability masses associated with Class 1 and Class 2 are maximized (or, equivalently, the masses of Empty and Both are minimized); and $\varepsilon = \frac{1}{1+1/\sqrt{2}} \approx 59\%$, where the Empty and Both regions have equal mass of $3 - 2\sqrt{2} \approx 17\%$. In that case, Class 1 and Class 2 have mass $2\sqrt{2} - 5/2 \approx 33\%$.

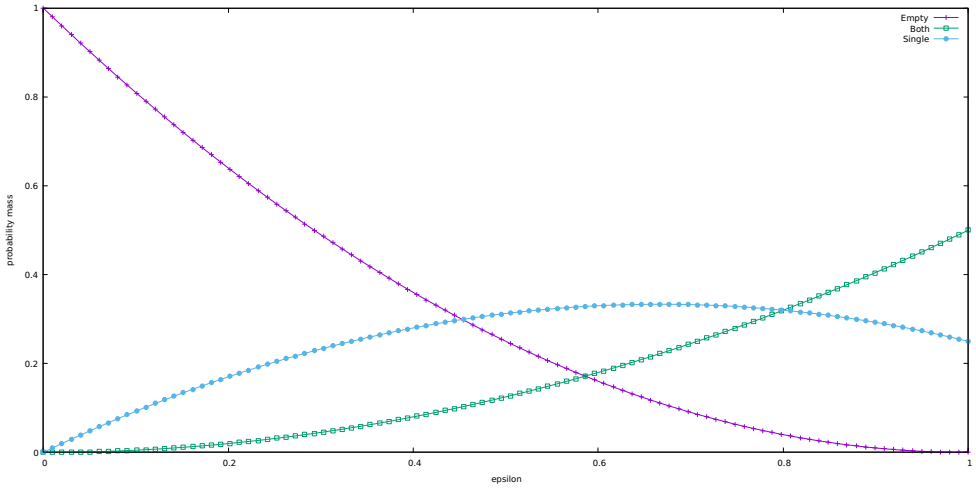


Figure 4.2: Plots of the probability masses as a function of ε for the MultIVAP outputs in a simplified binary classification setting.

We plot the probability masses as a function of ε in figure 4.2. At the low end of $\varepsilon = 0$, naturally we have 100% rejection and the MultIVAP never outputs any labels. At the other extreme, $\varepsilon = 1$, we have 0% rejection, 50% probability of outputting both labels and each individual label has a probability of 25%. Extrapolating from these findings in a simplified binary classification setting, we can conclude that the MultIVAP is a rather conservative algorithm: the probability of rejection scales quadratically with ε, the probability of any individual class never exceeds 1/3 and at low rejection rates there is a comparatively high chance of a useless prediction (*i.e.*, outputting both classes). Of course, these observations must be appropriately nuanced, mainly because we have assumed a uniform prior for the lower bounds $p_0^1(x)$ and $p_0^2(x)$, which is likely not realistic in most practical settings. In a sense, this is the "worst-case" scenario for the MultIVAP, as the lower bounds can be very uninformative.

4.5 Adaptive attack

To enable a fair and thorough evaluation of the MultIVAP, we also design a custom defense-aware white-box attack specifically for fooling this defense.[2] Given that the MultIVAP is a multi-probabilistic predictor returning a set of labels rather than a single prediction, we must slightly modify the typical goal of causing a misclassification since that is ill-defined here. On the one hand, we do not want the correct label to be present in the resulting label set; on the other hand, we do not want the MultIVAP to output too many or too few labels either as this can also raise suspicion. We settle on the following optimization problem:[3]

$$\min_{\tilde{x} \in \mathcal{X}} \|x - \tilde{x}\|_\infty + \lambda \|F(\tilde{x}) - s\|_\infty. \tag{4.7}$$

[2] See Carlini et al. [2019] for an overview of the various desiderata that an adversarial defense evaluation should satisfy.

[3] We use the ℓ_∞ norm everywhere as this is recommended by Madry et al. [2017]. However, the attack can be trivially adapted to any other norm.

Here, $F : \mathcal{X} \to \mathbb{R}^k$ is the scoring classifier, $\lambda \in \mathbb{R}$ is a parameter and $s \in \mathbb{R}^k$ is a target vector of scores. The scalar λ is optimized via binary search so that the magnitude of the perturbation $\|x - \tilde{x}\|_\infty$ is as small as possible. The score vector s is determined by searching the calibration set of the MultIVAP for all calibration samples (x', y') such that y' does not belong to the prediction region of x. Among these candidates, we randomly sample one element x' and take its calibration score vector as the target $s = F(x')$.

An adversarial example produced by (4.7) is accepted only if the following conditions are met:

1. The perturbation stays within the budget, that is, $\|x - \tilde{x}\|_\infty \leq \eta$ where η is a user-specified perturbation bound.
2. The MultIVAP does not return an empty prediction region for the adversarial sample.
3. The true label is not present in the prediction region returned by the MultIVAP.

Solutions to (4.7) satisfying these properties are counted as "successes", and the *success rate* of the attack is the fraction of samples that were counted as such.

The intuition behind our defense-aware attack is the following. The k IVAPs used internally by the MultIVAP will output the same upper and lower bounds for all samples that are assigned the same scores by the classifier F. Therefore, if we can corrupt a sample x into a sample $\tilde{x}$ which shares the score vector of another sample x', then $\tilde{x}$ will be associated with the same probabilities as x' and the optimization problem (4.6) will have the same solution sets for both. This will cause the MultIVAP to output the same prediction regions for $\tilde{x}$ and x'. If, furthermore, the distance between x and $\tilde{x}$ is small and if the true label of x is not present in the prediction region of x', then $\tilde{x}$ can be considered an adversarial example. This is what we aim to achieve in (4.7). In finding an appropriate sample x' to target, we try to minimize $\|F(x) - F(x')\|_\infty$ in order to "warm-start" the attack.

Note additionally that the objective (4.7) can be optimized using gradient descent without issue, as the scoring classifier F is a standard (fully differentiable) neural network. In particular, our defense-aware attack will not suffer from any so-called *gradient masking* [Athalye et al., 2018a]. This is a common problem affecting gradient-based adversarial attacks where a proposed defense method has "masked" or somehow corrupted the gradient signal of the model it is supposed to defend. As shown by Athalye et al. [2018a], this does not actually make models more robust. It can, however, create the illusion of robustness if one only evaluates against gradient-based attacks. Importantly, standard neural networks trained using typical methods generally do not exhibit gradient masking, making it not a concern for our attack.

4.6 Experiments

We perform experiments the following five data sets:

MNIST The MNIST data set was introduced by LeCun et al. [1998] and consists of 60,000 black-and-white handwritten digits. The images are 28x28 pixels in size and are divided into ten classes (digits 0 through 9).

Fashion-MNIST The Fashion-MNIST data set was proposed by Xiao et al. [2017] as a drop-in replacement for the MNIST data set. It has the same basic properties (black-and-white 28x28

pixel images divided into ten classes), but instead of handwritten digits it contains images of clothing items from the Zalando fashion catalog.

CIFAR-10 The CIFAR-10 data set contains 60,000 RGB images 32x32 pixels in size divided into ten classes. The classes correspond to certain everyday items, such as automobiles, dogs and cats. It was proposed by Krizhevsky and Hinton [2009] and originally derived from the (now retracted) 80 Million Tiny Images data set.

Asirra The Assira data set was introduced by Elson et al. [2007] and contains RGB images of cats and dogs of various sizes. Accordingly, it is divided into two classes. It was originally intended to replace classical CAPTCHA systems based on distorted text with the more difficult problem of distinguishing cats from dogs.[4]

SVHN The Street View House Numbers (SVHN) data set contains RGB images 32x32 pixels in size divided into ten classes (again, for the digits 0 through 9). The digits are not handwritten like MNIST; instead, they come from house numbers.

For each data set, we normalize the pixel values to the interval $[0,1]$. For the Asirra data set, we also resized all images to 64×64 pixels to facilitate processing by our pipeline as the images come in various irregular sizes. We used 64% of the data for training the neural networks, 16% for testing, 4% for validation and 16% for calibrating the MultIVAP. These splits were constructed by randomly shuffling the entire data set and subdividing the samples accordingly.

We train convolutional neural networks and apply algorithm 4.1 to obtain a calibrated multi-probabilistic predictor. We used TensorFlow Keras [Abadi et al., 2015, Chollet et al., 2015] for training the neural networks. Each network was optimized using the Adam optimization algorithm [Kingma and Ba, 2014] and trained for 10 epochs, except for Asirra which was trained for 50 epochs. For the IVAPs, we made use of an implementation by Toccaceli [2017].

Proper evaluation of the performance of conformal predictors such as the IVAP poses its own unique challenges and differs significantly from the evaluation of typical supervised classification algorithms. In general, conformal predictors output *set-valued* predictions, *i.e.*, their output is always a set containing zero or more labels. To assess the performance of the MultIVAP, we report the following metrics:

- The accuracy (ACC) of the original model. We give both the accuracy of the original model on the full test set as well as the accuracy of the model on the subset of predictions accepted by the MultIVAP. This is referred to as the "corrected accuracy" (COR).
- The average *Jaccard index* (JAC) of the MultIVAP on the test set [Jaccard, 1912]. This index, also known as the *intersection over union* (IoU), is defined as

$$J(x) = \frac{|L(x) \cap V(x)|}{|L(x) \cup V(x)|}$$

  where $L(x)$ is the true label set for the sample x and $V(x)$ is the MultIVAP prediction region. In the special case where $L(x)$ is a singleton (single-label prediction), this formula

[4] Of course, with the advent of deep learning, in 2022 one could no longer rely on such a system to create reliable CAPTCHAs.

reduces to

$$J(x) = \begin{cases} 1/|V(x)| & \text{if } L(x) \subseteq V(x), \\ 0 & \text{otherwise.} \end{cases}$$

That is, it is inversely proportional to the size of the prediction region $V(x)$ if this region contains the correct label. If it does not, then a score of zero is assigned. In particular, rejected predictions where $V(x) = \emptyset$ are treated as if they are erroneous. The average Jaccard index over the test set is then given by

$$\overline{J} = \frac{1}{m} \sum_{i:L(x_i) \subseteq V(x_i)} \frac{1}{|V|}.$$

If the MultIVAP always returns exactly one label, then the average Jaccard index coincides with the typical notion of test accuracy. In general, however, the MultIVAP can return multiple labels or even no labels at all in case of rejection. Therefore, $\overline{J}$ will always be bounded by the accuracy and it is expected that it will often be smaller.

- The *predictive efficiency* (EFF), which is the average number of labels the MultIVAP outputs across all samples. Ideally, this number should be very close to one.
- The significance level ε at which the results for the MultIVAP were obtained. This level was determined by tuning the threshold on a held-out validation set such that the predictive efficiency was as close to one as possible.
- The true rejection rate (TRR) along with the false rejection rate (FRR) and the overall rejection rate (REJ). Formally, these are computed as follows:

$$\text{TRR} = \frac{\text{TR}}{\text{TR} + \text{FA}}, \qquad \text{FRR} = \frac{\text{FR}}{\text{FR} + \text{TA}}, \qquad \text{REJ} = \frac{\text{TR} + \text{FR}}{\text{TR} + \text{FR} + \text{TA} + \text{FA}}.$$

The quantities appearing in these equations are the number of true rejections (TR), false acceptances (FA), false rejections (FR) and true acceptances (TA). Samples are rejected if the MultIVAP returns an empty prediction region. A rejection is true if the underlying scoring classifier F would have returned an erroneous prediction and false otherwise. Table 4.1 summarizes these quantities in a more visual manner.

It is of particular importance for us to study the Jaccard index, efficiency, TRR and FRR of the MultIVAP together and not simply the accuracy score. This is because the MultIVAP is a *multi-label* predictor, meaning it yields a set of possible labels for each sample instead of a single point prediction like most models do. When considering the performance of such a method compared to a standard classifier that outputs an argmax over a set of estimated probabilities, we must ask ourselves the following questions:

1. How well do the label sets predicted by the MultIVAP correspond to the true label sets? This is measured by the Jaccard index, which is one possible generalization of the accuracy score specifically for multi-label predictors.
2. How many labels does the MultIVAP output on average? This is the predictive efficiency. We do not want this quantity to be much higher or lower than the ground truth (which is usually one), otherwise the MultIVAP can create confusion.

3. When the MultIVAP rejects a prediction, what is the probability that this rejection was "justified", in the sense that the underlying model used by the MultIVAP was wrong? This is quantified by the true rejection rate and the false rejection rate.

Naturally, we want the Jaccard index at 100%, the predictive efficiency close to one, the TRR close to 100% and the FRR close to 0%. We can exercise some amount of control over these metrics by varying the significance level ε. In our experiments, we used a held-out validation set to tune ε in order to have a predictive efficiency as close to one as possible. Depending on the particular application, however, one can use many other methods to tune ε. For instance, in cybersecurity settings, it is considered highly undesirable to unjustly flag benign samples as malicious because this interferes with legitimate users' business. In such cases, we might be more concerned with minimizing the FRR than we are with maximizing predictive efficiency (or we may want to specify some sort of trade-off between these two metrics) and so we might tune ε so that the FRR does not exceed some specified threshold instead.

	Accept	Reject
Correct	True accept (TA)	False reject (FR)
Incorrect	False accept (FA)	True reject (TR)

Table 4.1: Confusion matrix for the detector.

To assess the robustness of the MultIVAP against adversarial perturbations, we consider two threat models:

Defense-oblivious. Here, we evaluate the defense against non-adaptive transfer attacks. That is, the attacks have no knowledge of the defense; they are crafted against the baseline model and then simply presented to the MultIVAP. This is the bare minimum of robustness that any defense should hope to achieve [Carlini et al., 2019]. The attacks we tested against here are DeepFool [Moosavi-Dezfooli et al., 2016], projected gradient descent (PGD; [Madry et al., 2017]), fast gradient sign method (FGSM; [Goodfellow et al., 2014]), single pixel [Su et al., 2019] and local search [Narodytska and Kasiviswanathan, 2016]. All implementations were provided by the Foolbox library [Rauber et al., 2020]. We chose this particular set of attacks because of their diversity: there are gradient-based attacks (DeepFool, PGD and FGSM), non-gradient-based attacks (Single pixel and local search), iterative (DeepFool, PGD, single pixel, local search) and single step attacks (FGSM).

Defense-aware. In this setting, we use our own adaptive attack to specifically bypass the MultIVAP. This evaluation should be a worst-case scenario for our defense and provides an estimation of a lower bound on its actual robustness.

Finally, we address the question of computational efficiency of our method. It is difficult to precisely quantify the complexity of the MultIVAP as it requires the solution of a MILP, which is NP-complete in general [Boyd and Vandenberghe, 2004]. As such, we perform timing experiments where we measure the average time per prediction for both the baseline model and the MultIVAP on the test sets of each task.

Results on clean data are reported in table 4.2. We can see from the corrected accuracy that the MultIVAP can increase the accuracy of the original model at the cost of rejecting a certain per-

	Baseline	MultIVAP					
Task	ACC (COR)	JAC	$1-\varepsilon$	EFF	TRR	FRR	REJ
MNIST	99.10% (99.59%)	94.16%	33.75%	0.99	56.94%	4.54%	2.90%
FMNIST	91.81% (93.84%)	83.44%	24.20%	1.04	29.62%	4.45%	6.75%
CIFAR-10	76.40% (81.52%)	61.86%	20.77%	1.05	32.73%	8.36%	15.54%
Asirra	88.82% (89.04%)	83.65%	41.86%	1.00	2.69%	0.48%	1.10%
SVHN	92.22% (96.40%)	80.77%	25.23%	1.01	59.22%	7.70%	10.42%

Table 4.2: Results of the baseline models and the MultIVAPs on the different data sets.

		Baseline	MultIVAP				
Task	Attack	ACC (COR)	JAC	EFF	TRR	FRR	REJ
MNIST ($\eta = 0.3$)	DeepFool	35.86% (35.83%)	27.18%	0.52	59.23%	60.86%	59.81%
	PGD	0.38% (0.33%)	37.23%	0.64	52.32%	46.67%	52.30%
	FGSM	4.39% (4.57%)	41.13%	1.10	39.64%	2.85%	38.03%
	Single pixel	98.01% (99.01%)	93.96%	0.99	62.89%	4.04%	5.21%
	LocalSearch	25.48% (26.13%)	43.24%	0.41	84.02%	2.60%	63.28%
FMNIST ($\eta = 0.3$)	DeepFool	27.98% (28.14%)	29.32%	0.66	43.60%	39.90%	42.56%
	PGD	0.00% (0.00%)	35.19%	0.69	41.59%	—	41.59%
	FGSM	1.10% (1.11%)	28.27%	0.53	56.76%	2.38%	56.19%
	Single pixel	85.05% (88.25%)	83.66%	1.02	35.28%	3.45%	8.21%
	LocalSearch	51.10% (52.46%)	73.33%	0.76	60.46%	2.84%	31.01%
CIFAR-10 ($\eta = 0.03$)	DeepFool	0.01% (0.01%)	37.47%	0.95	24.16%	16.10%	24.10%
	PGD	0.00% (0.00%)	35.28%	0.91	28.13%	—	28.13%
	FGSM	0.00% (0.00%)	32.38%	0.83	32.36%	0.00%	32.25%
	Single pixel	60.68% (68.12%)	59.43%	1.01	37.53%	4.39%	17.43%
	LocalSearch	18.99% (21.39%)	52.00%	0.73	48.62%	4.10%	40.16%
Asirra ($\eta = 0.03$)	DeepFool	0.00% (0.00%)	37.01%	0.99	1.18%	12.50%	12.04%
	PGD	0.00% (0.00%)	41.37%	0.98	1.60%	—	1.60%
	FGSM	0.00% (0.00%)	37.71%	0.98	1.64%	0.00%	1.63%
	Single pixel	75.40% (75.50%)	77.01%	1.00	1.63%	0.00%	0.40%
	LocalSearch	57.80% (57.87%)	73.14%	1.00	1.49%	0.00%	0.63%
SVHN ($\eta = 0.03$)	DeepFool	0.01% (0.01%)	31.69%	0.82	36.00%	16.34%	35.86%
	PGD	0.00% (0.00%)	27.98%	0.80	41.33%	0.00%	41.33%
	FGSM	0.00% (0.00%)	19.16%	0.54	58.72%	0.00%	58.55%
	Single pixel	69.59% (77.38%)	72.00%	0.96	58.50%	1.99%	15.78%
	LocalSearch	22.91% (25.47%)	56.55%	0.79	71.61%	2.01%	30.32%

Table 4.3: Performance metrics of the baseline models and the MultIVAPs on adversarial transfer attacks.

centage of the predictions. These percentages can vary significantly depending on the underlying model and the data set: on Asirra, we reject 1.10% of predictions whereas on CIFAR-10 we reject 15.54%. If one counts rejection as misclassification, then the corrected accuracy minus rejections (COR - REJ) can sometimes be lower than the baseline accuracy (ACC). However, depending on the specific application, it may not be appropriate to treat rejections as if they are errors: if correctness of the output is of paramount importance — such as when machine learning is used for medical applications, industrial control systems or autonomous vehicles — then it can be preferable to reject a relatively large number of predictions and return control to human moderators instead of allowing the model to continue functioning based on erroneous output. The risk of false rejection can be outweighed by the risks associated with misclassification, especially if the false rejection rate is relatively low (as it is in table 4.2).

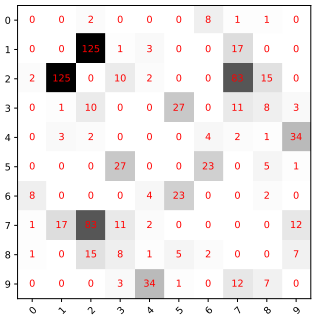
The Jaccard indices are relatively high, although (as expected) it is lower than the baseline accuracy because of the rejected predictions. The predictive efficiency is very close to one on all tasks, with high TRRs and relatively low FRRs. The significance levels $1 - \varepsilon$ are rather low, however, meaning the statistical guarantees provided by our method are relatively weak. This may be due to our specific way of tuning ε or it may be due to looseness of the bounds. Improving these bounds and obtaining better guarantees is left to future work.

Results for adversarial transfer attacks are presented in table 4.3. For MNIST and Fashion-MNIST, we chose $\eta = 0.3$; for all other data sets, we let $\eta = 0.03$ as these bounds are fairly typical in the literature. We can see that the MultIVAP is significantly more accurate than the original models and that it generally has a high TRR on adversarial samples.

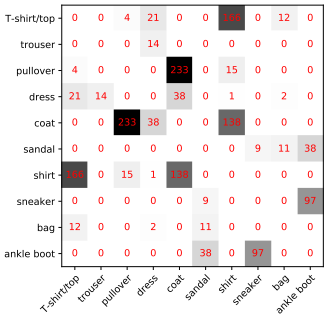
Since the MultIVAP inevitably returns multiple labels sometimes, it is interesting to study which labels it tends to output together. This gives us an idea of the classes that the MultIVAP has difficulty separating and may be a useful tool for diagnosing model shortcomings. Figure 4.3 plots co-occurence matrices as heatmaps for each classification task, where the cell at row i, column j contains the number of times that labels i and j were returned together in a prediction on the proper test set. The diagonals are left blank for clarity since every label trivially co-occurs with itself each time. We see from these heatmaps that the classes that are most difficult for the MultIVAP to separate are also the most perceptually similar ones: on CIFAR-10, automobiles are confused with trucks and deers are confused with horses, for example. On the other hand, birds are never confused with horses and frogs are never confused with trucks. Similar results hold for the other data sets.

Figure 4.4 shows the success rate of our defense-aware adversarial attack as a function of the number of iterations of gradient descent. We used the same values of η for the defense-aware attack as we did for the transfer attacks. The maximum number of iterations was limited to 5,000 as we noticed that the success rates for all tasks seemed to plateau after this point. The highest rate we were able to achieve on any task is less than 40%. For CIFAR-10 in particular, we have a success rate of less than 30%. By contrast, at the time of this writing, state of the art robust accuracy on CIFAR-10 is less than 50% for $\eta = 0.03$ [Shafahi et al., 2019]. On MNIST and Fashion-MNIST, our success rates of 15% and 30% at $\eta = 0.3$ seem to be competitive with existing methods, which achieve robust accuracies of 87.54% and 76.83% respectively [Andriushchenko and Hein, 2019]. Note also that our ℓ_∞ bound of 0.3 for Fashion-MNIST is three times higher than the 0.1 bound used by Andriushchenko and Hein [2019].

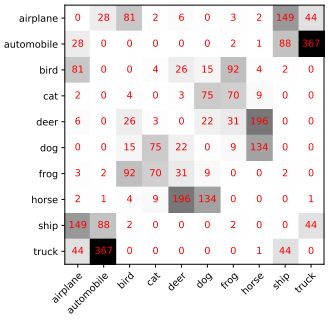
To gain more insights into the precise nature of our defense-aware adversarial examples, we plot the associated confusion matrices in figure 4.5. We see that these bear a striking similarity to the



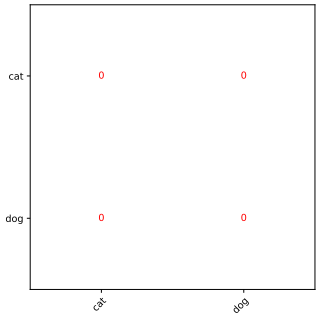
(a) MNIST



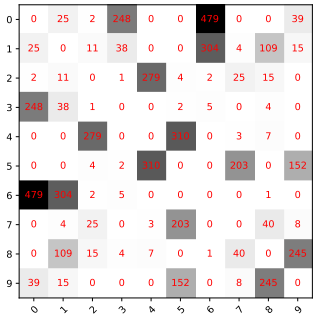
(b) Fashion-MNIST



(c) CIFAR-10



(d) Asirra



(e) SVHN

Figure 4.3: Co-occurence matrices of the MultIVAP for the different classification tasks.

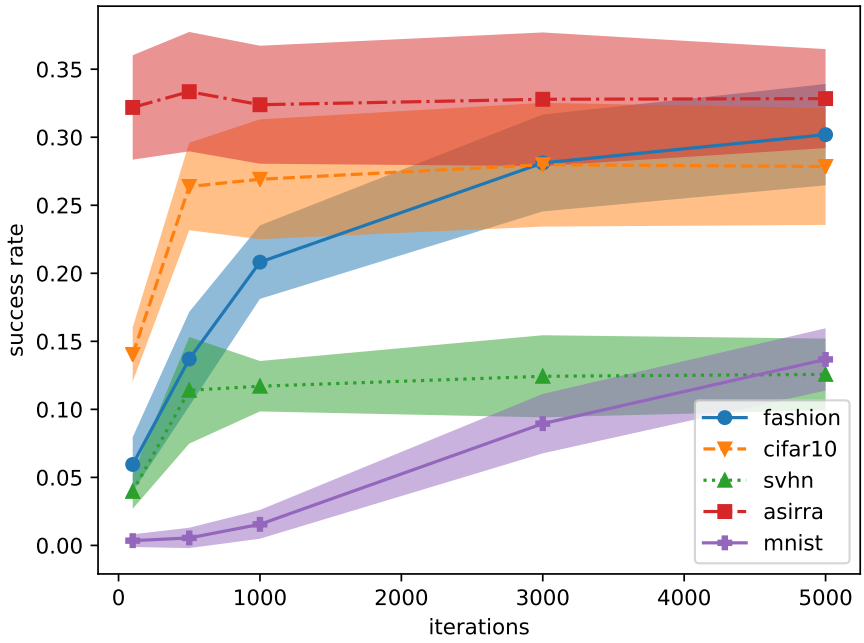


Figure 4.4: Success rate of the defense-aware adversarial attack as a function of the number of iterations of optimization. The lines represent the mean success rate based on ten random restarts of the attack. The area between the 5th and 95th percentiles is shaded.

co-occurence matrices from figure 4.3, where we observed that the MultIVAP mainly confuses perceptually similar classes. This means that when the defense-aware adversarial attack succeeds in fooling the MultIVAP, the resulting adversarial sample is usually misclassified into a class that is visually close to the original correct class. This is an important finding for two reasons:

- It represents a definite improvement over the state of the art in adversarial defense. Currently, even the most robust models can still be tricked into misclassifying a given sample into almost any other class regardless of similarity. The fact that the MultIVAP mostly confuses only perceptually similar classes is a good indicator that it is fundamentally more robust than other models.
- The mistakes that the MultIVAP makes on the adversarial examples are similar to the ones it makes on the test set. This suggests that we can make the model more robust via "traditional" means, that is, by decreasing test error using data augmentation or other tricks. Co-occurence matrices such as figure 4.3 can show us which classes are the most difficult for the MultIVAP to separate. This information can be used to guide efforts for reducing test error. More advanced and computationally expensive methods such as adversarial training may not be required to increase robustness; standard techniques for improving the generalization performance might already suffice.

Figure 4.6 shows the results of our timing experiments. Here, we compute the average time per prediction for the baselines as well as the MultIVAPs and calculate the base-10 log of the ratio between these two quantities over the proper test set:

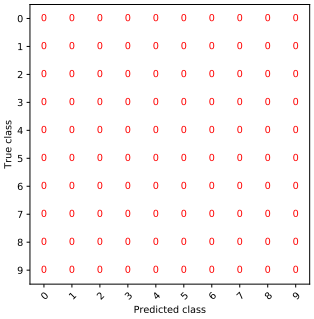
$$\text{LogRatio} = \log_{10}\left(\frac{\text{average MultIVAP inference time}}{\text{average baseline inference time}}\right).$$

We find that the log-ratio is close to one for problems with ten classes whereas it is close to 0.2 for the binary classification task. This translates to a slow-down factor of ten and 1.6 respectively. In practice, we found that this overhead was negligible in wall clock time for the models and data sets we tested.

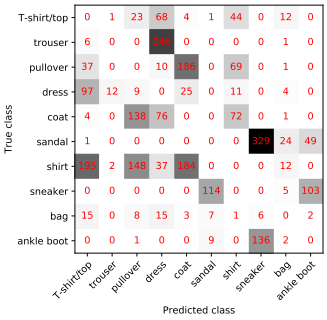
4.7 Conclusions

We have proposed a computationally efficient multiclass generalization of the inductive Venn-ABERS prediction algorithm which we have called the *MultIVAP*. In our experiments, we have found that this method significantly increases both the accuracy as well as the adversarial robustness of any model with which it is instantiated. The MultIVAP takes two hyperparameters, a calibration data set and a significance level ε, which can be tuned according to various objectives. The method enjoys the theoretical guarantee that its label sets will contain the true label with a probability of at least $1-\varepsilon$.

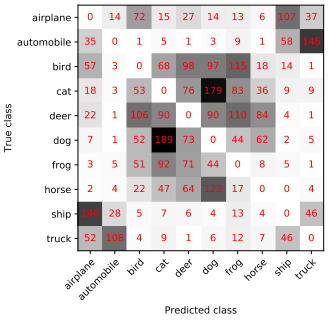
Several avenues for future work are possible. For one, the probabilistic bounds we have obtained for the error probability of our method have the advantage that they reduce inference using the MultIVAP to a MILP, which can be solved relatively efficiently. However, this efficiency comes at the cost of looseness: the tuned values of $1-\varepsilon$ we computed in our experiments are very low (always less than 50% and often less than 25%). Obtaining tighter probabilistic bounds with stronger guarantees on the label set is perhaps the most interesting improvement that could be made. The looseness we suffer with the MultIVAP is likely due to our reliance on the intersection



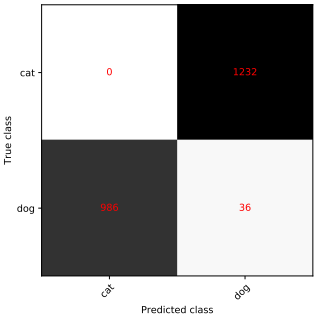
(a) MNIST



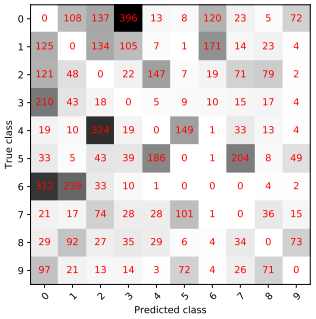
(b) Fashion-MNIST



(c) CIFAR-10



(d) Asirra



(e) SVHN

Figure 4.5: Confusion matrices of the adversarial examples for each task.

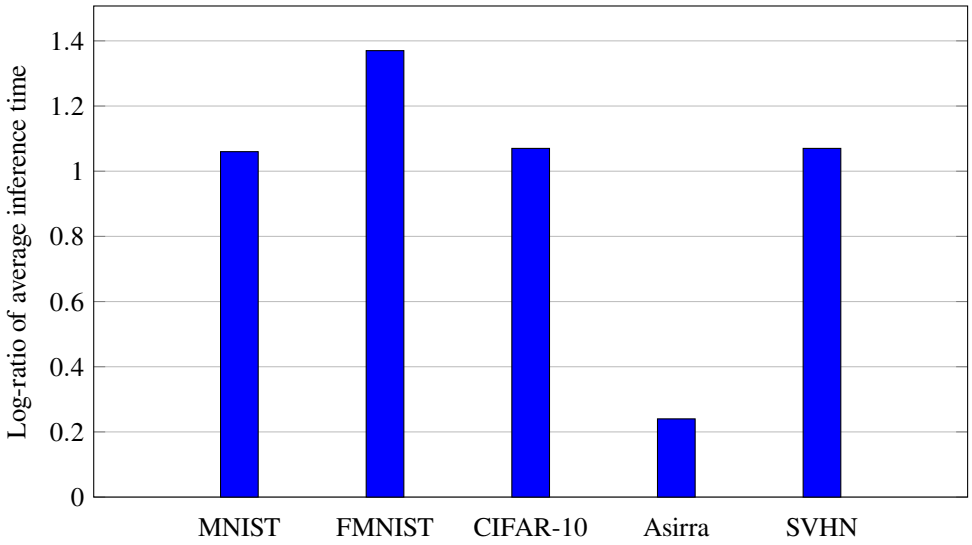


Figure 4.6: Comparison of the computational efficiency of the baseline models and the MultIVAPs. These measurements were taken on a single machine utilizing one NVIDIA Titan Xp GPU alongside four Intel Core i5-6600 CPUs @ 3.30GHz each.

bound, which means the resulting theoretical guarantees will become weaker as the number of classes increases and the approximation error introduced by the intersection bound grows larger. We would need to find other ways of bounding the probability that $V \subseteq L$ given the input data that do not rely on the intersection bound.

Our method is closest in spirit to the work of Manokhin [2017], who extended the IVAP using a one-vs-one approach. By contrast, we employ a one-vs-all strategy. This seems to allow us to derive more straightforward theoretical guarantees on the calibration properties of the MultIVAP, whereas these guarantees are not so easy to give for Manokhin's method. On the other hand, a one-vs-one approach is less sensitive to class imbalance, since each pair of classes is compared individually; for one-vs-all, every individual class is compared to all other classes. This makes the resulting binary classification problems heavily imbalanced, which can be exacerbated if the data itself is already imbalanced. In scenarios of high class imbalance, it is likely that the MultIVAP will always include certain labels corresponding to minority classes in order to hedge the probabilities. Studying how the MultIVAP behaves in such cases could also be interesting future work.

Improving the computational efficiency of this method when the number of classes becomes very high is definitely a worthwhile avenue for future work as well. Currently, the MultIVAP appears to increase inference time by a factor that is approximately equal to the number of classes. In applications that are very time-sensitive and which have a large number of classes (*e.g.*, extreme classification as in Jain et al. [2019]), this might be unacceptable. The main bottleneck here is the optimization problem; the other part of the algorithm which gathers the outputs of the different IVAPs is straightforward to parallelize as the different classes can be treated independently. One straightforward solution would be to use off-the-shelf MILP solvers

that have already been parallelized. We refer the reader to Ralphs et al. [2018] for a survey.

Furthermore, in data-limited settings, it may be undesirable to sacrifice a relatively large portion of the data set for calibration of the MultIVAP, especially if a separate validation split is required for tuning other hyperparameters. We did not study the effect of the size of the calibration set in detail here, but it can prove useful to test how small this set can be made before the MultIVAP is unable to meet certain guarantees on robustness and accuracy.

It may also be possible to develop stronger defense-aware attacks against the MultIVAP, since we only studied one attack in this work. We recall *Schneier's Law* [Schneier, 2011]:

> *Anyone, from the most clueless amateur to the best cryptographer, can create an algorithm that they themselves cannot break.*

We therefore invite the community to scrutinize our defense and develop stronger attacks against it. Nevertheless, even if our particular defense is ever broken, we believe there is a broader point to be made here: to the best of our knowledge, there exist almost no papers that attempt to tackle adversarial machine learning using techniques from the field of conformal prediction. Given that this entire field is dedicated precisely to the problem of trustworthiness of probabilistic predictions, it would seem there is still much unexplored potential in this area. We refer the interested reader to the works of Shafer and Vovk [2008], Vovk et al. [2005, 2015].

Chapter 5

The CANN detector

> Γλυκόπικρη, περίεργη εφηβεία, γεμάτη μύθους, φόβους και απαντοχή. Κρύβαμε το άγχος μας κάτω από ένα ψεύτικο ρομαντισμό και την περιέργεια πίσω από μια μάσκα πολυγνωσίας. Δεν ξέραμε τίποτε. Αλλά οι μύθοι μας ήταν τόσο αληθινοί για μας, που νομίζαμε πως τα ξέραμε όλα.
>
> *Παρ' όλα αυτά*
> Νίκος Δήμου

The binary IVAP algorithm 3.3 and the MultIVAP algorithm 4.1 are adversarial defenses based on the original IVAP proposed by Vovk et al. [2015], which at its heart is simply an isotonic regression computed on the scores generated by the underlying model. While simple and effective, there is little architectural leeway in the design of the defense. There are very few hyperparameters that can be tuned: the binary IVAP has the β threshold and the MultIVAP has the ε confidence parameter. If no setting of these scalar parameters yields acceptable performance according to relevant metrics, the designer has no choice other than to improve the underlying classifier, for instance by adversarial training. Depending on the particular task and problem constraints, this may not be feasible or desirable. Moreover, although we have empirically assessed the computational overhead of the MultIVAP to be approximately linear in the number of classes, its major bottleneck remains the optimization problem that must be solved for each individual sample at inference time. For k classes, the number of unknowns to be solved for in (4.6) is precisely $k+1$, which may become prohibitive at scale for certain data sets.

In an effort to eliminate these shortcomings, we turn to the general conformal prediction algorithm 3.1. This algorithm requires a non-conformity measure $\mathcal{A}$, which is a function that takes as input a bag B of previously-seen examples (*e.g.*, the training set or a part thereof) as well as a new sample z and outputs a non-negative real number $\mathcal{A}(B,z) \in \mathbb{R}$. The main desideratum of $\mathcal{A}$ is that $\mathcal{A}(B,z)$ should increase as z becomes more "anomalous" (or *non-conformal*) with respect to the bag B. Algorithm 3.1 employs the given non-conformity measure in a simple statistical test which, in the case of classification, yields a p-value for each possible label. It then returns

the set of labels that have a sufficiently high p-value as specified by the confidence parameter ε.

The idea in this section is to develop an adversarial defense based on algorithm 3.1 for multi-class classification with greater flexibility and efficiency than the MultIVAP. We achieve this by letting the non-conformity measure be determined by a neural network trained in a specific manner to facilitate detection of anomalous samples. For this, we make use of an old statistical tool that lends itself well to anomaly detection: *correspondence analysis* (CA) [Greenacre, 2010]. This method, which we describe in detail below, was originally proposed as a form of dimensionality reduction for categorical data. More recently, Hsu et al. [2019] generalized CA to continuous random variables and developed a neural network architecture with which to compute it, known as the *correspondence analysis neural network* (CANN). The CANN allows us to apply CA to large-scale data sets containing continuous variables, such as those commonly of interest in the field of computer vision and adversarial robustness.

We use this method along with certain mathematical properties of CA to define a non-conformity measure that can be computed efficiently given the output of the CANN. This yields a new adversarial detector that inherits the efficiency of the conformal prediction algorithm while being much more malleable, as the practitioner can now define an entire neural network instead of only specifying a single scalar hyperparameter. Of course, this increased flexibility also confers a higher responsibility to the designer to create and train an effective neural network. That said, the resulting CANN detector algorithm only depends on the data set and can treat the underlying classifier to be protected as a complete black-box (like the MultIVAP). As such, research can be dedicated to the construction of well-performing CANN detectors for specific data sets, after which these models can be released to the public and subsequently re-used "off the shelf" by ML engineers who do not have the resources to construct these detectors by themselves.

The contents of this chapter were not yet published elsewhere at the time this thesis was written.

5.1 Correspondence analysis

Given two random variables X and Y with support on $\mathcal{X}$ and $\mathcal{Y}$ respectively, the goal of *correspondence analysis* is to find functions $f_1, \ldots, f_d : \mathcal{X} \to \mathbb{R}$ and $g_1, \ldots, g_d : \mathcal{Y} \to \mathbb{R}$ such that the correlations between $f_i(X)$ and $g_i(Y)$ are maximized but the different functions f_i and g_i are mutually uncorrelated. More formally, these functions are defined by

$$(f_i, g_i) = \operatorname{argmax} \mathbb{E}[f(X)g(Y)]^2$$
$$\text{such that} \begin{cases} f \in \mathcal{L}_2(\mathcal{X}), \\ g \in \mathcal{L}_2(\mathcal{Y}), \\ \forall j < i : \mathbb{E}[f_j(X)f(X)] = \mathbb{E}[g_j(Y)g(Y)] = 0. \end{cases}$$

Here, $\mathcal{L}_2(\mathcal{X})$ and $\mathcal{L}_2(\mathcal{Y})$ are the function spaces of zero-mean unit-variance $\mathcal{X} \to \mathbb{R}$ and $\mathcal{Y} \to \mathbb{R}$ functions:

$$\mathcal{L}_2(\mathcal{X}) = \{f : \mathcal{X} \to \mathbb{R} \mid \mathbb{E}[f(X)] = 0, \mathbb{E}[f(X)^2] = 1\},$$
$$\mathcal{L}_2(\mathcal{Y}) = \{g : \mathcal{Y} \to \mathbb{R} \mid \mathbb{E}[g(Y)] = 0, \mathbb{E}[g(Y)^2] = 1\}.$$

This iterative process of optimization yields a set of pairs $(f_1, g_1), \ldots, (f_d, g_d)$ known as the *principal functions* [Calmon et al., 2017]. The *principal inertia components* are the absolute

values of the expectations themselves:

$$\sigma_i = |\,\mathbb{E}[f_i(X)g_i(Y)]|.$$

The total number d of principal functions is equal to the minimal dimension of the function spaces $\mathcal{L}_2(\mathcal{X})$ and $\mathcal{L}_2(\mathcal{Y})$. In our case, $\mathcal{X}$ is a subset of $\mathbb{R}^n$ and $\mathcal{Y}$ is a finite set of labels $\{1,\ldots,k\}$. As such, we have $d = k-1$.

The principal functions and principal inertia components can equivalently be defined as the diagonalization of the conditional expectation operator. Specifically, for any $f \in \mathcal{L}_2(\mathcal{X})$,

$$\mathbb{E}[f(X) \mid Y = y] = \sum_{i=1}^{d} \sigma_i g_i(y)\, \mathbb{E}[f_i(X)f(X)].$$

Similarly, for any $g \in \mathcal{L}_2(\mathcal{Y})$,

$$\mathbb{E}[g(Y) \mid X = x] = \sum_{i=1}^{d} \sigma_i f_i(x)\, \mathbb{E}[g_i(Y)g(Y)].$$

As such, they have found many applications in machine learning, information theory and even cryptography [Bell, 1962, Calmon et al., 2017, Hardoon et al., 2004, Makur et al., 2015, Rényi, 1959]. One property that is relevant to us here is the fact that the sum of squares of the principal inertia components equals the Pearson χ^2-divergence between the joint distribution (X,Y) and the product distribution $X \times Y$:

$$\chi^2(X;Y) = \sum_{i=1}^{d} \sigma_i^2 = \mathbb{E}\left[\left(\frac{\Pr[X,Y]}{\Pr[X]\Pr[Y]}\right)^2\right] - 1.$$

This is a special case of an f-divergence with $f(t) = t^2 - 1$. It is in some sense proportional to the dependence between X and Y. It is bounded by $0 \leq \chi^2(X;Y) \leq d$, with $\chi^2(X;Y) = 0$ indicating that the random variables are statistically independent and $\chi^2(X;Y) = d$ indicating that X and Y are deterministic measurable transformations of each other. Specifically, we have $\chi^2(X;Y) = d$ if and only if $\Pr[f_i(X) = g_i(Y)] = 1$ for all $i = 1,\ldots,d$. Essentially, the principal functions $\{f_i\}$ and $\{g_i\}$ can be seen as the most informative features of the data [Makur et al., 2015]. Due to these desirable properties, the principal functions are an attractive foundation for a non-conformity measure for use in a conformal predictor.

For large-scale problems, Hsu et al. [2019] have shown that the principal functions and the principal inertia components can be estimated accurately using neural networks. Their construction, the *correspondence analysis neural network* (CANN), is depicted in figure 5.1. It in fact consists of two neural networks: an F-Net which estimates the functions $f_1,\ldots,f_d$ and a G-Net which estimates $g_1,\ldots,g_d$. The parameters of these networks are jointly optimized via gradient descent on the *PIC loss*, a custom loss function which causes the networks to converge to the principal functions. We will make use of this construction in our detector approach.

5.2 Conformal CANN detector

The basic idea behind our detector is the fact that the principal functions $\{f_j\}$ and $\{g_j\}$ are minimum mean-squared estimators of each other [Calmon et al., 2017, Makur et al., 2015].

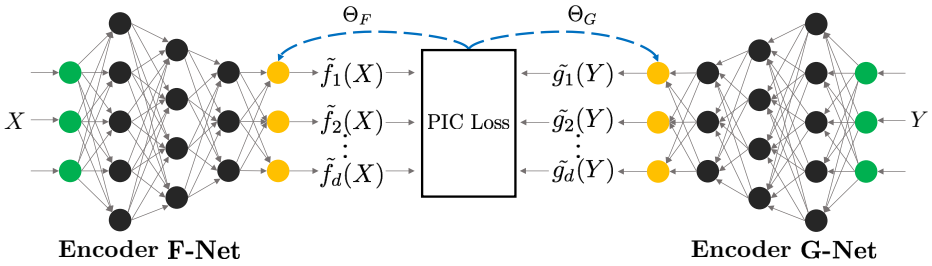


Figure 5.1: The architecture of the Correspondence Analysis Neural Network (CANN) from [Hsu et al., 2019].

That is,

$$\mathbb{E}[f_j(X) \mid Y = y] = \sigma_j g_j(y), \qquad \mathbb{E}[g_j(Y) \mid X = x] = \sigma_j f_j(x).$$

Therefore, we expect the squared difference between $f_j(x)$ and $\sigma_j g_j(y)$ to be small for all j whenever x and y are sampled from the joint distribution (X,Y). If these values are large, then this is an indication that x and y may have been sampled from a different distribution. We consider the mean-squared errors of these estimates for a given sample (x,y):

$$\Delta(x,y) = \frac{1}{d}\sum_{j=1}^{d}(f_j(x) - \sigma_j g_j(y))^2.$$

Given a data set of samples $D = \{(x_1,y_1),\ldots,(x_m,y_m)\}$, we can define a non-conformity measure based on $\Delta(x,y)$ for use in a conformal predictor:

$$\mathcal{A}(D,x,y) = \left(\Delta(x,y) - \frac{1}{|D|}\sum_{(x_i,y_i)\in D}\Delta(x_i,y_i)\right)^2.$$

This measure is simply the squared deviation of $\Delta(x,y)$ from the mean over the given data set. Algorithm 5.1 shows the full conformal prediction algorithm based on this measure. The data set D which we use here to determine the values α_i is a held-out validation set containing the ground-truth labels.

Algorithm 5.1 will compute a p-value $p \in [0,1]$ for each sample x that we provide to it. As is typical in frequentist statistics, this p-value must be "sufficiently small" in order for us to consider the result significant. A significant result here translates into *rejecting* the prediction of the model. Therefore, we must determine some threshold τ such that as many correct predictions are accepted as possible while minimizing the number of accepted incorrect predictions. Samples with incorrect predictions should be rejected whereas samples with correct predictions should be accepted. This leads to a confusion matrix such as table 4.1, which we also used to evaluate the MultIVAP in chapter 4. By plotting the true rejection rate (TRR) against the false rejection rate (FRR), we obtain the *Receiver Operating Characteristic* (ROC) curve [Murphy, 2012]. Ideally, we would choose a point on the ROC so that we maximize the TRR and minimize the FRR. One way of achieving this is to find a threshold τ which maximizes Youden's J index [Youden, 1950], similarly to the IVAP defense in chapter 3:

$$J = \text{TRR} - \text{FRR}.$$

Algorithm 5.1: Conformal CANN detector

Data: data set of samples D, new sample x, model f_θ, threshold τ
Result: `ACCEPT` or `REJECT`

Obtain the predicted label $\hat{y} \leftarrow f_\theta(x)$.
for *i from 1 to* $|D|$ **do**
  $\alpha_i \leftarrow \mathcal{A}(D \setminus \{(x_i, y_i)\}, x_i, y_i)$.
end
$\alpha_{|D|+1} \leftarrow \mathcal{A}(D, x, \hat{y})$.
$p \leftarrow \frac{1}{|D|+1} \#\{i = 1, \ldots, |D|+1 \mid \alpha_i \geq \alpha_{|D|+1}\}$.
if $p > \tau$ **then**
  return *ACCEPT*
else
  return *REJECT*
end

We tune τ to maximize J via a held-out validation data set of both clean and adversarial samples. More details can be found in section 5.4.

5.3 Adaptive attack

In order to properly evaluate the robustness of our conformal CANN detector approach, we design an adaptive attack specifically crafted to evade our detector and fool the underlying model. This means we must find samples $\tilde{x}$ in an ε-ball around the original x such that the model misclassifies $\tilde{x}$ and the non-conformity score $\mathcal{A}(D, \tilde{x}, y)$ is sufficiently low so that the detector does not reject the sample. We follow Carlini and Wagner [2017a] in the design of our attack. That is, we solve the following optimization problem:

$$\min_{\tilde{x}} \; \mathcal{D}(x, \tilde{x}) + \lambda g(\tilde{x}). \tag{5.1}$$

The quantity $\mathcal{D}$ is a distance metric between the original x and the adversarial $\tilde{x}$. As discussed in Carlini and Wagner [2017c], the L_∞ metric we use here is not easily optimized by gradient descent. Therefore, the following proxy is recommended:

$$\mathcal{D}(x, \tilde{x}) = \sum_{i=1}^{n} \max\{\tilde{x}_i - x_i - \varepsilon, 0\}.$$

The function g must have the property that $g(\tilde{x}) \leq 0$ if and only if both of the following conditions hold:

1. $f_\theta(\tilde{x}) = y'$ for some target class $y' \neq y$;
2. the detector accepts the prediction $(\tilde{x}, y')$.

The detector accepts a prediction $(\tilde{x}, y')$ if and only if the non-conformity measure $\mathcal{A}(D, \tilde{x}, y')$ is below some threshold t which depends only on the data set D. We can therefore define the objective function as follows:

$$g(\tilde{x}) = \max\{h(\tilde{x}), \mathcal{A}(D, \tilde{x}, y') - t\},$$

where h can be any of the objective functions proposed by Carlini and Wagner [2017c] which have the property that $f_\theta(\tilde{x}) = y'$ if and only if $h(\tilde{x}) \leq 0$. In our implementation, we use

$$h(\tilde{x}) = \max\left\{0, \max_{i \neq y'} \hat{p}_i(\tilde{x};\theta) - \hat{p}_{y'}(\tilde{x};\theta)\right\}.$$

In other words, h penalizes the margin between the highest-scoring class and our target class in terms of the estimated model probabilities. We note that the authors of Carlini and Wagner [2017c] actually advocate for the use of the model *logits* $z_\theta(x)$ instead of the estimated probabilities $\hat{p}(x;\theta)$ as this has turned out to be more effective. However, we deliberately chose to optimize the adversarials with respect to the probability space instead of the logit space. The reason for this is that logit-based adversarials tend to be heavily over-optimized [Ozbulak et al., 2019]: the softmax transform "squeezes" the unbounded logits into the compact space $[0,1]^k$, so that large differences in logit values will translate into only small differences in softmax output. This makes it easier to create adversarial examples, but the resulting adversarials can easily be detected using simple techniques such as IQR thresholding [Ozbulak et al., 2019] because their logit values are highly anomalous. Therefore, we optimize the adversarials in softmax space.

As in Carlini and Wagner [2017c], the parameter λ which provides the trade-off between the distance metric $\mathcal{D}$ and the adversarial loss g is tuned during the optimization itself to obtain the highest possible success rate. Specifically, we start with a low value of $\lambda = 1\text{e}{-3}$ and successively double the value until the adversarial is successful or we reach an upper bound of 2,048. The objective function (5.1) is optimized using gradient descent, where we initialize the optimization with adversarial samples generated by the L_∞ projected gradient descent attack from Madry et al. [2017]. As such, the optimization proceeds in two stages:

1. Generate an L_∞ PGD adversarial to ensure the model misclassifies the generated input without perceptible alterations to the original sample.
2. Refine the initial adversarial so that the non-conformity score is minimized and the detector is bypassed. The target class y' used for this stage of the attack is the predicted class on the adversarial generated in the first stage.

Algorithm 5.2 summarizes the attack in pseudo-code. Note that in the computation of the loss ℓ_t, we reduce the non-conformity threshold by 1% so that the non-conformity of the adversarial $\mathcal{A}(D, \tilde{x}_t, y')$ is optimized to lie just above the acceptance threshold. This is necessary because the inequality on line 7 of algorithm 5.1 is strict.

5.4 Experiments

We carry out experiments on four different data sets, described below:

MNIST. The MNIST data set consists of 28×28 grayscale images of handwritten digits [LeCun, 1998]. The total of 60,000 samples is divided equally into ten classes (from 0 to 9).

Fashion-MNIST. The Fashion-MNIST data set was designed by Xiao et al. [2017] as a drop-in replacement for MNIST. It consists of 28×28 grayscale images of articles of clothing. This data set has 60,000 samples just like MNIST.

Algorithm 5.2: Adaptive adversarial attack

Data: data set of samples D, sample (x, y), perturbation budget $\varepsilon > 0$, step size $\alpha > 0$, parameter $\lambda > 0$, number of iterations T

Result: adversarial sample $\tilde{x}$

Run the L_∞ PGD attack to obtain an adversarial example $\tilde{x}_0$ with $\|x - \tilde{x}_0\| \leq \varepsilon$.

Let y' be the predicted class on the adversarial: $y' \leftarrow f_\theta(\tilde{x}_0)$.

Compute the non-conformity threshold r.

for *t from 0 to* $T-1$ **do**

  Compute the loss:

$$\ell_t \leftarrow \mathcal{D}(x, \tilde{x}_t) + \lambda \max\{h(\tilde{x}_t, y', \theta), \mathcal{A}(D, \tilde{x}_t, y') - 0.99r\}.$$

  Compute the update:

$$\tilde{x}_{t+1} \leftarrow \tilde{x}_t - \alpha \nabla_{\tilde{x}_t} \ell_t.$$

end

return $\text{clip}(\tilde{x}_T, x - \varepsilon, x + \varepsilon)$

CIFAR-10. The CIFAR-10 data set was created by Krizhevsky and Hinton [2009] and consists of 32×32 RGB images of common everyday objects and animals. There are 60,000 samples in total.

SVHN. The Street View House Numbers (SVHN) data set contains pictures of house numbers collected by Netzer et al. [2011]. There are 99,289 samples in total.

For each data set, we test four models: a simple convolutional neural network (CNN) of our own design, ResNet50 [He et al., 2016], DenseNet121 [Huang et al., 2017] and Xception [Chollet, 2017]. The architecture of the simple CNN is illustrated in figure 5.2. We fit these models and evaluate them against adversarial attacks at different perturbation budgets. We plot the robust accuracy as a function of the perturbation budget along with the true rejection rate (TRR) and false rejection rate (FRR).

All of our experiments were carried out using the TensorFlow Keras framework [Abadi et al., 2015, Chollet et al., 2015]. The L_∞ PGD attack implementation was provided by the Foolbox library [Rauber et al., 2020]. For the correspondence analysis, we started from the original implementation[1] by Hsu et al. [2019]. However, we had trouble scaling this implementation to data sets such as CIFAR-10 and SVHN due to numerical errors. These problems mainly arose from the use of explicit matrix inverses and naive estimation of empirical covariance matrices. We resolved these issues, making use of the scikit-learn implementation of the Ledoit-Wolf covariance estimator [Ledoit and Wolf, 2004, Pedregosa et al., 2011].

Figure 5.3 shows the different data splits used for the experiments. We split the full data set into 70% training data, 20% testing data and 10% validation data. We run the PGD attack on the testing and validation data sets to create sets of adversarial samples. The training data is used to fit the baseline model, the testing data is used to evaluate it and the validation data is used for early stopping. We measure the robust accuracy by evaluating the model on the adversarial test

[1] `https://github.com/HsiangHsu/2019-AISTATS-CA`. Accessed: 2020-07-06.

```
import tensorflow as tf

# Note: input shape depends on the data set
input_shape = (32, 32, 3)

model = tf.keras.models.Sequential()
model.add(tf.keras.layers.Conv2D(32, (3, 3), padding='same',
            input_shape=input_shape))
model.add(tf.keras.layers.Activation('relu'))
model.add(tf.keras.layers.Conv2D(32, (3, 3)))
model.add(tf.keras.layers.Activation('relu'))
model.add(tf.keras.layers.MaxPooling2D(pool_size=(2, 2)))
model.add(tf.keras.layers.Dropout(0.25))

model.add(tf.keras.layers.Conv2D(64, (3, 3), padding='same'))
model.add(tf.keras.layers.Activation('relu'))
model.add(tf.keras.layers.Conv2D(64, (3, 3)))
model.add(tf.keras.layers.Activation('relu'))
model.add(tf.keras.layers.MaxPooling2D(pool_size=(2, 2)))
model.add(tf.keras.layers.Dropout(0.25))

model.add(tf.keras.layers.Flatten())
model.add(tf.keras.layers.Dense(512))
model.add(tf.keras.layers.Activation('relu'))
model.add(tf.keras.layers.Dropout(0.5))
model.add(tf.keras.layers.Dense(output_shape))
model.add(tf.keras.layers.Activation('softmax'))
```

Figure 5.2: TensorFlow Keras specification of the Simple CNN architecture.

set. The threshold τ is tuned to maximize Youden's index on the clean and adversarial validation sets. That is, we optimize τ so that as many samples as possible from the clean validation set pass through the detector whereas we try to reject as many samples as possible from the adversarial validation set. We then determine the clean and adversarial rejection rates by evaluating the detector on the clean and adversarial test sets, respectively.

Accuracy scores on clean test data of the different baseline models are given in table 5.1. To apply our detector, we specify a CANN for each data set and fit it according to the method described by Hsu et al. [2019]. The resulting estimates of the principal functions

$$(f_1, g_1), \ldots, (f_{k-1}, g_{k-1})$$

and principal inertia components

$$\sigma_1, \ldots, \sigma_{k-1}$$

are then used to compute the non-conformity scores for algorithm 5.1.

Table 5.2 shows the mean values of the deviation $\Delta(x_i, y_i)$ across the training data sets. This is the value used by the conformal predictor to compute the non-conformity score: if $\Delta(x, y)$

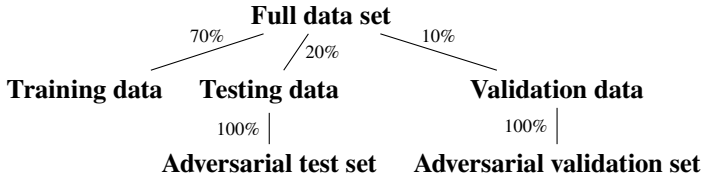


Figure 5.3: Illustration of the different data splits used in our experiments.

Data set	Simple CNN	ResNet50	DenseNet121	Xception
MNIST	99.24%	99.29%	99.34%	99.11%
Fashion-MNIST	93.21%	91.53%	91.33%	92.04%
CIFAR-10	78.40%	73.83%	78.82%	75.44%
SVHN	93.72%	92.65%	93.47%	93.83%

Table 5.1: Clean accuracy scores of the baseline models.

differs too much from this mean, the prediction is rejected. Note that we trained a separate CANN for each of the baseline models, so the values differ slightly across models.

5.4.1 Defense-oblivious attack

We first evaluate the robustness of our detector using a *defense-oblivious* attack where we generate adversarials using the L_∞ PGD attack for the underlying model without taking the defense into account. We set the perturbation budgets to 30%, 10%, 3% and 3% of the range for MNIST, Fashion-MNIST, CIFAR-10 and SVHN respectively since these values are common in the literature. The results are shown in table 5.3. AUROC indicates the area under the ROC curve, a common measure for these types of detectors that is independent of any particular threshold. The TRR, FRR and ACC values are reported for our tuned threshold and correspond to the true rejection rate, false rejection rate and overall detector accuracy.

We see from these results that our detector achieves relatively high AUROC, TRR and ACC values at low FRR across all models and data sets we tested. In some applications, the FRR values may still be too high and so one may wish to tune the thresholds using a different metric than Youden's index. In cybersecurity settings, for example, it is common to tune the threshold for a specific false rejection rate (e.g., 0.01%) regardless of the accuracy or true rejection rate [Peck et al., 2019b]. To put these numbers in perspective, in section 5.6, we compare our method to other state-of-the-art detector approaches in the defense-oblivious setting.

It is important to note that defense-oblivious attacks have limited practical use, as one cannot safely assume that the attacker is unaware of the defensive measures that have been put in place [Carlini et al., 2019]. However, they demonstrate the potential viability of the method, since robustness against defense-oblivious attacks is a necessary condition for any serious defense.

Data set	SimpleCNN	ResNet50	DenseNet121	Xception
MNIST	0.0804	0.0735	0.0848	0.0888
Fashion-MNIST	0.0583	0.0695	0.0644	0.0707
CIFAR-10	0.4527	0.3665	0.3708	0.3872
SVHN	0.1126	0.1135	0.1133	0.1174

Table 5.2: Mean deviation values for the conformal predictor.

Data set	Network	AUROC	TRR	FRR	ACC
MNIST	SimpleCNN	85.89%	80.51%	9.01%	85.71%
	ResNet50	83.54%	78.67%	10.52%	84.10%
	DenseNet121	87.09%	82.70%	9.64%	86.51%
	Xception	81.88%	73.69%	8.11%	82.71%
Fashion-MNIST	SimpleCNN	69.22%	46.65%	6.17%	68.64%
	ResNet50	70.13%	47.99%	4.96%	69.97%
	DenseNet121	62.29%	47.57%	6.98%	68.33%
	Xception	70.42%	55.03%	8.73%	71.71%
CIFAR10	SimpleCNN	70.16%	56.40%	7.27%	70.68%
	ResNet50	65.96%	57.19%	8.86%	70.40%
	DenseNet121	65.01%	55.73%	11.13%	69.04%
	Xception	69.82%	60.67%	12.57%	70.83%
SVHN	SimpleCNN	79.79%	69.40%	6.17%	82.00%
	ResNet50	79.85%	69.35%	5.05%	82.04%
	DenseNet121	81.68%	73.82%	5.42%	84.31%
	Xception	85.49%	80.43%	4.78%	87.94%

Table 5.3: Performance metrics for the defense-oblivious attack.

5.4.2 Defense-aware attack

To lend more credibility to our claims of robustness, we also evaluate the detector against our *defense-aware* attack specified in section 5.3. Note that we do not tune the rejection threshold τ on validation adversarials produced by our adaptive attack; instead, we reuse the thresholds that were tuned on the L_∞ PGD adversarials. In that sense, the defense is tuned on the L_∞ PGD attack and subsequently evaluated against our adaptive attack. We made this choice for two reasons:

- The adaptive attack depends on the tuned rejection threshold τ. Therefore, there is a circular dependency where the threshold first needs to be tuned on adversarials, but these adversarials can only be generated if the threshold has already been tuned.
- We do not wish to "overpower" our defense against the adaptive attack. The idea behind adaptive evaluations is that the defense should be fixed beforehand and the adversary should tailor the attack for the defense as it is specifically deployed. In practice, one will tune the rejection threshold based on a data set of adversarials generated by existing

attacks. The defense will then be subjected to adaptive attacks which the practitioners may not have foreseen. As a result, we keep our defense oblivious to the adaptive attack.

The results of these experiments are shown in table 5.4. As before, the perturbation budgets are expressed relative to the range of pixel values. We fixed the total number of iterations of optimization T to 100 here, as we observed no substantial improvements in performance at higher numbers of iterations.

Data set	Network	AUROC	TRR	FRR	ACC
MNIST	SimpleCNN	87.28%	81.66%	9.01%	86.29%
	ResNet50	93.51%	93.63%	10.75%	91.43%
	DenseNet121	94.77%	94.47%	9.64%	92.43%
	Xception	84.13%	75.24%	8.11%	83.49%
Fashion-MNIST	SimpleCNN	72.83%	57.69%	6.18%	74.53%
	ResNet50	63.63%	46.16%	6.31%	68.54%
	DenseNet121	64.59%	47.18%	6.99%	68.11%
	Xception	85.98%	79.83%	8.75%	85.09%
CIFAR10	SimpleCNN	76.45%	63.03%	7.22%	74.82%
	ResNet50	72.78%	65.12%	9.01%	75.36%
	DenseNet121	65.86%	56.89%	11.12%	70.04%
	Xception	75.78%	68.04%	12.55%	75.43%
SVHN	SimpleCNN	79.40%	70.92%	6.54%	83.20%
	ResNet50	79.73%	70.93%	5.19%	83.23%
	DenseNet121	85.20%	78.68%	5.60%	86.97%
	Xception	84.77%	79.77%	4.95%	87.69%

Table 5.4: Performance metrics for the defense-aware attack.

The results for the defense-aware adversarials are comparable to the defense-oblivious ones: we have relatively high AUROC, TRR and ACC values at low FRR. Of course, we must appropriately nuance these results. The defense-aware attack we evaluated against here is just one of many that could potentially be applied. However, the results are still surprising: in the design of our attack, we followed a methodology similar to Carlini and Wagner [2017c], which has turned out to be highly effective against virtually all defenses proposed thus far [Athalye and Carlini, 2018, Carlini and Wagner, 2017a, Carlini et al., 2019]. The fact that this method of constructing adaptive attacks — which has been so successful against past defenses — seems to fail when applied to our defense provides at least some cause for optimism. However, as is well-known in the field of cryptography,[2] anyone can develop a security system that they themselves cannot break. Therefore, we invite the community to scrutinize our defense and attempt to devise more effective attacks against it.

[2] `https://www.schneier.com/blog/archives/2011/04/schneiers_law.html`. Accessed 2020-06-09.

5.5 Discussion

Based on the above results, it appears that our detector approach could be a viable defense against adversarial attacks both in the defense-oblivious as well as the defense-aware settings. One question that we still wish to address is the matter of computational efficiency in terms of memory consumption and runtime overhead. To reiterate, our detector requires two components:

1. a correspondence analysis neural network for computing the non-conformity;
2. a conformal predictor.

In our experiments, the CANN was implemented using convolutional autoencoders for the F-Net and shallow fully-connected networks for the G-Net. As such, the computational overhead of the CANN is roughly proportional to that of an autoencoder on the given data set. However, this network only needs to be trained once, after which it can be evaluated efficiently. This part of the pipeline is also highly flexible, as one can design the architecture freely according to the computational resources that are available. Moreover, since the CANN depends only on the data set itself, it is possible to use pre-trained CANNs shared by other parties with larger computational resources. These pre-trained CANNs could then be fine-tuned in case the specific data set under consideration is not entirely the same as the one for which the CANN was originally trained. This practice of sharing pre-trained models and fine-tuning for specific data sets has become commonplace in modern machine learning, especially in the field of natural language processing (NLP) where state of the art models usually have so many parameters that only a handful of companies could feasibly train these networks from scratch [Vaswani et al., 2017].

The conformal predictor requires us to store an additional data set D for calibration. This algorithm can then be divided into two stages:

1. Initialization. Here, the values of α_i are computed on D and stored for later use. This takes $O(|D|d)$ time and memory provided that the evaluation of the non-conformity measure takes $O(d)$ time and memory. In our case, this assumption is realistic since the non-conformity measure is computed using a fixed neural network and a mean squared deviation value based on the d outputs of this network.
2. Inference. When the conformal predictor is initialized with a data set D, it can compute p-values in time $O(|D|\log|D|)$ since we can assume the values of α_i are sorted. This relies on the assumption that $d \leq |D|$, so that searching through the sorted list of $|D|$ values will dominate the computations.

To summarize, the conformal predictor incurs an overhead of $O(d|D|)$ in memory requirements and $O(|D|\log|D|)$ in execution time. We used our validation data split for D, which is relatively small (10% of the entire data set; see figure 5.3). In wall-clock time, the overhead of the inference phase of our detector was negligible.

We wish to stress that the data set D does not need to be stored fully in its original form in order for our detector to work. We require only the trained CANN, the mean deviation

$$\frac{1}{|D|}\sum_{(x,y)\in D}\Delta(x,y)$$

as well as the values $\alpha_1, \ldots, \alpha_{|D|}$. Given these statistics, the data set D itself can be discarded. This is important in case the detector needs to be deployed in limited-memory settings where the full data set D does not fit into memory. It is also relevant in privacy-sensitive situations where D contains data that cannot be publicly released. Of course, the CANN itself encodes this data set in a certain form, but this problem is common to all machine learning methods. It is a central object of study in the field of privacy-preserving machine learning [Xu et al., 2015] and we consider it out of scope for the present work.

For the sake of completeness, we also mention that the architecture of the CANN depends on the number of classes k, since the network must produce $k-1$ output values to compute the full set of principal functions for use in the deviation measure $\Delta(x,y)$. However, the same is true for the underlying model: it too must have $k-1$ outputs to perform the classification. The number of classes will generally be much smaller than the dimensionality of the input, so this is not really a practical concern. Furthermore, the PIC loss function proposed by Hsu et al. [2019] is flexible enough that we can approximate the first d principal functions for $d < k-1$ by simply setting the number of outputs of the CANN to d. In this way, we approximate the deviation Δ using fewer than the full set of $k-1$ principal functions. In the most extreme case, we could set $d = 1$ and therefore only compute the deviation based on the first pair of principal functions. These are also known as the *Rényi correlation functions* and are interesting in their own right [Rényi, 1959], but from our perspective the question is mainly one of computational efficiency. In general, a practitioner will want to set d to the smallest value that still obtains sufficiently high adversarial robustness. We set $d = k-1$ in all of our experiments, but in principle any value from 1 to $k-1$ could be used.

5.6 Comparison to other methods

Detector approaches appear to have largely fallen out of favor after the work of Carlini and Wagner [2017a], who showed that most detectors proposed in the past were not effective against adaptive attacks. As a result, the vast majority of new papers on adversarial robustness focus on hardening approaches. Indeed, most of the detector approaches we are aware of have either not been thoroughly evaluated yet or have been broken by adaptive attacks. There are a few exceptions, however. The *deep K-nearest neighbors* defense proposed by Papernot and McDaniel [2018] has been evaluated by Sitawarin and Wagner [2019] and found to be moderately robust against adaptive attacks. Ma et al. [2018] propose a different detector based on the *local intrinsic dimensionality* (LID) characteristic which also seems to work well. Lee et al. [2018] developed a very simple method based on Gaussian discriminant analysis which uses the Mahalanobis distance between features learned by the DNN.

Similar to our own detector, all of these methods can be applied to any pre-trained softmax classifier without the need to re-train the model. The deep K-nearest neighbor method is of particular interest to us, since it is also inspired by conformal prediction techniques and is therefore conceptually closest to our own detector. We are aware that there exist still other detector methods that appear promising, such as the work by Pang et al. [2018]. However, these usually seem to involve novel training procedures which would require re-training of the underlying model. The comparison with our method would therefore not be fair, since we wish to focus on "wrapper methods" that do not need to change the underlying model at all.

Mahalanobis distance-based confidence score. The method proposed by Lee et al. [2018] uses the following score to measure the confidence of a classification result:

$$M(x) = \max_y \; -(z_\theta(x) - \hat{\boldsymbol{\mu}}_y)^\intercal \hat{\boldsymbol{\Sigma}}^{-1} (z_\theta(x) - \hat{\boldsymbol{\mu}}_y).$$

Here, $z_\theta(x)$ denotes the output of the penultimate layer of the DNN (i.e., the logits). The vector $\hat{\boldsymbol{\mu}}_y$ is the sample mean of the logit vectors for all samples belonging to class y and the matrix $\hat{\boldsymbol{\Sigma}}$ is the sample covariance of the logits across all classes:

$$\hat{\boldsymbol{\mu}}_y = \frac{1}{N_y} \sum_{i:y_i=y} z_\theta(x_i),$$

$$\hat{\boldsymbol{\Sigma}} = \frac{1}{N} \sum_{y=1}^{k} \sum_{i:y_i=y} (z_\theta(x_i) - \hat{\boldsymbol{\mu}}_y)(z_\theta(x_i) - \hat{\boldsymbol{\mu}}_y)^\intercal.$$

Once the matrix $\hat{\boldsymbol{\Sigma}}$ and the vectors $\hat{\boldsymbol{\mu}}_1, \ldots, \hat{\boldsymbol{\mu}}_k$ have been computed, we can easily create a detector by thresholding $M(x)$ for new samples x. The results are given in table 5.5. Note that these are for defense-oblivious adversarials and so table 5.5 must be compared to table 5.3.

Data set	Network	AUROC	TRR	FRR	ACC
MNIST	SimpleCNN	64.68%	63.92%	42.97%	60.50%
	ResNet50	86.29%	87.99%	28.44%	79.76%
	DenseNet121	84.83%	85.39%	27.95%	78.77%
	Xception	96.64%	90.10%	8.57%	90.76%
Fashion-MNIST	SimpleCNN	79.10%	65.35%	23.81%	70.40%
	ResNet50	87.55%	77.01%	17.33%	79.65%
	DenseNet121	88.40%	85.13%	19.50%	83.02%
	Xception	94.86%	89.60%	1.52%	93.68%
CIFAR10	SimpleCNN	75.18%	63.07%	23.08%	68.52%
	ResNet50	55.64%	67.37%	55.42%	58.50%
	DenseNet121	61.11%	59.73%	42.49%	58.84%
	Xception	79.45%	68.32%	17.47%	73.71%
SVHN	SimpleCNN	57.56%	73.69%	58.78%	56.95%
	ResNet50	77.22%	70.40%	28.51%	70.94%
	DenseNet121	75.38%	65.02%	26.28%	69.41%
	Xception	68.96%	43.06%	12.00%	65.89%

Table 5.5: Performance metrics for the Mahalanobis distance-based detector.

Deep K-nearest neighbor. The deep K-nearest neighbor algorithm proposed by Papernot and McDaniel [2018] is another instantiation of the conformal prediction algorithm described in Shafer and Vovk [2008]. The non-conformity measure used by deep K-NN can be written as follows:

$$\mathcal{A}(D, (x, y)) = \sum_{l=1}^{L} \#\{\hat{y} \in \Omega_l(x) \mid \hat{y} \neq y\}.$$

Here, L is the number of layers of the neural network and $\Omega_l(x)$ is a multiset of labels associated with layer l on input x. This set is constructed by performing a k-nearest search on the output of layer l: for a test point x, we compute the output of layer l and collect the k training data points nearest to x in the space defined by layer l. The labels of these samples are then collected in the multiset $\Omega_l(x)$. Since the intermediate layers of a neural network can be high-dimensional, the locality-sensitive hashing (LSH) scheme FALCONN by Andoni et al. [2015] is used to reduce the dimensionality of the outputs before the nearest neighbor search is executed. Since deep K-NN is also a conformal predictor just like our method, it serves as an easy drop-in replacement in our experiments. The results are shown in table 5.6 for $K = 75$ as in Papernot and McDaniel [2018]. Again, these are for the defense-oblivious setting. It is important to note that the original deep K-NN algorithm requires the computation of Ω_l for *every* layer l, but this was not computationally feasible with our hardware. Even for relatively shallow networks, this method requires an exceedingly high amount of memory. Our compromise was to restrict the algorithm to the final eight layers of each network (excluding the softmax layer). We are aware that this must have skewed the comparison in our favor, but we believe the memory and time requirements of deep K-NN are rather excessive. As such, even if deep K-NN outperforms our method using a larger number of layers, our method still has the comparative advantage of requiring less memory and less computation time to achieve the same performance.

Data set	Network	AUROC	TRR	FRR	ACC
MNIST	SimpleCNN	92.68%	89.87%	17.77%	86.08%
	ResNet50	66.86%	86.16%	56.02%	65.03%
	DenseNet121	61.02%	43.93%	22.46%	60.63%
	Xception	45.83%	2.18%	1.82%	49.75%
Fashion-MNIST	SimpleCNN	77.28%	78.97%	34.44%	72.72%
	ResNet50	68.58%	76.87%	49.39%	64.61%
	DenseNet121	85.05%	82.89%	21.83%	80.73%
	Xception	48.92%	58.58%	56.45%	51.66%
CIFAR10	SimpleCNN	49.62%	77.04%	73.25%	57.27%
	ResNet50	48.74%	92.12%	86.17%	61.69%
	DenseNet121	44.14%	3.98%	2.53%	41.50%
	Xception	32.59%	3.50%	1.72%	39.45%
SVHN	SimpleCNN	75.81%	86.19%	43.21%	71.01%
	ResNet50	66.07%	73.78%	50.45%	61.76%
	DenseNet121	61.05%	43.21%	24.57%	59.48%
	Xception	48.24%	5.58%	2.94%	52.04%

Table 5.6: Performance metrics for the deep K-NN detector.

Discussion. The Mahalanobis distance-based detector does not appear competitive to our method for the SimpleCNN, ResNet50 and DenseNet121 models as it has a much higher FRR and significantly lower AUROC, TRR and ACC scores. The Xception network does not follow this trend, however: the Mahalanobis method using Xception appears to be competitive with our detector on all data sets except SVHN. This suggests that the latent space learned by the Xception network is more meaningful than that learned by the other networks. The Ma-

halanobis method may therefore be preferable in cases where the network has learned a very good representation of the data, although this depends on the specifics of both the data set as well as the network. The Mahalanobis detector overall tends to have much higher FRR than our method, however, so it seems to be a lot more conservative: it is much more likely than the CANN detector to reject a prediction that was actually correct. The deep K-NN detector, on the other hand, performs significantly worse than our method across almost all tests. It is not surprising that the performance of deep K-NN deteriorates as the models become deeper, since we only used a limited subset of layers due to computational constraints. This indicates a particular shortcoming of deep K-NN: it can only perform well when taking into account a large fraction of the layers. For deep architectures such as Xception, this makes deep K-NN very computationally demanding. We conclude that even if deep K-NN outperformed our method when taking the entire network into account, the CANN detector would still compare favorably because of its much lower computational overhead.

5.7 Conclusions

We have proposed a technique for detecting adversarial examples based on a combination of ideas from conformal prediction as well as correspondence analysis. Specifically, we derived a novel measure of non-conformity based on the theory of correspondence analysis which we then incorporated into a conformal predictor. The resulting algorithm appears to be a powerful and efficient method for detecting adversarial examples both in the defense-oblivious as well as the defense-aware settings. Aside from its effectiveness at detection, one of the main advantages of our detector is its model-independence: the parameters of the detector depend only on the specifics of the data set but do not require any information about the underlying model to be protected. As such, pre-trained detectors can be constructed for a variety of data sets, which can then be shared and further fine-tuned by other users.

Although the results presented here can be considered promising, the adaptive attack we evaluated against is just one of potentially many different evasion techniques that could be tried. Designing more effective attacks against our specific defense is therefore a primary avenue for future work. Aside from developing stronger attacks, we could also test how well our detector is capable of filtering out *universal adversarial perturbations* (UAPs) such as those crafted by Moosavi-Dezfooli et al. [2017]. A UAP is a single perturbation which can be added to any sample from a given data set and which will result in misclassification with high probability for any model trained on that data. These perturbations are particularly interesting to study due to the ease with which they can be abused in practice, since an attacker requires almost no knowledge of the machine learning system in order to use them effectively.

A larger-scale evaluation on more realistic data sets such as ImageNet [Deng et al., 2009] using state of the art classifiers would also be informative. In this work, we have evaluated against MNIST, Fashion-MNIST, CIFAR-10 and SVHN, which are standard benchmarks in the field of machine learning. However, they are also known to be relatively small data sets that are not representative of many real-world use cases. Moreover, since the CANN is parameterized by a neural network, there is still a lot of room for experimentation with regards to different architectures and methods of training which could vastly improve the results.

The frameworks of conformal prediction and correspondence analysis on which our approach relies are well-studied and have many appealing theoretical guarantees. Therefore, it may be

productive to try to prove formal robustness bounds for our defense in the L_p norm threat model similar to those provided by e.g. Fawzi et al. [2018], Hein and Andriushchenko [2017], Peck et al. [2017], Wong and Kolter [2018].

Chapter 6

Adversarial density filtering

> It's as though my dreams are a mirror of my waking world, like finding myself walking down the street where I could have sworn I caught a glimpse of you, only to look again and realize it wasn't you after all.
>
> *Sea of Strangers*
> Lang Leav

In chapters 3 to 5 we proposed several algorithms that allow for the *detection* of adversarial examples. Although detector methods can certainly be useful, their main drawback is the increased need for human supervision. Typically, when the detector flags an input as suspicious, there are only a few reasonable courses of action:

- Inform the user that their data cannot be processed at this time. Notify a human moderator and have them manually process the samples. This can be slow and requires manual intervention.
- Pass the sample to a more complicated model. This can be unreliable and requires an additional, more complex model to be deployed.

The second option of course runs the risk that the input may be rejected again (or unduly accepted). The first option is likely the most desirable in the majority of use-cases, but it involves potentially burdensome manual labor. In this chapter, we shift our focus to a *hardening* method that aims to make an existing black-box model intrinsically more robust, so that it is not fooled by adversarial examples at all. This obviates the need for any manual inspection and might be more useful in practice.

The contents of this chapter were not yet published elsewhere at the time of this writing.

6.1 Preliminaries

We consider the supervised learning setting where the goal is to find a function $F : \mathcal{X} \to \mathcal{Y}$ which maps inputs $x \in \mathcal{X}$ to appropriate outputs $y \in \mathcal{Y}$. The relationship between the inputs and outputs

may be stochastic, and so we model it as an unknown joint distribution on random variables X and Y with respective supports $\mathcal{X} \subseteq \mathbb{R}^n$ and $\mathcal{Y} = \{1, \ldots, k\}$. Estimation of F proceeds by parameterizing it using a set of parameters θ and solving the optimization problem

$$\theta^\star = \underset{\theta}{\operatorname{argmin}}\ \mathbb{E}[\ell(X, Y, \theta)]$$

where ℓ is an appropriate loss function. In practice, we approximate this objective based on a finite sample of data points $(x_1, y_1), \ldots, (x_m, y_m)$:

$$\hat{\theta} = \underset{\theta}{\operatorname{argmin}}\ \frac{1}{m} \sum_{i=1}^{m} \ell(x_i, y_i, \theta).$$

This technique is known as *empirical risk minimization* (ERM) [Shalev-Shwartz and Ben-David, 2014], since it is based on minimizing an empirical approximation of the *risk functional*

$$R(F_\theta) = \mathbb{E}[\ell(X, Y, \theta)].$$

Much work in adversarial machine learning casts the problem as one of *robust optimization* [Madry et al., 2017]. That is, we employ the standard ERM framework but with a modified *adversarial risk functional*

$$R_{\text{adv}}(F_\theta; \varepsilon) = \mathbb{E}\left[\sup_{\boldsymbol{\delta} \in \mathrm{B}(\varepsilon)} \ell(X + \boldsymbol{\delta}, Y, \theta)\right]$$

where ε is the perturbation budget and $\mathrm{B}(\varepsilon)$ is the ε-ball around the origin. To minimize the adversarial risk R_{adv}, we therefore need to minimize the expected *worst-case* loss under bounded additive perturbations of the input. From a theoretical point of view, we are interested in so-called *risk gaps* of the form

$$R_{\text{adv}}(F_\theta; \varepsilon) \leq R(F_\theta) + h(\varepsilon),$$

where the adversarial risk of a model F_θ can be bounded in terms of its standard risk $R(F_\theta)$ and some function $h(\varepsilon)$ of the perturbation budget. Such bounds have attracted much research interest, as they allow us to provide theoretical performance guarantees in the adversarial setting independently of any specific attack [Gourdeau et al., 2019, Pinot et al., 2021]. The method we propose in this work admits very favorable risk gaps, where $h(\varepsilon)$ grows sublinearly with the perturbation budget ε. Other works that prove theoretical risk gaps of this kind usually have at least linear or quadratic dependencies on the perturbation budget [Pinot et al., 2021].

Aside from accuracy, another important property of a machine learning model is its *calibration*. Intuitively, calibration measures the degree to which a model's predictions can be trusted. For example, a calibrated classifier should output the correct class for 80% of inputs on which it predicts said class with a probability of 80%. There exist different metrics for measuring calibration, but the most common one is *expected calibration error* (ECE) [Guo et al., 2017, Naeini et al., 2015], defined as follows:

$$\mathrm{ECE} = \mathbb{E}\left[\left|\hat{P} - \Pr[\hat{Y} = Y \mid \hat{P}]\right|\right].$$

Here, $\hat{Y}$ represents the estimated class of a random input sample X and $\hat{P}$ is the probability assigned by the classifier to $\hat{Y}$. Calibration is an additional interesting property for us to study here in the context of adversarial robustness, since prior work has shown that robust neural networks tend to be less well-calibrated than their non-robust counterparts [Wen et al., 2020].

In the sequel, we will also make extensive use of the notion of a *modulus of continuity*. Let (U, d_U) and (V, d_V) be two metric spaces. A function $\omega : \mathbb{R} \to \mathbb{R}$ is a modulus of continuity of another function $f : U \to V$ if it satisfies the following two properties:

1. Continuous and vanishing at zero:
$$\lim_{t \to 0} \omega(t) = \omega(0) = 0.$$
2. For every $x_1, x_2 \in U$,
$$d_V(f(x_1), f(x_2)) \leq \omega(d_U(x_1, x_2)).$$

Sets of functions sharing the same modulus of continuity form so-called *equicontinuous families*. For example, if $\omega(t) = Kt$ for some $K > 0$, we recover the class of Lipschitz continuous functions. Furthermore, a function is uniformly continuous if and only if it admits a modulus of continuity.

6.2 The Adversarial Density Filter (AdvDF)

We motivate our approach by considering a multivariate Gaussian density $q \sim \mathcal{N}(0, \boldsymbol{\Sigma}(x))$ with zero mean and covariance $\boldsymbol{\Sigma}(x) \in \mathbb{R}^{t \times t}$. The $t \times t$ covariance matrix is a function of an observation $x \in \mathbb{R}^n$, and hence the density q is conditioned on x. Fix such an observation x as well as some additive perturbation $\boldsymbol{\delta} \in \mathbb{R}^n$. Let us denote $q \sim \mathcal{N}(0, \boldsymbol{\Sigma}(x)) = \mathcal{N}(0, \boldsymbol{\Sigma})$ and $\tilde{q} \sim \mathcal{N}(0, \boldsymbol{\Sigma}(x + \boldsymbol{\delta})) = \mathcal{N}(0, \tilde{\boldsymbol{\Sigma}})$. As Gaussians with equal means, the KL divergence between $\tilde{q}$ and q has a simple analytical form. If, furthermore, the covariance matrices are diagonal with $\boldsymbol{\Sigma} = \operatorname{diag}(\sigma_1, \ldots, \sigma_t)$ and $\tilde{\boldsymbol{\Sigma}} = \operatorname{diag}(\tilde{\sigma}_1, \ldots, \tilde{\sigma}_t)$, then the KL divergence reduces to

$$\mathrm{D}_{\mathrm{KL}}(\tilde{q} \| q) = \frac{1}{2} \left(\sum_{i=1}^{t} \log \frac{\sigma_i}{\tilde{\sigma}_i} - t + \sum_{i=1}^{t} \frac{\tilde{\sigma}_i}{\sigma_i} \right).$$

Suppose that the maps $x \mapsto \sigma_i(x)$ are all uniformly continuous with local modulus of continuity ω_i and that $\sigma_i(x) > \omega_i(\|\boldsymbol{\delta}\|)$. Then,

$$\mathrm{D}_{\mathrm{KL}}(\tilde{q} \| q) \leq \frac{1}{2} \left(\sum_{i=1}^{t} \frac{\omega_i(\|\boldsymbol{\delta}\|)}{\sigma_i} - \sum_{i=1}^{t} \log \left(1 - \frac{\omega_i(\|\boldsymbol{\delta}\|)}{\sigma_i} \right) \right). \tag{6.1}$$

Hence, if σ_i is large with respect to $\omega_i(\|\boldsymbol{\delta}\|)$ for all i, then $\mathrm{D}_{\mathrm{KL}}(\tilde{q} \| q) \approx 0$ and the two densities correspond to almost the same distribution. In such a regime, small additive perturbations $\boldsymbol{\delta}$ to the observation x will not have much effect on the resulting conditional density q.

Our proposal, therefore, is to pass input samples $x \in \mathcal{X}$ through such a density in order to "filter out" adversarial perturbations that may have been added to the sample. To achieve this, we introduce the *adversarial density filter* (AdvDF) as a pre-processing step for the machine learning model to be protected. The AdvDF is essentially an autoencoder and hence consists of two main components:

Encoder. The *encoder* E_ϕ, a neural network parameterized by a vector of weights ϕ. It is an $\mathcal{X} \to \mathbb{R}^t_+$ function which maps input samples $x \in \mathcal{X}$ to variance vectors $E_\phi(x) \in \mathbb{R}^t_+$. The latent vectors $z \in \mathcal{Z}$ are created by sampling from the distribution

$$q_\phi(z \mid x) = \mathcal{N}(z \mid 0, \operatorname{diag} E_\phi(x)). \tag{6.2}$$

Decoder. The *decoder* D_ψ, a neural network parameterized by a weight vector ψ. It takes the latent vectors $z \in \mathcal{Z}$ from the encoder and maps them back into the input space $\mathcal{X}$.

In this way, input samples x are mapped to a Gaussian distribution, and the goal of the decoder is to reconstruct the original input x from samples of $q_\phi(z \mid x)$. The decoder therefore aims to minimize the expected reconstruction error $\mathbb{E}[\|X - D_\psi(Z)\|^2]$ where $Z \sim q_\phi$. On the other hand, the encoder aims to make q_ϕ resistant to small additive perturbations of the input. By the upper bound (6.1), this can be achieved by fulfilling the following requirements:

1. E_ϕ must be uniformly continuous
2. Each component of $E_\phi(x)$ must be large with respect to its modulus of continuity

The first constraint can be satisfied through architectural design choices in the specification of the encoder. The second constraint can be encouraged by regularizing the optimization process. Specifically, the parameters ϕ and ψ are optimized by minimizing the following loss:

$$\mathcal{L}(x, \phi, \psi) = \|x - D_\psi(z_\phi(x))\|^2 - \beta \sum_{i=1}^{t} \log E_{\phi,i}(x). \tag{6.3}$$

Here, we write $z_\phi(x)$ to represent a sample from the conditional distribution $q_\phi(\cdot \mid x)$. The parameter $\beta > 0$ balances the competing objectives of the encoder and decoder. Note that maximizing the sum of the log-variances in the Gaussian case is equivalent to maximizing its entropy. The reconstruction error can be bounded from below in terms of the entropy (see Cover [1999]), and so these objectives must indeed be appropriately balanced: if the entropy is too high, reconstruction will degrade and performance on downstream tasks will suffer; on the other hand, if the entropy is too low, there may not be any real increase in robustness and the AdvDF will be ineffective.

Curiously, we find that the regularization term in (6.3) is superfluous. We performed several experiments where we tuned β to achieve an optimal result, but we found no significant improvements in accuracy or robustness. Therefore, we set $\beta = 0$ in all our experiments, essentially disabling the regularization. This phenomenon is further investigated in section 6.3.2.

The theoretical properties of the AdvDF bring us to our main theorem:

Theorem 6.1. *The success rate of any adversary against the AdvDF scales like*

$$\sqrt{1 - \exp(-O(\omega(\varepsilon)^2))}$$

where ω is the modulus of continuity of the encoder network.

A more formal statement as well as a proof of this result is given in section 6.4.1. We have the following immediate corollary:

Corollary 6.2. *If the encoder network of the AdvDF is Lipschitz continuous, then the success rate of any adversary scales like $o(\varepsilon)$, i.e., sublinearly with the perturbation budget.*

We note that corollary 6.2 represents an asymptotic improvement of the dependency on the perturbation budget ε compared to related work [Pinot et al., 2021]: our bounds are *sublinear* in ε whereas others are at least linear (and usually even quadratic) in ε. Moreover, we have assumed only *uniform* continuity of the AdvDF encoder component, whereas most prior work

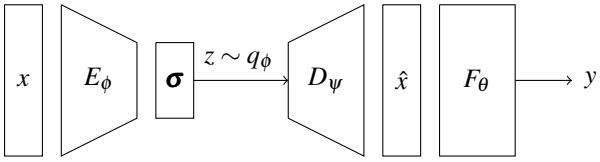


Figure 6.1: Graphical illustration of the AdvDF defense. The input sample x is first fed to the encoder network E_ϕ, which outputs a variance vector $\boldsymbol{\sigma}$. This variance vector is used to sample the latent representation $z \sim \mathcal{N}(0, \operatorname{diag} \boldsymbol{\sigma})$. The decoder takes this latent vector z and reconstructs the original input, obtaining an approximation $\hat{x}$. This reconstructed input is then fed to the model F_θ, yielding the output $y = F_\theta(\hat{x})$.

assumes *Lipschitz* continuity of the entire auto-encoder [Barrett et al., 2021] or even of the entire network [Fazlyab et al., 2019], which are much stricter requirements. In section 6.4.3, we provide a more detailed comparison of our work with that of Barrett et al. [2021], as their results are highly relevant to us as well.

We also wish to stress that the proofs of theorem 6.1 and corollary 6.2 work with *any* ℓ_p norm. Although we use ℓ_2 throughout our work to evaluate the defense in practice, the theoretical results suggest that the AdvDF should also work well for other norms, such as ℓ_∞ or ℓ_1. Furthermore, we find that similar results may also hold when the sampling distribution is generalized to the exponential family rather than restricted to a Gaussian. In section 6.4.4, we discuss such a generalization of the AdvDF to the exponential family of distributions.

The defense. Given a trained AdvDF (E_ϕ, D_ψ) and a model F_θ parameterized by θ, we defend the model by constructing the following composition:

$$G_\theta = F_\theta \circ D_\psi \circ z_\phi .$$

The function $z_\phi : \mathcal{X} \to \mathcal{Z}$ represents the sampling step where an input vector x is mapped onto a latent vector $z_\phi(x)$ by sampling from $q_\phi(\cdot \mid x) = \mathcal{N}(0, \operatorname{diag} E_\phi(x))$. These samples are then reconstructed by the decoder D_ψ and subsequently fed to the underlying model F_θ. The full model G_θ can then be trained end-to-end as usual, keeping the parameters ϕ and ψ of the AdvDF fixed. Figure 6.1 illustrates this construction graphically.

Note that the AdvDF is fully *unsupervised*: it does not require any labels in order to function. It therefore has the useful property of being amenable to a great variety of downstream tasks including classification, regression, clustering or generative modeling. As such, in our defensive strategy, the underlying model F_θ does not need to be a classifier, although we do restrict ourselves to that setting in this work. To the best of our knowledge, there exist very few adversarial defenses that satisfy this property, since most work in this area appears to be focused exclusively on supervised learning.

Finally, using the results from Pinot et al. [2021], we can also obtain the following risk gap for the AdvDF:

Theorem 6.3. *The adversarial risk of the AdvDF model $G_\theta = F_\theta \circ D_\psi \circ z_\phi$ satisfies*

$$R_{\text{adv}}(G_\theta; \varepsilon) \leq R(G_\theta) + \sqrt{1 - \exp(-O(\omega(\varepsilon)^2))}.$$

The proof of this statement can be found in section 6.4.2.

6.2.1 Computing the risk bounds

Given that the AdvDF has certified bounds on the adversarial risk, it would be interesting to compute these and compare them with empirical estimates of the robustness. Empirical estimates given by adversarial attacks give lower bounds on the adversarial risk, but these bounds can be very loose if the attack is not well-suited to the defense [Athalye et al., 2018a, Carlini and Wagner, 2017c]. To obtain a more reliable estimate of the true robustness of the method, we can compute the risk bound provided by theorem 6.3 numerically. To achieve this, we first recall the full explicit bound:

$$R_{\text{adv}}(G_\theta;\varepsilon) \leq R(G_\theta) + \underbrace{\mathbb{E}\left[\sqrt{1-\exp\left(-\frac{1}{2}\left(\sum_{i=1}^{t}\frac{\omega_i(\varepsilon)}{\sigma_i(X)} - \sum_{i=1}^{t}\log\left(1-\frac{\omega_i(\varepsilon)}{\sigma_i(X)}\right)\right)\right)}\right]}_{R_{\text{gap}}(G_\theta;\varepsilon)}.$$

Here, the expectation is taken over the data sample X. The risk gap $R_{\text{gap}}(G_\theta;\varepsilon)$ can be approximated using a sufficiently large data set:

$$\hat{R}_{\text{gap}}(G_\theta;\varepsilon) = \frac{1}{|S|}\sum_{x\in S}\sqrt{1-\exp\left(-\frac{1}{2}\left(\sum_{i=1}^{t}\frac{\omega_i(\varepsilon)}{\sigma_i(x)} - \sum_{i=1}^{t}\log\left(1-\frac{\omega_i(\varepsilon)}{\sigma_i(x)}\right)\right)\right)}. \tag{6.4}$$

As a result, the only difficulty in computing (6.4) are the moduli of continuity $\omega_1,\dots,\omega_t$ associated with the partial functions $\sigma_1(x),\dots,\sigma_t(x)$ that form the components of the variance vectors $\boldsymbol{\sigma}(x)$.

Let us consider for a moment an arbitrary uniformly continuous function $f:\mathbb{R}\to\mathbb{R}$. Our goal is to estimate its modulus of continuity ω at a given distance ε. By definition of uniform continuity, it holds that

$$\omega(\varepsilon) = \sup_{x,x'\in\mathbb{R}}\{|f(x)-f(x')| : |x-x'| = \varepsilon\}.$$

Equivalently, we can write

$$\omega(\varepsilon) = \sup_{x\in\mathbb{R}}\{|f(x)-f(x+\delta)| : |\delta| = \varepsilon\}. \tag{6.5}$$

Specializing (6.5) to our use-case, we have

$$\omega_i(\varepsilon) = \sup_{x\in\mathcal{X}}\{|\sigma_i(x)-\sigma_i(x+\boldsymbol{\delta})| : \|\boldsymbol{\delta}\| = \varepsilon\}. \tag{6.6}$$

We could therefore try to solve (6.6) for every $i=1,\dots,t$. Of course, this would quickly become inefficient for large t, so we instead rely on the following simple fact. Given moduli of continuity $\omega_1,\dots,\omega_t$ for $\sigma_1,\dots,\sigma_t$, the function

$$\omega(\varepsilon) = \max\{\omega_1(\varepsilon),\dots,\omega_t(\varepsilon)\}$$

is clearly a modulus of continuity for $\boldsymbol{\sigma}$ regardless of the norm used to measure distances. We can therefore simplify (6.6) to

$$\omega(\varepsilon) = \sup_{x\in\mathcal{X}}\{\|\boldsymbol{\sigma}(x)-\boldsymbol{\sigma}(x+\boldsymbol{\delta})\|_\infty : \|\boldsymbol{\delta}\| = \varepsilon\}. \tag{6.7}$$

The norm constraint on the perturbation $\boldsymbol{\delta}$ must be measured according to the norm used in the adversarial threat model, which is L_2 in our case. The distance between the variance vectors, on the other hand, is always measured in L_∞.

Note that we do not need to optimize over x and $\boldsymbol{\delta}$ simultaneously: the data sample x can be drawn from the data set and the optimization (6.7) can then be solved for each data sample individually. The final value $\omega(\varepsilon)$ is then computed by taking a maximum over these data-dependent results. Formally, we first solve

$$\omega(\varepsilon;x) = \max_{\boldsymbol{\delta}\in\mathbb{R}^n} \|\boldsymbol{\sigma}(x) - \boldsymbol{\sigma}(x+\boldsymbol{\delta})\|_\infty \text{ subject to } \|\boldsymbol{\delta}\| = \varepsilon \tag{6.8}$$

for a set of data samples $x \in S$. We then take

$$\omega(\varepsilon) = \max_{x\in S} \omega(\varepsilon;x).$$

This value can then be used in (6.4) to compute our empirical estimate $\hat{R}_{\text{gap}}(G_\theta;\varepsilon)$ of the adversarial risk gap.

6.2.2 Robustness evaluation

In this section, we detail the adversarial attacks used to evaluate the robustness of both our baseline (non-robust) models as well as the models defended using the AdvDF.

AutoAttack

At ICML 2020, Croce and Hein [2020b] proposed the *AutoAttack* framework, which is a software library that runs a diverse ensemble of adversarial attacks with optimized parameters to maximize their effectiveness. The code is available online,[1] and can be easily deployed to attack our models in the ℓ_2 setting. Concretely, AutoAttack runs an ensemble of four adversarial attacks against the given model and data, with no free parameters the user needs to specify. The following attacks are run sequentially against all models:

1. APGD-CE. A step-size free version of projected gradient descent (PGD) on the cross-entropy loss.
2. APGD-DLR. A step-size free version of PGD on the *Difference of Logits Ratio* (DLR) loss introduced by Croce and Hein [2020b]. The DLR loss eliminates a common problem with the cross-entropy loss where the scaling of the logits causes misleading estimates of model robustness. DLR is thus constructed to be invariant to the scale of the logits.
3. FAB. The *fast adaptive boundary* attack introduced by Croce and Hein [2020a].
4. Square. A query-efficient black-box attack proposed by Andriushchenko et al. [2020].

Since our defense is randomized, we run AutoAttack combined with Expectation over Transformation [Athalye et al., 2018a], a feature that the library supports natively. This further eliminates the possibility that the increase in robustness could be due to gradient masking.

Note that AutoAttack is not an *adaptive* robustness evaluation, *i.e.*, the adversarial attacks have no specific knowledge of the AdvDF except that it is randomized. Although AutoAttack is an

[1] `https://github.com/fra31/auto-attack`. Accessed 2021-09-03.

excellent baseline for an adversarial defense, a key component of a good robustness evaluation is an *adaptive attack*, *i.e.*, an adversarial attack that is specifically tailored to the proposed defense [Carlini et al., 2019, Tramer et al., 2020]. We therefore also develop three adaptive attacks, which are detailed below, to complement the defense-oblivious AutoAttack.

Maximum damage attack

Our first idea for an adaptive attack against the AdvDF is to attempt to find small perturbations $\boldsymbol{\delta}$ in the input which cause large deviations in the reconstructions produced by the decoder. This naturally leads us to the *maximum damage attack* originally proposed by Camuto et al. [2021] to assess the robustness of variational auto-encoders (VAEs):

$$\max_{\boldsymbol{\delta}\in\mathbb{R}^n} d(D_\psi(z_\phi(x+\boldsymbol{\delta})), D_\psi(z_\phi(x))) \text{ subject to } d(x+\boldsymbol{\delta}, x) \leq \varepsilon.$$

We maintain a strict norm constraint on the perturbation $\boldsymbol{\delta}$. This attack was shown to be highly effective against VAEs and is therefore a logical candidate for our defense as well, given that the AdvDF is *architecturally*[2] nothing more than a VAE where the mean vectors are always set to zero.

Maximum variance attack

In order to accurately reconstruct an input sample x from a sample z of its latent distribution $\mathcal{N}(0, \operatorname{diag} E_\phi(x))$, the decoder can only use information encoded in the variance vector $E_\phi(x)$. Hence, similarly to the maximum damage attack, a natural idea is to attack the encoder network E_ϕ so that it produces a very different variance vector. More specifically, in order to generate an adversarial $\tilde{x}$ for a given input x, we solve the following optimization problem:

$$\max_{\boldsymbol{\delta}\in\mathbb{R}^n} d(E_\phi(x+\boldsymbol{\delta}), E_\phi(x)) \text{ subject to } d(x+\boldsymbol{\delta}, x) \leq \varepsilon. \tag{6.9}$$

That is, we attempt to find a small additive perturbation $\boldsymbol{\delta}$ such that the variance $E_\phi(x+\boldsymbol{\delta})$ is far from the original variance $E_\phi(x)$ while maintaining a strict norm constraint on $\boldsymbol{\delta}$. If the metric $d(\cdot,\cdot)$ is the Euclidean distance, the objective (6.9) is fully differentiable and can be minimized iteratively using gradient descent.

Latent space attack

To achieve robustness, the AdvDF crucially relies on a small KL divergence $\mathrm{D}_{\mathrm{KL}}(\tilde{q}_\phi \| q_\phi)$ between the latent distributions of the corrupted samples and the clean samples. Hence, another method of attack specifically tailored to this defense could proceed as follows. We start from an original sample x_o that we wish to perturb and a target sample x_t from a different class. Then, we optimize a perturbation $\boldsymbol{\delta}$ such that the KL divergence between the latent distribution associated with $x_o + \boldsymbol{\delta}$ and the distribution associated with x_t is minimized. This is the idea behind the *latent space attack* for VAEs [Gondim-Ribeiro et al., 2018, Kos et al., 2018, Tabacof et al., 2016], which attempts to solve the following optimization problem:

$$\min_{\boldsymbol{\delta}\in\mathbb{R}^n} \mathrm{D}_{\mathrm{KL}}\left(\tilde{q}_\phi \middle\| q_\phi^{(t)}\right) \text{ subject to } d(x_o+\boldsymbol{\delta}, x_o) \leq \varepsilon.$$

[2] Aside from the architecture, the AdvDF is trained differently from the VAE: we minimize the ℓ_2 reconstruction loss instead of maximizing the evidence lower bound [Kingma and Welling, 2013, Zhang et al., 2019a], and the AdvDF is additionally regularized to maximize the variance components.

Here, $\tilde{q}_\phi$ refers to the latent distribution associated with the perturbed sample $x_o + \boldsymbol{\delta}$ and $q_\phi^{(t)}$ refers to the distribution associated with the target sample x_t.

Implementation details. For all adaptive attacks, we used 1,000 iterations of gradient descent in the optimization loops with the Adam optimizer [Kingma and Ba, 2014].

6.2.3 Black-box attack

The adaptive attacks we described in section 6.2.2 are all *gradient-based*, which means they rely upon the gradients of the model in order to perform a successful attack. As shown by Athalye et al. [2018a], gradient-based attacks can fail for various reasons, collectively called "gradient masking." To ensure that our model remains robust even against attacks that do not suffer from gradient masking, we also implement a gradient-free approach inspired by the Simba attack[3] from Guo et al. [2019]. The attack functions in the ℓ_0 regime, that is, it attempts to minimize the *number of pixels* perturbed instead of the global perturbation magnitude. Pseudo-code is given in algorithm 6.1.

Algorithm 6.1: Pseudo-code for the black-box attack

Data: model F, sample $x \in \mathbb{R}^n$, budget $\varepsilon \in (0,1)$, iterations N, pixel bounds (L,U)
Result: sample $\tilde{x} \in \mathbb{R}^n$

foreach *iteration in* $\{1,\ldots,N\}$ **do**
  Sample a random fraction of pixels I of x with $|I| = \lceil n\varepsilon \rceil$ (without replacement).
  Initialize $\tilde{x} \leftarrow x$.
  foreach *pixel* $i \in I$ **do**
    if $x_i \leq (L+U)/2$ **then**
      $\tilde{x}_i \leftarrow U$
    else
      $\tilde{x}_i \leftarrow L$
    end
  end
  if $F(x) \neq F(\tilde{x})$ **then**
    return $\tilde{x}$
  end
end
return x

The attack is instantiated with a perturbation budget $\varepsilon \in (0,1)$, which specifies the fraction of pixels we are allowed to perturb, as well as a number of iterations N. For instance, on a data set like MNIST where the input samples are 28×28 grayscale images LeCun et al. [1998], $\varepsilon = 1\%$ would correspond to an ℓ_0 budget of eight pixels (rounded up). The magnitude of the perturbation applied to these pixels does not matter; only that they are modified with respect to the original sample. For each input sample x, the attack chooses a fraction ε of the pixels from x at random and changes their values to either the minimum or maximum value observed in the data set, whichever causes the largest deviation. If the classification changes as a result of these

[3] In fact, our attack is equivalent to Simba using the standard basis together with our slightly different procedure for selecting the pixel-wise perturbations.

modifications, the modified sample is returned as the adversarial. Otherwise, this procedure is repeated with fresh random pixels until a successful adversarial is found or the number of iterations N is exhausted.

6.3 Experiments

In this section, we empirically evaluate the usefulness of the AdvDF for robust machine learning. We utilize various adversarial attacks for this purpose and vary their perturbation budgets to show the deterioration in accuracy of the baselines compared to the AdvDF in section 6.3.2. The experiments were carried out using the TensorFlow Keras framework [Abadi et al., 2015, Chollet et al., 2015] along with the Foolbox and ART libraries [Nicolae et al., 2018, Rauber et al., 2020]. We used one NVIDIA Titan Xp GPU and two NVIDIA GeForce RTX 2080 Ti GPUs to run all experiments.

Gradient masking. To prevent gradient masking, we use the Backwards Pass Differentiable Approximation (BPDA) technique by Athalye et al. [2018a] for all adversarial attacks. That is, we replace the AdvDF component $D_\psi \circ z_\phi$ by the identity in the backward pass. In essence, this amounts to approximating the gradient of G_θ at $\hat{x}$ using $\nabla_x F_\theta(x) \restriction_{x=D_\psi(z_\phi(\hat{x}))}$. The forward pass remains unchanged. This is the recommended approach for defenses like ours, which belong to the class of "straight-through estimators."

6.3.1 Data sets and models

We perform all our experiments on the four image classification data sets detailed below. For each data set, we utilize a particular model architecture and set of hyperparameters for the AdvDF. The influence of these hyperparameters is further elaborated on in section 6.3.2. All data sets were made available in the TensorFlow Datasets catalog [4].

Fashion-MNIST. The Fashion-MNIST data set [Xiao et al., 2017] consists of images from the Zalando catalog of clothing articles. The training set contains 60,000 examples and the test set contains 10,000 examples. Each example is a 28×28 grayscale image belonging to one of ten possible classes. The latent dimensionality t of the AdvDF is set to 128 for this data set.

CIFAR-10. The CIFAR-10 data set [Krizhevsky and Hinton, 2009] consists of 60,000 RGB images 32×32 pixels in size. The samples are divided into 10 classes, with 6,000 images per class. We set $t = 2048$ here. The model architecture for this data set was based on ResNet [He et al., 2016].

Cats vs dogs. The cats vs dogs data set [Elson et al., 2007] contains RGB images of cats and dogs in various sizes. To streamline the processing of the samples, we resize them to 64×64 pixels. We let $t = 8192$.

[4] `https://www.tensorflow.org/datasets`. Accessed 2021-09-07.

Rock, paper, scissors (RPS). The rock, paper, scissors data set [Moroney, 2019] consists of 300×300 RGB images of human hands playing the rock, paper, scissors game. The goal is to classify the pose of each hand into one of the three classes: rock, paper or scissors. For our purposes, we resize the images to 80×80 pixels. We let $t = 5184$.

For each data set, all samples are normalized so the pixel values lie in the range $[0, 1]$. We must also remark that the DLR loss as the authors proposed it in Croce and Hein [2020b] is undefined for classification problems that are comprised of fewer than four classes. We therefore excluded this attack from the evaluation when running AutoAttack on the Cats vs dogs and Rock, paper, scissors data sets; we include it for CIFAR-10 and Fashion-MNIST.

6.3.2 Results

To evaluate the robust accuracy, we subject G_θ to several adversarial attacks. First, we run the AutoAttack benchmark [Croce and Hein, 2020b] which consists of an ensemble of diverse adversarial attacks. Then, we consider the following non-adaptive gradient-based attacks which are not included in the AutoAttack evaluation: the fast gradient method (FGM) [Goodfellow et al., 2014], the projected gradient descent attack (PGD) [Madry et al., 2017] and the Brendel-Bethge attack [Brendel et al., 2019]. We also test the AdvDF against our own adaptive gradient-based attacks detailed in section 6.2.2. For each of these attacks, we utilize the ℓ_2 norm to measure the perturbation budget. Finally, we assess the performance of the AdvDF against the gradient-free black-box attack described in section 6.2.3. Note that this attack measures the perturbation budget in terms of the *number* of perturbed pixels and ignores the magnitude of the global perturbation. In this case, the budget ε represents the fraction of pixels perturbed. We also compute the relative squared error (RSE) values for the reconstructed samples. For a given sample x with reconstruction $\hat{x} = D_\psi(z_\phi(x))$, the RSE is computed as $\mathrm{RSE}(x, \hat{x}) = \|x - \hat{x}\|_2^2 / \|x\|_2^2$. The AdvDF is compared against the randomized smoothing defense [Awasthi et al., 2020, Cohen et al., 2019, Salman et al., 2020b] using the implementation provided by the Adversarial Robustness Toolbox (ART) [Nicolae et al., 2018].

Data set	Baseline	AdvDF	RSE
Fashion-MNIST	89.40 ± 0.13	88.47 ± 0.12	1.20 ± 0.05
CIFAR-10	85.61 ± 0.14	74.67 ± 0.31	1.70 ± 0.04
Cats vs dogs	88.60 ± 0.28	82.08 ± 0.42	2.41 ± 0.02
RPS	91.91 ± 3.36	94.19 ± 1.10	0.97 ± 0.04

Table 6.1: Accuracy and relative squared error (RSE) of the different models.

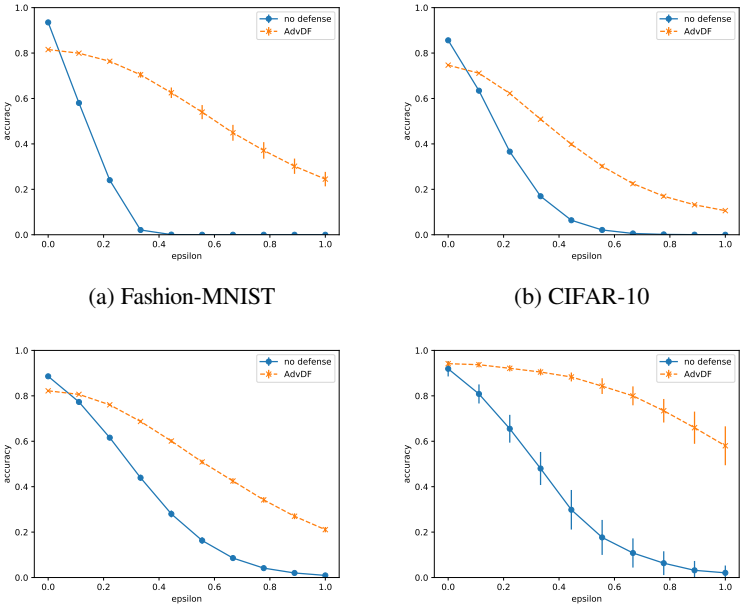
Table 6.1 reports the accuracy and RSE scores of the baseline and AdvDF models. The reported results are the means of ten different runs of the experiments with the standard errors indicated. We observe that the AdvDF often incurs a drop in clean test accuracy. This is typical of adversarial defenses in practice, especially given that all models were trained under identical circumstances (using the same optimizer, number of epochs, etc). This phenomenon is known as the accuracy-robustness trade-off, and there is some theoretical evidence that it cannot be avoided in general [Gourdeau et al., 2019, Schmidt et al., 2018]. However, when plotting the robustness curves (figures 6.2 and 6.3 in the main text and figures 6.5 to 6.11 in section 6.5), we see that the robust accuracy of the AdvDF seems to degrade much more slowly than the base-

lines within the perturbation budgets we tested. In fact, in most cases, the robust accuracy of the AdvDF is significantly higher than the baselines and randomized smoothing defenses for certain intervals of the perturbation budget. A more extensive comparison to randomized smoothing is given in section 6.3.3.

The RSE also appears to be very low across all data sets. For a more qualitative assessment, we give examples of randomly selected reconstructions in section 6.5. There, it can indeed be observed that the reconstructions tend to resemble blurred versions of the original samples, as is common for autoencoders trained with an ℓ_2 loss.

Risk gap. Unfortunately, when we compute the adversarial risk gap $\hat{R}_{\text{gap}}$ for the models in table 6.1, the estimates of the moduli of continuity given by (6.7) are very large and cause the resulting risk bounds to become vacuous (*i.e.*, greater than 100%). This is similar to the problem faced by classical generalization bounds when applied to deep neural networks [Zhang et al., 2021a]. The main quantities used to bound the generalization gap, such as the Vapnik-Chervonenkis dimension or the fat shattering dimension, become too large to obtain meaningful bounds. In statistical learning theory, the aim is typically to control some notion of "representational power" of the network. The modulus of continuity is similar in that regard since it measures how much the neural network can "stretch" or "compress" its input space. The optimization problem (6.8) essentially involves computing adversarial examples for the encoder network $\boldsymbol{\sigma}$, which explains why it can be so successful if special care is not taken. In particular, one must ensure that the modulus of continuity of the encoder is small, a condition that does not appear to have been studied in great detail before. There is an increasing amount of related work on *Lipschitz* regularisation [Gouk et al., 2021], but this is a much stricter constraint than the uniform continuity that we require. Moreover, we only need to constrain the *encoder* network; the decoder component and the downstream classifier are not constrained at all, a rather remarkable property which may significantly reduce the overhead of the AdvDF compared to other approaches which must constrain the entire network end-to-end. That said, it may come as a surprise that our AdvDF constructions are still able to provide significant robustness despite the large moduli. The main objective of our future work in this area is to find ways to effectively constrain the uniform continuity and obtain non-vacuous risk gaps.

Sanity check. As an additional sanity check for our defense, we verify that the AdvDF does not provide any significant robustness on a task where this is provably impossible. Specifically, we take the binary classification problem constructed by Tsipras et al. [2019] and train both a baseline and AdvDF model. We mimic the CIFAR-10 data set in our setup: the data dimensionality is $32 \times 32 \times 3$ and we generate 60k samples for training and 10k samples for testing. We let $p = 0.2$ and $\eta = 3/\sqrt{3071}$. This way, standard classification should achieve very high accuracy (over 90%) but robust classification against an ℓ_∞-bounded adversary should not perform better than random guessing for $\varepsilon = 2\eta \approx 0.11$. We evaluate the models against the ℓ_∞ FGM attack, since this method is most susceptible to gradient masking. The results confirm our expectations: the baseline and AdvDF models achieve $98.29\% \pm 0.20\%$ and $92.20\% \pm 1.53\%$ standard accuracy respectively (averaged across ten different runs) whereas their corresponding robust accuracy is $2.77\% \pm 0.28\%$ and $21.16\% \pm 2.00\%$. These figures are significantly worse than the level of random guessing, which is 50% in this case.



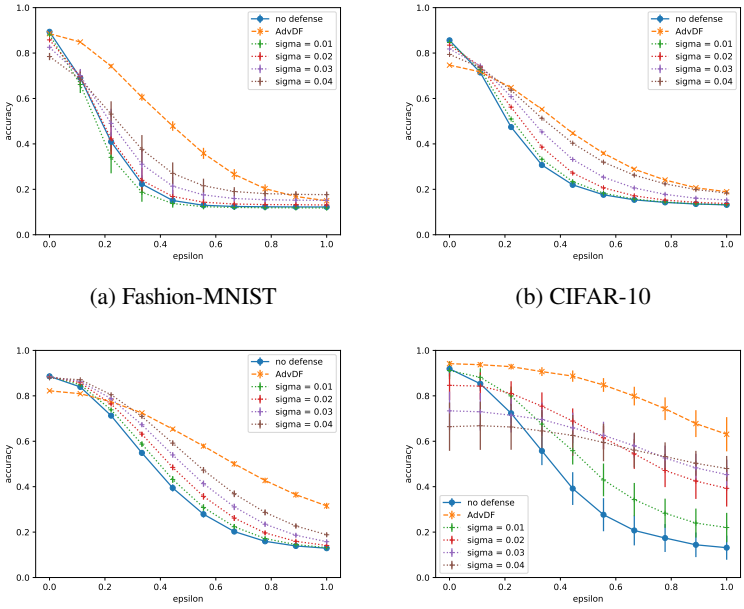
(a) Fashion-MNIST (b) CIFAR-10

(c) Cats vs dogs (d) RPS

Figure 6.2: Robustness curves for the AutoAttack benchmark on the different models.

Plausibility of the theoretical assumptions. Theorem 6.1 relies on several theoretical assumptions about the AdvDF networks. Specifically, we require that the network be uniformly continuous and that the components of the variance vectors be large with respect to the modulus of continuity. In section 6.5.5, we verify these assumptions by studying the magnitude of components of the variance vectors and plotting the KL divergence of the latent densities when subjected to adversarial attacks. We confirm that the variances are large and that the KL divergence appears to scale quadratically with the perturbation budget ε. These results lend empirical support to theorem 6.1 and corollary 6.2.

Hyperparameters. In section 6.5.6, we experiment with varying the latent dimensionality t of the AdvDF to study its effects on standard and robust accuracy. We find that a larger t increases both standard accuracy and robust accuracy up to a certain point, after which standard accuracy levels off but robust accuracy tends to decrease. This suggests that there exists an optimal "filter bottleneck" where the AdvDF removes just enough information so that the downstream model can be both robust and accurate. Below this level, we erase too much information and the model is no longer accurate; above it, we retain too much information and the model becomes fragile. These findings are in line with the theory of robustly useful features proposed by Ilyas et al. [2019]: for a well-chosen latent dimensionality t, the AdvDF appears to make it harder to learn non-robust features of the input, forcing the downstream model to rely exclusively on robustly useful features.



(a) Fashion-MNIST

(b) CIFAR-10

(c) Cats vs dogs

(d) RPS

Figure 6.3: Comparison of the baselines to the AdvDF and randomized smoothing defenses for the ℓ_2 PGD attack.

Calibration. In section 6.5.3, we plot ECE values of the baseline, AdvDF and randomized smoothing methods for different perturbation budgets against the ℓ_2 PGD attack. We observe that the AdvDF tends to have a significantly lower ECE for small perturbation budgets (the common scenario). For larger budgets, depending on the data set, randomized smoothing may be better calibrated than the AdvDF.

Efficiency. Aside from increased robustness, *computational efficiency* is another strength of the AdvDF. Timing experiments in figure 6.17 show that inference using the AdvDF is on average 31x faster than randomized smoothing for the tasks we considered. For training, we do require that the AdvDF be trained first, independently of the downstream model. Then, the downstream model must be retrained (or at least finetuned) using the fixed AdvDF. However, we believe that the overhead incurred during training will often be offset by the improved robustness and fast inference times.

Robustness to natural corruptions. Aside from adversarial perturbations, which are *worst-case* corruptions of the input, deep neural networks are also expected to be robust to *natural* corruptions. Natural corruptions can be very different from adversarial ones; indeed, prior work has identified that robustness to adversarial perturbations often comes at the cost of robustness against other distortions [Taori et al., 2020, Yin et al., 2019b]. To assess the generalization of the AdvDF to naturally occurring perturbations, we evaluate our CIFAR-10 models on the CIFAR-10-C data set proposed by Hendrycks and Dietterich [2019]. The CIFAR-10-C data set

consists of samples from the CIFAR-10 test set which have been subjected to various types of noise and distortions of varying severity levels. We discuss these experiments in more detail in section 6.5.4. To summarize, for 14 out of 19 corruption types considered in the CIFAR-10-C data set, we find that the AdvDF is significantly more robust than the baselines. By the standards of a defense against adversarial perturbations, the AdvDF exhibits a remarkable resilience to natural corruptions as well.

6.3.3 Comparison to randomized smoothing

At the time of this writing, the state of the art in certified adversarial defenses is randomized smoothing [Awasthi et al., 2020, Cohen et al., 2019, Salman et al., 2020b]. It is therefore interesting to contrast theorem 6.1 with the theoretical guarantee given by the randomized smoothing method [Cohen et al., 2019]. Given a variance σ^2 and a sample x, randomized smoothing is provably robust within the ℓ_2-radius

$$\rho(x) = \frac{\sigma}{2}\left(\Phi^{-1}(p_1) - \Phi^{-1}(p_2)\right),$$

where Φ^{-1} is the inverse Gaussian CDF and p_1, p_2 are the top-2 prediction probabilities for x. Note that the guarantees of randomized smoothing do not naturally generalize to other ℓ_p norms; this requires some additional work [Awasthi et al., 2020]. Furthermore, although $\rho(x)$ is easy to compute in practice, it is not clear how this function behaves exactly with varying σ, since p_1 and p_2 are computed using the randomized smoothing model with variance σ^2 on the input x. These probabilities may therefore exhibit highly non-linear behavior and they will lead to a trivial robustness radius when $p_1 \approx p_2$. By contrast, our guarantee does not directly depend on the probability margin; instead, it depends on the standard accuracy of the model as well as the continuity properties of the encoder network.

It is important to keep in mind that the nature of our theoretical guarantee is fundamentally different from that of randomized smoothing. In a sense, randomized smoothing yields a "hard" guarantee: the model prediction will remain constant within the *entire* ball of radius $\rho(x)$ around x. On the other hand, our guarantee is "soft" in the sense that it bounds the probability that an adversary will succeed in attacking the model using samples drawn from the original data distribution. It is entirely possible that a model protected by the AdvDF still has many adversarial examples. However, we claim that the probability of drawing a sample x from the data distribution that can be successfully perturbed into an adversarial $\tilde{x} = x + \boldsymbol{\delta}$ such that $\|\boldsymbol{\delta}\| \leq \varepsilon$ and $G_\theta(x) \neq G_\theta(\tilde{x})$ grows very slowly with ε. This means that, although strong adversarial examples likely still exist for the AdvDF, they will be *hard to find*. This is conceptually similar to the defense of Madry et al. [2017], which also does not give any "hard" robustness claims but rather relies on an argument that finding adversarials for the defended model becomes computationally prohibitive.

The AdvDF is clearly not *universally* better than randomized smoothing for every data set, model and adversarial attack. For some combinations of model, data set, adversarial attack and perturbation budget, the AdvDF can perform significantly worse and randomized smoothing may be the better option. However, we do see that the robust accuracy of the AdvDF tends to decrease more slowly with the perturbation budget than the robust accuracy of randomized smoothing, and that there are certainly cases when the AdvDF is much more robust. Furthermore, the AdvDF incurs almost no overhead during inference, whereas randomized smoothing can slow

AdvDF	Randomized smoothing
+ Fast inference times	- Slow inference times
+ Clear sublinear dependence on ε	- No clear analytical dependence on ε
+ Amenable to unsupervised learning	- Only used for supervised learning (*)
- "Soft" robustness guarantee	+ "Hard" robustness guarantee
- Need to retrain the downstream model	+ No need for retraining
- Need to specify an additional model	+ Only need to specify σ

Table 6.2: Comparison of the relative advantages and disadvantages of the AdvDF and randomized smoothing methods. (*) To the best of our knowledge

down the inference times by two orders of magnitude. Hence, the choice between our method and randomized smoothing is not clear-cut but depends on the specific design considerations of the concrete machine learning system in which it is to be deployed. We list some advantages and disadvantages of each approach in table 6.2. This list is not necessarily exhaustive, and of course it does not take into account the precise performance characteristics of the methods for a given machine learning problem.

6.4 Theoretical results

In this appendix, we discuss our main theoretical results in more detail. Section 6.4.1 presents our proof of theorem 6.1 and corollary 6.2. We also discuss the relationship to the closely related work of Pinot et al. [2021] in section 6.4.2. Finally, in section 6.4.4, we study a possible generalization of the AdvDF to the case where the encoder samples from a distribution within the exponential family.

6.4.1 Proof of theorem 6.1 and corollary 6.2

In this section, we present the proof of our main result (theorem 6.1). First, we introduce some notation. Let Y_θ denote the output of the model on a random sample from the data distribution. Similarly, $\tilde{Y}_\theta$ denotes the output of the model on a corrupted sample from the data distribution. Specifically, if X is a random variable representing a sample from the data distribution and if G_θ is the classifier (protected using the AdvDF), then $Y_\theta = G_\theta(X)$ and $\tilde{Y}_\theta = G_\theta(X + \boldsymbol{\delta})$. Here, $\boldsymbol{\delta} \in \mathbb{R}^n$ is only required to satisfy $\|\boldsymbol{\delta}\| \leq \varepsilon$ for some $\varepsilon \geq 0$; it is otherwise unconstrained. In particular, it can (and usually will) depend on X. We let Y denote the true label of the data distribution.

We are most interested in $\Pr[\tilde{Y}_\theta \neq Y \mid Y_\theta = Y]$, that is, the probability that an adversary can take a sample that was initially classified correctly and transform it into an incorrectly classified sample using a bounded additive perturbation.

The data processing inequality (DPI) [Cover, 1999] immediately yields[5]

$$\mathrm{D}_{\mathrm{KL}}(\tilde{Y}_\theta \| Y_\theta) \leq \mathrm{D}_{\mathrm{KL}}(\tilde{q}_\phi \| q_\phi).$$

[5]Note that this divergence is defined here for Y_θ and $\tilde{Y}_\theta$ *conditioned on* $X = x$. We will also use $\delta(\tilde{Y}_\theta, Y_\theta)$ for the total variation distance between $\tilde{Y}_\theta$ and Y_θ conditioned on $X = x$ and similarly for $\mathrm{D}_{\mathrm{KL}}(\tilde{q}_\phi \| q_\phi)$. For the sake of brevity, we prefer to omit the explicit conditioning in this notation since we will never consider the KL divergence or the total variation distance between the marginal distributions.

We note that, in the case of classification, the random variables Y_θ and $\tilde{Y}_\theta$ follow categorical distributions conditioned on $X = x$:

$$Y_\theta \sim \text{Cat}(p_1(x), \ldots, p_k(x)), \qquad \tilde{Y}_\theta \sim \text{Cat}(\tilde{p}_1(x), \ldots, \tilde{p}_k(x)).$$

As such,[6]

$$\Pr[Y_\theta \neq \tilde{Y}_\theta \mid X = x] = 1 - \Pr[Y_\theta = \tilde{Y}_\theta \mid X = x] = 1 - \sum_{i=1}^{k} p_i(x)\tilde{p}_i(x).$$

The total variation distance is

$$\delta(\tilde{Y}_\theta, Y_\theta) = \max_i \ |\tilde{p}_i(x) - p_i(x)|.$$

This implies $\tilde{p}_i(x) \geq p_i(x) - \delta(\tilde{Y}_\theta, Y_\theta)$ and so

$$\Pr[Y_\theta \neq \tilde{Y}_\theta \mid X = x] \leq 1 - \sum_{i=1}^{k} p_i(x)(p_i(x) - \delta(\tilde{Y}_\theta, Y_\theta)).$$

The Bretagnolle-Huber inequality [Bretagnolle and Huber, 1979] states that

$$\delta(\tilde{Y}_\theta, Y_\theta) \leq \sqrt{1 - \exp(-\text{D}_{\text{KL}}(\tilde{Y}_\theta \| Y_\theta))}.$$

Therefore,

$$\Pr[Y_\theta \neq \tilde{Y}_\theta \mid X = x] \leq 1 - \sum_{i=1}^{k} p_i(x) \left(p_i(x) - \sqrt{1 - \exp(-\text{D}_{\text{KL}}(\tilde{Y}_\theta \| Y_\theta))} \right).$$

Using the DPI, we obtain

$$\Pr[Y_\theta \neq \tilde{Y}_\theta \mid X = x] \leq 1 - \sum_{i=1}^{k} p_i(x) \left(p_i(x) - \sqrt{1 - \exp(-\text{D}_{\text{KL}}(\tilde{q}_\phi \| q_\phi))} \right).$$

In the special case where the model outputs a one-hot encoding of the label, this bound simplifies to

$$\Pr[Y_\theta \neq \tilde{Y}_\theta \mid X = x] \leq \sqrt{1 - \exp(-\text{D}_{\text{KL}}(\tilde{q}_\phi \| q_\phi))}.$$

Contrary to the more complicated bound that includes all probabilities p_i, this one is never vacuous. Furthermore, this setting is closer to typical practice in machine learning where the label is chosen deterministically (usually via an argmax over the logits) rather than probabilistically (via sampling from the softmax probabilities).

Now,

$$\begin{aligned} \Pr[\tilde{Y}_\theta \neq Y \mid X = x, Y_\theta = Y] &= \frac{\Pr[\tilde{Y}_\theta \neq Y, Y_\theta = Y \mid X = x]}{\Pr[Y_\theta = Y \mid X = x]} \\ &\leq \frac{\Pr[\tilde{Y}_\theta \neq Y_\theta \mid X = x]}{\Pr[Y_\theta = Y \mid X = x]}. \end{aligned}$$

[6] We made use here of the fact that Y_θ and $\tilde{Y}_\theta$ are conditionally independent given X, since we have the fork $Y_\theta \leftarrow X \rightarrow \tilde{Y}_\theta$.

If everything remains fixed except the perturbation budget ε, then from (6.1) we obtain

$$\mathrm{D}_{\mathrm{KL}}(\tilde{q}_\phi \| q_\phi) = O(\omega(\varepsilon)^2)$$

where ω is the modulus of continuity of the encoder. Hence

$$\Pr[\tilde{Y}_\theta \neq Y \mid Y_\theta = Y, X = x] = \sqrt{1 - \exp(-O(\omega(\varepsilon)^2))}.$$

Note that this bound holds for any $x \in \mathcal{X}$. To get rid of the conditioning on X, we make use of a simple auxiliary lemma:

Lemma 6.4. *Let $f : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ and $g : \mathbb{R} \to \mathbb{R}$ be functions. Let U be a random variable. If $f(u,v) = O(g(v))$ for every $u \in \mathbb{R}$, then $\mathbb{E}[f(U,v)] = O(g(v))$.*

Proof. We consider the full expression of the expectation:

$$\mathbb{E}[f(U,v)] = \int_{\mathbb{R}} p_U(u) f(u,v) \mathrm{d}u.$$

Since $f(u,v) = O(g(v))$, it holds that

$$\exists k > 0 : \exists v_0 \in \mathbb{R} : \forall v > v_0 : |f(u,v)| < k g(v).$$

Fix an appropriate $k > 0$ and $v > v_0$. Then

$$\mathbb{E}[f(U,v)] \leq \mathbb{E}[|f(U,v)|] = \int_{\mathbb{R}} p_U(u) |f(u,v)| \mathrm{d}u < \int_{\mathbb{R}} p_U(u) k g(v) \mathrm{d}u = k g(v).$$

We conclude that the function $v \mapsto \mathbb{E}[f(U,v)]$ also satisfies $O(g(v))$. □

Marginalizing over X and using lemma 6.4 therefore yields

$$\Pr[\tilde{Y}_\theta \neq Y \mid Y_\theta = Y] = \sqrt{1 - \exp(-O(\omega(\varepsilon)^2))},$$

which is the stated result. In particular, if the encoder network is Lipschitz continuous, then $\omega(\varepsilon) = L\varepsilon$ for some constant $L > 0$. In that case, the robust accuracy of the AdvDF degrades *sublinearly* with the perturbation budget, as $\sqrt{1 - \exp(-O(\varepsilon^2))} = \mathrm{o}(\varepsilon)$.

The above is a succinct asymptotic characterization of the more explicit bound

$$\Pr[\tilde{Y}_\theta \neq Y \mid Y_\theta = Y] = \sqrt{1 - \exp(-O(\omega(\varepsilon)^2))}\, \mathbb{E}\left[\frac{1}{\Pr[Y_\theta = Y \mid X]}\right].$$

Note that the term within the expectation is independent of ε and so does not affect the asymptotic behavior of the bound as a function of the perturbation budget. It is, however, interesting to note that this quantity is inversely proportional to the clean accuracy of the model. The AdvDF therefore has no inherent trade-off between accuracy and robustness: as long as the modulus of continuity ω can be kept under control, the robustness of the AdvDF should improve along with its accuracy. This is another feature which distinguishes it from many existing adversarial defenses, which usually have to sacrifice considerable accuracy in order to obtain improved robustness.

Finally, remark that we did not need to assume that the norm $\|\cdot\|$ was ℓ_2. Hence, this result applies to *any* ℓ_p threat model.

6.4.2 Relationship to Pinot et al. [2021]

Pinot et al. [2021] provide (to our knowledge) the only other theoretical study of the robustness of randomized classifiers using probability metrics. Their work is particularly relevant to us, since their theoretical results generalize the ones we presented here specifically for the AdvDF.

A *randomized classifier* is a classifier G_θ that outputs a categorical random variable

$$Y_\theta \sim \mathrm{Cat}(P_1, \ldots, P_k).$$

Pinot et al. [2021] call such a classifier (α_p, ε)-robust if for all $x, x' \in \mathcal{X}$,

$$\|x - x'\|_p \leq \alpha_p \implies D(G_\theta(x), G_\theta(x')) \leq \varepsilon.$$

Here, D can be any probability *metric*, *e.g.*, the Renyi divergence or the total variation distance. We have mainly focused on the KL divergence in our own work, which is not a metric. However, we have a bound on the total variation distance between Y_θ and $\tilde{Y}_\theta$:

$$\delta(Y_\theta, \tilde{Y}_\theta) \leq \sqrt{1 - \exp(-\mathrm{D}_{\mathrm{KL}}(\tilde{Y}_\theta \| Y_\theta))} = \sqrt{1 - \exp(-O(\omega(\varepsilon)^2))}.$$

Hence, in the language of Pinot et al. [2021], we have that classifiers constructed using the AdvDF are $(\varepsilon, \sqrt{1 - \exp(-O(\omega(\varepsilon)^2))})$-robust. In particular, if the encoder network of the AdvDF is Lipschitz continuous, the classifier is $(\varepsilon, \mathrm{o}(\varepsilon))$-robust. By [Pinot et al., 2021, theorem 1], this leads to the risk bound

$$R_{\mathrm{adv}}(G_\theta; \varepsilon) \leq R(G_\theta) + \sqrt{1 - \exp(-O(\omega(\varepsilon)^2))}.$$

6.4.3 Relationship to Barrett et al. [2021]

Barrett et al. [2021] investigate certified robustness of variational auto-encoders, a topic that is highly relevant to us given the similarities between the AdvDF and VAEs. Since VAEs are non-deterministic, Barrett et al. introduce the notion of *r-robustness*: a model F is r-robust (in the ℓ_2 norm) on a sample x and perturbation $\boldsymbol{\delta}$ with probability q if

$$\Pr[\|F(x + \boldsymbol{\delta}) - F(x)\|_2 \leq r] \geq q.$$

The model F under consideration here is supposed to be a generative model such as a VAE, so that $F(x)$ produces an approximate reconstruction of the sample x. In Barrett et al. [2021, theorem 1], the authors prove a bound on the r-robustness probability of Lipschitz continuous VAEs of the form

$$\Pr[\|F(x + \boldsymbol{\delta}) - F(x)\|_2 \leq r] \geq 1 - O\left(\frac{L^2 \varepsilon^2}{r^2}\right).$$

Here, L is the Lipschitz constant of the VAE and ε is the perturbation budget. We can prove a similar result if we assume the entire AdvDF is Lipschitz continuous:

Theorem 6.5. *If the entire AdvDF (both the encoder and decoder network) is Lipschitz continuous with constant L, then we have the following r-robustness probability bound:*

$$\Pr[\|D_\psi(z) - D_\psi(\tilde{z})\|_2 \leq r] \geq 1 - O\left(\frac{L\varepsilon}{r}\right).$$

Proof. Suppose D_ψ is Lipschitz continuous. Consider any input x and perturbation $\boldsymbol{\delta}$ such that $d(x+\boldsymbol{\delta},x) \leq \varepsilon$. Let $q_\phi = \mathcal{N}(0, \operatorname{diag} E_\phi(x))$ and $\tilde{q}_\phi = \mathcal{N}(0, \operatorname{diag} E_\phi(x+\boldsymbol{\delta}))$. By simple computations, we know that $\mathrm{D}_{\mathrm{KL}}(\tilde{q}_\phi \| q_\phi) = O(\omega(\varepsilon)^2)$.

Now let $z \sim q_\phi$ and $\tilde{z} \sim \tilde{q}_\phi$. If D_ψ is Lipschitz with constant L,

$$d(D_\psi(z), D_\psi(\tilde{z})) \leq L d(z, \tilde{z}).$$

If $d(\cdot,\cdot)$ is the Euclidean distance, then

$$d(z,\tilde{z}) = \sqrt{\sum_{i=1}^{t} (z_i - \tilde{z}_i)^2}.$$

Since z_i and $\tilde{z}_i$ are independent Gaussians for all i, their difference $z_i - \tilde{z}_i$ is also Gaussian. In particular, $z_i - \tilde{z}_i \sim \mathcal{N}(0, E_{\phi,i}(x)^2 + E_{\phi,i}(x+\boldsymbol{\delta})^2)$. Hence we may write

$$d(z,\tilde{z})^2 = \sum_{i=1}^{t} (E_{\phi,i}(x)^2 + E_{\phi,i}(x+\boldsymbol{\delta})^2) v_i^2,$$

where $v_1, \ldots, v_t \sim \mathcal{N}(0,1)$. This quantity can be modeled by a generalized chi-squared distribution,

$$\xi = \sum_{i=1}^{t} w_i y_i,$$

with weights $w_i = E_{\phi,i}(x)^2 + E_{\phi,i}(x+\boldsymbol{\delta})^2$ and chi-squared variables $y_i \sim \chi^2(1,0)$. The expectation is given by

$$\mathbb{E}[d(z,\tilde{z})^2] = \sum_{i=1}^{t} (E_{\phi,i}(x)^2 + E_{\phi,i}(x+\boldsymbol{\delta})^2).$$

Linearity of the expectation thus yields

$$\mathbb{E}[d(D_\psi(z), D_\psi(\tilde{z}))^2] \leq L^2 \sum_{i=1}^{t} (E_{\phi,i}(x)^2 + E_{\phi,i}(x+\boldsymbol{\delta})^2).$$

Given the uniform continuity of the encoder and the norm constraint on $\boldsymbol{\delta}$, we have

$$E_{\phi,i}(x+\boldsymbol{\delta}) \leq E_{\phi,i}(x) + \omega(\varepsilon).$$

As such,

$$\mathbb{E}[d(D_\psi(z), D_\psi(\tilde{z}))^2] \leq L^2 \sum_{i=1}^{t} (2E_{\phi,i}(x)^2 + 2E_{\phi,i}(x)\omega(\varepsilon) + \omega(\varepsilon)^2).$$

Hence $\mathbb{E}[d(D_\psi(z), D_\psi(\tilde{z}))^2] = O(L^2\omega(\varepsilon)^2)$. Concavity of the square root together with Jensen's inequality then yields $\mathbb{E}[d(D_\psi(z), D_\psi(\tilde{z}))] = O(L\omega(\varepsilon))$. Applying Markov's inequality, we find

$$\Pr[d(D_\psi(z), D_\psi(\tilde{z})) \leq r] \geq 1 - \frac{O(L\omega(\varepsilon))}{r}.$$

In particular, if the encoder is Lipschitz with $\omega(\varepsilon) = M\varepsilon$, this bound reduces to

$$\Pr[d(D_\psi(z), D_\psi(\tilde{z})) \leq r] \geq 1 - \frac{O(ML\varepsilon)}{r},$$

which completes the proof. □

Theorem 6.5 shows that the AdvDF can be certified in the framework of Barrett et al. [2021] with an r-robustness probability that depends *linearly* on the Lipschitz constant L of the full AdvDF network, the perturbation budget ε and the margin r.

6.4.4 Generalization to the exponential family

The results we obtained for the AdvDF combined with the existing literature on the robustness of randomized classifiers suggest a general pre-processing scheme where the input to a classifier F is first transformed by a randomized auto-encoder (E,D). The AdvDF is a special case of this where the encoder E maps the input to a Gaussian random variable with zero mean. Given that this construction has such favorable theoretical guarantees, it may be of interest to consider generalizations beyond the Gaussian. The most straightforward generalization is the *exponential family*. Therefore, in this section, we consider the robustness of a generalized AdvDF where the encoder maps the input to a random variable from the exponential family.

A probability density f is said to belong to the exponential family if it takes the form

$$f(z \mid \boldsymbol{\theta}) = h(z)\exp(\eta(\boldsymbol{\theta})^\intercal T(z) - A(\boldsymbol{\theta})),$$

where h, η, T and A are known functions. For a given input x the encoder E defines a random variable $E(x) \sim f(z \mid \boldsymbol{\theta}(x))$. Let $\boldsymbol{\theta} = \boldsymbol{\theta}(x)$ and $\tilde{\boldsymbol{\theta}} = \boldsymbol{\theta}(x+\boldsymbol{\tau})$ for $\|\boldsymbol{\tau}\| \leq \varepsilon$. We are then interested in the quantity

$$\begin{aligned}
&\mathrm{D}_{\mathrm{KL}}(f(z \mid \tilde{\boldsymbol{\theta}}) \| f(z \mid \boldsymbol{\theta})) \\
&= \int f(z \mid \tilde{\boldsymbol{\theta}}) \log \frac{f(z \mid \tilde{\boldsymbol{\theta}})}{f(z \mid \boldsymbol{\theta})} \mathrm{d}z \\
&= \int f(z \mid \tilde{\boldsymbol{\theta}}) \log \frac{\exp(\eta(\tilde{\boldsymbol{\theta}})^\intercal T(z) - A(\tilde{\boldsymbol{\theta}}))}{\exp(\eta(\boldsymbol{\theta})^\intercal T(z) - A(\boldsymbol{\theta}))} \mathrm{d}z \\
&= \int f(z \mid \tilde{\boldsymbol{\theta}}) \left[\eta(\tilde{\boldsymbol{\theta}})^\intercal T(z) - A(\tilde{\boldsymbol{\theta}}) - \eta(\boldsymbol{\theta})^\intercal T(z) + A(\boldsymbol{\theta})\right] \mathrm{d}z \\
&= \int f(z \mid \tilde{\boldsymbol{\theta}}) \eta(\tilde{\boldsymbol{\theta}})^\intercal T(z) \mathrm{d}z - \int f(z \mid \tilde{\boldsymbol{\theta}}) \eta(\boldsymbol{\theta})^\intercal T(z) \mathrm{d}z + A(\boldsymbol{\theta}) - A(\tilde{\boldsymbol{\theta}}) \\
&= (\eta(\tilde{\boldsymbol{\theta}}) - \eta(\boldsymbol{\theta}))^\intercal \int f(z \mid \tilde{\boldsymbol{\theta}}) T(z) \mathrm{d}z + A(\boldsymbol{\theta}) - A(\tilde{\boldsymbol{\theta}}).
\end{aligned}$$

Letting $\tilde{Z} \sim f(z \mid \tilde{\boldsymbol{\theta}})$, this yields

$$\mathrm{D}_{\mathrm{KL}}(f(z \mid \tilde{\boldsymbol{\theta}}) \| f(z \mid \boldsymbol{\theta})) = (\eta(\tilde{\boldsymbol{\theta}}) - \eta(\boldsymbol{\theta}))^\intercal \mathbb{E}[T(\tilde{Z})] + A(\boldsymbol{\theta}) - A(\tilde{\boldsymbol{\theta}}).$$

As a result, we have the following risk gap for any AdvDF where the encoder samples from a distribution within the exponential family:

$$R_{\mathrm{adv}}(G;\varepsilon) \leq R(G) + \sup_{\boldsymbol{\tau} \in \mathrm{B}(\varepsilon)} \sqrt{1 - \exp\left(A(\tilde{\boldsymbol{\theta}}) - A(\boldsymbol{\theta}) + (\eta(\boldsymbol{\theta}) - \eta(\tilde{\boldsymbol{\theta}}))^\intercal \mathbb{E}[T(\tilde{Z})]\right)}.$$

Note that this gap vanishes when $\varepsilon = 0$, as expected. Furthermore, there is still a dependency on the original set of parameters $\boldsymbol{\theta}$ (which depends on the input sample x) and the perturbation $\boldsymbol{\tau}$ (which results in an optimization problem). Therefore, to make this bound useful in practice,

one has to specialize the functions A, η and T for the chosen distribution and prove some lower bound of the form

$$A(\tilde{\boldsymbol{\theta}}) - A(\boldsymbol{\theta}) + (\eta(\boldsymbol{\theta}) - \eta(\tilde{\boldsymbol{\theta}}))^{\intercal}\, \mathbb{E}[T(\tilde{Z})] \geq h(\varepsilon),$$

where $h : \mathbb{R} \to \mathbb{R}$ depends only on the perturbation budget ε. This would then yield the risk gap

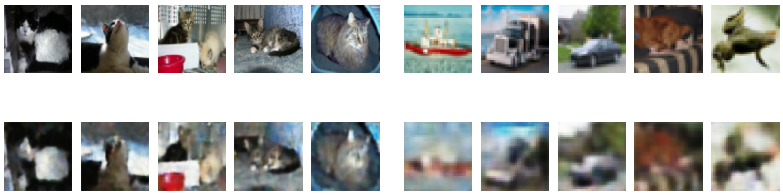
$$R_{\text{adv}}(G;\varepsilon) \leq R(G) + \sqrt{1 - \exp\left(h(\varepsilon)\right)}.$$

In the case of our AdvDF construction, we have $h(\varepsilon) = o(\omega(\varepsilon)^2)$.

6.5 Additional figures

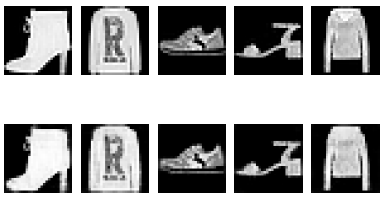
In this section, we supply additional visualizations produced by our experiments. Section 6.5.1 shows examples of input samples reconstructed by the AdvDF autoencoder. Section 6.5.2 is perhaps the most interesting section, as it presents full robustness curves of all models against all attacks. In the case of the PGD attack, we also provide comparisons with the state-of-the-art randomized smoothing defense as implemented by ART [Nicolae et al., 2018]. Section 6.5.3 plots the ECE values of the baselines, AdvDF and randomized smoothing methods for different perturbation budgets under the ℓ_2 PGD attack. Section 6.5.5 collects some additional results lending empirical support for the theoretical assumptions made about the AdvDF construction in order to prove theorem 6.1. In section 6.5.6, we experiment with the hyperparameters of the AdvDF, specifically the latent dimensionality t, to study its effects on the robust accuracy. Finally, in section 6.5.7 we perform timing experiments to compare the computational efficiency of the AdvDF to randomized smoothing.

6.5.1 Examples of reconstructed inputs

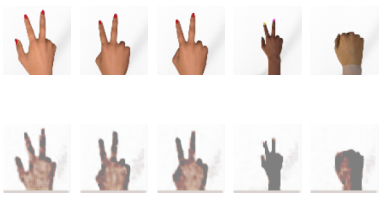


(a) Cats vs dogs

(b) CIFAR-10



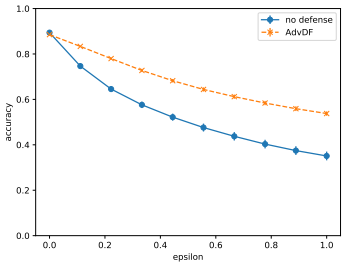
(c) Fashion-MNIST



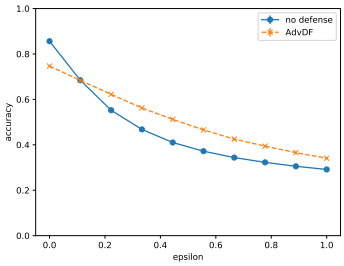
(d) Rock, paper, scissors

Figure 6.4: Examples of reconstructions made by the AdvDF decoder. For each data set, the top row shows a random selection of original samples and the bottom row shows their reconstructions by the decoder network of the AdvDF.

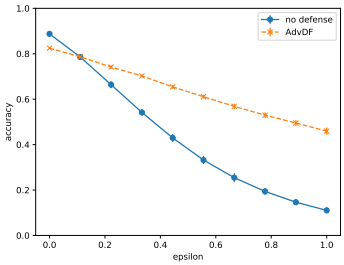
6.5.2 Robustness curves



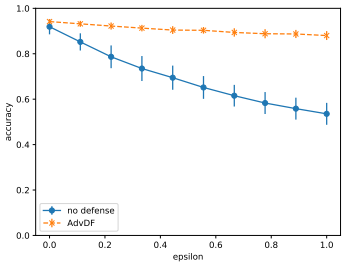
(a) Fashion-MNIST



(b) CIFAR-10

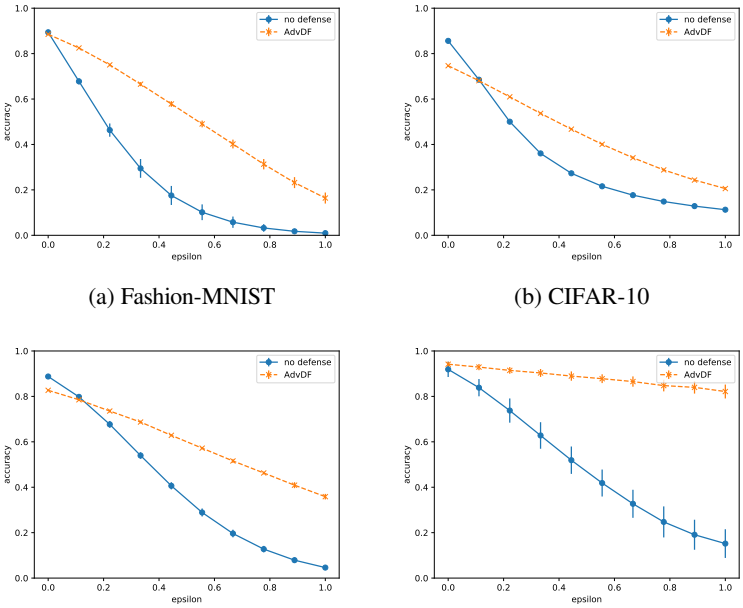


(c) Cats vs dogs



(d) RPS

Figure 6.5: Robustness curves for the ℓ_2 FGM attack on the different models.



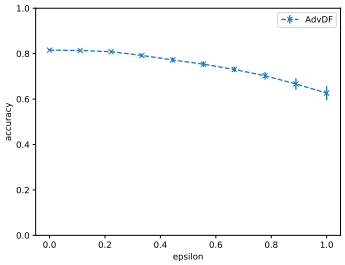
(a) Fashion-MNIST

(b) CIFAR-10

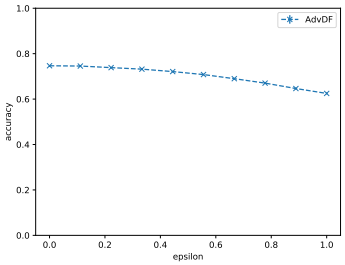
(c) Cats vs dogs

(d) RPS

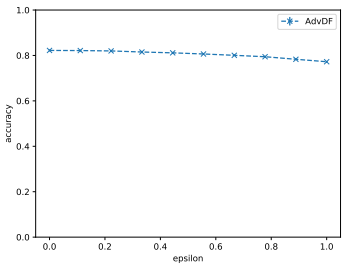
Figure 6.6: Robustness curves for the ℓ_2 PGD attack on the different models.



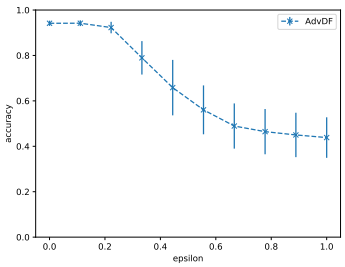
(a) Fashion-MNIST



(b) CIFAR-10

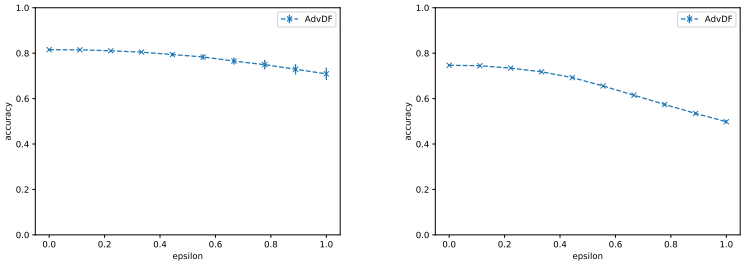


(c) Cats vs dogs

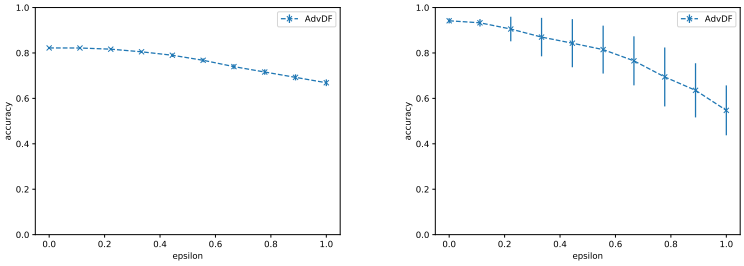


(d) RPS

Figure 6.7: Robustness curves for the adaptive ℓ_2 maximum damage attack on the different models.

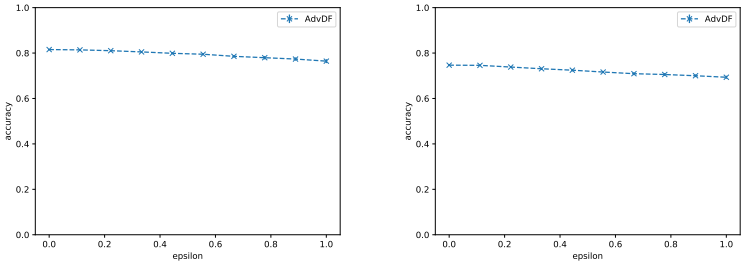


(a) Fashion-MNIST (b) CIFAR-10

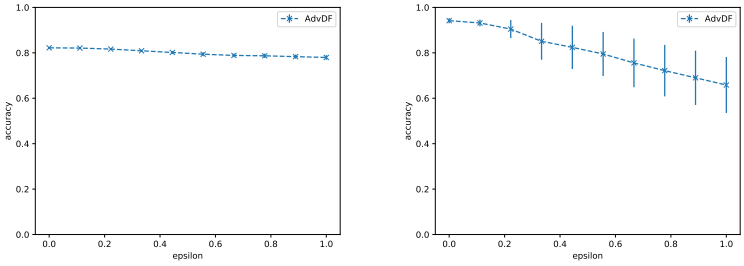


(c) Cats vs dogs (d) RPS

Figure 6.8: Robustness curves for the adaptive ℓ_2 variance attack on the different models.

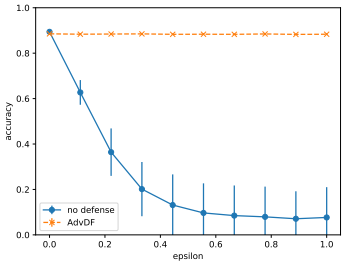


(a) Fashion-MNIST (b) CIFAR-10

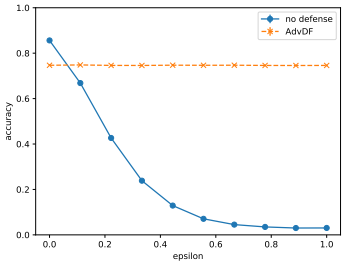


(c) Cats vs dogs (d) RPS

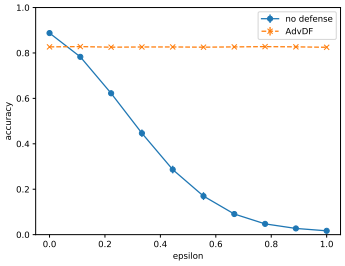
Figure 6.9: Robustness curves for the adaptive ℓ_2 latent space attack on the different models.



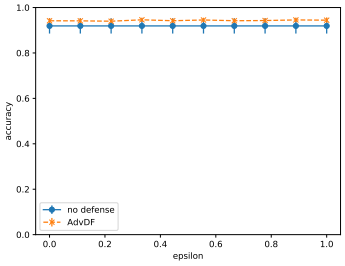
(a) Fashion-MNIST



(b) CIFAR-10

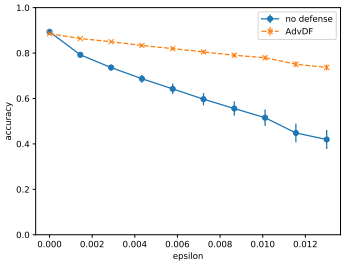


(c) Cats vs dogs

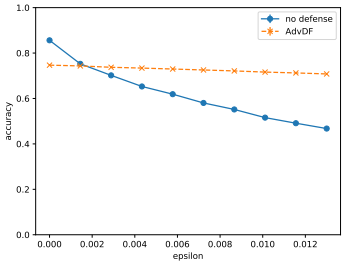


(d) RPS

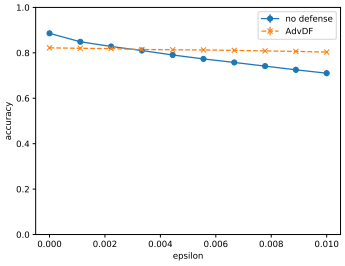
Figure 6.10: Robustness curves for the ℓ_2 Brendel-Bethge attack on the different models.



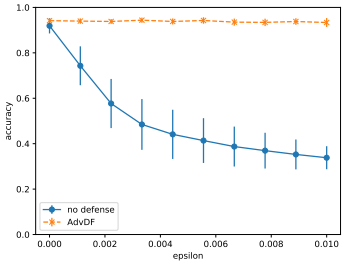
(a) Fashion-MNIST



(b) CIFAR-10



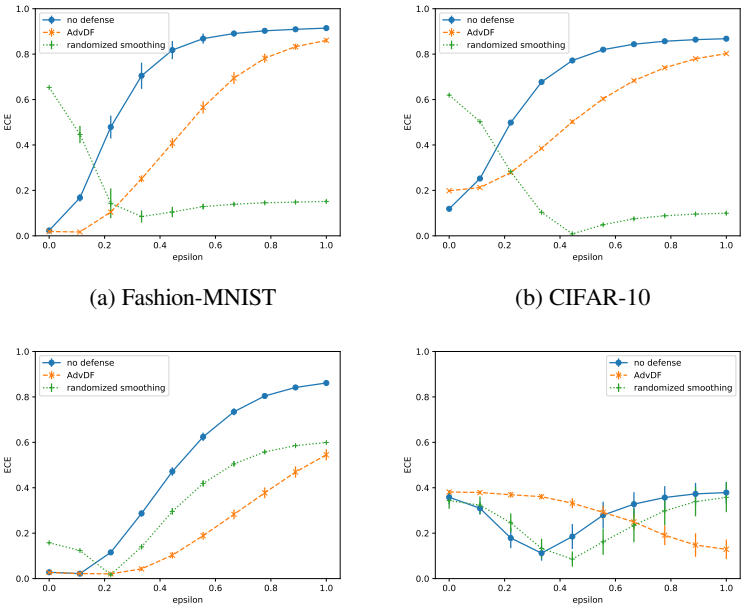
(c) Cats vs dogs



(d) RPS

Figure 6.11: Robustness curves for the black-box attack (algorithm 6.1) on the different models.

6.5.3 Calibration curves



(a) Fashion-MNIST

(b) CIFAR-10

(c) Cats vs dogs

(d) RPS

Figure 6.12: Comparison of the ECE for the baseline and AdvDF models against the ℓ_2 PGD attack.

6.5.4 Performance on CIFAR-10-C

The CIFAR-10-C data set was introduced by Hendrycks and Dietterich [2019] in order to benchmark the robustness of neural networks against *common* corruptions and perturbations. This is distinct from adversarial perturbations, which are designed to be *worst-case* and hence can differ substantially from naturally occurring distortions. Indeed, it has been observed that robustness to adversarial perturbations often comes at the cost of robustness to natural corruptions [Taori et al., 2020, Yin et al., 2019b]. Yet it is of course desirable for neural networks to be robust to *both* classes of perturbations (worst-case and natural). Therefore, as an additional benchmark for our method, we evaluate the AdvDF on the CIFAR-10-C data set and compare to the undefended baselines.

The CIFAR-10-C data set consists of samples from the CIFAR-10 test set which have been subjected to about twenty different forms of noise and corruption, divided into five severity levels. For a given model f and corruption type c, we report the *relative corruption error* (RCE) as defined by Hendrycks and Dietterich [2019]:

$$\mathrm{RCE}_c = \frac{\sum_{s=1}^{5} \left(E_{s,c}(f') - E_{\mathrm{clean}}(f')\right)}{\sum_{s=1}^{5} \left(E_{s,c}(f) - E_{\mathrm{clean}}(f)\right)}.$$

Here, f' is the robust equivalent of f (*i.e.*, defended using the AdvDF), $E_{s,c}(f)$ is the error of model f against corruption type c with severity level s and $E_{\mathrm{clean}}(f)$ is the error of f on the clean test data. The RCE thus measures the ratio of the drop in accuracy of the AdvDF f' and the original model f, summed across all severity levels. An RCE value greater than 1 indicates that the robust model suffers a larger relative drop in accuracy than the baseline, so it is desired that $\mathrm{RCE}_c < 1$ for all corruption types c.

Note that, as a relative measure, the RCE is not sensitive to the absolute accuracy of the models on the data; it only depends on the ratios of the differences between the accuracies on the original and corrupted data. As an extreme example, if f' does no better than random guessing, then $E_{s,c}(f') \approx E_{\mathrm{clean}}(f')$ and $\mathrm{RCE}_c \approx 0$ regardless of the accuracy of f. The RCE only gives information about the resilience of the model to corruptions; it is entirely possible that low RCE values may be achieved using models that are undesirable to deploy in practice. In order to be meaningful, one must take into account the absolute accuracy along with the RCE.

Figure 6.13 plots the RCE of our CIFAR-10 models for each corruption type. We computed the RCE for every model trained on CIFAR-10, *i.e.*, ten models initialized with unique random seeds, and indicated the standard error with black bars. For 14 out of the 19 corruption types included in the data set, we find that the AdvDF has significantly lower RCE than the baselines. There are only five types where the AdvDF incurs a significantly higher drop in accuracy. From these five, we notice two major outliers: `brightness` and `fog`, both with an RCE of around 200%. The `saturate` and `contrast` corruptions follow with an RCE of about 150%. It therefore appears that the AdvDF is more susceptible to corruptions that influence the brightness, contrast and color saturation of the image. It is *least* susceptible to corruptions such as Gaussian noise, blurring, compression artifacts, zooming, etc.

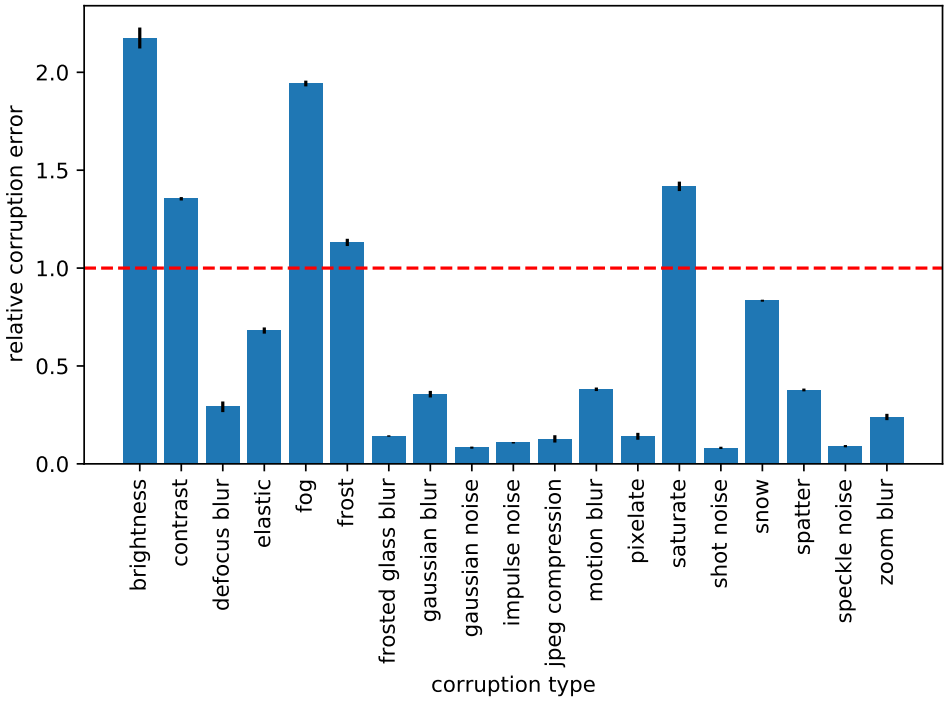


Figure 6.13: Bar plot of the relative corruption error (RCE) of the CIFAR-10 models against all corruption types and severity levels included in the CIFAR-10-C data set. The dashed red line marks 100% RCE.

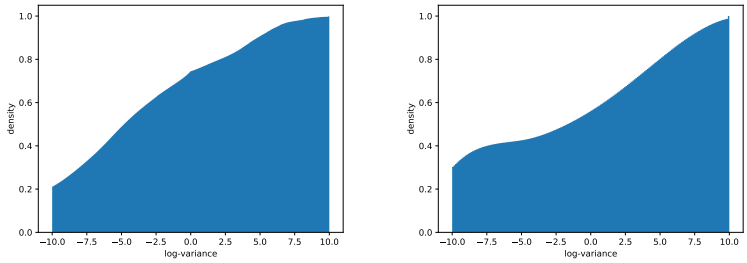
6.5.5 Theoretical assumptions

In this section, we perform some additional experiments regarding the assumptions underlying theorem 6.1. We first verify that the components of the variance vectors produced by the encoder networks are relatively large. We also plot the KL divergence values of the latent densities when subjected to adversarial attacks, in order to check whether the bound (6.1) holds in practice.

Variance vectors. In figure 6.14, we plot cumulative distributions of the log-variances produced by the encoder networks. We find that, across all data sets, the components of the variance vectors $E_\phi(x)$ tend to be either very large (at the maximum of the permissible range) or very close to zero; only relatively few components lie somewhere in between. Note that this result holds despite the fact that we did not utilize any regularization in the loss (6.3). This points to a curious property of the AdvDF: the encoder network will naturally tend towards high entropy regardless of whether we encourage this behavior or not, and this phenomenon is reproducible across architectures and data sets.

KL divergence values. Theorem 6.1 relies on several assumptions that may not be satisfied in practice. To confirm that our AdvDF models follow these assumptions and hence are subject to theorem 6.1, we plot the KL divergence values along with a quadratic polynomial approximation in figure 6.15. For each data set, we subject our trained AdvDF models to the ℓ_2 FGM attack for $\varepsilon \in [0, 1]$ and compute the average KL divergence across all samples for the original densities $q_\phi(\cdot \mid x)$ and perturbed densities $q_\phi(\cdot \mid \tilde{x})$. The resulting curves appear to grow quadratically with the budget: the quadratic fits lie close to the average KL divergence values and the residual sum of squares (RSS) values are very low.

These findings about the magnitude of the variance components as well as the KL divergence values of the latent densities provide empirical support for the assumptions made in corollary 6.2. We therefore expect the success rate of the adversarial attacks to scale sublinearly with ε. This appears to be confirmed by our empirical robustness curves: the robust accuracy of the AdvDF models seems to degrade very slowly as a function of the perturbation budget.



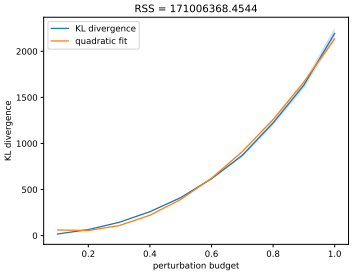
(a) Fashion-MNIST

(b) CIFAR-10

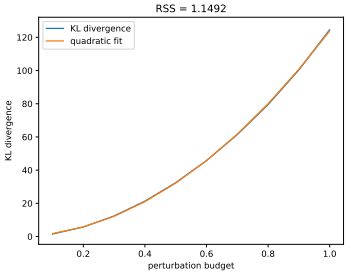
(c) Cats vs dogs

(d) RPS

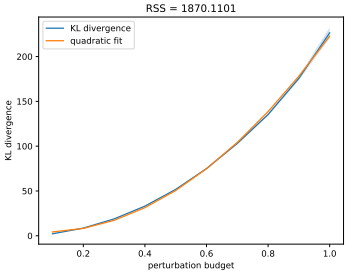
Figure 6.14: Cumulative distributions of the log-variance components of the AdvDF encoder networks.



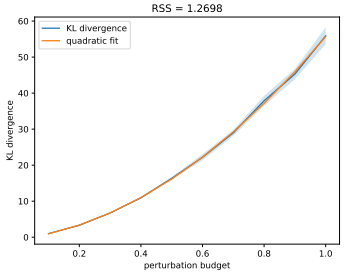
(a) Fashion-MNIST



(b) CIFAR-10



(c) Cats vs dogs



(d) RPS

Figure 6.15: Plots of the KL divergence values for the different AdvDF models against the ℓ_2 FGM attack. The residual sum of squares (RSS) values for the quadratic fits are also given.

6.5.6 Hyperparameters

Aside from the model architectures themselves, the main tunable hyperparameter of the AdvDF is the dimensionality t of the latent space. We experiment with varying t for the CIFAR-10 data set, choosing $t \in \{64, 128, 256, 512, 1024\}$. For each value, we retrain the AdvDF model and report its clean test accuracy as well as the robust accuracy against the ℓ_2 PGD attack at $\varepsilon = 0.4$. The results are given in figure 6.16. We see that a smaller value of t leads to a smaller gap between the clean and robust accuracy scores. However, at a certain point, the clean accuracy appears to plateau whereas the robust accuracy continues to decrease gradually. To tune t, one should therefore select the smallest possible value that still yields acceptable clean accuracy.

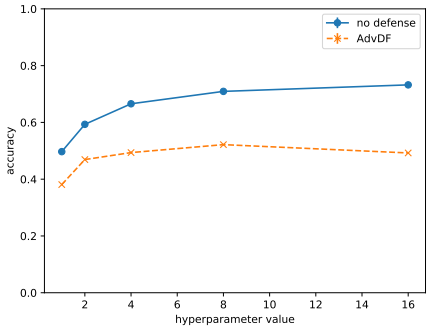


Figure 6.16: Clean and robust accuracies when varying the latent dimensionality t of the AdvDF for the CIFAR-10 data set. Note that t appears to take values in the set $\{1, 2, 4, 8, 16\}$ here, but this is because the last layer of our CIFAR-10 encoder network is a convolutional layer with outputs of dimension $(8, 8, r)$. Hence, $t = 64r$ and we let $r \in \{1, 2, 4, 8, 16\}$ so that $t \in \{64, 128, 256, 512, 1024\}$.

6.5.7 Computational efficiency

Figure 6.17 shows the relative increase in inference time with respect to the baseline models for the AdvDF and randomized smoothing methods. These ratios are shown on a log-10 scale, that is, we plot

$$\log_{10}\left(\frac{\text{inference time of the defense}}{\text{inference time of the baseline}}\right).$$

We measure the inference time across all samples in the test set of the respective task. The results clearly show that the AdvDF incurs virtually no overhead with respect to the baselines, while randomized smoothing is approximately two orders of magnitude slower.

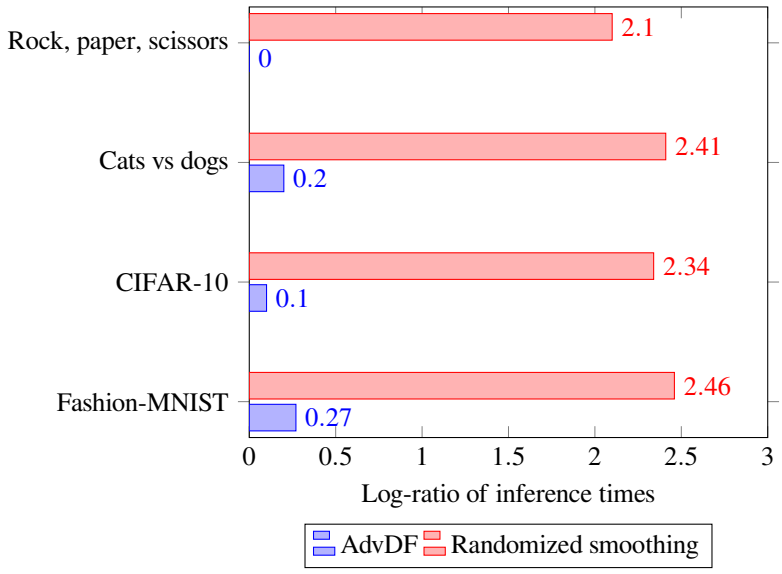


Figure 6.17: Comparison of the inference times for the different methods.

6.6 Conclusions

We have proposed the adversarial density filter (AdvDF), a simple and scalable defense against adversarial examples that appears to offer high robustness at relatively low computational cost. We have experimentally verified the effectiveness of the defense using a variety of adversarial attacks, including the AutoAttack benchmark and three adaptive attacks. The AdvDF significantly outperforms the baselines and compares favorably with the state-of-the-art randomized smoothing method of defense, both in terms of robustness as well as overhead at inference time. The experimental findings support our theory that the robust accuracy of the AdvDF decays *sublinearly* with the perturbation budget, whereas prior work usually scales at least linearly or quadratically.

The AdvDF construction requires an additional auto-encoder to be trained on the data, but it makes no further assumptions on the downstream model to be defended. In fact, only the encoder module of the AdvDF is subject to any real constraint, namely that it must have a small

modulus of continuity. This implies that the encoder must be a uniformly continuous function, which is a much less stringent requirement than the Lipschitz continuity that is commonly assumed in related work [Gouk et al., 2021]. Indeed, uniform continuity is already satisfied by many neural networks used in practice. However, in this work, we were not yet able to construct encoder modules with small moduli of continuity: when estimated via our optimization procedure (6.8), the resulting moduli are too large and cause the risk bounds to become vacuous. The main avenue for future work, therefore, is to find methods of effectively constraining the modulus of continuity of the encoder networks so that we may give non-trivial generalization bounds. That said, it is worth noting that our AdvDF models still obtained significant improvements in robustness despite our inability to provide meaningful theoretical bounds. This may indicate that our bounds are too pessimistic, or it could imply that our current benchmarks (including AutoAttack) still significantly over-estimate robustness even when there is no gradient masking or other apparent obstacles to robustness evaluation.

Architecturally speaking, the AdvDF is identical to a VAE [Kingma and Welling, 2013]. The important difference is that the AdvDF is also trained to maximize the entropy of its latent distribution, which we accomplish here by instantiating it with an isotropic Gaussian and maximizing the variance. The VAE, on the other hand, penalizes the KL divergence of its latent distribution from a *unit* Gaussian. Theoretically, we could inherit all the advantages of the AdvDF by simply using a VAE where the unit Gaussian in the KL divergence term is replaced by an isotropic Gaussian with high variance. This variance could either by fixed beforehand or it could also be learned, in which case a penalty may be added to encourage the variance to be large. Moreover, VAEs typically do not have zero-mean latent distributions, the mean being determined by the encoder neural network. However, if the mean is non-zero, this would introduce a quadratic dependency on the modulus of continuity of the mean function in the KL divergence bound (6.1). To suppress this effect, we would either have to use a zero-mean variant of the VAE or the mean would have to be carefully constructed to have small modulus of continuity. Exploring this relationship further could be interesting given the success of VAE methods.

Another avenue for future work could be to evaluate the AdvDF in the unsupervised setting, as we have only considered the supervised setting in this work. Theorem 6.5 already provides a theoretical foundation for such work. Furthermore, combining the AdvDF with model weight averaging and data augmentation strategies such as those employed by Rebuffi et al. [2021] would likely boost the performance of the models considerably. These techniques have recently shown great promise, are straightforward to implement alongside the AdvDF and would come at little additional cost. Using them to improve the robustness of the encoder module of the AdvDF may be a first step towards a solution to the problem of large moduli of continuity, since we expect that robust models composed of uniformly continuous functions would naturally have small moduli. If they do not, then this likely means our adversarial attacks are not strong enough.

Chapter 7

Conclusions

With them the Seed of Wisdom did I sow,
And with my own hand labour'd it to grow:
And this was all the Harvest that I reap'd-
"I came like Water, and like Wind I go."

Rubaiyat
Omar Khayyam

7.1 Summary of contributions

This work has focused on several avenues for defending deep neural networks against adversarial perturbations. The result of this PhD has been a number of different algorithms that improve the reliability of machine learning models. The obvious question that arises now is when and why to use each of these algorithms. In this section, we recapitulate the proposed algorithms and list their particular use-cases. These observations are also summarized in table 7.1.

The binary IVAP defense. The binary IVAP (algorithm 3.3) is an efficient and effective detector-type defense for binary classification tasks where a pre-existing scoring classifier is already available. It requires two hyperparameters: a calibration set and a threshold β. The calibration set may be taken to be (part of) the training data of the scoring classifier, or it might be entirely new data. The threshold β is an intuitive scalar parameter which is easy to set manually based on some empirical tests, but it can also be tuned automatically to optimize various metrics of interest. As a detector defense, the binary IVAP is only able to signal when a model prediction might be unreliable, but it cannot (in general) correct it. To deploy this defense, therefore, there needs to be some mechanism that deals with rejected predictions. This may be as simple as notifying the user that their submission has been rejected or must be subjected to manual review. This could be appropriate in content filtering systems such as those employed by YouTube. Alternatively, rejections may be handled by a more complicated model that requires more computational resources.

The MultIVAP defense. The MultIVAP (algorithm 4.1) was proposed as a generalization of the concept of the binary IVAP defense to the multiclass case. Its efficiency and effectiveness are comparable to the binary IVAP; its main computational bottleneck seems to be the number of classes: inference using the MultIVAP requires the solution of an optimization problem the complexity of which seems to scale linearly with the number of classes. Like the binary IVAP defense, the MultIVAP assumes a pre-existing scoring classifier on which it can be calibrated. It also has two hyperparameters: the calibration data and a confidence parameter ε. Again, the calibration data may simply be the original training data; this does not matter much as long as it constitutes an i.i.d. sample of the data distribution. The confidence parameter can be tuned manually or automatically, *e.g.*, to target a specific Jaccard index or predictive efficiency value. The MultIVAP can be considered a detector-type defense, as it has several ways of expressing unreliability of a prediction: it can output nothing at all or it can output multiple labels. To deploy this algorithm, one must therefore be able to handle multi-valued and even empty outputs. Empty outputs can be handled the same way as rejections in the case of the binary IVAP, but multiple labels could be approached differently. There might be a logical preference ordering on the labels, so that multi-label predictions can be reduced to single labels; there might also be a threshold on the maximum number of labels allowed before the prediction is considered too unreliable and treated as a rejection. It is also possible that the problem itself is indeed inherently multi-label, in which case it can be entirely appropriate to apply more than one label to a single input.

The conformal CANN detector. Whereas the IVAP and MultIVAP defenses are based on the IVAP algorithm by Vovk et al. [2015], the CANN detector we proposed in algorithm 5.1 is a direct instantiation of the general conformal prediction algorithm 3.1. The non-conformity measure was chosen based on the mean-squared difference between the principal functions of the underlying data distribution because of their desirable properties [Hardoon et al., 2004, Makur et al., 2015]. The CANN detector makes use of a separate neural network based on Hsu et al. [2019] which estimates the principal functions and is trained independently of the model to be defended. Similarly to the IVAP and MultIVAP, the CANN detector is can be readily applied to any model without the need for retraining or finetuning. The neural network on which the detector is based is simultaneously its main advantage as well as its main disadvantage: the neural network model gives the practitioner much greater control over the detector, since its architectural details and specifics of training can be heavily modified as desired to fit various requirements such as memory, speed and accuracy. On the other hand, this greater degree of control also means that appropriate tuning of these hyperparameters may be much more intensive. That said, construction of the CANN detector depends only on the underlying data distribution. As such, the defense is modular and re-usable: practitioners can develop different pre-trained CANN detectors for various data sets and practical settings in advance, which can then be deployed off the shelf by practitioners who do not have the resources to construct them from scratch. In case of distribution shift, the CANN detector may be adapted using lightweight finetuning, a technique that is well-established in, *e.g.*, the NLP community, where large language models that are infeasible to retrain are often finetuned and quickly adapted to novel problems [Brown et al., 2020].

The adversarial density filter. The adversarial density filter (AdvDF) was proposed in chapter 6 as a certifiable purification defense. It has the main advantage of admitting specific bounds on the adversarial risk that grow sublinearly as a function of the perturbation budget. The de-

Algorithm	Type	Certified	Randomized	Remarks
Binary IVAP	Detector	✗	✗	1. binary classification only 2. possibility of rejection
MultIVAP	Detector	✗	✗	1. possibility of rejection 2. multi-label outputs 3. inference time scales linearly with number of classes
CANN detector	Detector	✗	✗	1. possibility of rejection 2. requires an additional neural network 3. modular and re-usable
AdvDF	Purification	✓	✓	1. requires an additional neural network 2. may require re-training

Table 7.1: Summary of the salient properties of the different defenses proposed in this work.

fense itself is essentially a type of variational auto-encoder trained using a specific loss function that encourages high entropy in the latent distribution, thus preventing small modifications of the input from having a large impact on the latent code and, subsequently, the reconstructions. This also makes it relatively efficient, as inference using the AdvDF merely requires an additional forward pass through a neural network. Its main drawback is that it requires the practitioner to specify an entire auto-encoder network, but this might not be a significant hurdle since there exist many published auto-encoder architectures that perform well on many tasks. The AdvDF can be trained on its own, without any knowledge of downstream models, making it a modular defense that can potentially be applied in many different tasks at once. However, depending on the specific downstream task, the defended model may have to be retrained or fine-tuned. As a purification defense, the AdvDF will never reject any input or flag any prediction as unreliable, making it suitable for use-cases where rejections or multiple labels are undesirable.

7.2 Future research directions

To conclude this work, let us take a step back and analyze where the field of adversarial machine learning has come from, what its current status is and where it could go from here. First, allow me to emphasize a point that is sometimes contentious within the wider machine learning community: adversarial robustness *is* important and useful to study, but not for the reasons that most people seem to believe. The problem is often cast as one of cybersecurity, where a malicious adversary could gain all sorts of unauthorized access and control by manipulating machine learning systems in imperceptible manners. As I have said before, I do not believe this setting to be very compelling, because a well-engineered security-sensitive system should not crucially rely on any single component. This is especially true when it comes to machine learning, since we know that even our most accurate models have a non-negligible probability of error even in the complete absence of any sort of adversary. This is a simple statistical fact. Therefore, any system that incorporates machine learning should also have redundancies and fail-safes in place to ensure the system does not malfunction when the ML component breaks down. This simple principle of good engineering immediately rules out most of the common adversarial attack scenarios, such as attacks on self-driving cars which could be compensated by LIDAR systems

and GPS data. A case can also be made that in a good portion of the remaining adversarial attack scenarios which cannot be adequately compensated by redundancy and fail-safes, there is often no need to respect any form of imperceptibility requirement: an attacker can introduce arbitrarily large distortions, because no human will be looking at the data anyway (or at least, not until it's too late). The objective of most attackers would hence simply be to fool the system however they can, even if this means crafting inputs that are highly anomalous. They have this advantage by virtue of the fact that most machine learning applications are developed specifically for use-cases where manual inspection of the data by human moderators is prohibitively expensive or inefficient.

Therefore, what makes adversarial machine learning a compelling problem to study, is because of what it means for *generalization*. In specific cases, of course, this may have security implications as well, but that is not our main concern. What research into adversarial robustness *has* taught us, is that our models are generally unreliable. Consider, for example, a classifier tasked with screening resumés. By now, there exist many companies that provide such automated resumé screening as a commercial service. But can we trust these services if we know that the decision of the model may very well be entirely reversed by manipulating a specific character in the name of the candidate, or by manipulating a handful of pixels in their profile picture? What has the model actually learned if it can so easily make such egregious errors?

7.2.1 Explainable AI

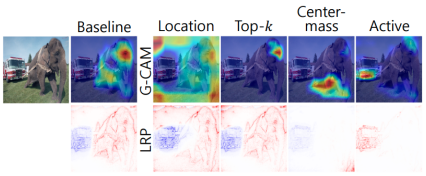
These questions naturally bring us to the domain of *explainable AI* (XAI). There are obvious connections between adversarial robustness and XAI, and I believe further exploring them should be one of our principal future research directions. Specifically, there is a glaring contradiction that must be resolved: if the goal (broadly speaking) of XAI is to understand *why* a model made a certain prediction on a certain sample, how is it that XAI techniques have failed to independently confirm the existence of adversarial perturbations? Indeed, we know that many XAI techniques *themselves* can be fooled by adversarial samples: this is shown, for instance, in the works by Dombrowski et al. [2019] and Heo et al. [2019]. Figure 7.1 shows some striking examples of their results.

We would intuitively expect that any "good" XAI technique should be able to discover that a model may be making a correct prediction for the wrong reasons, such as overly relying on specific values of certain pixels in an image, but to my knowledge this is not the case: when XAI techniques are applied to models known to be fragile against adversarial perturbations, the methods generally do not detect this vulnerability, and defenses based on XAI have been shown to fail [Carlini, 2019a]. Moreover, for many XAI techniques, it is possible to generate adversarials that simultaneously fool the models and also allow us to manipulate the generated explanations however we want. This seems to point toward a deep flaw in our methodologies when it comes to explaining model predictions, but it also represents a certain opportunity. Indeed, it shows that the fields of XAI and AML have valuable things to learn from each other.

7.2.2 Moving beyond the toy problem

Another important research direction that ought to be explored further, is the refinement of the adversarial robustness problem itself. As it stands, researchers are much too heavily focused on computer vision and the L_∞ threat model. Although one of course has to start somewhere,

(a) Dombrowski et al. [2019]



(b) Heo et al. [2019]

Figure 7.1: Examples of how the explanations generated by XAI methods can be arbitrarily manipulated by adversarial attacks.

at the time of this writing it has been almost ten years since the work of Szegedy et al. [2013] introduced the deep learning community to this phenomenon, yet we are still studying essentially the same problem despite its known shortcomings. The L_p threat model is a particularly poor fit for computer vision: large L_p perturbations do not necessarily translate to heavily altered images, nor do small L_p perturbations imply small distortion. A small rotation, for instance, could imply a very large L_p perturbation but would not actually change much of what is seen in the image. Furthermore, adversarial robustness is a problem for virtually *all* machine learning models, not just deep neural networks for image recognition. In the case of natural language processing, for example, it is unclear what exactly constitutes a "small" or "imperceptible" perturbation. Most work in this area considers the introduction of typographical errors, where one or more letters of a word are replaced, thus causing slight misspellings in the text. In essence, this is the equivalent of the L_p threat model using the Levenshtein distance as the metric instead of an L_p norm. However, one could also justify replacing words with synonyms or homophones, which could drastically increase the Levenshtein distance but would nevertheless only make a very small difference to human readers.

As a first step towards overcoming the limitations of the toy problem, we can take the robust optimization approach by Madry et al. [2017] as our starting point:

$$\theta^\star = \underset{\theta}{\operatorname{argmin}}\ \mathbb{E}\left[\max_{\boldsymbol{\delta}\in B_\varepsilon} \mathcal{L}(X+\boldsymbol{\delta}, Y, \theta)\right].$$

Suppose now we have a collection of transformations Φ of $\mathcal{X} \to \mathcal{X}$ functions and we wish to be robust to all of these. A straightforward generalization to consider would then be

$$\theta^\star = \underset{\theta}{\operatorname{argmin}}\ \mathbb{E}\left[\max_{\phi\in\Phi} \mathcal{L}(\phi(X), Y, \theta)\right].$$

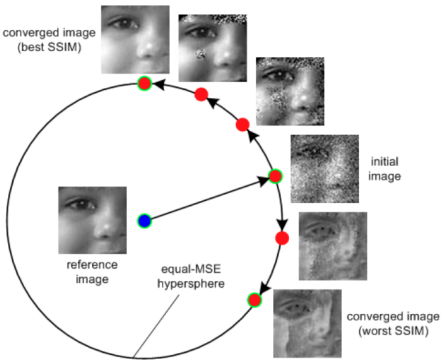


Figure 7.2: Images equally far away from a reference image in the L_p sense can be dramatically different in perceived distance as measured by the Structural Similarity Index (SSIM). Figure due to Wang and Bovik [2009].

We recover the original formulation by Madry et al. [2017] by simply setting

$$\Phi = \{x \mapsto x + \boldsymbol{\delta} \mid \boldsymbol{\delta} \in B_\varepsilon\}.$$

This is essentially what techniques such as Expectation over Transformation (EOT; Athalye et al. [2018b]) hope to achieve. EOT was, however, introduced with a much more narrow scope in mind: to produce physical adversarial examples that could be printed on stickers or 3D printed onto solid objects. This idea could certainly be generalized far beyond this one setting. However, there are significant technical barriers that must be resolved. The most obvious one being the question of how to choose Φ such that networks can still be trained efficiently and we obtain non-trivial robustness. The problem is already exceptionally hard in the case where Φ is restricted to norm-bounded additive perturbations, so extending it to more general transformations poses a great challenge. To make such optimization remotely practical, Φ would have to consist of a finite number of parametric families of functions $\Phi_1, \ldots, \Phi_k$, such that the loss $\mathcal{L}(\phi(X), Y, \theta)$ can be maximized using standard methods for any $\phi \in \Phi$. In the most extreme case, these families could be neural networks, so that each inner maximization would correspond to training an auto-encoder. However, such an approach likely would not scale. Moreover, even for relatively simple transformations there remains the question of how to guarantee that the transformed input retains its original label. For example, on the MNIST data set, rotating digits by 180 degrees can cause sixes to transform into nines and vice versa, so one must be careful to limit the maximum rotation degrees. In general, however, it is unclear how to constrain a complicated function class Φ such that $\phi(x)$ always has the same ground-truth label as x for any $\phi \in \Phi$. In fact, this is already nearly impossible in the L_p threat model for computer vision, as even very small additive perturbations in L_p space can dramatically alter the appearance of a given image. This was demonstrated quite effectively by Wang and Bovik [2009] and Gilmer et al. [2018], using figure 7.2.

I believe progress on this front can be made by taking inspiration from differential geometry. There is a somewhat folkloristic idea within the ML community, tacitly assumed by many existing algorithms, often referred to as the *manifold hypothesis*. This hypothesis posits that the

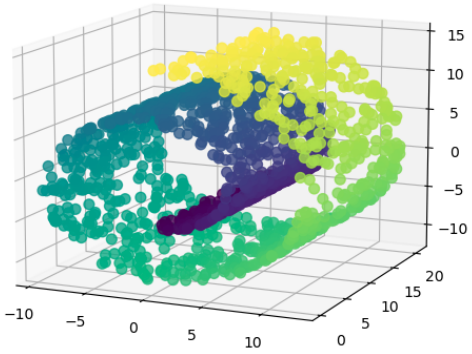


Figure 7.3: The Swiss roll, a typical example of a non-Euclidean manifold embedded in Euclidean space and a common benchmark data set for manifold learning algorithms. This image was generated using the scikit-learn library [Pedregosa et al., 2011].

input data of our ML algorithms, although they are typically given as vectors in $\mathbb{R}^n$ for some potentially large n, in fact almost always lie on an embedded submanifold of much lower dimensionality $d \ll n$. An archetypal illustration of the manifold hypothesis is given by the Swiss roll, depicted in figure 7.3. Although samples on this manifold would be most conveniently represented as real-valued vectors in $\mathbb{R}^3$, the actual structure of the data is more complicated. In particular, the Swiss roll is not well-modelled by three-dimensional Euclidean space and metric structure, as two points may lie very close to each other in the ambient $\mathbb{R}^3$ space but very far along any geodesic of the manifold. However, the Swiss roll can be easily transformed such that Euclidean metric structure is respected: indeed, by "unfolding" the Swiss roll, we can project it onto $\mathbb{R}^2$ in such a way that Euclidean distance is a good proxy for its geodesic distance. Some variants of the manifold hypothesis hold that similar transformations can be found for essentially any data set of interest, where the original data manifold can be "flattened" into some Euclidean space.[1] The manifold hypothesis tends to be an unavoidable requirement for most ML algorithms, particularly those dealing with dimensionality reductions [Ma and Fu, 2012]. The work of Fefferman et al. [2016] is perhaps one of the most important contributions in this area, as they not only gave a rigorous statement of the manifold hypothesis, but also gave explicit algorithms to test this hypothesis as well as learn the manifold in question, along with statistical sample complexity bounds.

A blog post by Steven Traversi gives a more realistic example of the manifold hypothesis in practice, on the MNIST data set.[2] He considers the classification accuracy of a k-nearest neighbor (kNN) algorithm on MNIST samples after dimensionality reduction with principal component analysis (PCA). PCA is a linear dimensionality reduction method, *i.e.*, it assumes that the embedded manifold is a linear subspace of the ambient space. In the case of MNIST, the ambient space is the Euclidean vector space $\mathbb{R}^{28\times 28} \cong \mathbb{R}^{784}$. Mathematically, the result of applying PCA to any n-dimensional data set is a matrix $W \in \mathbb{R}^{d\times n}$ where d is the chosen dimensionality of the

[1] See `http://colah.github.io/posts/2014-03-NN-Manifolds-Topology/` for a nice discussion of these ideas.

[2] `https://steven.codes/blog/ml/how-to-get-97-percent-on-MNIST-with-KNN/`. Accessed 2022-06-03.

reduced space. That is, we have a linear transformation that maps vectors $x \in \mathbb{R}^n$ in the ambient space to the reduced space $\mathbb{R}^d$. Using cross-validation, Traversi is able to obtain a maximum of 97% accuracy by reducing the 784-dimensional MNIST input space to a 45-dimensional space via PCA, and then applying kNN with $k = 6$. This shows that MNIST is well-modelled by a 45-dimensional Euclidean space despite actually living in a 784-dimensional space. This also suggests much potential for non-linear dimensionality reduction methods (*e.g.*, based on neural networks) on other, more complicated data sets.

To make the connection with adversarial robustness explicit, consider a continuously connected Riemannian manifold (M, g), where M is the topological manifold and g is the Riemannian metric tensor.[3] The metric tensor assigns an inner product to each of the tangent vector spaces T_pM, for all $p \in M$. These inner products are typically written as g_p, with corresponding norms

$$|v|_p = \sqrt{g_p(v, v)}$$

for $v \in T_pM$. Under the assumption of continuous connectedness, g induces a metric structure on M. Specifically, it gives rise to the distance function

$$d_g : M \times M : (p, q) \mapsto \inf\{L(\gamma) \mid \gamma \in \Gamma(p, q)\}.$$

Here, $\Gamma(p, q)$ is the set of all continuously differentiable curves connecting the points $p \in M$ and $q \in M$. That is, it consists of all continuously differentiable functions $\gamma : [0, 1] \to M$ where $\gamma(0) = p$ and $\gamma(1) = q$. In the language of differential geometry, the function $\gamma : [0, 1] \to M$ is differentiable if it assigns to every $t \in [0, 1]$ a vector $\gamma'(t)$ in the tangent space $T_{\gamma(t)}M$. The function $L(\gamma)$ computes the length of any differentiable curve, which we can define thanks to the metric tensor g:

$$L(\gamma) = \int_0^1 |\gamma'(t)|_{\gamma(t)} \mathrm{d}t = \int_0^1 \sqrt{g_{\gamma(t)}(\gamma'(t), \gamma'(t))} \mathrm{d}t.$$

Using this machinery of differential geometry, the adversarial robustness problem can be generalized as follows. Under the assumption of the manifold hypothesis, we posit that our input data in fact lie on a Riemannian manifold (M, g). Furthermore, this manifold is embedded in Euclidean space $\mathbb{R}^n$ via some map $\iota : M \to \mathbb{R}^n$, and its dimension may be much less than n. Now consider any classifier $f : \mathbb{R}^n \to \mathcal{Y}$ which maps the real vector representations of points on the manifold M to labels in some finite set $\mathcal{Y}$. We can then define point-wise robustness of f at some point $p \in M$ as

$$\rho(p; f) = \inf_{q \in M} \{d_g(p, q) \mid f(\iota(p)) \neq f(\iota(q))\}.$$

Its general robustness with respect to some probability density p_M on M can then be defined as

$$\rho(f) = \mathop{\mathbb{E}}_{p \sim p_M} [\rho(p; f)].$$

We can recover the typical L_p notion of robustness by simply assuming that ι is the identity map on $\mathbb{R}^n$ and M is Euclidean. In essence, the common L_p formulation of the adversarial robustness

[3] The exposition here is necessarily rather superficial. For a more thorough development of the theory of differential geometry, we refer the interested reader to the introductory textbook of Tu [2010] or the more advanced work by Lee [2010].

problem can be motivated from the perspective of differential geometry by virtue of the fact that Riemannian manifolds locally behave as Euclidean space. Indeed, for any $p \in M$ there exists an open subset $U_p \subseteq M$ that is diffeomorphic to $\mathbb{R}^n$. Let us denote this diffeomorphism by ψ and take any $q \in U_p$. Let $\gamma : [0,1] \to M$ be a curve connecting p and q. Since ψ is differentiable, we can apply Taylor's theorem to obtain

$$\psi(q) = \psi(\gamma(1)) \approx \psi(\gamma(0)) + \left[\frac{d}{d\tau}\psi(\gamma(\tau))\right]_{\tau=0} = \psi(p) + \left[\frac{d}{d\tau}\psi(\gamma(\tau))\right]_{\tau=0}.$$

In other words, if the manifold M is sufficiently "flat" around p, and q is sufficiently close to p, then $d_g(p,q) \approx \|\imath(p) - \imath(q)\|$ and the L_p threat model makes sense. However, it has been made abundantly clear by now that this approximation does not allow us to achieve the level of robustness we desire from our models. This is likely due to the fact that, although the data manifolds are locally Euclidean, this notion of locality is too strict, *i.e.*, the Euclidean structure already starts to break down for very small distances $\|\imath(p) - \imath(q)\|$. This happens, for example, when the manifold M is highly curved, something which seems realistic for many real-world data sets [Sritharan et al., 2021]. This may also explain, in part, why L_p adversarial robustness seems to be at odds with accuracy: small perturbations in the ambient Euclidean space can correspond to large perturbations in the embedded manifold. This is demonstrated by the Swiss roll in figure 7.3 or, more realistically, by figure 7.2. Depending on the exact structure of the data manifold, it might not make any sense at all to perform L_p adversarial training, because we may actually be training the model on incorrectly labeled examples or examples that are nonsensical, lying completely off the data manifold (such as those generated by Nguyen et al. [2015]).

For these reasons, I believe the future of adversarial ML may lie in manifold learning. Indeed, suppose that we can construct some accurate approximation of a data manifold M based on i.i.d. samples in Euclidean space $\mathbb{R}^n$, yielding a map $\psi : \mathbb{R}^n \to M$. This may be the latent space of some auto-encoder or generative adversarial network [Creswell et al., 2018], or the output of some normalizing flow [Kobyzev et al., 2020]. In that case, we can consider an alternative characterization of adversarial training that respects the underlying manifold structure:

$$\theta^\star = \min_\theta \mathbb{E}\left[\max_{\tilde{X} \in B_\varepsilon(X)} \mathcal{L}(\tilde{X}, Y, \theta)\right]. \tag{7.1}$$

Here, the neighborhood $B_\varepsilon(x)$ is defined as the set of all points $\tilde{x} \in \mathbb{R}^n$ whose geodesic distance from x on the manifold is bounded by ε:

$$B_\varepsilon(x) = \{\tilde{x} \in \mathbb{R}^n \mid d_g(\psi(x), \psi(\tilde{x})) \leq \varepsilon\}.$$

Equivalently, we can consider the set $L(p,\varepsilon)$ of all continuous curves $\gamma : [0,1] \to M$ with $\gamma(0) = p$ and $L(\gamma) \leq \varepsilon$. More formally,

$$L(p,\varepsilon) = \{\gamma \in \Gamma(p,q) \mid q \in M \text{ and } d_g(p,q) \leq \varepsilon\}.$$

We can then write (7.1) as

$$\theta^\star = \min_\theta \mathbb{E}\left[\max_{\gamma \in L(\psi(X),\varepsilon)} \mathcal{L}(\gamma(1), Y, \theta)\right]. \tag{7.2}$$

The formulation (7.2) may be more convenient than (7.1), as we may not have a useful characterization of the geodesic distance d_g but we might be able to define continuous curves starting from some given point $p \in M$. For example, in computer vision, it is generally hard to quantify the perceptual distance (according to the human vision system) between two arbitrary images. On the other hand, given some image p, we can define many continuous curves that start at p and end in some other image perceptually close to p: rotations where we let the angle vary continuously, translations where the offset varies continuously, etc.

We could even imagine a utopian scenario where the map ψ manages to induce Euclidean topology, similar to the unfolding of the Swiss roll, in which case we can take it to be an $\mathbb{R}^n \to \mathbb{R}^d$ function where d is the dimension of M. This would yield a variant of adversarial training that comes very close to the L_p version:

$$\theta^\star = \min_\theta \mathbb{E}\left[\max_{\boldsymbol{\eta} \in B_\varepsilon(\psi(X))} \mathcal{L}(\psi^{-1}(\psi(X)+\boldsymbol{\eta}), Y, \theta)\right]. \tag{7.3}$$

In this case, $B_\varepsilon(z)$ is the regular L_p norm ball around $z \in \mathbb{R}^d$. In other words, we are constructing adversarial examples by additively perturbing the latent code $z = \psi(x)$ of the given input x with vectors $\boldsymbol{\eta}$ of bounded L_p norm and then mapping the resulting perturbed code $\tilde{z} = z + \boldsymbol{\eta}$ back to the input space, obtaining $\tilde{x} = \psi^{-1}(\psi(x) + \boldsymbol{\eta})$. Of course, this imposes the additional requirement that ψ be invertible as well, but this is satisfied in the case of auto-encoders and normalizing flows. The works of Zhang et al. [2018b] and Zhang et al. [2019c] essentially follow the spirit of these ideas, but they do not appear to have made any real impact on the adversarial ML community. It would seem that differential geometry is in the same boat as conformal prediction in this particular case: the groundwork has been laid and the ideas have much potential, but they seem to be underappreciated by the broader ML community.

It is important to recognize that, should any of the above ideas be adopted, novel benchmarks also must be developed. Indeed, since many L_p perturbations may lie off the data manifold, they would not be explored by adversarial training methods based on (7.1). Consequently, with respect to current benchmarks such as AutoAttack [Croce and Hein, 2020b], models trained along these lines could still exhibit zero robustness. This could be at least partially mitigated by pairing the models with a detector that signals when inputs lie off the manifold. There is extensive work on such out-of-distribution detection, such as Lee et al. [2018], which could be applied here. On the other hand, Kanbak et al. [2018] have proposed ManiFool, an adversarial attack that explicitly tries to create examples that lie on the original data manifold. Attacks like ManiFool could be used as tractable approximations with which to instantiate (7.1), or they could be employed as benchmarks. Ideally, however, we would want our models to be able to gracefully handle out-of-distribution samples as well, *e.g.*, by rejecting the input. This means that any method for generalizing the L_p robustness problem should be a *true* generalization, in the sense that robustness in the generalized setting (whatever it may be) should imply robustness in the L_p setting. So far, it seems, we have not been able to accomplish this...

7.2.3 Benchmarks

How do we measure progress? The typical scientific answer is deceptively simple: formulate a metric M which takes any method A and outputs a real number $M(A) \in \mathbb{R}$. Method A is then defined to be "better" at solving our problem than method B if and only if $M(A) > M(B)$. As a typical scientific answer, of course, it is also wildly unhelpful in practice, as it deliberately

abstracts away all the important details. How should we choose the metric M? Does there even exist any *single* metric that can capture all variables of interest or do we need a whole collection of them? How do we appropriately define trade-offs between different metrics?

The AML community still has to contend with most of these questions. In particular, we lack good benchmarks. There is AutoAttack [Croce and Hein, 2020b] and RobustBench,[4] and they are certainly useful, but they do not address all of the important benchmarking issues. The current state of the art in benchmarking adversarial defenses at the time of this writing is as follows. We consider at most three data sets — CIFAR-10, CIFAR-100 and ImageNet — and we train models for them we hope will be robust. Then, we run AutoAttack or some other suite of attacks which function within either the L_2 or L_∞ threat model. For CIFAR-10 and CIFAR-100, we set the perturbation budget to $8/255$ in the L_∞ model and $1/2$ in the L_2 case; for ImageNet, we set the L_∞ budget to $4/255$. The L_2 threat model for ImageNet is not typically considered. Then we report standard and robust accuracy scores. There are a few issues with this entrenched methodology I believe should be addressed:

- Who decided these perturbation budgets? The $\varepsilon = 8/255$ in the L_∞ threat model for CIFAR-10 can be traced back to Madry et al. [2017], who observed that they couldn't make their models robust at higher values. However, to my knowledge, these experiments have never been replicated for other models, data sets and attacks, and this value is highly specific to the setting used by Madry et al. [2017]. They certainly weren't advocating that it be used as a general benchmark, but the community adopted it as such nonetheless. Yet it is easy to see that whatever perturbation budget we choose will strongly depend on the data set as well as the context in which the model is deployed. It makes little sense to hold this value fixed to some constant. This sentiment is echoed by Carlini et al. [2019], who argue that we need to look at curves that plot the robust accuracy as a function of the perturbation budget.

- Why only these three data sets? What makes us believe that an adversarial defense that works for CIFAR-10, CIFAR-100 and ImageNet will also work for every other data set? In particular, these data sets are all crafted specifically for image classification. Other domains, such as natural language processing, are neglected despite their obvious importance. This concern becomes especially relevant when we consider that most published defenses often require extensive tuning of hyperparameters tailored to each specific data set. To properly evaluate our defenses, we ought to adopt a much more diverse collection of data sets and tasks on which to test them.

- We need a more diverse set of metrics on which to score adversarial defenses. Standard and robust accuracy scores are obviously important, of course, but there are also other concerns. How much additional data is needed for the defense to work? How many more parameters and hyperparameters must be tuned? How does the defense affect the efficiency of inference with the model? These are all pertinent questions for practitioners that are not typically given much attention in the literature.

I think it would be a major benefit to the field if we started standardizing our methodologies to the point where we could create a single tool (for example, a website similar to RobustBench or a downloadable library such as AutoAttack) that can take any standardized implementation of a proposed defense and run it through a battery of tests where it has to defend different mod-

[4] `https://robustbench.github.io/`. Accessed 2022-01-06.

els trained on different data sets. The output could contain a plethora of information on how the defended model compares to undefended baselines as well as previously published defenses according to all sorts of different metrics, including but not limited to standard and robust accuracy, average inference speed, training time and memory consumption.

7.2.4 Lightweight solutions

An often overlooked aspect of adversarial defenses is its *computational complexity*. Researchers have been able to come up with some remarkably effective defenses over the past few years, but it appears every defense proposed thus far can satisfy at most two of the following three desiderata:

1. Efficiency. The defense should be efficient with respect to the number of parameters of the underlying model, the amount of additional data needed and the perturbation budget.
2. Certifiability. The defense should provide a certificate of robustness, proving (at least with high probability) that successful attacks against it do not exist.
3. Reasonable trade-off between accuracy and robustness. The defense should achieve reasonable levels of robustness without major penalties to standard accuracy.

To the best of my knowledge, no known defense satisfies all three of these properties. At best, we are able to achieve two out of three. Randomized smoothing, for instance, is a certifiable defense that achieves a reasonable trade-off between standard accuracy and robustness, but it is not efficient. The defense requires that, for each input sample x, thousands of Gaussian perturbations $x+\delta$ be computed and evaluated, thus slowing down inference times by several orders of magnitude. Although these queries can be run in parallel to some extent, the slowdown remains significant. Adversarial training is typically not certifiable or not efficient, requiring an adversarial attack be run on each sample within the mini-batches during the full phase of training. One particularly extreme example is the work of Schott et al. [2018], who created one of the most complex adversarial defenses I have ever seen, in the hopes of at least solving this problem for MNIST. Their defense remains seemingly undefeated to this day, but it suffers from two main shortcomings:

- It is overly complicated for a simple data set such as MNIST, and as it is their approach would barely scale to CIFAR-10, let alone ImageNet. It is just not practical.
- Owing to its complexity, properly evaluating robustness becomes more difficult. One of the major conclusions of Tramer et al. [2020] is that simplicity (both in attacking and defending) is a virtue. In particular, complicated defenses tend to have only a few simple components that actually matter for robustness, the others merely serving to confuse and obfuscate.

Therefore, it is highly likely that the defense by Schott et al. [2018] will never be deployed in practice, either because it doesn't scale to larger data sets or because the robustness it is claimed to have has been overestimated due to the complexity of the defense.

There is an important realization that needs to sink in with the AML community: since every DNN appears to be vulnerable to adversarial attacks, and since these attacks seriously undermine the trustworthiness of the models, we should strive to make *every* DNN robust. This means that robustness ought to become as easy to practitioners as the training of the networks itself, with no

noticeable overhead during training or inference and no measurable performance hit with respect to metrics of interest such as accuracy. Eventually, we want to find some set of defenses that can become widely adopted as community standards and integrated into the basic frameworks of machine learning. When a machine learning practitioner uses Keras, for example, to load in a data set, setup a model and run `model.fit()`, they should end up with a network that is as accurate and robust as possible without even having to think about robustness at all. Whatever mechanism exists to make the models robust should be as transparent and invisible as all the other internal house-keeping done by the framework. It should *just work*. That should be the goal of the AML community: to become invisible yet ever-present.

One can of course appreciate that robustness is a hard problem, and despite the significant progress made over the years we are still very far from a general-purpose solution that we could standardize and implement in regular frameworks. I am of course talking about long-term goals for the field, and this should not be taken as some meta-commentary about how the field has somehow failed in any way. It has not. In fact, we have achieved impressive results in a relatively short amount of time given the difficulty of the problem we are studying. But we must not lose sight of the ultimate goal: creating defenses that can be standardized and integrated into regular ML pipelines. At this time, there is no defense that could meet this requirement, either because they do not scale or because they are simply too finicky. Adversarial training, for example, sounds easy enough to implement in theory, and indeed one can get a basic implementation working without too much hassle. However, if one wishes to reach any sort of non-trivial robustness against adaptive attacks, one must be *very* particular about how the training is carried out: choosing any set of hyperparameters that is not entirely optimal can ruin everything. The model might not converge at all anymore, or it might even be *less* robust than its vanilla counterpart. I have been in this position plenty of times, where I sat behind my laptop, screaming internally as I witnessed my fiftieth iteration of an adversarial training implementation crash and burn with 1% robust accuracy against the AutoAttack suite. Turned out I had to adjust the learning rate scheduling ever so slightly, or tweak the step size of the attack. Or maybe it was pure dumb luck that eventually fixed it. Who knows? Our understanding of how adversarial training works, or indeed any robust training mechanism, is still very much lacking. Things go wrong far more often than not, and we rarely know how to fix it. If we were to silently incorporate adversarial training into Keras now, people would stop using Keras altogether. What we need are defenses that work well under a variety of circumstances without the need for excessive tuning of hyperparameters.

7.2.5 Conformal prediction

The majority of novel contributions in this work are squarely situated in the field of conformal prediction (CP). First proposed by Vovk et al. [2005], CP focuses mainly on quantifying the reliability of predictions made by statistical models. At a conceptual level, the premise of CP seems ideally suited to the development of adversarial defenses, since adversarial attacks by definition construct inputs on which the target model is unreliable. Researchers in CP have constructed various efficient algorithms for ascertaining the reliability of model predictions, some of which are in particular compatible with DNNs. The IVAP [Vovk et al., 2015] is one such example for which reference implementations are publicly available [Toccaceli, 2017].

Nevertheless, despite its promising connections to adversarial robustness, CP seems to be largely unknown to the broader ML community. The only other work we have been able to find that

explicitly takes inspiration from CP in the context of adversarial defense is Papernot and McDaniel [2018]. At the time of this writing, this work stands at 359 citations according to Google Scholar, but almost none of the works that cite it make use of conformal prediction themselves (at least not in the context of robustness). The fact that we have been able to construct several reasonably strong and efficient defenses based on CP in the course of this single Ph.D. suggests that far more can be done if a larger concerted effort were made by the community. This ought to be one of the main take-aways of our research: conformal prediction methods hold much promise in the field of adversarial ML and deserve greater attention from the community.

Bibliography

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, 2015. URL `https://www.tensorflow.org/`. Software available from tensorflow.org.

P-A Absil, Robert Mahony, and Rodolphe Sepulchre. *Optimization algorithms on matrix manifolds*. Princeton University Press, 2009.

Shun-ichi Amari. *Information geometry and its applications*, volume 194. Springer, 2016.

Amnesty International. Surveillance giants: How the business model of Google and Facebook threatens human rights, 2019.

Alexandr Andoni, Piotr Indyk, Thijs Laarhoven, Ilya Razenshteyn, and Ludwig Schmidt. Practical and optimal LSH for angular distance. In *Advances in Neural Information Processing Systems*, pages 1225–1233, 2015.

Maksym Andriushchenko and Matthias Hein. Provably robust boosted decision stumps and trees against adversarial attacks. In *Advances in Neural Information Processing Systems*, pages 12997–13008, 2019.

Maksym Andriushchenko, Francesco Croce, Nicolas Flammarion, and Matthias Hein. Square attack: a query-efficient black-box adversarial attack via random search. In *European Conference on Computer Vision*, pages 484–501. Springer, 2020.

Cem Anil, James Lucas, and Roger Grosse. Sorting out Lipschitz function approximation. In *International Conference on Machine Learning*, pages 291–301. PMLR, 2019.

Martin Anthony and Peter L Bartlett. *Neural network learning: Theoretical foundations*, volume 9. Cambridge University Press, 1999.

Raman Arora, Amitabh Basu, Poorya Mianjy, and Anirbit Mukherjee. Understanding deep neural networks with rectified linear units. *arXiv preprint arXiv:1611.01491*, 2016.

Sanjeev Arora and Boaz Barak. *Computational complexity: a modern approach*. Cambridge University Press, 2009.

Anish Athalye and Nicholas Carlini. On the robustness of the CVPR 2018 white-box adversarial example defenses. *arXiv preprint arXiv:1804.03286*, 2018.

Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *International conference on machine learning*, pages 274–283. PMLR, 2018a.

Anish Athalye, Logan Engstrom, Andrew Ilyas, and Kevin Kwok. Synthesizing robust adversarial examples. In *International conference on machine learning*, pages 284–293. PMLR, 2018b.

Pranjal Awasthi, Himanshu Jain, Ankit Singh Rawat, and Aravindan Vijayaraghavan. Adversarial robustness via robust low rank representations. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 11391–11403. Curran Associates, Inc., 2020. URL `https://proceedings.neurips.cc/paper/2020/file/837a7924b8c0aa866e41b2721f66135c-Paper.pdf`.

Lei Jimmy Ba and Rich Caruana. Do deep nets really need to be deep? *arXiv preprint arXiv:1312.6184*, 2013.

Shumeet Baluja and Ian Fischer. Adversarial transformation networks: Learning to generate adversarial examples. *arXiv preprint arXiv:1703.09387*, 2017.

Ben Barrett, Alexander Camuto, Matthew Willetts, and Tom Rainforth. Certifiably robust variational autoencoders. *arXiv preprint arXiv:2102.07559*, 2021.

Vahid Behzadan and Arslan Munir. Whatever does not kill deep reinforcement learning, makes it stronger. *arXiv preprint arXiv:1712.09344*, 2017.

CB Bell. Mutual information and maximal correlation as measures of dependence. *The Annals of Mathematical Statistics*, pages 587–595, 1962.

Aharon Ben-Tal and Arkadi Nemirovski. Robust optimization–methodology and applications. *Mathematical programming*, 92(3):453–480, 2002.

Emily M. Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. On the dangers of stochastic parrots: Can language models be too big? In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, FAccT '21, page 610–623, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450383097. doi: 10.1145/3442188.3445922. URL `https://doi.org/10.1145/3442188.3445922`.

Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8): 1798–1828, 2013.

Battista Biggio and Fabio Roli. Wild patterns: Ten years after the rise of adversarial machine learning. *Pattern Recognition*, 84:317–331, 2018.

Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Šrndić, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. Evasion attacks against machine learning at test time. In *Joint European conference on machine learning and knowledge discovery in databases*, pages 387–402. Springer, 2013.

Abeba Birhane and Vinay Uday Prabhu. Large image datasets: A pyrrhic win for computer vision? In *2021 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1536–1546. IEEE, 2021.

Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. YOLOv4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*, 2020.

Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge University Press, 2004.

Wieland Brendel, Jonas Rauber, and Matthias Bethge. Decision-based adversarial attacks: Reliable attacks against black-box machine learning models. *arXiv preprint arXiv:1712.04248*, 2017.

Wieland Brendel, Jonas Rauber, Matthias Kümmerer, Ivan Ustyuzhaninov, and Matthias Bethge. Accurate, reliable and fast robustness evaluation. *arXiv preprint arXiv:1907.01003*, 2019.

J Bretagnolle and C Huber. Estimation des densités: risque minimax. *Zeitschrift für Wahrscheinlichkeitstheorie und verwandte Gebiete*, 47(2):119–137, 1979.

Andrew Brock, Theodore Lim, James M Ritchie, and Nick Weston. Neural photo editing with introspective adversarial networks. *arXiv preprint arXiv:1609.07093*, 2016.

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

Antoine Buetti-Dinh, Vanni Galli, Sören Bellenberg, Olga Ilie, Malte Herold, Stephan Christel, Mariia Boretska, Igor V Pivkin, Paul Wilmes, Wolfgang Sand, et al. Deep neural networks outperform human expert's capacity in characterizing bioleaching bacterial biofilm composition. *Biotechnology Reports*, 22:e00321, 2019.

Flavio Calmon, Ali Makhdoumi, Muriel Médard, Mayank Varia, Mark Christiansen, and Ken R Duffy. Principal inertia components and applications. *IEEE Transactions on Information Theory*, 63(8):5011–5038, 2017.

Alexander Camuto, Matthew Willetts, Stephen Roberts, Chris Holmes, and Tom Rainforth. Towards a theoretical understanding of the robustness of variational autoencoders. In *International Conference on Artificial Intelligence and Statistics*, pages 3565–3573. PMLR, 2021.

Nicholas Carlini. Is AmI (Attacks meet Interpretability) robust to adversarial examples? *arXiv preprint arXiv:1902.02322*, 2019a.

Nicholas Carlini. A critique of the DeepSec platform for security analysis of deep learning models. *arXiv preprint arXiv:1905.07112*, 2019b.

Nicholas Carlini and David Wagner. Defensive distillation is not robust to adversarial examples. *arXiv preprint arXiv:1607.04311*, 2016.

Nicholas Carlini and David Wagner. Adversarial examples are not easily detected: Bypassing ten detection methods. In *Proceedings of the 10th ACM workshop on artificial intelligence and security*, pages 3–14, 2017a.

Nicholas Carlini and David Wagner. MagNet and "Efficient Defenses Against Adversarial Attacks" are not robust to adversarial examples. *arXiv preprint arXiv:1711.08478*, 2017b.

Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *IEEE Symposium on Security and Privacy*, pages 39–57. IEEE, 2017c.

Nicholas Carlini, Anish Athalye, Nicolas Papernot, Wieland Brendel, Jonas Rauber, Dimitris Tsipras, Ian Goodfellow, Aleksander Madry, and Alexey Kurakin. On evaluating adversarial robustness. *arXiv preprint arXiv:1902.06705*, 2019.

Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. SMOTE: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16: 321–357, 2002.

Hongge Chen, Huan Zhang, Si Si, Yang Li, Duane Boning, and Cho-Jui Hsieh. Robustness verification of tree-based models. *Advances in Neural Information Processing Systems*, 32, 2019.

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 *Proceedings of the 10th ACM workshop on artificial intelligence and security*, pages 15–26, 2017.

François Chollet. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1251–1258, 2017.

Francois Chollet et al. Keras. `https://github.com/fchollet/keras`, 2015.

Moustapha Cisse, Piotr Bojanowski, Edouard Grave, Yann Dauphin, and Nicolas Usunier. Parseval networks: Improving robustness to adversarial examples. In *International Conference on Machine Learning*, pages 854–863. PMLR, 2017a.

Moustapha M Cisse, Yossi Adi, Natalia Neverova, and Joseph Keshet. Houdini: Fooling deep structured visual and speech recognition models with adversarial examples. *Advances in Neural Information Processing Systems*, 30, 2017b.

Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (ELUs). *arXiv preprint arXiv:1511.07289*, 2015.

Jeremy Cohen, Elan Rosenfeld, and Zico Kolter. Certified adversarial robustness via randomized smoothing. In *International Conference on Machine Learning*, pages 1310–1320. PMLR, 2019.

Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3): 273–297, 1995.

Thomas M Cover. *Elements of information theory*. John Wiley & Sons, 1999.

Kate Crawford. *The atlas of AI*. Yale University Press, 2021.

Antonia Creswell, Tom White, Vincent Dumoulin, Kai Arulkumaran, Biswa Sengupta, and Anil A Bharath. Generative adversarial networks: An overview. *IEEE Signal Processing Magazine*, 35(1):53–65, 2018.

Francesco Croce and Matthias Hein. Minimally distorted adversarial examples with a fast adaptive boundary attack. In *International Conference on Machine Learning*, pages 2196–2205. PMLR, 2020a.

Francesco Croce and Matthias Hein. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *International Conference on Machine Learning*, pages 2206–2216. PMLR, 2020b.

Francesco Croce, Maksym Andriushchenko, and Matthias Hein. Provable robustness of ReLU networks via maximization of linear regions. In *the 22nd International Conference on Artificial Intelligence and Statistics*, pages 2057–2066. PMLR, 2019.

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*, 128(4):1028–1046, 2020.

Nilesh Dalvi, Pedro Domingos, Sumit Sanghai, and Deepak Verma. Adversarial classification. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 99–108, 2004.

Michael David-Fox. The myth of the Soviet Potemkin village. `https://histoire.ens.psl.eu/IMG/file/Coeure/David-Fox%20Potemkin%20villages.pdf`, 2013.

J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

René Descartes. Animals are machines. *Animal rights and human obligations*, pages 13–19, 1989.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Dimitrios I Diochnos, Saeed Mahloujifar, and Mohammad Mahmoody. Lower bounds for adversarially robust PAC learning. *arXiv preprint arXiv:1906.05815*, 2019.

Ann-Kathrin Dombrowski, Maximillian Alber, Christopher Anders, Marcel Ackermann, Klaus-Robert Müller, and Pan Kessel. Explanations can be manipulated and geometry is to blame. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL `https://proceedings.neurips.cc/paper/2019/file/bb836c01cdc9120a9c984c525e4b1a4a-Paper.pdf`.

Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Hang Su, Jun Zhu, Xiaolin Hu, and Jianguo Li. Boosting adversarial attacks with momentum. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9185–9193, 2018.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Timothy Dozat. Incorporating Nesterov momentum into Adam. *ICLR*, 2016.

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(7), 2011.

Javid Ebrahimi, Daniel Lowd, and Dejing Dou. On adversarial examples for character-level neural machine translation. *arXiv preprint arXiv:1806.09030*, 2018.

Stefan Elfwing, Eiji Uchibe, and Kenji Doya. Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. *Neural Networks*, 107:3–11, 2018.

Jeremy Elson, John (JD) Douceur, Jon Howell, and Jared Saul. Asirra: A CAPTCHA that exploits interest-aligned manual image categorization. In *Proceedings of 14th ACM Conference on Computer and Communications Security (CCS)*. Association for Computing Machinery, Inc., 10 2007.

Kevin Eykholt, Ivan Evtimov, Earlence Fernandes, Bo Li, Amir Rahmati, Chaowei Xiao, Atul Prakash, Tadayoshi Kohno, and Dawn Song. Robust physical-world attacks on deep learning visual classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1625–1634, 2018.

Alhussein Fawzi, Seyed-Mohsen Moosavi-Dezfooli, and Pascal Frossard. The robustness of deep networks: A geometrical perspective. *IEEE Signal Processing Magazine*, 34(6):50–62, 2017. doi: 10.1109/MSP.2017.2740965.

Alhussein Fawzi, Hamza Fawzi, and Omar Fawzi. Adversarial vulnerability for any classifier. In *Advances in Neural Information Processing Systems*, pages 1178–1187, 2018.

Mahyar Fazlyab, Alexander Robey, Hamed Hassani, Manfred Morari, and George J Pappas. Efficient and accurate estimation of Lipschitz constants for deep neural networks. *arXiv preprint arXiv:1906.04893*, 2019.

Charles Fefferman, Sanjoy Mitter, and Hariharan Narayanan. Testing the manifold hypothesis. *Journal of the American Mathematical Society*, 29(4):983–1049, 2016.

Li Fei-Fei, Jia Deng, and Kai Li. ImageNet: Constructing a large-scale image database. *Journal of vision*, 9(8):1037–1037, 2009.

Kunihiko Fukushima and Sei Miyake. Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition. In *Competition and cooperation in neural nets*, pages 267–285. Springer, 1982.

Yarin Gal. *Uncertainty in Deep Learning*. PhD thesis, University of Cambridge, 2016.

Yarin Gal and Zoubin Ghahramani. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059. PMLR, 2016.

Alexander Gammerman and Vladimir Vovk. Hedging predictions in machine learning. *The Computer Journal*, 50(2):151–163, 2007.

Ruize Gao, Feng Liu, Jingfeng Zhang, Bo Han, Tongliang Liu, Gang Niu, and Masashi Sugiyama. Maximum mean discrepancy is aware of adversarial attacks. *arXiv preprint arXiv:2010.11415*, 2020.

Robert Geirhos, Carlos R Medina Temme, Jonas Rauber, Heiko H Schütt, Matthias Bethge, and Felix A Wichmann. Generalisation in humans and deep neural networks. *arXiv preprint arXiv:1808.08750*, 2018.

Robert Geirhos, Jörn-Henrik Jacobsen, Claudio Michaelis, Richard Zemel, Wieland Brendel, Matthias Bethge, and Felix A Wichmann. Shortcut learning in deep neural networks. *Nature Machine Intelligence*, 2(11):665–673, 2020.

Carles Gelada and Jacob Buckman. Bayesian neural networks need not concentrate. `https://jacobbuckman.com/2020-01-22-bayesian-neural-networks-need-not-concentrate/`, 2020.

Justin Gilmer, Ryan P Adams, Ian Goodfellow, David Andersen, and George E Dahl. Motivating the rules of the game for adversarial example research. *arXiv preprint arXiv:1807.06732*, 2018.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. JMLR Workshop and Conference Proceedings, 2010.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 315–323. JMLR Workshop and Conference Proceedings, 2011.

George Gondim-Ribeiro, Pedro Tabacof, and Eduardo Valle. Adversarial attacks on variational autoencoders. *arXiv preprint arXiv:1806.04646*, 2018.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

Henry Gouk, Eibe Frank, Bernhard Pfahringer, and Michael J Cree. Regularisation of neural networks by enforcing lipschitz continuity. *Machine Learning*, 110(2):393–416, 2021.

Pascale Gourdeau, Varun Kanade, Marta Kwiatkowska, and James Worrell. On the hardness of robust classification. In *Advances in Neural Information Processing Systems*, pages 7444–7453, 2019.

Sven Gowal, Chongli Qin, Jonathan Uesato, Timothy Mann, and Pushmeet Kohli. Uncovering the limits of adversarial training against norm-bounded adversarial examples. *arXiv preprint arXiv:2010.03593*, 2020.

Michael J Greenacre. Correspondence analysis. *Wiley Interdisciplinary Reviews: Computational Statistics*, 2(5):613–619, 2010.

Kathrin Grosse, Praveen Manoharan, Nicolas Papernot, Michael Backes, and Patrick McDaniel. On the (statistical) detection of adversarial examples. *arXiv preprint arXiv:1702.06280*, 2017.

Charles Grumer, Jonathan Peck, Femi Olumofin, Anderson Nascimento, and Martine De Cock. Hardening DGA classifiers utilizing IVAP. In *2019 IEEE International Conference on Big Data (Big Data)*, pages 6063–6065. IEEE, 2019.

Shixiang Gu and Luca Rigazio. Towards deep neural network architectures robust to adversarial examples. *arXiv preprint arXiv:1412.5068*, 2014.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *International Conference on Machine Learning*, pages 1321–1330. PMLR, 2017.

Chuan Guo, Jacob Gardner, Yurong You, Andrew Gordon Wilson, and Kilian Weinberger. Simple black-box adversarial attacks. In *International Conference on Machine Learning*, pages 2484–2493. PMLR, 2019.

Karen Hao. Training a single AI model can emit as much carbon as five cars in their lifetimes. *MIT Technology Review*, 2019. URL `https://www.technologyreview.com/2019/06/06/239031/training-a-single-ai-model-can-emit-as-much-carbon-as-five-cars-in-their-lifetimes/`.

David R Hardoon, Sandor Szedmak, and John Shawe-Taylor. Canonical correlation analysis: An overview with application to learning methods. *Neural computation*, 16(12):2639–2664, 2004.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, December 2015.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

Michael T Heath. *Scientific Computing: An Introductory Survey, Revised Second Edition*. SIAM, 2018.

Matthias Hein and Maksym Andriushchenko. Formal guarantees on the robustness of a classifier against adversarial manipulation. *arXiv preprint arXiv:1705.08475*, 2017.

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. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 41–50, 2019.

Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. *arXiv preprint arXiv:1903.12261*, 2019.

Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (GELUs). *arXiv preprint arXiv:1606.08415*, 2016.

Dan Hendrycks, Steven Basart, Norman Mu, Saurav Kadavath, Frank Wang, Evan Dorundo, Rahul Desai, Tyler Zhu, Samyak Parajuli, Mike Guo, et al. The many faces of robustness: A critical analysis of out-of-distribution generalization. *arXiv preprint arXiv:2006.16241*, 2020.

Juyeon Heo, Sunghwan Joo, and Taesup Moon. Fooling neural network interpretations via adversarial model manipulation. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL `https://proceedings.neurips.cc/paper/2019/file/7fea637fd6d02b8f0adf6f7dc36aed93-Paper.pdf`.

Danny Hernandez and Tom B Brown. Measuring the algorithmic efficiency of neural networks. *arXiv preprint arXiv:2005.04305*, 2020.

Geoffrey Hinton, Nitish Srivastava, and Kevin Swersky. Neural networks for machine learning. `http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf`, 2012.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *arXiv preprint arxiv:2006.11239*, 2020.

Tin Kam Ho. Random decision forests. In *Proceedings of 3rd international conference on document analysis and recognition*, volume 1, pages 278–282. IEEE, 1995.

Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi, Jürgen Schmidhuber, et al. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies, 2001.

Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

Hsiang Hsu, Salman Salamatian, and Flavio P Calmon. Correspondence analysis using neural networks. *arXiv preprint arXiv:1902.07828*, 2019.

Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4700–4708, 2017.

Gary B Huang, Marwan Mattar, Tamara Berg, and Eric Learned-Miller. Labeled faces in the wild: A database forstudying face recognition in unconstrained environments. In *Workshop on faces in'Real-Life'Images: detection, alignment, and recognition*, 2008.

Hanxun Huang, Xingjun Ma, Sarah Monazam Erfani, James Bailey, and Yisen Wang. Unlearnable examples: Making personal data unexploitable. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=iAmZUo0DxC0`.

Borja Ibarz, Jan Leike, Tobias Pohlen, Geoffrey Irving, Shane Legg, and Dario Amodei. Reward learning from human preferences and demonstrations in Atari. *arXiv preprint arXiv:1811.06521*, 2018.

Andrew Ilyas, Logan Engstrom, Anish Athalye, and Jessy Lin. Black-box adversarial attacks with limited queries and information. In *International Conference on Machine Learning*, pages 2137–2146. PMLR, 2018.

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*, pages 125–136, 2019.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015.

Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. Averaging weights leads to wider optima and better generalization. *arXiv preprint arXiv:1803.05407*, 2018.

Paul Jaccard. The distribution of the flora in the alpine zone. *New Phytologist*, 11(2):37–50, 1912.

Himanshu Jain, Venkatesh Balasubramanian, Bhanu Chunduri, and Manik Varma. Slice: Scalable linear extreme classifiers trained on 100 million labels for related searches. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, pages 528–536. ACM, 2019.

Uyeong Jang, Xi Wu, and Somesh Jha. Objective metrics and gradient descent algorithms for adversarial examples in machine learning. In *Proceedings of the 33rd Annual Computer Security Applications Conference*, pages 262–277, 2017.

Matt Jordan and Alexandros G Dimakis. Exactly computing the local Lipschitz constant of RELU networks. *arXiv preprint arXiv:2003.01219*, 2020.

Michael I Jordan, Zoubin Ghahramani, Tommi S Jaakkola, and Lawrence K Saul. An introduction to variational methods for graphical models. *Machine learning*, 37(2):183–233, 1999.

John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with AlphaFold. *Nature*, page 1, 2021.

Samira Ebrahimi Kahou, Xavier Bouthillier, Pascal Lamblin, Caglar Gulcehre, Vincent Michalski, Kishore Konda, Sébastien Jean, Pierre Froumenty, Yann Dauphin, Nicolas Boulanger-Lewandowski, et al. Emonets: Multimodal deep learning approaches for emotion recognition in video. *Journal on Multimodal User Interfaces*, 10(2):99–111, 2016.

Can Kanbak, Seyed-Mohsen Moosavi-Dezfooli, and Pascal Frossard. Geometric robustness of deep networks: analysis and improvement. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4441–4449, 2018.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Michael J Kearns, Umesh Virkumar Vazirani, and Umesh Vazirani. *An introduction to computational learning theory*. MIT press, 1994.

Henry J Kelley. Gradient theory of optimal flight paths. *Ars Journal*, 30(10):947–954, 1960.

Norbert L Kerr. HARKing: Hypothesizing after the results are known. *Personality and social psychology review*, 2(3):196–217, 1998.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Diederik P Kingma and Max Welling. Auto-encoding variational Bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. Self-normalizing neural networks. In *Proceedings of the 31st international conference on Neural Information Processing Systems*, pages 972–981, 2017.

B Klaus and K Strimmer. fdrtool: Estimation of (local) false discovery rates and higher criticism. *R package version*, 1:15, 2015.

Will Knight. Researchers blur faces that launched a thousand algorithms. `https://www.wired.com/story/researchers-blur-faces-launched-thousand-algorithms/`, 2021.

Ivan Kobyzev, Simon JD Prince, and Marcus A Brubaker. Normalizing flows: An introduction and review of current methods. *IEEE transactions on pattern analysis and machine intelligence*, 43(11):3964–3979, 2020.

Zelun Kong, Junfeng Guo, Ang Li, and Cong Liu. PhysGAN: Generating physical-world-resilient adversarial examples for autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14254–14263, 2020.

Jernej Kos, Ian Fischer, and Dawn Song. Adversarial examples for generative models. In *2018 ieee security and privacy workshops (spw)*, pages 36–42. IEEE, 2018.

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, 25: 1097–1105, 2012.

Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial machine learning at scale. *arXiv preprint arXiv:1611.01236*, 2016.

Alexey Kurakin, Ian Goodfellow, Samy Bengio, Yinpeng Dong, Fangzhou Liao, Ming Liang, Tianyu Pang, Jun Zhu, Xiaolin Hu, Cihang Xie, et al. Adversarial attacks and defences competition. In *The NIPS'17 Competition: Building Intelligent Systems*, pages 195–231. Springer, 2018.

Peter Langenberg, Emilio Balda, Arash Behboodi, and Rudolf Mathar. On the robustness of support vector machines against adversarial examples. In *2019 13th International Conference on Signal Processing and Communication Systems (ICSPCS)*, pages 1–6. IEEE, 2019.

Yann LeCun. The MNIST database of handwritten digits. `http://yann.lecun.com/exdb/mnist/`, 1998.

Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Olivier Ledoit and Michael Wolf. A well-conditioned estimator for large-dimensional covariance matrices. *Journal of multivariate analysis*, 88(2):365–411, 2004.

Jeffrey M Lee. *Manifolds and Differential Geometry*. Graduate studies in mathematics. American Mathematical Society, Providence, RI, January 2010.

Kimin Lee, Kibok Lee, Honglak Lee, and Jinwoo Shin. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. *Advances in Neural Information Processing Systems*, 31, 2018.

Jeremy Lent. *The web of meaning*. New Society Publishers, 2021.

Alexander Levine and Soheil Feizi. (De)Randomized smoothing for certifiable defense against patch attacks. *Advances in Neural Information Processing Systems*, 33:6465–6475, 2020.

Bai Li, Changyou Chen, Wenlin Wang, and Lawrence Carin. Certified adversarial robustness with additive noise. *Advances in Neural Information Processing Systems*, 32, 2019.

Dong C Liu and Jorge Nocedal. On the limited memory BFGS method for large scale optimization. *Mathematical programming*, 45(1):503–528, 1989.

Weiyang Liu, Yandong Wen, Zhiding Yu, and Meng Yang. Large-margin softmax loss for convolutional neural networks. In *ICML*, volume 2, page 7, 2016a.

Yanpei Liu, Xinyun Chen, Chang Liu, and Dawn Song. Delving into transferable adversarial examples and black-box attacks. *arXiv preprint arXiv:1611.02770*, 2016b.

Daniel Lowd and Christopher Meek. Adversarial learning. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 641–647, 2005.

Zhou Lu, Hongming Pu, Feicheng Wang, Zhiqiang Hu, and Liwei Wang. The expressive power of neural networks: A view from the width. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 6232–6240, 2017.

Xingjun Ma, Bo Li, Yisen Wang, Sarah M Erfani, Sudanthi Wijewickrema, Grant Schoenebeck, Dawn Song, Michael E Houle, and James Bailey. Characterizing adversarial subspaces using local intrinsic dimensionality. *arXiv preprint arXiv:1801.02613*, 2018.

Yunqian Ma and Yun Fu. *Manifold learning theory and applications*, volume 434. CRC press Boca Raton, 2012.

Andrew L Maas, Awni Y Hannun, Andrew Y Ng, et al. Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, volume 30, page 3. Citeseer, 2013.

Wesley J Maddox, Pavel Izmailov, Timur Garipov, Dmitry P Vetrov, and Andrew Gordon Wilson. A simple baseline for Bayesian uncertainty in deep learning. *Advances in Neural Information Processing Systems*, 32, 2019.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

Anuran Makur, Fabián Kozynski, Shao-Lun Huang, and Lizhong Zheng. An efficient algorithm for information decomposition and extraction. In *2015 53rd Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 972–979. IEEE, 2015.

Stéphane Mallat. Understanding deep convolutional networks. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 374(2065):20150203, 2016.

Valery Manokhin. Multi-class probabilistic classification using inductive and cross Venn-Abers predictors. In *Conformal and Probabilistic Prediction and Applications*, pages 228–240. PMLR, 2017.

Gary Marcus. Deep learning: A critical appraisal. *arXiv preprint arXiv:1801.00631*, 2018.

Gary Marcus and Ernest Davis. *Rebooting AI: Building artificial intelligence we can trust*. Vintage, 2019.

Alexander Meinke and Matthias Hein. Towards neural networks that provably know when they don't know. *arXiv preprint arXiv:1909.12180*, 2019.

Marco Melis, Ambra Demontis, Battista Biggio, Gavin Brown, Giorgio Fumera, and Fabio Roli. Is deep learning safe for robot vision? Adversarial examples against the iCub humanoid. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 751–759, 2017.

Omar Montasser, Steve Hanneke, and Nathan Srebro. VC classes are adversarially robustly learnable, but only improperly. In *Conference on Learning Theory*, pages 2512–2530. PMLR, 2019.

Guido F Montufar, Razvan Pascanu, Kyunghyun Cho, and Yoshua Bengio. On the number of linear regions of deep neural networks. *Advances in Neural Information Processing Systems*, 27, 2014.

Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2574–2582, 2016.

Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. Universal adversarial perturbations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1765–1773, 2017.

Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Jonathan Uesato, and Pascal Frossard. Robustness via curvature regularization, and vice versa. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

Laurence Moroney. Rock, paper, scissors dataset. `http://laurencemoroney.com/rock-paper-scissors-dataset`, feb 2019.

Kevin P Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.

Kevin P. Murphy. *Probabilistic Machine Learning: An introduction*. MIT Press, 2022. URL `probml.ai`.

Mahdi Pakdaman Naeini, Gregory Cooper, and Milos Hauskrecht. Obtaining well calibrated probabilities using Bayesian binning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29, 2015.

Thomas Nagel. *The view from nowhere*. oxford university press, 1989.

Nina Narodytska and Shiva Prasad Kasiviswanathan. Simple black-box adversarial perturbations for deep networks. *arXiv preprint arXiv:1612.06299*, 2016.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011.

Anh Nguyen, Jason Yosinski, and Jeff Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 427–436, 2015.

Quynh Nguyen, Mahesh Chandra Mukkamala, and Matthias Hein. Neural networks should be wide enough to learn disconnected decision regions. In *International Conference on Machine Learning*, pages 3740–3749. PMLR, 2018.

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. *CoRR*, 1807.01069, 2018. URL `https://arxiv.org/pdf/1807.01069`.

Brian A Nosek, Charles R Ebersole, Alexander C DeHaven, and David T Mellor. The preregistration revolution. *Proceedings of the National Academy of Sciences*, 115(11):2600–2606, 2018.

Cathy O'Neil. *Weapons of math destruction: How big data increases inequality and threatens democracy*. Crown, 2016.

Utku Ozbulak, Arnout Van Messem, and Wesley De Neve. Not all adversarial examples require a complex defense: Identifying over-optimized adversarial examples with IQR-based logit thresholding. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2019.

Tianyu Pang, Chao Du, Yinpeng Dong, and Jun Zhu. Towards robust detection of adversarial examples. In *Advances in Neural Information Processing Systems*, pages 4579–4589, 2018.

Nicolas Papernot and Patrick McDaniel. Deep *k*-nearest neighbors: Towards confident, interpretable and robust deep learning. *arXiv preprint arXiv:1803.04765*, 2018.

Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. In *2016 IEEE European symposium on security and privacy (EuroS&P)*, pages 372–387. IEEE, 2016a.

Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a defense to adversarial perturbations against deep neural networks. In *2016 IEEE symposium on security and privacy (SP)*, pages 582–597. IEEE, 2016b.

Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia conference on computer and communications security*, pages 506–519, 2017.

Jonathan Peck, Joris Roels, Bart Goossens, and Yvan Saeys. Lower bounds on the robustness to adversarial perturbations. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL `https://proceedings.neurips.cc/paper/2017/file/298f95e1bf9136124592c8d4825a06fc-Paper.pdf`.

Jonathan Peck, Bart Goossens, and Yvan Saeys. Calibrated multi-probabilistic prediction as a defense against adversarial attacks. In *Artificial Intelligence and Machine Learning*, pages 85–125. Springer, 2019a.

Jonathan Peck, Claire Nie, Raaghavi Sivaguru, Charles Grumer, Femi Olumofin, Bin Yu, Anderson Nascimento, and Martine De Cock. CharBot: a simple and effective method for evading DGA classifiers. *IEEE Access*, 7:91759–91771, 2019b.

Jonathan Peck, Bart Goossens, and Yvan Saeys. Detecting adversarial manipulation using inductive Venn-ABERS predictors. *Neurocomputing*, 416:202–217, 2020.

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*, 12:2825–2830, 2011.

Rafael Pinot, Laurent Meunier, Florian Yger, Cédric Gouy-Pailler, Yann Chevaleyre, and Jamal Atif. On the robustness of randomized classifiers to adversarial examples. *arXiv preprint arXiv:2102.10875*, 2021.

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. *arXiv preprint arXiv:2106.09947*, 2021.

John Platt et al. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers*, 10(3):61–74, 1999.

Daniel Plohmann, Khaled Yakdan, Michael Klatt, Johannes Bader, and Elmar Gerhards-Padilla. A comprehensive measurement study of domain generating malware. In *25th USENIX Security Symposium (USENIX Security 16)*, pages 263–278, 2016.

Ben Poole, Subhaneil Lahiri, Maithra Raghu, Jascha Sohl-Dickstein, and Surya Ganguli. Exponential expressivity in deep neural networks through transient chaos. *Advances in Neural Information Processing Systems*, 29, 2016.

Omid Poursaeed, Isay Katsman, Bicheng Gao, and Serge Belongie. Generative adversarial perturbations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4422–4431, 2018.

David Price, Stefan Knerr, Léon Personnaz, and Gérard Dreyfus. Pairwise neural network classifiers with probabilistic outputs. *Advances in Neural Information Processing Systems*, 7, 1994.

Yao Qin, Xuezhi Wang, Alex Beutel, and Ed Chi. Improving calibration through the relationship with adversarial robustness. *Advances in Neural Information Processing Systems*, 34, 2021.

R Core Team. R: A language and environment for statistical computing, 2015.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. *Preprint*, 2018.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Aditi Raghunathan, Jacob Steinhardt, and Percy Liang. Certified defenses against adversarial examples. *arXiv preprint arXiv:1801.09344*, 2018.

Ali Rahmati, Seyed-Mohsen Moosavi-Dezfooli, Pascal Frossard, and Huaiyu Dai. GeoDA: a geometric framework for black-box adversarial attacks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8446–8455, 2020.

Ted Ralphs, Yuji Shinano, Timo Berthold, and Thorsten Koch. Parallel solvers for mixed integer linear optimization. In *Handbook of parallel constraint reasoning*, pages 283–336. Springer, 2018.

Prajit Ramachandran, Barret Zoph, and Quoc V Le. Searching for activation functions. *arXiv preprint arXiv:1710.05941*, 2017.

Hannah Rashkin, Eric Michael Smith, Margaret Li, and Y-Lan Boureau. Towards empathetic open-domain conversation models: A new benchmark and dataset. *arXiv preprint arXiv:1811.00207*, 2018.

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. doi: 10.21105/joss.02607. URL `https://doi.org/10.21105/joss.02607`.

Sylvestre-Alvise Rebuffi, Sven Gowal, Dan A Calian, Florian Stimberg, Olivia Wiles, and Timothy Mann. Fixing data augmentation to improve adversarial robustness. *arXiv preprint arXiv:2103.01946*, 2021.

Joseph Redmon and Ali Farhadi. YOLO9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7263–7271, 2017.

Alfréd Rényi. On measures of dependence. *Acta Mathematica Hungarica*, 10(3-4):441–451, 1959.

Leslie Rice, Eric Wong, and Zico Kolter. Overfitting in adversarially robust deep learning. In *International Conference on Machine Learning*, pages 8093–8104. PMLR, 2020.

Michael Roberts, Derek Driggs, Matthew Thorpe, Julian Gilbey, Michael Yeung, Stephan Ursprung, Angelica I Aviles-Rivero, Christian Etmann, Cathal McCague, Lucian Beer, et al. Common pitfalls and recommendations for using machine learning to detect and prognosticate for COVID-19 using chest radiographs and CT scans. *Nature Machine Intelligence*, 3(3): 199–217, 2021.

Herbert Roitblat. *Algorithms are not enough*. MIT Press, 2020.

Kevin Roth, Yannic Kilcher, and Thomas Hofmann. The odds are odd: A statistical test for detecting adversarial examples. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 5498–5507. PMLR, 6 2019. URL `https://proceedings.mlr.press/v97/roth19a.html`.

Itay Safran and Ohad Shamir. Depth-width tradeoffs in approximating natural functions with neural networks. In *International Conference on Machine Learning*, pages 2979–2987. PMLR, 2017.

Tim Salimans and Durk P Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. *Advances in Neural Information Processing Systems*, 29:901–909, 2016.

Hadi Salman, Andrew Ilyas, Logan Engstrom, Ashish Kapoor, and Aleksander Madry. Do adversarially robust imagenet models transfer better? *Advances in Neural Information Processing Systems*, 33:3533–3545, 2020a.

Hadi Salman, Mingjie Sun, Greg Yang, Ashish Kapoor, and J. Zico Kolter. Denoised smoothing: A provable defense for pretrained classifiers. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 21945–21957. Curran Associates, Inc., 2020b. URL `https://proceedings.neurips.cc/paper/2020/file/f9fd2624beefbc7808e4e405d73f57ab-Paper.pdf`.

Shibani Santurkar, Andrew Ilyas, Dimitris Tsipras, Logan Engstrom, Brandon Tran, and Aleksander Madry. Image synthesis with a single (robust) classifier. *Advances in Neural Information Processing Systems*, 32, 2019.

Motoki Sato, Jun Suzuki, Hiroyuki Shindo, and Yuji Matsumoto. Interpretable adversarial perturbation in input embedding space for text. *arXiv preprint arXiv:1805.02917*, 2018.

C. Saunders, A. Gammerman, and V. Vovk. Transduction with confidence and credibility. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'99, pages 722–726, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.

Andrew M Saxe, James L McClelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv preprint arXiv:1312.6120*, 2013.

Joshua Saxe and Konstantin Berlin. eXpose: A character-level convolutional neural network with embeddings for detecting malicious URLs, file paths and registry keys. *arXiv preprint arXiv:1702.08568*, 2017.

Kevin Scaman and Aladin Virmaux. Lipschitz regularity of deep neural networks: analysis and efficient estimation. *arXiv preprint arXiv:1805.10965*, 2018.

Ludwig Schmidt, Shibani Santurkar, Dimitris Tsipras, Kunal Talwar, and Aleksander Madry. Adversarially robust generalization requires more data. In *Advances in Neural Information Processing Systems*, pages 5014–5026, 2018.

Bruce Schneier. Schneier's law. `https://www.schneier.com/blog/archives/2011/04/schneiers_law.html`, 2011.

Lukas Schott, Jonas Rauber, Matthias Bethge, and Wieland Brendel. Towards the first adversarially robust neural network model on MNIST. *arXiv preprint arXiv:1805.09190*, 2018.

Samuel Schüppen, Dominik Teubert, Patrick Herrmann, and Ulrike Meyer. FANCI: Feature-based automated NXDomain classification and intelligence. In *27th USENIX Security Symposium (USENIX Security 18)*, pages 1165–1181, 2018.

Karen Scouller. The influence of assessment method on students' learning approaches: Multiple choice question examination versus assignment essay. *Higher Education*, 35(4):453–472, 1998.

David Sculley, Gary Holt, Daniel Golovin, Eugene Davydov, Todd Phillips, Dietmar Ebner, Vinay Chaudhary, Michael Young, Jean-Francois Crespo, and Dan Dennison. Hidden technical debt in machine learning systems. *Advances in Neural Information Processing Systems*, 28:2503–2511, 2015.

Ali Shafahi, Mahyar Najibi, Amin Ghiasi, Zheng Xu, John Dickerson, Christoph Studer, Larry S Davis, Gavin Taylor, and Tom Goldstein. Adversarial training for free! *arXiv preprint arXiv:1904.12843*, 2019.

Glenn Shafer and Vladimir Vovk. A tutorial on conformal prediction. *Journal of Machine Learning Research*, 9(3), 2008.

Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

Shreya Shankar, Yoni Halpern, Eric Breck, James Atwood, Jimbo Wilson, and D Sculley. No classification without representation: Assessing geodiversity issues in open data sets for the developing world. *arXiv preprint arXiv:1711.08536*, 2017.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of Go with deep neural networks and tree search. *nature*, 529 (7587):484–489, 2016.

David Silver, Satinder Singh, Doina Precup, and Richard S Sutton. Reward is enough. *Artificial Intelligence*, page 103535, 2021.

Aman Sinha, Hongseok Namkoong, Riccardo Volpi, and John Duchi. Certifying some distributional robustness with principled adversarial training. *arXiv preprint arXiv:1710.10571*, 2017.

Chawin Sitawarin and David Wagner. On the robustness of deep k-nearest neighbors. In *2019 IEEE Security and Privacy Workshops (SPW)*, pages 1–7. IEEE, 2019.

Raaghavi Sivaguru, Chhaya Choudhary, Bin Yu, Vadym Tymchenko, Anderson Nascimento, and Martine De Cock. An evaluation of DGA classifiers. In *2018 IEEE International Conference on Big Data (Big Data)*, pages 5058–5067. IEEE, 2018.

Raaghavi Sivaguru, Jonathan Peck, Femi Olumofin, Anderson Nascimento, and Martine De Cock. Inline detection of DGA domains using side information. *IEEE Access*, 8: 141910–141922, 2020.

George Davey Smith and Shah Ebrahim. Data dredging, bias, or confounding: They can all get you into the BMJ and the Friday papers, 2002.

Michael Spivak. *Calculus on manifolds: a modern approach to classical theorems of advanced calculus*. CRC press, 2018.

Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*, 2014.

Duluxan Sritharan, Shu Wang, and Sahand Hormoz. Computing the Riemannian curvature of image patch and single-cell RNA sequencing data manifolds using extrinsic differential geometry. *Proceedings of the National Academy of Sciences*, 118(29), 2021.

Nedim Šrndic and Pavel Laskov. Detection of malicious PDF files based on hierarchical document structure. In *Proceedings of the 20th Annual Network & Distributed System Security Symposium*, pages 1–16. Citeseer, 2013.

David Stutz, Matthias Hein, and Bernt Schiele. Confidence-calibrated adversarial training: Generalizing to unseen attacks. In *International Conference on Machine Learning*, pages 9155–9166. PMLR, 2020.

Jiawei Su, Danilo Vasconcellos Vargas, and Kouichi Sakurai. One pixel attack for fooling deep neural networks. *IEEE Transactions on Evolutionary Computation*, 2019.

Ruoyu Sun. Optimization for deep learning: theory and algorithms. *arXiv preprint arXiv:1912.08957*, 2019.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the Inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2818–2826, 2016.

Pedro Tabacof, Julia Tavares, and Eduardo Valle. Adversarial images for variational autoencoders. *arXiv preprint arXiv:1612.00155*, 2016.

Thomas Tanay and Lewis Griffin. A boundary tilting persepective on the phenomenon of adversarial examples. *arXiv preprint arXiv:1608.07690*, 2016.

Rohan Taori, Achal Dave, Vaishaal Shankar, Nicholas Carlini, Benjamin Recht, and Ludwig Schmidt. Measuring robustness to natural distribution shifts in image classification. *arXiv preprint arXiv:2007.00644*, 2020.

Simen Thys, Wiebe Van Ranst, and Toon Goedemé. Fooling automated surveillance cameras: adversarial patches to attack person detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pages 0–0, 2019.

Paolo Toccaceli. Venn-ABERS predictor. `https://github.com/ptocca/VennABERS`, 2017.

Antonio Torralba, Rob Fergus, and William T Freeman. 80 million tiny images: A large data set for nonparametric object and scene recognition. *IEEE transactions on pattern analysis and machine intelligence*, 30(11):1958–1970, 2008.

Florian Tramèr. Detecting adversarial examples is (nearly) as hard as classifying them. *arXiv preprint arXiv:2107.11630*, 2021.

Florian Tramèr, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. The space of transferable adversarial examples. *arXiv preprint arXiv:1704.03453*, 2017.

Florian Tramer, Nicholas Carlini, Wieland Brendel, and Aleksander Madry. On adaptive attacks to adversarial example defenses. *arXiv preprint arXiv:2002.08347*, 2020.

Duc Tran, Hieu Mac, Van Tong, Hai Anh Tran, and Linh Giang Nguyen. A LSTM based framework for handling multiclass imbalance in DGA botnet detection. *Neurocomputing*, 275:2401–2413, 2018.

Lloyd N Trefethen and David Bau III. *Numerical linear algebra*, volume 50. Siam, 1997.

Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. Robustness may be at odds with accuracy. In *International Conference on Learning Representations*, 2019.

Loring W Tu. *An introduction to manifolds*. Universitext. Springer, New York, NY, 2 edition, October 2010.

Hoang Tuy. *Convex analysis and global optimization*. Springer, 1998.

Francisco Utrera, Evan Kravitz, N Benjamin Erichson, Rajiv Khanna, and Michael W Mahoney. Adversarially-trained deep nets transfer better: Illustration on image classification. *arXiv preprint arXiv:2007.05869*, 2020.

Leslie G Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.

Lieven Vandenberghe and Stephen Boyd. Semidefinite programming. *SIAM review*, 38(1): 49–95, 1996.

Vladimir N Vapnik. An overview of statistical learning theory. *IEEE Transactions on Neural Networks*, 10(5):988–999, 1999.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017.

Nicholas Vincent, Hanlin Li, Nicole Tilly, Stevie Chancellor, and Brent Hecht. Data leverage: A framework for empowering the public in its relationship with technology companies. In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, pages 215–227, 2021.

Yevgeniy Vorobeychik and Murat Kantarcioglu. Adversarial machine learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 12(3):1–169, 2018.

Soroush Vosoughi, Prashanth Vijayaraghavan, and Deb Roy. Tweet2vec: Learning tweet embeddings using character-level CNN-LSTM encoder-decoder. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 1041–1044, 2016.

Vladimir Vovk, Alexander Gammerman, and Glenn Shafer. *Algorithmic learning in a random world*. Springer Science & Business Media, 2005.

Vladimir Vovk, Ivan Petej, and Valentina Fedorova. Large-scale probabilistic predictors with and without guarantees of validity. *Advances in Neural Information Processing Systems*, 28, 2015.

Wei Wang, Feixiang He, and Qijun Zhao. Facial ethnicity classification with deep convolutional neural networks. In *Chinese Conference on Biometric Recognition*, pages 176–185. Springer, 2016.

Yilun Wang and Michal Kosinski. Deep neural networks are more accurate than humans at detecting sexual orientation from facial images. *Journal of personality and social psychology*, 114(2):246, 2018.

Zhou Wang and Alan C Bovik. Mean squared error: Love it or leave it? A new look at signal fidelity measures. *IEEE signal processing magazine*, 26(1):98–117, 2009.

Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4): 600–612, 2004.

Larry Wasserman. Frasian inference. *Statistical Science*, 26(3):322–325, 2011.

Yandong Wen, Bhiksha Raj, and Rita Singh. Face reconstruction from voice using generative adversarial networks. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/file/eb9fc349601c69352c859c1faa287874-Paper.pdf.

Yuxin Wen, Shuai Li, and Kui Jia. Towards understanding the regularization of adversarial robustness on neural networks. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 10225–10235. PMLR, 13–18 Jul 2020. URL http://proceedings.mlr.press/v119/wen20c.html.

Lily Weng, Huan Zhang, Hongge Chen, Zhao Song, Cho-Jui Hsieh, Luca Daniel, Duane Boning, and Inderjit Dhillon. Towards fast computation of certified robustness for RELU networks. In *International Conference on Machine Learning*, pages 5276–5285. PMLR, 2018a.

Tsui-Wei Weng, Huan Zhang, Pin-Yu Chen, Jinfeng Yi, Dong Su, Yupeng Gao, Cho-Jui Hsieh, and Luca Daniel. Evaluating the robustness of neural networks: An extreme value theory approach. *arXiv preprint arXiv:1801.10578*, 2018b.

Daan Wierstra, Tom Schaul, Tobias Glasmachers, Yi Sun, Jan Peters, and Jürgen Schmidhuber. Natural evolution strategies. *Journal of Machine Learning Research*, 15(1):949–980, 2014.

Eric Wong and Zico Kolter. Provable defenses against adversarial examples via the convex outer adversarial polytope. In *International Conference on Machine Learning*, pages 5286–5295. PMLR, 2018.

Eric Wong, Leslie Rice, and J Zico Kolter. Fast is better than free: Revisiting adversarial training. *arXiv preprint arXiv:2001.03994*, 2020.

Jonathan Woodbridge, Hyrum S Anderson, Anjum Ahuja, and Daniel Grant. Predicting domain generation algorithms with long short-term memory networks. *arXiv preprint arXiv:1611.00791*, 2016.

Yuxin Wu and Kaiming He. Group normalization. In *Proceedings of the European conference on computer vision (ECCV)*, pages 3–19, 2018.

Chaowei Xiao, Bo Li, Jun-Yan Zhu, Warren He, Mingyan Liu, and Dawn Song. Generating adversarial examples with adversarial networks. *arXiv preprint arXiv:1801.02610*, 2018.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms. https://github.com/zalandoresearch/fashion-mnist, 2017.

Cihang Xie, Mingxing Tan, Boqing Gong, Jiang Wang, Alan L Yuille, and Quoc V Le. Adversarial examples improve image recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 819–828, 2020.

Duorui Xie, Lingyu Liang, Lianwen Jin, Jie Xu, and Mengru Li. SCUT-FBP: A benchmark dataset for facial beauty perception. In *2015 IEEE International Conference on Systems, Man, and Cybernetics*, pages 1821–1826. IEEE, 2015.

Jie Xu, Lianwen Jin, Lingyu Liang, Ziyong Feng, Duorui Xie, and Huiyun Mao. Facial attractiveness prediction using psychologically inspired convolutional neural network (PI-CNN). In *2017 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 1657–1661. IEEE, 2017.

Kaihe Xu, Hao Yue, Linke Guo, Yuanxiong Guo, and Yuguang Fang. Privacy-preserving machine learning algorithms for big data systems. In *2015 IEEE 35th international conference on distributed computing systems*, pages 318–327. IEEE, 2015.

Jong Chul Ye. *Geometry of Deep Learning: A Signal Processing Perspective*, volume 37. Springer Nature, 2022.

Dong Yi, Zhen Lei, Shengcai Liao, and Stan Z Li. Learning face representation from scratch. *arXiv preprint arXiv:1411.7923*, 2014.

Dong Yin, Ramchandran Kannan, and Peter Bartlett. Rademacher complexity for adversarially robust generalization. In *International conference on machine learning*, pages 7085–7094. PMLR, 2019a.

Dong Yin, Raphael Gontijo Lopes, Jonathon Shlens, Ekin D Cubuk, and Justin Gilmer. A Fourier perspective on model robustness in computer vision. *arXiv preprint arXiv:1906.08988*, 2019b.

William J Youden. Index for rating diagnostic tests. *Cancer*, 3(1):32–35, 1950.

Bin Yu, Jie Pan, Jiaming Hu, Anderson Nascimento, and Martine De Cock. Character level based detection of DGA domain names. In *2018 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2018.

Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. CutMix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6023–6032, 2019.

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

Matthew Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European Conference on Computer Vision*, pages 818–833. Springer, 2014.

Matthew Zeiler, Graham Taylor, and Rob Fergus. Adaptive deconvolutional networks for mid and high level feature learning. In *International Conference on Computer Vision*, pages 2018–2025. IEEE, 2011.

Matthew D Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

Cheng Zhang, Judith Bütepage, Hedvig Kjellström, and Stephan Mandt. Advances in variational inference. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(8):2008–2026, 2019a. doi: 10.1109/TPAMI.2018.2889774.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning (still) requires rethinking generalization. *Communications of the ACM*, 64 (3):107–115, 2021a.

Daniel Zhang, Saurabh Mishra, Erik Brynjolfsson, John Etchemendy, Deep Ganguli, Barbara Grosz, Terah Lyons, James Manyika, Juan Carlos Niebles, Michael Sellitto, et al. The AI index 2021 annual report. *arXiv preprint arXiv:2103.06312*, 2021b.

Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric Xing, Laurent El Ghaoui, and Michael Jordan. Theoretically principled trade-off between robustness and accuracy. In *International Conference on Machine Learning*, pages 7472–7482. PMLR, 2019b.

Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. Mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017.

Huan Zhang, Tsui-Wei Weng, Pin-Yu Chen, Cho-Jui Hsieh, and Luca Daniel. Efficient neural network robustness certification with general activation functions. *Advances in Neural Information Processing Systems*, 31, 2018a.

Shufei Zhang, Kaizhu Huang, Jianke Zhu, and Yang Liu. Manifold adversarial learning. *arXiv preprint arXiv:1807.05832*, 2018b.

Shufei Zhang, Kaizhu Huang, Rui Zhang, and Amir Hussain. Generalized adversarial training in Riemannian space. In *2019 IEEE International Conference on Data Mining (ICDM)*, pages 826–835. IEEE, 2019c.

Chenxiao Zhao, P. Thomas Fletcher, Mixue Yu, Yaxin Peng, Guixu Zhang, and Chaomin Shen. The adversarial attack and detection under the Fisher information metric. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):5869–5876, Jul. 2019. doi: 10.1609/aaai.v33i01.33015869. URL `https://ojs.aaai.org/index.php/AAAI/article/view/4536`.

Haizhong Zheng, Ziqi Zhang, Juncheng Gu, Honglak Lee, and Atul Prakash. Efficient adversarial training with transferable adversarial examples. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1181–1190, 2020.

Wen Zhou, Xin Hou, Yongjun Chen, Mengyun Tang, Xiangqi Huang, Xiang Gan, and Yong Yang. Transferable adversarial perturbations. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 452–467, 2018.

Chen Zhu, Yu Cheng, Zhe Gan, Siqi Sun, Tom Goldstein, and Jingjing Liu. FreeLB: Enhanced adversarial training for natural language understanding. *arXiv preprint arXiv:1909.11764*, 2019.

Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232, 2017.

Epilogue

> Early on she ceased to allow her image to be recorded. You worked with what you had. You justified her image, rotated her through planes of light, planes of shadow, generated models, mapped her skull in grids of neon. You used special programs to age her images according to statistical models, animation systems to bring your mature Marie-France to life. You reduced her image to a vast but finite number of points and stirred them, let new forms emerge, chose those that seemed to speak to you... And then you went on to the others, to Ashpool and the daughter whose face frames your work, its first and final image.
>
> *Mona Lisa Overdrive*
> William Gibson

If the reader will indulge me, I'd like to take some time here to reflect on my personal experiences doing this Ph.D. in a section of the book that is allowed to be less formal and less professional than its main matter. I'm just going to speak my mind frankly here, in order to get a few things off my chest as well as to give the manuscript a personal touch. The rest of the book is dry enough as it is. After all, it is supposed to be a work of science.

But what actually *is* a work of science? Doing this Ph.D. has been fascinating, because it serves in some sense as an "initiation" into the world of science. A master thesis gives a student a certain taste of this world as well, of course, but there are several important differences to a Ph.D., the most obvious one being its length: a master thesis usually only takes a single academic year, and in some circumstances even less. It is also not generally expected of a master thesis to be worthy of publication in a serious scientific venue, though that is something students and their advisors will sometimes strive for. Even so, a master student must receive a great deal of direction from their advisor, and the nature of the scientific project cannot be too elaborate due to the time constraints. If the thesis is published at all, it is often the only scientific publication ever undertaken by the student. A Ph.D., by contrast, takes much longer: often four years or more. During this time, a Ph.D. student is expected to work on a much more sophisticated research agenda that could not feasibly be accomplished in a single year. The results of our investigations are usually published in multiple papers over several years, and we are expected to present our ideas and results regularly at academic conferences. Ph.D. students therefore form much stronger connections to the scientific community than any thesis student ever could. We set up our own collaborations and manage our own projects almost fully independently, relying on our advisors mostly to double-check our results, brainstorm new ideas, make sure we remain on track to graduate and (as our direct superiors) approve our absences in the HR system when we're suffering another mental breakdown.

A Ph.D. basically marks the *start* of a potential scientific career, if one wishes to pursue it. Through this process, we are initiated into our chosen field of science and become recognized as worthy contributors to it. We are introduced into what I like to call a *culture of knowledge production*. I deliberately avoid the term *science* here, because it tends to obscure the very human aspects of our communities. Doing research in my particular field (machine learning) has been fascinating in this regard, because it has clearly revealed to me said human nature of the scientific enterprise: far from being a well-oiled machine that consumes coffee and tax money to produce objective and useful truths, science as a whole really is a patchwork of different cultures as diverse and idiosyncratic as the individual scientists themselves. This is only natural: all scientists are human, after all, and any community of people that interact with each other over a sufficiently long period of time will develop their own culture that may be alien and perhaps even hostile to outsiders. Although, in the abstract, we remain singularly fixated on thoroughly studying interesting problems and faithfully reporting our well-reasoned conclusions, cultural and personal factors significantly impact how science is done in individual fields. I am of course not suggesting that the validity of, say, mathematical proofs of specific theorems is a matter of personal taste. Rather, the cultural influences that pervade the scientific world tend to be much more subtle: they determine, for example, the specific research questions the community deems sufficiently interesting to pursue, and which questions we dare not even ask. They can also determine the precise methodology employed to study any given problem. Let me give a few concrete examples:

- At the time of this writing, the field of deep learning is very interested in methods to efficiently train ever larger models. The OpenAI GPT-3 paper [Brown et al., 2020], for example, uses as one of its main selling points the fact that GPT-3 has 175 billion parameters, ten times larger than the state of the art at the time. To the authors' credit, they discuss various potentially negative effects of this model in their impact statement. However, despite the growing awareness of the issues affecting large models — chief among which are bias, fairness and environmental concerns — the community has not really evolved beyond paying lip service to these causes. Compared to papers like Brown et al. [2020], there is an absolute paucity of research into how to make these models environmentally sustainable, economically fair and socially just. As discussed at length by Crawford [2021], the cost of developing and maintaining large AI systems goes far beyond what is typically considered in papers: from the reliance on conflict minerals to the increasing power of the surveillance state and erosion of human rights, our field is beset with problems to which we remain strategically blind. In fact, the community sometimes even responds to this type of research with outright hostility, as the infamous firings of ex-Google researchers Timnit Gebru and Margaret Mitchell painfully illustrate.[5]
- In recent years, there has been a major surge of research interest into Bayesian statistical methods to study various machine learning problems. However, the Bayesian school is not the only possible foundation for statistics, its main competitor being the frequentist school. It is not necessary to get into the specifics of the ideological divide between these differing schools of thought; suffice to say that they approach statistical questions in slightly different ways, obtaining sometimes very different and incompatible results. To me, there is no clear reason why the field of machine learning has grown to prefer Bayesian methods over the frequentist ones. Indeed, both fields have a long history and both are respected as mature scientific subdisciplines of statistics. Yet almost all recent

[5] `https://www.bbc.com/news/technology-56135817`. Accessed 2022-05-09.

> papers in the field are decidedly Bayesian in methodology. It's almost as if frequentist statistics don't exist anymore. Scientifically speaking, there is no reason why one should, in general, absolutely prefer one school over the other, so I am inclined to conclude that the current resurgence of Bayesian methods in machine learning are merely a trend of fashion: the Bayesian methods are simply more appealing on an intuitive level to the current generation of machine learning scientists and practitioners. This has the obvious consequence of collectively blinding the field to interesting findings that could more easily (or perhaps exclusively) be achieved via frequentist means.

My experience in the scientific world over the past few years has shown that the scientific community, as much as any human community, is subject to certain arbitrary whims, fashion trends, rivalries and politics. I make no secret of the fact that many scientists, including myself, have a love-hate relationship with the scientific community. On the one hand, it can be positively delightful to collaborate with like-minded people on an interesting problem, write a nice paper about it, get it published with favorable reviews from other experts and then present the work at a conference where the audience literally applauds your efforts and actively wants to discuss your work further in detail. I have had several such experiences over the course of my Ph.D., and these memories I will carry with me forever. They never fail to lift my spirits. These are the good parts of our culture: the ones where we support each other, take an interest in each other's work and collaborate for the benefit of mankind.

Naturally, there is a flip-side to this story, and I have several grievances with the machine learning community that I think I must discuss. The broader issues affecting the entire scientific community at large — poor work-life balance, broken peer review, publish-or-perish mentality, etc. — are well-known,[6] and I have nothing to add to that discussion. Instead, I wish to focus on what I think are major problems specific to my own field of research. This is, in my opinion, a crucial aspect of what makes someone an expert in a certain area: the ability to critique it *intelligently*. An expert is supposed to be more than simply someone who knows a lot about a given subject. An expert must also know how the subject is practiced by other experts, and must be able to critically examine this process. Experts are not glorified parrots; they are supposed to have a thorough knowledge of their field of study, of course, but they must also have their own well-informed opinions on how to *improve* the field. We are not passive bystanders but active participants in this culture of knowledge production who collectively shape the future direction of the field. The following, therefore, is my personal take on how the field of machine learning could be improved.

It is important to keep in mind that, although some people may feel "called out" or attacked in some way by the things I write here, none of this is intended to be taken personally. Like I said, science is not a machine but a *culture*, and it can be very difficult for people embedded in a certain culture to come to terms with its problems or even realize these problems exist at all. The views I express here are also by no means mine alone; they have been raised by several other experts as well in recent years.

That said, there are two major problems I believe the field of machine learning must reckon with. The first and perhaps most surprising issue is the pervasiveness of outright pseudo-scientific ideas in contemporary research papers that are nonetheless taken seriously by respectable scientific publishers and conferences such as IEEE, Springer and NeurIPS. I have collected some concrete examples here, in no particular order:

[6] `https://slowscience.be`

- A NeurIPS 2019 paper claiming that we can reconstruct a person's face based purely on audio samples of their voice [Wen et al., 2019].

  Aside from the problematic implications regarding individual privacy, there are obvious confounders here that the authors do not consider at all. In particular, most data sets of this kind consist mostly of white people. The observation that facial features can be accurately reconstructed from a biased data set that is not representative of the general population is not surprising at all, nor is it interesting or useful. This racial bias is obviously present based on the authors' own examples in figures 3-5. The majority of the data set also appears to consist of women, as women clearly dominate all of the examples given by the authors. The data sets used are `Voxceleb` for the audio recordings and `VGGFace` for the facial images. I can't speak to the `Voxceleb` data set, but `VGGFace` is known to be heavily racially biased. This fact is even admitted by the authors of the data set themselves,[7] but this obvious source of confounding does not appear to be discussed anywhere in the paper. One would expect NeurIPS to hold its authors to higher standards.

  Aside from the statistical confounders, the scientific evidence considered by the authors to support the idea that facial appearance is related to voice is either weak or incorrectly cited, and they attempt to cover this up with the misleading statement that "[a] person's voice is incontrovertibly statistically related to their facial structure." Be that as it may, the research cited in the paper only provides evidence of a relationship between voice and *skeletal structure* of certain parts of the face (such as the throat or mouth). The idea that one can create an accurate approximation of the person themselves is, in my opinion, not sufficiently supported. Moreover, in the case of transgender individuals, the predicted gender and facial reconstruction is likely to be entirely wrong depending on how far along the individual is in transition. A similar failure case occurs for gender-nonconforming individuals, men with feminine-sounding voices or women with masculine-sounding voices. These important and obvious potential problems are not considered anywhere in the work.

- A widely-publicized 2017 paper claiming that machine learning can deduce sexual orientation from facial images [Wang and Kosinski, 2018].

  To put it bluntly, this work is promoting outright *phrenology* or *physiognomy*, the debunked pseudo-scientific idea that personality traits can be inferred from facial features. There are way too many papers just like this that try to infer sexual orientation or other personal characteristics from facial images alone, despite the fact that phrenology has long been debunked. It is the machine learning equivalent of chemists who still publish works on phlogiston theory, or medical doctors who still investigate the miasma theory of disease (except such research likely would not involve the gross violations of privacy that have become so normalized in machine learning). The irony is that the authors begin the manuscript with a discussion of physiognomy and state that it is nowadays rightly rejected as pseudo-science, before going on to argue why their specific brand of physiognomy is not pseudo-scientific after all. For this they employ a common fallacious rhetorical tactic, where they strawman their detractors as claiming that no links between personality traits and physical appearance exist at all. This is clearly nonsensical, as it would imply phenotypes do not exist and genes do not correlate with behavior. Obviously they do, but the mere fact that *some* personality traits *may* correlate with certain aspects of physical appearance does not justify a link between any *particular* trait and physical attribute. This

[7] `https://www.robots.ox.ac.uk/~vgg/data/vgg_face/`

would need to be investigated as a separate scientific question, but the authors make no such case. Instead, they see the mere fact that their algorithm works on a very specific data set as all the evidence required to prove the existence of a relationship between sexuality and facial attributes, ignoring all of the pitfalls known to be associated with such methodologies.

The data collection process employed by the authors is fraught with issues as well. First, they obtained their data by scraping profile pictures from unnamed U.S. dating websites.[8] This is perhaps the most obvious source of bias: because the pictures came from a *dating* website, users are encouraged to post material that overtly communicates their sexual orientation. The kind of images one would upload of themselves on a dating website is generally not the same as what one would upload on Facebook or Instagram, and is certainly not representative of how one acts in public. It is therefore not surprising at all that sexual orientation can be deduced from such pictures, especially by someone who is familiar with dating apps. The set of images was then processed using Amazon Mechanical Turk workers "to verify that the faces were adult, Caucasian, fully visible, and of a gender that matched the one reported on the user's profile." Here the authors introduce overt biases against gender-nonconforming individuals as well as any non-Caucasian people. The authors never seem to realize that all these factors significantly compromise the validity and generality of their conclusions.

This work is sloppy at best and harmful pseudo-science at worst, as works like this could serve no earthly purpose other than to label people as homosexuals against their own wishes. This can be potentially life-threatening: at the time of this writing, over 70 countries criminalize homosexuality, imposing prison sentences or even the death penalty.[9] Even if this work were scientifically sound (which it certainly is not), its creation constitutes nothing short of an act of overt aggression against the LGBTQ+ community, who are often mortally dependent on keeping their identities secret. A tool that reveals people's sexual orientation without their consent can *never* have *any* legitimate use-cases, making one wonder what the motivation behind this work could possible have been. Perhaps it is an illustration of the Arendtian notion of the banality of evil, where evil arises not through the edicts of malevolent dictators but as a side-effect of the baffling ignorance and complacency of otherwise intelligent people.

The most ironic thing about this work is that it didn't have to be pseudo-scientific at all: as I've argued earlier, sexual orientation can likely be deduced very easily from a good majority of profile pictures uploaded to dating apps and websites because of the obvious incentive people have to clearly signal their sexual preferences in these contexts. If the authors had gone no further than this conclusion, that would have been a valid result in my opinion, but of course in that case the work would not have been interesting enough for publication. In other words, the authors fell victim to the same publish-or-perish mentality that affects all scientists: they had to overstate their claims to make their work appear more interesting than it actually is. To achieve this, they had to argue that their work proves the existence of a genetic link between sexual orientation and facial features, a hypothesis that is not well-supported at all given the evidence they provide. This still

[8] They do not mention whether any of the people whose pictures were used gave their consent to participate in this study, so I'm guessing the authors never bothered to contact any of them. This makes the work a non-consensual study involving unaware human participants, something every reasonable ethics board would object to.

[9] `https://76crimes.com/76-countries-where-homosexuality-is-illegal/`

does not answer the question as to *why* this work exists in the first place, however, and the potential damage it can do to the LGBTQ+ community is simply ignored.

- A 2017 IEEE paper claiming that machine learning can predict a woman's attractiveness [Xu et al., 2017].

  The paper markets itself as a general method for "facial beauty prediction," but the data set used, SCUT-FBP [Xie et al., 2015], consists *exclusively* of pictures of women. Once again, many papers exactly like this one can be found published in reputable venues with a little searching. Indeed, going through the references of this paper as well as other papers that cite it reveals an unsettling sub-community of male researchers who are strangely obsessed with ranking women according to their attractiveness. What this shows, perhaps unsurprisingly, is a significant presence of misogynist men in our field who use the air of scientific authority as an excuse to objectify women.

  If one were being generous, one could defend this paper as a study into the reality of how people generally understand facial beauty. However, I have found no significant insights in any of these papers that reveal anything about traditional beauty standards we did not already know. This type of research therefore adds nothing novel to the scientific record. Moreover, as discussed by O'Neil [2016], by formalizing these sorts of overtly subjective value judgements in an opaque algorithm, the public is deceived into thinking that the resulting scale somehow reflects an objective reality rooted in biology or physics rather than being exposed for what it is: an arbitrary human social construct. This only serves to prevent productive conversations about the ways in which traditional beauty standards harm and oppress women and certain minority groups.

  The fact that certain groups of researchers apparently take seriously the idea that there could (or *should*) be a simple scale "objectively" ranking female attractiveness from 1 to 5 is likely a symptom of the deeper underlying patriarchal pathologies that deter women from participating in this field.

- A 2016 Springer paper claiming that emotions can be accurately inferred from short video clips [Kahou et al., 2016].

  Although this work dates back to 2016, papers attempting to infer emotions from facial images or videos are published on a regular basis to this day. The basic fallacy all of these papers have in common is that they assume emotions are primarily and accurately expressed via facial expressions or other visible mannerisms. This is barely a toddler's level of understanding of how emotions work, and the consensus among psychologists is that such tools couldn't possibly be accurate.[10] However, as is tradition, the machine learning community ignores these inconvenient nuances provided by subject matter experts in favor of flashy reductionist soundbites that make clickbait headlines in the mainstream media.

The list could go on, but I believe my point has been made. It seems clear to me that, in each case, researchers did not stop to first consider the scientific plausibility of their hypotheses. Instead, they likely started out with a data set already at hand for reasons unrelated to the actual research and simply tried to predict various things based on this data indiscriminately. At no point in most of these works do the authors ever stop to critically examine their data and methodology

[10] `https://www.nature.com/articles/d41586-020-00507-5`

for potential biases. Indeed, if biases are considered at all, this is usually done as a virtue signalling tactic with no real intent to reckon with the implications.

I believe this issue is caused in large part by the fact that machine learning is considered a figurative hammer and hence every scientific question is seen as a potential nail, a process that is enabled by the vast indiscriminate data collection to which we are subjected despite all the efforts by privacy advocates and legal reforms. Consider how the scientific method (at least in the abstract) is *supposed* to work:

1. Formulate a hypothesis regarding the answer to some question of interest.
2. Gather data that could validate or refute this hypothesis.
3. Critically examine the data and draw well-founded conclusions.

The crucial aspects that need to be stressed here are the fact that data is gathered *after* the hypothesis has been formulated and that we are not explicitly aiming to *verify* the hypothesis; we are prepared to let the chips fall where they may. If the data does not support the hypothesis, we are expected to be honest about this. In machine learning, however, this process sometimes appears to be distorted. While certainly not true of all research in this field, many papers appear to follow this methodology instead:

1. Gather data about anything and everything you can get your hands on.
2. Iterate over all attributes $X_1, \ldots, X_n$ present in this data set. For each attribute X_i, use machine learning to infer X_i from all of the other attributes.
3. If this process is successful for some X_i, claim that there is a causal link between the other attributes and X_i which can be predicted by machine learning.

Instead of starting with an interesting scientific question, certain researchers tend to start with the data they were able to collect. Moreover, they tend to only report *positive* results which verify the hypothesis, neglecting to report when the data do not support it. In other words, the fundamental question the scientific method tries to answer is reversed. The question we *should* be asking regarding any hypothesis is the following:

> *What data could plausibly validate or refute our hypothesis?*

The question we appear to be asking in practice is

> *What hypothesis could we validate using this data?*

No serious effort appears to be made to refute the hypothesis or to consider potential problems in the methodology. Instead of critically examining their assumptions and data sets for potential biases and statistical confounders, the mere fact that they were able to train a successful algorithm on their specific data is taken as all the evidence that is necessary to prove that the hypothesized relationship actually exists. Researchers believe, for example, that they can infer sexuality based on facial images merely because they were able to do this on certain data sets. The fact that sexuality is not known to have any significant phenotypical expression of this kind, and that the existence of such expressions is a necessary pre-condition for their hypothesis to make sense, is conveniently ignored or obfuscated. It is certainly no coincidence that most papers of this sort perform experiments on just a single data set.

This phenomenon is well-known within the statistical sciences under various names: *data dredging*, *data snooping*, *p-hacking*, etc. It has received extensive scholarly critique [Smith and Ebrahim, 2002], and it clearly undermines the validity of any scientific conclusions. When testing a scientific hypothesis, we obviously cannot use the same data that was employed to formulate the hypothesis in order to validate it. As a trivial example, consider a sequence of five coin tosses resulting in three heads and two tails. One might then conclude that the coin is biased in favor of heads, but this pattern could be explained purely by chance. What we *should* do in order to conclude whether or not the coin is biased is to do one sequence of tosses, compute the potential bias (in this case, 3/5 probability of heads and 2/5 probability of tails), then do *another* sequence of tosses and verify whether the bias is still present in this new *independent* sequence. In practice, when all we have is a single finite data set of samples, this means we need to split the data set into one part (the "training data") on which we formulate our hypothesis and another data set (the "validation data") on which we verify said hypothesis. Hypothesizing after the results are known ("HARKing"; Kerr [1998]) introduces a circularity to the scientific process which clearly undermines the generality of the conclusions, yet it appears to be commonplace within machine learning.

In recent years, however, the scientific community has begun to take these issues much more seriously and introduced the notion of *pre-registration* [Nosek et al., 2018]. When a study is pre-registered, the research questions and analysis plan are defined and stored in a central registry *before* the outcomes are actually observed, *i.e.*, before the study really begins. Although the reliability of pre-registration is obviously lessened in cases where data has already been collected (as is often the case in contemporary machine learning), it is still a valuable tool and presents a much-needed innovation towards the prevention of HARKing and other related problems mentioned above. In my opinion, the ML community should seriously consider pre-registration, at least for studies that have potentially harmful societal implications.

A curious observation common to many machine learning papers that struggle with HARKing is that they tend to use data sets containing highly specific and sensitive personal information, begging the question how (and *why*) this data even came to exist in the first place. This leads me to the second issue facing the field of machine learning: its almost complete lack of any ethical considerations. Researchers collect whatever data they can, often using automated scripts to scrape content off websites despite the fact that many jurisdictions would consider this an illegal abuse of digital access. In many cases, the people whose data is scraped quite obviously never intended for anyone to collect and use it in this manner, and would not give their consent if it were asked. Machine learning researchers rarely bother with consent, however. These practices have led to public outcry, leading to censoring[11] and even retraction[12] of certain data sets.

Aside from the problematic data collection practices, many of the most popular applications of modern machine learning techniques are inherently dubious and ethically questionable. Consider that the most popular research topic *by far* in the machine learning community at this time is the development of *facial recognition algorithms* that classify people into various groups based on pictures of their faces. One struggles to understand why this could possibly be such an interesting problem to researchers. Why would we, as a society, want to invest such enormous sums of money into research grants for projects that essentially serve no other purpose but to allow governments and corporations to more efficiently invade our privacy and violate our

[11] `https://www.wired.com/story/researchers-blur-faces-launched-thousand-algorithms/`
[12] `https://groups.csail.mit.edu/vision/TinyImages/`

rights in everyday life? The answer is that we don't want this at all,[13] but clearly the interests and incentives of the machine learning community are not aligned with the interests of society. Particularly egregious examples can be found in the Chinese machine learning community, who publish a plethora of papers annually on ethnicity recognition specifically to differentiate between Han Chinese, non-Han Chinese and Uyghur people [Wang et al., 2016]. These works are not pseudo-scientific in nature, but they clearly serve no other purpose than to facilitate the Chinese government's programs of discrimination and genocide against non-Han Chinese minority groups in China. The Uyghur are an especially vulnerable community in this regard, and it is known that the Chinese government employs a vast system of surveillance (including automated facial recognition algorithms) to identify, track and detain Uyghur people and other minorities for essentially no reason.[14] These people are then put into "re-education centers" reminiscent of Nazi concentration camps. In this case, machine learning researchers are actively enabling atrocious human rights violations by a fascist regime, all in the name of citation counts, and the scientific peers that are supposed to review these works appear to be either oblivious to these issues or apathetic to them.

This apathy is too often justified by an argument which, in my opinion, borders on the criminally insane: the idea that scientists should not be "political." This is a dangerous misconception: it should be obvious that many scientific findings (such as the examples I have given here) clearly only exist to serve very particular political agendas. Ignoring these implications is irresponsible at best and reprehensible at worst. Should we uncritically accept the development of facial recognition algorithms that can serve no other purpose but to be used by governments to make "undesirable" people vanish from the face of the Earth without any form of due process, because objecting to the research on these grounds would be "too political"? Should we ignore the tangible harms caused by certain lines of research, because it might make some powerful people uncomfortable or because it might lead to lower citation counts? What a depraved argument. Even if we accept that scientists should not be political, it requires an astonishing amount of mental gymnastics to refuse to see that the mere conception of such tools in the first place constitutes an overtly political act. If nothing else, this position is self-defeating because it precludes doing any science that could possibly have political implications. On such grounds we might as well refuse to do any further vaccine research, because vaccines have been so heavily politicized. Indeed, engaging in politics is not a bane but a part of the *duty* of a scientist. Politics, by themselves, are not the problem; politics are inevitable. The problems arise when these politics, whatever they are, are not allowed to be challenged or are actively obfuscated.

The harsh conclusion I am led to, then, is that we — the machine learning community — have betrayed this most fundamental value every scientist should have: that our work should aim to *improve* our living conditions. By contrast, the primary technologies developed by machine learning researchers tend to make people *worse off*, both in the way they are constructed (requiring a lot of personal data to be collected) and the way they are deployed (shifting power away from regular people and towards already very powerful individuals and companies). Instead of trying to improve people's lives, we worry about our citation counts, and this leads us to publishing papers which (either directly or indirectly) contribute to the erosion of our collective human rights and the pollution of the scientific record. We are well past the era that scientists could pretend to be nothing but passive observers, reporting the facts of the world without any biases

[13] `https://www.amnesty.org/en/tech/surveillance-giants/`

[14] `https://thediplomat.com/2019/07/which-countries-are-for-or-against-chinas-xinjiang-policies/`

or politics. The fabled "view from nowhere" described by Nagel [1989] is exactly that: a fable, a lie complacent scientists like to tell themselves in order to shirk their societal responsibilities and feign innocence as they actively participate in horrendous acts of destruction and aggression.

The good news is that I believe it doesn't have to be this way at all. To move forward and make meaningful change, I think at least the following recommendations should be adopted by governments, funding agencies and scientific publishing venues around the world:

1. Institute mandatory ethical review boards for all machine learning research involving the collection of personal data. Reject research proposals where the cost of data collection is clearly not outweighed by the benefits provided by the proposed novel technology. If any personal data is collected at all, *every* individual involved must have provided *informed* consent. They must also be able to retract this consent at any point during the research process.

2. Reject papers that do not critically examine their own methodology. If there are clear statistical confounders or other sources of bias that might very well compromise the validity of the results, the authors *must* clearly and thoroughly address these issues before publication is possible. Pre-registration as well as encouraging the publication of reproduction studies should facilitate this objective.

3. If the authors made use of Amazon Mechanical Turk or similar services to enable their experiments, workers must be adequately compensated for their labor. Funding agencies and review boards would do well to impose reasonable minimum wages for these workers. These services have become immensely popular for ML research, especially for the labeling of training data sets such as ImageNet [Deng et al., 2009] as well as natural language processing [Rashkin et al., 2018]. It is never made clear how much these workers were paid, and services such as Mechanical Turk raise considerable labor rights concerns [Crawford, 2021].

These guidelines could put a stop to the indiscriminate collection of sensitive data about millions of individuals around the world without their consent. They would ensure that underprivileged workers who provide valuable assistance to scientific experiments are appropriately compensated instead of being outright exploited. Researchers would be encouraged to engage more critically with their work, making honest attempts at refutation and appropriately qualifying their conclusions with clear caveats and limitations instead of emphasizing reductionist and misleading claims for the sake of clickbait.

Some might balk at these conditions, arguing that they are much too heavy a burden for projects that require large-scale data sets in order to function. Current state-of-the-art large language models are an excellent example of this: these models only appear to work because its developers were able to train them on billions of documents, and they were only able to construct such an incredibly large repository of data because they could easily scrape all of it from the internet without bothering to differentiate between benign and problematic data or even to respect copyright. Indeed, the data sets involved are just too large for any sort of screening to be feasible, and the data sets must be big, because neural networks are big! My response to such criticism is this: perhaps there are only very few, if any, *legitimate* scientific projects that require data of this kind. If you find yourself needing to violate the human rights and dignity of untold people in the name of science, if your project requires the exploitation of underpaid workers, then you're doing the wrong kind of science. For large language models, perhaps the number

one research priority *should* be to design model architectures and training algorithms that are much less data hungry than the techniques we have now. Decreasing the required sample size of our AI systems would already go a long way, and this is indeed an active area of research. However, the efforts put into this extremely important research program that could bring about *real* progress are completely dwarfed by the number of researchers and companies that simply take large sample sizes as a given, and who abuse their position of power to obtain them.

Could we call this meaningful progress? In what direction are we progressing if we allow such malpractice to continue? Certainly not a direction I care for. I will also point out that many other scientific fields that work with sensitive data on such large scales, such as the medical community, *are* able to stick to stringent ethical requirements. The only cost is that it slows the research down, but the research is better off because of this. Other fields emphasize quality over quantity, and this perhaps reveals the fundamental problem with our field: we want to publish as many papers as possible, as fast as possible. We don't care if we break things and we don't care to pause and critically examine our work. What matters is the citation count, the impact factor, the h-index. To quote Kate Crawford [Crawford, 2021]:

> *It is a failure of imagination and methodology to claim that it is necessary to experiment on millions of people without their consent in order to produce good data science.*

Of course, the grievances I have discussed here should not be taken as blanket statements about every single scientist working in machine learning. I have met many great people over the course of my Ph.D. whose work I very much respect, and I am not suggesting that our scientific record is rife with pseudo-scientific nonsense. But these trends cannot be denied, and the field is in need of urgent reform. We can no longer practice machine learning as if we're still in the 1990s, back when we were only using machine learning to classify hand-written digits or flag spam e-mails, and personal data was much harder to collect. We have a real impact on the world now that we are clearly ill-equipped to handle. Machine learning needs to grow up.