Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE

Corso di Laurea in Informatica

ASYMMETRIES IN ADVERSARIAL SETTINGS

Relatore:
MICHELE LOMBARDI

Presentata da:
SAMUELE MARRO

I Sessione
A.A. 2021-22

To my family,
which never stopped
believing in me.

Abstract

Even without formal guarantees of their effectiveness, adversarial attacks against Machine Learning models frequently fool new defenses. We identify six key asymmetries that contribute to this phenomenon and formulate four guidelines to build future-proof defenses by preventing such asymmetries. We also prove that attacking a classifier is NP-complete, while defending from such attacks is Σ_2^P-complete. We then introduce Counter-Attack (CA), an asymmetry-free metadefense that determines whether a model is robust on a given input by estimating its distance from the decision boundary. Under specific assumptions CA can provide theoretical detection guarantees. Additionally, we prove that while CA is NP-complete, fooling CA is Σ_2^P-complete. Even when using heuristic relaxations, we show that our method can reliably identify non-robust points. As part of our experimental evaluation, we introduce UG100, a new dataset obtained by applying a provably optimal attack to six limited-scale networks (three for MNIST and three for CIFAR10), each trained in three different manners.

Sommario

Pur non fornendo garanzie formali sulla loro efficacia, gli attacchi avversariali ingannano frequentemente le difese più recenti. Identifichiamo sei asimmetrie chiave che contribuiscono a questo fenomeno e formuliamo quattro linee guida per costruire difese prive di queste debolezze. Dimostriamo inoltre che attaccare un modello è un problema NP-completo, mentre difenderlo è un problema Σ_2^P-completo. Illustriamo quindi Counter-Attack (CA), una metadifesa priva di asimmetrie che stabilisce la distanza di un input dal suo decision boundary, determinando quindi la robustezza del modello su quell'input. Sotto ipotesi specifiche, CA offre garanzie formali della sua efficacia. Dimostriamo inoltre che, mentre eseguire CA è un problema NP-completo, ingannare CA è un problema Σ_2^P-completo. Mostriamo infine che, anche quando vengono utilizzati rilassamenti euristici, CA riesce a identificare consistentemente input non robusti. Come parte del nostro studio sperimentale, costruiamo UG100, un nuovo dataset ottenuto applicando attacchi esatti a sei reti di scala ridotta (tre per MNIST e tre per CIFAR10), ciascuna addestrata con tre tecniche diverse.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Machine Learning (ML) models have seen widespread application in numerous fields, from natural language generation [23] and image classification [37] to autonomous driving [26] and drug development [68].

However, adversarial attacks, i.e. algorithms designed to fool such models, represent a significant threat to the applicability of ML in real-world contexts [8, 9, 95]. Despite several years of research effort, countermeasures (i.e. "defenses") to adversarial attacks are frequently fooled by applying small tweaks to existing techniques [11, 12, 16, 31, 34, 85]. This phenomenon is present even when neither attacks nor defenses provide provable guarantees of their effectiveness.

We argue that this pattern is due to differences between the fundamental mathematical problems that defenses and attacks need to tackle. We identify six of such differences, which we refer to as *asymmetries*. Additionally, we formulate four guidelines that can help to even out the field w.r.t. attacks.

We formalize these observations by showing that, under reasonable assumptions, attacking a classifier is NP-complete, while defending from an attack is Σ_2^P-complete.

Motivated by this analysis, we introduce a new metadefense, named Counter-Attack (CA), that can determine if a defended model is not robust on a given input. The main idea in CA is to mitigate asymmetries by relying on attacks themselves as a detection tool. The use of adversarial attacks also ensures that CA can be applied in any setting where at least one untargeted adversarial attack is known, while also allowing one to capitalize on future improvements in the field of adversarial attacks. Moreover, we prove that running CA with exact attacks is NP-complete, while fooling CA is Σ_2^P-complete.

While using CA in combination with exact attacks is not scalable, existing attacks may serve as an effective proxy. We provide empirical evidence for this claim via the introduction of UG100, a dataset of provably optimal adversarial examples found on 2.2k samples of MNIST and CIFAR10 for six differently configured Neural Networks (NNs). We also benchmark seven heuristic adversarial attacks and find that they provide an accurate (average over-estimate between 2.04% and 4.65%) and predictable (average $R^2 > 0.99$) approximation of the true optimum. UG100 can also represent a benchmark for future attacks, and allow for a comparison with both exact and heuristic approaches. All our code, datasets, pretrained weights and results are available at https://github.com/samuelemarro/counter-attack under MIT license.

Chapter 2

Preliminaries

2.1 Background and Related Work

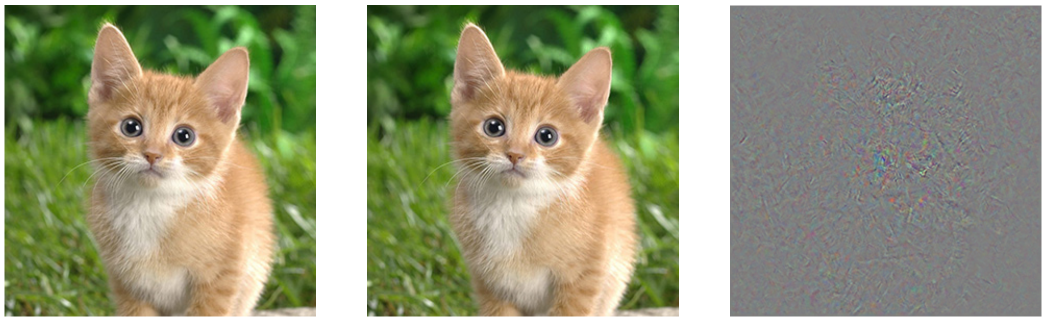
2.1.1 Adversarial Examples

Adversarial examples can be informally defined as inputs that cause an undesirable behavior in a target model. For instance, a spam email that fools a spam detector is a type of adversarial example, since it achieves a certain behavior (getting classified as legitimate) against the wishes of the recipient. In the context of Machine Learning, the most common adversarial examples in the literature are those against image classifiers. These are inputs that are visually indistinguishable from regular images (also known as *genuine* inputs) but are classified incorrectly. Refer to Figure 2.1 for an example. Adversarial examples were first identified in image classifiers by [81]. The same paper also introduced what is now known as the *L-BFGS attack*, the first adversarial attack against a NN.

2.1.2 Adversarial Attacks

An adversarial attack is an algorithm designed to find adversarial examples. Attacks against classifiers are commonly classified depending on three factors:

- Admissible input constraints;



(a) Genuine Predicted class: Tiger Cat (b) Adversarial Predicted class: Frying Pan (c) Difference

Figure 2.1: Adversarial example against the Inception V3 ImageNet-1000 classifier. For ease of visualization, the difference has been multiplied by a factor of 15.

- Objective;
- Information requirements.

Admissible Input Constraints Attacks are usually constrained on which inputs can be used as adversarial examples. This is due to the fact that only some inputs are beneficial to the attacker. For example, in the context of attacks against spam detectors, an email that does not convey the intended malicious message is likely to be of little use to the spammer. For image classifiers, the most common utility metric is the perceptive similarity with a given starting image. This is frequently approximated by the L_p norm of the difference between the adversarial example and the starting point. Depending on the setting, p is usually 0 [80], 2 [47, 51] or ∞ [25, 47].

Objective In the contest of image classification, the objective of the attack is usually to find an input that is classified with a label other than the correct one (i.e. the class of the starting point). If there are no additional restrictions on the predicted class, these attacks are referred to as *untargeted* attacks.

On the other hand, if the objective of the attack is to find an input with a specific (and incorrect) predicted class, such an attack is said to be *targeted*.

Information Requirements Not all adversarial attacks require the same level of information regarding the target. While most attacks (known as *white-box* attacks) require unrestricted access to both the predictions and the gradients of the model, *gray-* and *black-box* attacks require only partial information, such as predictions alone. [15] and [53] are two examples of black-box attacks.

2.1.3 Defenses

The threat of adversarial attacks has led to the development of adversarial defenses, i.e. techniques aimed at preventing models from being fooled. Some of the first defenses focused on hiding information (e.g. gradients) from the attacker [25, 28, 63]. These defenses were broken in [3, 4, 59].

A different line of work focused on detecting adversarial examples, either by making assumptions regarding their nature [33, 71] or by training detectors to identify common properties [24, 27]. These defenses were broken in [12, 34, 85].

Defensive distillation [58, 60] attempted to improve robustness by making a student network learn to imitate a temperature-scaled teacher model. The defense was broken by adjusting existing techniques in [11]. Early attempts to combine multiple defenses were also found to be ineffective [31].

The same pattern repeated for MagNet [48] and Bounded ReLU [103] (broken in [13]), as well as relatively recent defenses such as k-Winners Take All [96] and Mixup Inference [57] (broken in [85]).

One of the few consistently effective defenses in the literature is adversarial training [7, 81] (specifically, PGD adversarial training [47]). In adversarial training, the model is trained on a dataset augmented with adversarial examples. While this approach provides notable robustness gains, it often comes at the expense of other properties such as accuracy [47] and fairness [99].

Refer to [2] for an in-depth overview of recent developments in the field.

2.1.4 Provable Methods

Robustness bounds for NNs were first provided in [81], followed by [32] and [92]. One major breakthrough was the introduction of automatic verification tools, such as the Reluplex solver [36]. However, the same work also showed that proving arbitrary properties of a ReLU network is NP-complete.

Researchers tried to address this issue by working in three directions. The first is building more efficient solvers based on alternative formulations [22, 76, 83]. The second involves training models that can be verified with less computational effort [43, 97] or provide inherent robustness bounds [77]. The third focuses on guaranteeing robustness under specific threat models [30] or input distribution assumptions [17, 65, 77]. Since all these approaches have limitations that reduce their applicability [75], heuristic defenses tend to be more common in practice.

Exact approaches can also be used to compute provably optimal adversarial examples [10, 83]. Such optimal attacks are subject to the same limitations as provable defenses and thus have limited applicability.

2.1.5 Robustness Studies

Another line of research has focused on understanding the nature of robustness and adversarial attacks. Frameworks such as [21], [62] and [64] focused on formalizing the concept of adversarial robustness. Some studies have also highlighted trade-offs between robustness (under specific definitions) and properties such as accuracy [20, 105], generalization [50] and invariance [84]. However, some of these results have been recently questioned, suggesting that these trade-offs might not be inherent in considered approaches [100, 107].

Adversarial attacks have also been studied from the point of view of Bayesian learning, which has been used to derive robustness bounds and provide insight into the role of uncertainty [67, 70, 88].

Finally, adversarial attacks have also been studied in the context of game theory [69], identifying Nash equilibria between attackers and defenders [54, 108].

2.2 Formalization

In this section, we provide key definitions and notations for our analysis. Compared to existing frameworks, we focus on obtaining formulations for strict forms of defenses and attacks: while less practical for designing algorithms, this approach will clarify the motivations for our identified asymmetries.

Let $f_\theta : X \to \{1, \dots, N\}$ be a discrete classifier with parameter vector θ. Let $d : X \times X \to \mathbb{R}^+$ be a function representing a measure of the similarity between two examples. Let $B(x, \varepsilon) = \{x' \in X \mid d(x', x) \leq \varepsilon\}$ be a ball of radius ε and center x. We define an adversarial example as follows:

Definition 1 (Adversarial Example)**.** *Given an input x, an adversarial example is an input $x' \in B(x, \varepsilon)$ such that $f_\theta(x') \in C(x)$, where $C(x) \subseteq \{1, \dots, N\} \setminus \{f_\theta(x)\}$*

An *adversarial attack* for a classifier f_θ can then be viewed as follows:

Definition 2 (Adversarial Attack)**.** *An adversarial attack is a function $a_{f,\theta} : X \to X$ that solves (in an heuristic or exact fashion) the following optimization problem:*

$$\arg\min_{x' \in X} \{d(x', x) \mid f_\theta(x') \in C(x)\} \tag{2.1}$$

The attack is then deemed successful if the returned solution $x' = a_{f,\theta}(x)$ also satisfies $d(x', x) \leq \varepsilon$. We say that an attack is *exact* if it finds an optimal solution to Equation (2.1); otherwise, we say that the attack is *heuristic*. An attack is said to be targeted if $C(x) = C_{t,y'}(x) = \{y'\}$ with $y' \neq f_\theta(x)$; it is instead untargeted if $C_u(x) = \{1, \dots, N\} \setminus \{f_\theta(x)\})$.

Additionally, we define the *decision boundary distance* as follows:

Definition 3 (Decision Boundary Distance)**.** *The decision boundary distance of a given input x is the minimum distance between x and another input x' such that $f_\theta(x) \neq f_\theta(x')$.*

Note that this is also the value of $d(a_{f,\theta}(x), x)$ for an exact, untargeted, attack.

Finally, we formalize the concept of robustness. Intuitively, a classifier is robust w.r.t. an example x iff x cannot be successfully attacked. More formally:

Definition 4 (Local Robustness)**.** *f_θ is locally robust w.r.t. an example $x \in X$ iff $\forall x' \in B(x, \varepsilon)$ we have $f_\theta(x') = f_\theta(x)$.*

By extending the reasoning to all possible genuine examples we obtain a definition of global robustness:

Definition 5 (Global Robustness)**.** *Let $\mathcal{D}$ be the genuine data distribution and let $X(\mathcal{D})$ be the projection of its support on X. f_θ is robust w.r.t. $\mathcal{D}$ iff f_θ is locally robust $\forall x \in X(\mathcal{D})$.*

Chapter 3

Adversarial Asymmetries

3.1 Overview

We now proceed to describe the asymmetries between attacks and defenses. Note that not all defenses are affected by all asymmetries. Moreover, a defense that is subject to an asymmetry is not necessarily ineffective, but it will have a higher chance of being circumvented. We start by observing that, based on Definition 1, the problem that an *attack* needs to actually solve to be considered successful is:

$$\text{find } x' \in B(x, \varepsilon) \text{ s.t. } \bigvee_{c \in C(x)} f_\theta(x') = c \tag{3.1}$$

By building over Definition 5 and using a similar rationale, we can determine the problem that a *defense* needs to solve:

$$\text{find } \theta \text{ s.t. } \underbrace{\mathbb{E}_{x,y \sim \mathcal{D}} \left[\mathcal{L}(y, f_\theta(x))\right] \leq \kappa}_{\text{acceptable performance}} \wedge \underbrace{\forall x \in X(\mathcal{D}), \forall x' \in B(x, \varepsilon) : f_\theta(x') = f_\theta(x)}_{\text{global robustness}} \tag{3.2}$$

where $\mathcal{D}$ is the data distribution and κ is a threshold value. The condition is a conjunction of two terms: the left-most one captures the fact that the "defended" model still needs to be acceptably accurate, while the right-most term corresponds to global robustness. Note that here f_θ refers to the combination of the model and any attached defense mechanisms.

Comparing Equation (3.1) and Equation (3.2) suggests that, to some degree, *adversarial attacks can be viewed as a relaxation of adversarial defenses*. This provides the basis for our analysis, throughout which we will assume that both attacks and defenses rely on the same definition of d and ε.

Asymmetry 1 (Target Information Asymmetry)**.** *The attacker can always obtain more information about the defender than the defender can obtain about the attacker.*

Based on Equation (3.1), the attacker needs information about the target model f_θ. However, this can be usually be obtained, since the model necessarily *predates* the attacker. For example, the attacker can use genuine queries to collect samples and train a surrogate model [4, 59]. Defenses often employ information about attacks to reduce the computational cost of the robustness check, i.e. they replace $B(x, \varepsilon)$ in Equation (3.2) with a subset $B_a(x, \varepsilon)$. For instance, they may assume that adversarial examples satisfy some properties or that they follow a specific distribution. However, since attacks are deployed *after* defenses, they can actively try to violate such assumptions [4, 59]. Note that the reasoning stands even if a defense is updated in response to new attacks.

Asymmetry 2 (Data Asymmetry)**.** *Defenses need information regarding the genuine data distribution, while attacks do not.*

This is reflected in Equation (3.1), which contains no reference to the distribution $\mathcal{D}$. In fact, an attack can be executed even if the prediction $f_\theta(x)$ is incorrect. Conversely, information on the genuine distribution (labeled or unlabeled) is needed specifically by some defenses, and in general for training the classifier. Such information typically comes in the form of samples (e.g. a training set), thus leading to generalization errors that increase the success chance of attacks. This is the case for MagNet [48], which was broken in [13] by finding adversarial examples that are also classified as in-distribution. A similar issue would arise by using a small sample size in adversarial training.

Asymmetry 3 (Objective Asymmetry)**.** *While the attacker only needs to satisfy the constraints mentioned in Equation* (3.1)*, the defender also needs to account for the model loss.*

The natural loss $\mathcal{L}$ is targeted explicitly when the defense is stated as an optimization problem (e.g. adversarial training) or by adding a constraint (such as in Equation (3.2)). Even when this is not the case, no defense mechanism can compromise the accuracy of a model to the point that it becomes useless. This additional requirement can be at odds with robustness, as discussed in [86]. Additionally, solving Equation (3.2) can be expected to be more challenging than solving Equation (3.1).

Asymmetry 4 (Input Space Asymmetry)**.** *Defended models are expected to be robust on the input distribution, while attacks only focus on a single provided starting point.*

The defender does not know in advance which inputs will be provided to the model and will therefore need to aim for global robustness, or more likely to an approximation due to the involved computational cost. Typically, the term $\forall x \in X(\mathcal{D})$ is replaced with $\forall x \in \hat{X}$, where $\hat{X}$ is the training set. Such an approach may lead to exploitable generalization errors, while still leaving a computational advantage to the attacker as long as $|\hat{X}| > 1$.

Asymmetry 5 (Success Condition Asymmetry)**.** *While a defense needs to ensure that* every *input close to the original has the same prediction, an attack only needs to find* one *adversarial example.*

In other words, even one blind spot is sufficient for the attack to succeed and the defense to fail. Formally, Equation (3.1) has the term $\forall x' \in B(x, \varepsilon)$, while its counterpart in Equation (3.1) lacks universal quantification. This asymmetry is particularly relevant for high-dimensional input spaces.

Asymmetry 6 (Constraint Asymmetry)**.** *A defense needs to ensure a specific prediction for inputs close to the original, while an attack may succeed with more than one prediction value.*

Specifically, the strictest possible constraint for Equation (3.1) is obtained for $|C(x)| = 1$, which will be typically as restrictive as the constraint $f_\theta(x') = f_\theta(x)$ in Equation (3.2). As a consequence, untargeted attacks can be expected to have an additional advantage over defenses. Note that this observation may not hold in cases where certain classes are rarer than others (e.g. anomaly detection) and generally depends on the shape of the decision boundaries over the input space X.

3.2 Computational Robustness

A direct consequence of our definition of heuristic defense is that, given enough computational effort, *any heuristic defense can be fooled.* In the worst case, a motivated attacker could create an almost-perfect clone of f_θ and run a brute force attack on every input in $B(x, \varepsilon)$. We thus argue that heuristic defenses should instead focus on *computational* robustness. We define computational robustness informally as follows:

Definition 6. *A defense is computationally robust if the computational resources required to fool it represent a higher cost than the potential gain.*

A game-theoretical analysis of the game between the attacker and the defender would be heavily influenced by the choice of utility functions, which are fundamentally dependent on the specific application context of the classifier. For this reason, we instead focus on the more general (and tractable) problem of maximizing the difference between the computational complexity of defending and attacking.

In other words, if f_θ is a defended model, achieving computational robustness involves increasing the complexity of the following constraint satisfaction problem:

$$\text{find } x' \in B(x, \varepsilon) \text{ s.t. } f_\theta(x') \in C(x) \wedge d(x', a_{f,\theta}(x')) - b(x', a_{f,\theta}(x')) > \varepsilon \quad (3.3)$$

Unlike provable robustness, computational robustness is compatible with efficient defenses and is thus more suitable for real-world applications.

A trivial analysis from the point of view of computational robustness involves the sample efficiency of the attack. If a defense has complexity $O(f)$ and an attack requires $O(g)$ calls to the model to find an adversarial example, then the complexity of fooling the defense is in $O(fg)$. However, a custom-crafted attack could exploit the structure of the defense in order to achieve a higher sample efficiency. Proving a lower bound on the number of required samples to fool a defense can thus provide evidence in favor of its computational robustness.

A more effective argument involves proving that running a defense and fooling it are complete w.r.t to different levels of the polynomial hierarchy. Under reasonable assumptions (i.e. strict inclusion between the two levels), this difference leads to a significant computational gap between the two problems.

3.3 Guidelines

Based on our analysis, we formulate four guidelines to mitigate the effect of the asymmetries and achieve computational robustness. These guidelines can be used as a "checklist" when designing or evaluating a defense. The checklist can also be used by attackers in order to identify potential weak points of a defense, thus representing a potential ethical concern. We argue that, due to the repeated failures of "security by obscurity" in the context of adversarial machine learning [3, 4, 59], the benefits of exposing potential issues of adversarial defenses outweigh the risks.

Guideline 1. *In order to not be affected by Asymmetry 1 and Asymmetry 2, a defense should not rely on information outside of the target model itself.*

Guideline 2. *In order to not be affected by Asymmetry 3, a defense should not take into account (explicitly or implicitly) the natural loss. Alternatively, the defense should offer an explicit trade-off between robustness and loss so as to align itself with the priorities of the defender.*

Guideline 3. *In order to not be affected by Asymmetry 4, a defense should take the input x as a parameter at inference time.*

Guideline 4. *In order to not be affected by Asymmetry 5 and Asymmetry 6, a defense should solve a problem with success condition and constraints that are not stricter than those of the attack.*

Note that Guideline 4 is often difficult to verify, mostly due to two reasons:

- The problem formulation might not be comparable with the one used by attacks;
- The success condition is often context-dependent.

3.4 Analysis of Existing Defenses

We now analyze whether existing defenses in the literature follow our guidelines in Tables 3.2 and 3.3. Due to the large number of defenses and the frequency of the same observations, we provide a legend for our findings in Table 3.1.

Table 3.1: Reference for observations in Tables 3.2 and 3.3.

Letter	**Observation**
A	Assumes or learns properties of adversarial examples
B	Relies on the assumption that the attacker does not have full information
C	Relies on the assumption that the attacker cannot deal with randomness
D	Requires genuine data
E	Explicit role of loss
F	Implicit loss constraint (effect on loss or accuracy)
G	Provides an explicit accuracy-robustness trade-off
H	A non-trivial part of the computation is performed before the input is known
I	Problem statements are not directly comparable
J	Partially broken
K	Classifies *sets* of inputs

Table 3.2: Relation between existing defenses and guidelines.

Defense	**Follows...**				**Broken?**
	G1	**G2**	**G3**	**G4**	
AAT (detector) [102]	No (D)	No (F)	No	No	[85]
AAT (ensemble) [102]	No (D)	No (F)	No	No	[85]
Adversarial Training [47]	No (D)	No (E)	No	No	No
Adversarial Retraining [27]	No (A, D)	Yes	No	No	[12]
APE-GAN [74]	No (A, D)	No (F)	$\sim$ (H)	N/A (I)	[13]
Are Generative Classifiers... [45]	No (A, D)	Yes	No	N/A (I)	[85]
Artifact Detection [24]	No (A, D)	Yes	No	No	[12]
Attacking Adversarial Attacks [93]	No (A)	No (F)	Yes	Yes	[16]
Beating Attackers... [94]	No (A, D)	Yes	Yes	Yes	No
Bounded ReLU [106]	No (D)	No (F)	Yes	Yes	[13]
Cascade Adversarial Training [52]	No (D)	No (E)	No	No	No
Counteracting Adversarial... [104]	No (A)	No (F)	Yes	Yes	No
Defense-GAN [72]	No (D)	No (F)	$\sim$ (H)	N/A (I)	[4] (J)
Defensive Distillation [60]	No (D)	No (F)	No	N/A (I)	[11]
Defensive Dropout [90]	Yes	No (F)	Yes	Yes	[4] (J)
Dimensionality Reduction [6]	No (A)	No (F)	Yes	N/A (I)	[12]
Dropout Randomization [24]	No (A)	Yes	Yes	Yes	[12] (J)
EMPIR [73]	No (D)	No (E)	No	No	[85]
Ensemble Diversity [56]	No (D)	No (E)	No	No	[85]
Error Correcting Codes [87]	No (A, D)	No (E)	No	No	[85]
Examining Convolutional... [49]	No (A, D)	Yes	No	No	[12] (J)
Gaussian Data Augmentation [106]	No (D)	No (F)	No	N/A (I)	[13]
Hidden Layer PCA [44]	No (A, D)	Yes	No	No	[12]
Input Image PCA [33]	No (A, D)	Yes	No	No	[12]
Input Transformations [28]	No (A, B)	No (F)	Yes	Yes	[4]
k-Winners Take All [96]	No (B)	No (F)	No	Yes	[85]
Local Intrinsic Dimensionality [46]	No (A, D)	Yes	No	No	[4]

Table 3.3: Relation between existing defenses and guidelines (cont.).

Defense	**Follows...**				**Broken?**
	G1	**G2**	**G3**	**G4 (L)**	
MagNet (classifier) [48]	No (A, D)	No (F)	∼ (H)	No	[13]
MagNet (detector) [48]	No (A, D)	Yes	∼ (H)	N/A (I)	[13]
Maximum Mean Discrepancy [27]	No (A, D)	No	N/A (K)	No	[12]
ME-Net [101]	No (A, D)	No (F)	∼ (H)	N/A (I)	[85]
Mean Blur [44]	No (A)	Yes	Yes	No	[12] (J)
Mixup Inference [57]	Yes	No (F)	Yes	Yes	[85]
Mitigating Through... [98]	No (A, C)	No (F)	Yes	Yes	[4]
OOAT [89]	No (D)	Yes (G)	No	No	No
PixelDefend [78]	No (B, D)	No (F)	∼ (H)	N/A (I)	[4]
Re-Attacking [1]	No (A, D)	Yes	∼ (H)	N/A (I)	No
Rethinking Softmax.. [55]	No (D)	No (E)	No	No	[85]
RSFT [5]	No (A)	No (F)	Yes	Yes	[85]
ShieldNets (classifier) [82]	No (D)	No	∼ (H)	N/A (I)	No
ShieldNets (detector) [82]	No (D)	Yes	∼ (H)	N/A (I)	No
Stochastic Activation Pruning [18]	No (C)	No (F)	Yes	Yes	[4]
The Odds are Odd [71]	No (A, B, D)	Yes	No	No	[85]
Thermometer Encoding [25]	No (A, B)	No (F)	No	Yes	[4]
Turning a Weakness... [35]	No (A, D)	Yes	Yes	Yes	[85]

Chapter 4

Counter-Attack

We now introduce Counter-Attack, an asymmetry-free technique to determine whether a model is robust on a given input. We start by observing that, according to Definition 4, a classifier f_θ is *not* locally robust w.r.t. an example x iff:

$$\exists x' \in X : d(x', x) \leq \varepsilon \ \wedge \bigvee_{c \in C_u(x)} f_\theta(x') = c \tag{4.1}$$

If d is symmetric, this equation can be rewritten as:

$$\exists x' \in X : d(x, x') \leq \varepsilon \ \wedge \bigvee_{c \in C_u(x)} f_\theta(x') = c \tag{4.2}$$

This condition is true if and only if x can be successfully attacked using an untargeted approach. For an asymmetric d, one can rework the attack formulation from Equation (2.1) by replacing $d(x', x)$ with $d(x, x')$ and obtain an analogous result. In both cases, we obtain by negation that f_θ is locally robust w.r.t. x *if and only if* x *cannot be successfully attacked*. This is the core idea behind Counter-Attack.

From here we can obtain the actual method by introducing a “buffer” function to account for the gap between the example found by an attack method and the actual optimal solution of Equation (3.1). Formally, let f_θ be a discrete classifier, let $a_{f,\theta} : X \to X$ be an untargeted attack method and let $b : X \times X \to [0, \infty)$ be the buffer function. We define Counter-Attack as follows:

Definition 7 (Counter-Attack)**.** *The CA method marks an example x as not robust if and only if:*

$$d(x, a_{f,\theta}(x)) - b(x, a_{f,\theta}(x)) \leq \varepsilon \tag{4.3}$$

In other words, after finding an adversarial example, we compute its similarity w.r.t. x, we subtract a buffer value and we compare the result with the threshold ε.

The action taken in case an input is marked as non-robust depends on the specific context. Examples include analyzing with slower but more robust tools (e.g. a human being), requesting additional information from the submitter or rejecting the example altogether. Additionally, the flagged examples can be then used to increase the robustness of the model (e.g. through adversarial training).

4.1 Properties

The properties of the attack function $a_{f,\theta}$ and the buffer function b affect those of the method as a whole. Let $d^*(x)$ be the decision boundary distance for x. Then an example x is not robust iff $d^*(x) \leq \varepsilon$. As a consequence:

Statement 1. *If $a_{f,\theta}$ is an an exact attack and b is constant and equal to 0, then CA is an exact robustness check.*

This property is a direct consequence of the fact that for an exact attack we have $d(x, a_{f,\theta}(x)) = d^*(x)$.

If instead the buffer function systematically *under-estimates* the gap w.r.t. the decision boundary distance, we obtain the following property:

Statement 2. *If $\forall x \in X$ we have that $d^*(x) + b(x, a_{f,\theta}(x)) \leq d(x, a_{f,\theta}(x))$, then CA cannot lead to false positives.*

With the stated assumption $d(x, a_{f,\theta}(x)) - b(x, a_{f,\theta}(x))$ will always over-estimate $d^*(x)$, so that any point that is reported as non-robust is guaranteed to be such. This scenario is easy to reproduce in practice by setting $b(x_1, x_2) = 0 \; \forall x_1, x_2 \in X$ and using any attack method.

Conversely, if the buffer function systematically *over-estimates* the similarity gap, we have that any example marked as robust by CA is guaranteed to be robust:

Statement 3. *If $\forall x \in X$ we have that $d^*(x) + b(x, a_{f,\theta}(x)) \geq d(x, a_{f,\theta}(x))$, then CA cannot lead to false negatives.*

This scenario could be achieved by using as an attack method an algorithm with a guaranteed approximation factor, i.e. one such that $d(x, a_{f,\theta}(x)) \leq \alpha d^*(x)$ for a given α. Therefore, the development of such attacks represents a promising research direction towards improving the robustness of future models.

CA does not rely on information outside of the model itself (Guideline 1) and takes x as parameter (Guideline 3). Moreover, it does not affect accuracy (thus preventing the need for implicit constraints related to the loss, as per Guideline 2), although it can increase reliance on other robust classifiers. Finally, using the same constraints as an untargeted attack trivially satisfies Guideline 4.

4.2 Fooling CA

If the buffer function is not guaranteed to be either an over- or under-estimator (e.g. because it has been empirically calibrated), the method offers no theoretical guarantees. Therefore, an attacker could design a method $a^s_{f,\theta}$ to find adversarial examples with an over-estimated decision boundary distance. Formally, the attacker would need to solve the following feasibility problem:

$$\text{find } x' \in B(x, \varepsilon) \text{ s.t. } f_\theta(x') \in C(x) \wedge d(x', a_{f,\theta}(x')) - b(x', a_{f,\theta}(x')) > \varepsilon \quad (4.4)$$

However, even in this situation CA offers two major advantages.

Asymmetry 7 (Defense Information Asymmetry). *The attack used by the attacker has at most as much information regarding the target model as the attack used by the defender.*

In other words, while CA is guaranteed access to the true model, the attacker will obtain a perfect replica only as a best case.

Asymmetry 8 (Defense Complexity Asymmetry). *Counter-Attack can be viewed as a relaxation of the problem of fooling Counter-Attack.*

Specifically, verifying the feasibility of a single input x' in Equation (4.4) involves solving Equation (3.1) while at the same time determining if CA fails to solve an equivalent problem. Depending on how efficiently the attacker can simulate CA, this can significantly increase the computational load.

While neither asymmetry guarantees that Equation (4.4) is infeasible, they both provide a reasonable advantage to the defender, increasing the cost of fooling CA. We also provide a formal analysis of the computational complexity of CA and $a^s_{f,\theta}$ in Chapter 5.

Note that improving CA does not necessarily involve developing stronger attacks. Collecting more information about the behavior of existing attacks, designing weak but predictable attacks and training models that are easier to attack (such as in [97]) are all reasonable ways to decrease the chances of an unexpected over-estimate.

4.3 Buffer Models

We provide an overview of potential buffer models, focusing on their strengths and weaknesses in the context of achieving robustness.

Note that all the listed models require some form of calibration or training, which, if done using sampled data, can represent a violation of Guideline 1. In order to reduce the impact of this violation, designers should thus take into account the potential impact of misplaced assumptions and adapt the model accordingly (e.g. by multiplying $b(x, a_{f,\theta}(x))$ by a fixed correction factor).

4.3.1 Least-Squares Linear Model

A least-squares linear model aims to predict the true decision boundary distance given the heuristic one by minimizing the mean squared error on a given dataset. While easy to train and understand, this type of model tends to be significantly influenced by outliers, which means that a few unusually over-estimated heuristic distances can cause the buffer model to under-estimate the true decision boundary distance.

4.3.2 Quantile Linear Model

A more reliable procedure involves modeling a certain quantile τ of the true decision boundary distance through quantile regression [39]. Such a technique allows the designer to pick the acceptable fraction and type of mistakes on the training data. For certain choices of τ (e.g. 0.5), the model also tends to be more robust to outliers, since unusually high over-estimates have a significantly smaller impact on such quantiles of the distance distribution.

4.3.3 Chance-Constrained Least-Squares Model

A hybrid approach involves solving a constrained least-squares problem, where the model is required to under- or over-estimate a given quantile of the distance distribution:

$$\begin{aligned} &\arg\min_{\alpha} \mathbb{E}_{x \sim \mathcal{D}} \left[\left(d(x, a_{f,\theta}(x)) + b_\alpha(x, a_{f,\theta}(x)) - d^*(x) \right)^2 \right] \\ &\text{s.t. } P(d^*(x) + b_\alpha(x, a_{f,\theta}(x)) \leq d(x, a_{f,\theta}(x))) \geq \tau \end{aligned} \tag{4.5}$$

where α is the parameter vector for b. While partially influenced by outliers, this approach is more robust than regular least-squares models, while also providing a similar intuition.

4.3.4 Beyond Linear Models

Linear models based on the heuristic distance are by no means the only possible approach. It is possible to both use more sophisticated models (e.g. binary trees or NNs) and rely on additional information (e.g. the output of the model or the density of the genuine distribution). However, increasing the complexity of the buffer model carries the risk of introducing flaws that can be exploited by the attacker. Designers should thus be mindful of such a risk when employing large, non-explainable buffer models.

4.4 Comparison with Attack-Based Defenses

Counter-Attack shares some similarities with attack-based defenses, i.e. defenses that employ adversarial attacks. Both CA and attack-based defenses run an adversarial attack on the input that should be classified by the target model. However, these defenses use adversarial attacks not to verify local robustness, but rather to check properties that are assumed to be correlated with adversarial examples (thus violating Guideline 1). Specifically:

- [35] assumes that adversarial examples can be corrupted by Gaussian noise and/or require a high number of iterations to be attacked;
- [93] postulates that the loss function w.r.t. false classes has a greater local Lipschitzness compared to the ground-truth class;
- [94] assumes that perturbing an adversarial example leads to an anomalous variation of the output of the classifier;
- [1] assumes that the difference between the features of an input x and those of $a_{f,\theta}(x)$ can be used to identify adversarial examples;
- [104] assumes that adding targeted perturbations for non-predicted labels to an adversarial example causes the score of the ground-truth class to increase.

At the moment of writing, [35] has been fooled in [85], while [93] has been fooled in [16]. In both cases, the attackers used an approach similar to the one described in Section 3.1. We postulate that applying the same technique to [1], [94] and [104] would lead to similar results.

Chapter 5

Complexity Analysis

In this chapter, we provide theoretical bounds for several classes of relevant problems. We will focus on L_∞-bounded attacks against *ReLU networks*, i.e. networks containing only linear combinations and ReLUs. In the context of CA, we will also restrict our analysis to polynomial-bounded buffer models, i.e. buffer models that can be used in polynomial time w.r.t the size of x.

We prove the following statements:

- All of the following problems are NP-complete:
  - Running an exact attack;
  - Running an exact untargeted attack;
  - Running Counter-Attack in its exact form;
- All of the following problems are $coNP$-complete:
  - Verifying the local robustness of a classifier;
  - Verifying the global robustness of a classifier;
- All of the following problems are Σ_2^P-complete:
  - Fooling CA;
  - Finding a locally robust classifier;

  - Finding a globally robust classifier;
  - Training a defense with constrained loss;

- Training a defense with optimal loss is Σ_2^P-hard.

5.1 Preliminaries

5.1.1 Notation

We use f_i to denote the i-th output of a network. We define f as

$$f(x) = \arg\max_i \{f_i(x)\} \tag{5.1}$$

for situations where multiple outputs are equal to the maximum, we use the class with the lowest index.

5.1.2 Polynomial-Bounded Input Spaces

We focus on polynomial-bounded input spaces, i.e. spaces for which there exists a polynomial p such that $\forall x . \forall \varepsilon . \forall x' \in B(x, \varepsilon) . |x'| \leq p(|x| + |\varepsilon|)$, where $|x|$ is the size of the representation of x. Examples of polynomial-bounded input spaces (using L_∞ balls) include:

- Integers;
- Fixed-point numbers;
- IEE754 floating-point numbers;
- Fixed-length vectors of all of the above;
- Images with known size and bit depth;
- Audio files and videos with known duration and bitrate.

A consequence of this definition is that the cardinality of $B(x, \varepsilon)$ is finite for all choices of x and ε. This implies that, for all values of ϵ, given $X' \subseteq B(x, \varepsilon)$ there exists a value μ such that $\forall x', x'' \in X'. \mu \leq d(x', x'')$.

Note that, given a sufficiently small choice of μ, it is possible to compute the value μ' of a polynomial bounded space that has been transformed by the following functions:

- Sum;
- Multiplication by constants;
- ReLU.

in polynomial time w.r.t. the size of the inputs.

5.1.3 Functions

We now provide an overview of several functions that can be obtained by using linear combinations and ReLUs.

max [10] showed that we can implement the max function using linear combinations and ReLUs as follows:

$$\max(x, y) = ReLU(x - y) + y \tag{5.2}$$

We can also obtain an n-ary version of max by chaining multiple instances together.

step If X is a polynomial-bounded space and $x_s \in X$, then $\forall \varepsilon > 0$ and $\forall x \in B(x_s, \varepsilon)$ the following function:

$$step_0(x) = \frac{1}{\mu}\left(ReLU(x) - ReLU(x - \mu)\right) \tag{5.3}$$

is such that $step_0(x) = 0$ for all representable $x \leq 0$ and $step_0(x) = 1$ for all representable $x > 0$.

Similarly, let $step_1$ be defined as follows:

$$step_1(x) = \frac{1}{\mu}\left(ReLU(x+\mu) - ReLU(x)\right) \tag{5.4}$$

Note that $step_1(x) = 0$ for all representable $x < 0$ and $step_1(x) = 1$ for all representable $x \geq 0$.

Boolean Functions We then define the Boolean functions $neg : \{0,1\} \to \{0,1\}$, $and : \{0,1\} \times \{0,1\} \to \{0,1\}$ and $or : \{0,1\} \times \{0,1\} \to \{0,1\}$ as follows:

$$not(x) = 1 - x \tag{5.5}$$

$$and(x,y) = step_1(x + y - 2) \tag{5.6}$$

$$or(x,y) = step_1(x + y) \tag{5.7}$$

Note that we can obtain n-ary variants of *and* and *or* by chaining multiple instances together.

***cnf*₃** Given a set $z = \{\{z_{1,1}, \dots, z_{1,3}\}, \dots, \{z_{n,1}, z_{n,3}\}\}$ of Boolean atoms (i.e. $z_{i,j}(x) = x_k$ or $\neg x_k$ for a certain k) defined on an n-long Boolean vector, $cnf_3(z)$ returns the following Boolean function:

$$cnf_3'(x) = \bigwedge_{i=1,\dots,n} \bigvee_{j=1,\dots,3} z_{i,j}(x) \tag{5.8}$$

We refer to z as a 3CNF formula.

Since cnf_3' only uses negation, conjunction and disjunction, it can be implemented using respectively *neg*, *and* and *or*. Note that, given z, we can build cnf_3' in polynomial time.

Comparison Functions We can use $step_0$, $step_1$ and neg to obtain comparison functions as follows:

$$geq(x,k) = step_1(x-k) \tag{5.9}$$

$$gt(x,k) = step_0(x,k) \tag{5.10}$$

$$leq(x,k) = not(gt(x,k)) \tag{5.11}$$

$$lt(x,k) = not(geq(x,k)) \tag{5.12}$$

$$eq(x,k) = and(geq(x,k), leq(x,k)) \tag{5.13}$$

Moreover, we define *open* as follows:

$$open(x,a,b) = and(gt(x,a), lt(x,b)) \tag{5.14}$$

5.2 Untargeted Attacks are NP-Complete

We will now prove that running an untargeted attack with L_∞-bounded perturbations (which we will refer to as U-ATT) is NP-complete.

5.2.1 Definition of U-ATT

We define U-ATT as the set of all tuples $\langle x, \varepsilon, f \rangle$ such that:

$$\exists x' \in B(x,\varepsilon). f(x') \neq f(x) \tag{5.15}$$

where:

- $x \in X$;
- X is a polynomial-bounded space;
- f is a ReLU classifier;
- $B(x,\varepsilon) = \{x' \in X \mid \|x - x'\|_\infty \leq \varepsilon\}$, where $\|\cdot\|_\infty$ is the L^∞ norm,

Theorem 1. *U-ATT is NP-complete.*

5.2.2 *U-ATT* $\in$ *NP*

To prove that U-$ATT \in NP$, we show that there exists a polynomial-bounded certificate for U-ATT that can be checked in polynomial time. The certificate is the value of x', which will have polynomial size w.r.t to the size of x (due to the polynomial-bounded space assumption) and can be checked by verifying:

- $\|x - x'\|_\infty \leq \varepsilon$, which can be checked in linear time;
- $f_\theta(x') \neq f(x)$, which can be checked in polynomial time.

5.2.3 *U-ATT* is *NP*-Hard

We will prove that ATT is NP-Hard by showing that $3SAT \leq U$-ATT.

Given a set of 3CNF clauses $z = \{\{z_{11}, z_{12}, z_{13}\}, \ldots, \{z_{m1}, z_{m2}, z_{m3}\}\}$ defined on n Boolean variables $x_1, \ldots, x_n$, we construct the following query $q(z)$ for U-ATT:

$$q(z) = \langle x_s, \frac{1}{2}, f \rangle \tag{5.16}$$

where $x_s = \left(\frac{1}{2}, \ldots, \frac{1}{2}\right)$ is a vector with n elements. Verifying $q(z) \in U$-ATT is equivalent to checking:

$$\exists x' \in B\left(x_s, \frac{1}{2}\right) . f(x') \neq f(x_s) \tag{5.17}$$

Note that $x \in B\left(x_s, \frac{1}{2}\right)$ is equivalent to $x \in [0, 1]^n$.

Truth Values We will encode the truth values of $\hat{x}$ as follows:

$$x'_i \in \left[0, \frac{1}{2}\right] \iff \hat{x}_i = 0 \tag{5.18}$$

$$x'_i \in \left(\frac{1}{2}, 1\right] \iff \hat{x}_i = 1 \tag{5.19}$$

We can obtain the truth value of a scalar variable by using $isT(x_i) = gt\left(x_i, \frac{1}{2}\right)$. Let $bin(x) = or(isT(x_1), \ldots, isT(x_n))$.

Definition of *f* We define f as follows:

$$f_1(x) = and(not(isx_s(x)), cnf_3'(bin(x))) \tag{5.20}$$

$$f_0(x) = not(f_1(x)) \tag{5.21}$$

where $cnf_3' = cnf_3(z)$ and isx_s is defined as follows:

$$isx_s(x) = and\left(eq\left(x_1, \frac{1}{2}\right), \ldots, eq\left(x_n, \frac{1}{2}\right)\right) \tag{5.22}$$

Note that f is designed such that $f(x_s) = 0$.

Lemma 1.1. $z \in 3SAT \implies q(z) \in U\text{-}ATT$

Proof. Let $z \in 3SAT$. Therefore $\exists x^* \in \{0,1\}^n$ such that $cnf_3(z)(x^*) = 1$. Since $bin(x^*) = x^*$ and $x^* \neq x_s$, $f(x^*) = 1$, which means that it is a valid solution for Equation (5.17). From this we can conclude that $q(z) \in U\text{-}ATT$. □

Lemma 1.2. $q(z) \in U\text{-}ATT \implies z \in 3SAT$

Proof. Since $q(z) \in U\text{-}ATT$, $\exists x^* \in [0,1]^n \setminus \{x_s\}$ that is a solution to Equation (5.17). Then $cnf_3'(bin(x^*)) = 1$, which means that there exists a $\hat{x}$ (i.e. $bin(x^*)$) such that $cnf_3'(\hat{x}) = 1$. From this we can conclude that $z \in 3SAT$. □

Since:

- $q(z)$ can be computed in polynomial time;
- $z \in 3SAT \implies q(z) \in U\text{-}ATT$;
- $q(z) \in U\text{-}ATT \implies z \in 3SAT$.

we can conclude that $3SAT \leq U\text{-}ATT$.

5.2.4 Corollaries

Corollary 1.1 (Attacks are $\boldsymbol{NP}$-Complete)**.** *Let ATT be the set of all tuples $\langle x, \varepsilon, f, C\rangle$ such that:*

$$\exists x' \in B(x, \varepsilon). f(x') \in C(x) \tag{5.23}$$

where $f(x') \in C(x)$ can be verified in polynomial time. Then ATT is NP-complete.

Proof. ATT is NP-hard due to U-ATT being a special case of ATT where $C = C_u$. Moreover, since $f(x') \in C(x)$ can be verified in polynomial time, there exists a polynomial-bounded certificate (the same as the one for ATT) that can be checked in polynomial time. □

Corollary 1.2 (Counter-Attack is $\boldsymbol{NP}$-Complete)**.** *Let CA be the set of all tuples $\langle x, \varepsilon, f, b, a_f\rangle$ such that:*

$$\exists x' \in B(x, \varepsilon + b(x, a_f(x))). f_\theta(x') \neq f(x) \tag{5.24}$$

where b and a_f can be computed in polynomial time. Then CA is NP-complete.

Proof. Since a certificate for CA can be verified in polynomial time, $CA \in NP$. Additionally, CA is NP-hard due to the fact that U-ATT is a special case of CA where $b(x, a_f(x)) = 0$. Therefore, CA is NP-complete. □

Note that since a_f must be computable in polynomial time, and since exact attacks are NP-complete, a_f cannot be an exact attack, unless $P = NP$.

Corollary 1.3 (Verifying Local Robustness is $\boldsymbol{coNP}$-Complete)**.** *Let L-ROB be the set of all tuples $\langle x, \varepsilon, f\rangle$ such that:*

$$\forall x' \in B(x, \varepsilon). f(x') = f(x) \tag{5.25}$$

Then L-ROB is coNP-complete.

Proof. *L-ROB* is *coNP*-complete due to being the complement of U-ATT, which is NP-complete. □

Corollary 1.4 (Verifying Global Robustness is $\boldsymbol{coNP}$-Complete)**.** *Let G-ROB be the set of all tuples* $\langle \chi, \varepsilon, f \rangle$ *such that:*

$$\forall x \in X. \left(\chi(x) \implies (\forall x' \in B(x, \varepsilon). f(x') = f(x))\right) \tag{5.26}$$

where X *is a polynomial-bounded space and* $\chi(x)$ *can be verified in polynomial time. Then G-ROB is* $coNP$*-complete.*

Proof. We first prove that *G-ROB* $\in coNP$. Let *coG-ROB* be the set of all tuples $\langle \chi, \varepsilon, f \rangle$ such that:

$$\exists x. \chi(x) \wedge (\exists x' \in B(x, \varepsilon). f(x') \neq f(x)) \tag{5.27}$$

Since there exists a polynomial-boundeded certificate for *coG-ROB* that can be verified in polynomial time, *coG-ROB* $\in NP$ and thus *G-ROB* $\in coNP$.

We now prove that *L-ROB* is $coNP$-hard. This is a consequence of the fact that *L-ROB* is a special case of *G-ROB* where $\chi(x)$ for only one x.

Therefore, *G-ROB* is $coNP$-complete. □

5.3 Fooling CA is $\boldsymbol{\Sigma_2^P}$-Complete

We now show that the problem of fooling CA (which we will refer to as CCA) is Σ_2^P-complete.

5.3.1 Definition of $\boldsymbol{CCA}$

We formalize CCA as the set of all tuples $\langle x, \varepsilon, C, f, b, a_f \rangle$ such that:

$$\exists x' \in B(x, \varepsilon). \left(f(x') \in C(x) \wedge d(x', a_f(x')) - b(x', a_f(x')) > \varepsilon\right) \tag{5.28}$$

where:

- $x \in X$;
- X is a polynomial-bounded space;
- f is a ReLU classifier;

- Whether an output is in $C(x^*)$ for some x^* can be decided in polynomial time;
- b can be computed in polynomial time;
- a_f can be computed in polynomial time;
- $B(x, \varepsilon) = \{x' \in X \mid \|x - x'\|_\infty \leq \varepsilon\}$.

Theorem 2. *CCA is Σ_2^P-complete.*

5.3.2 Preliminaries

Alternative Formulation For ease of reading, we rewrite Equation (5.28) as:

$$\exists x' \in B(x, \varepsilon).f(x') \in C(x) \land \forall x'' \in B(x', \varepsilon + b(x', a_f(x'))).f(x'') = f(x') \tag{5.29}$$

$\Sigma_2 3SAT$ $\forall\exists 3SAT$ is the set of all z such that:

$$\forall \hat{x} \exists \hat{y}.R(\hat{x}, \hat{y}) \tag{5.30}$$

where $R(\hat{x}, \hat{y}) = cnf_3(z)(\hat{x}_1, \dots, \hat{x}_n, \hat{y}_1, \dots, \hat{y}_n)$.

[79] showed that $\Pi_2 3SAT$ is Π_2^P-complete. Therefore, $\Sigma_2 3SAT$, which is defined as the set of all z such that:

$$\exists \hat{x} \forall \hat{y} \neg R(\hat{x}, \hat{y}) \tag{5.31}$$

is Σ_2^P-complete.

5.3.3 $CCA \in \Sigma_2^P$

By the Kuratowski-Tarski Theorem, $CCA \in \Sigma_2^P$ iff there exists a problem $A \in P$ and a polynomial p such that $\forall \Gamma = \langle x, \varepsilon, C, f, b \rangle$:

$$\Gamma \in CCA \iff \exists y.|y| \leq p\left(|\Gamma|\right) \land (\forall z.(|z| \leq p(|\Gamma|) \implies \langle \Gamma, y, z \rangle \in A)) \tag{5.32}$$

This can be proven by setting $y = x'$,$z = x''$ and A as the set of all triplets $\langle \Gamma, x', x'' \rangle$ such that all of the following are true:

- $\|x - x'\|_\infty \leq \varepsilon$
- $f(x') \in C(x)$
- $\|x'' - x'\|_\infty \leq \varepsilon + b(x', a_f(x'))$
- $f(x'') = f(x')$

Since all properties can be checked in polynomial time, $A \in P$.

5.3.4 CCA is Σ_2^P-Hard

We will show that CCA is Σ_2^P-hard by proving that $\Sigma_2 3SAT \leq CCA$.

First, suppose that the length of $\hat{x}$ and $\hat{y}$ differ. In that case, we pad the shortest one with additional variables that will not be used.

Let n be the maximum of the lengths of $\hat{x}$ and $\hat{y}$.

Given a set z of 3CNF clauses, we construct the following query $q(z)$ for CCA:

$$q(z) = \langle x_s, \frac{1}{2}, C, h, b \rangle \tag{5.33}$$

where $x_s = \left(\frac{1}{2}, \dots, \frac{1}{2}\right)$ is a vector with n elements and where b is defined as:

$$b(x', a_{f,\theta}(x')) = \begin{cases} -\varepsilon + 2\gamma & x' \in \{0,1\}^n \\ -\varepsilon + \delta & \text{otherwise} \end{cases} \tag{5.34}$$

where $0 < \gamma < \frac{1}{4}$ and $\delta > 1 + 2\gamma$. Verifying $q(z) \in CCA$ is equivalent to checking:

$$\exists x' \in B\left(x_s, \frac{1}{2}\right). \left(h(x') \in C(x) \wedge \left(\forall x'' \in B(x', \varepsilon + b(x', a_{f,\theta}(x'))).\ h(x'') = h(x')\right)\right) \tag{5.35}$$

Note that $x' \in [0,1]^n$.

Truth Values We will encode the truth values of $\hat{x}$ and $\hat{y}$ as follows:

$$\begin{aligned} x_i'' = -2\gamma &\iff \hat{x}_i = 0 \wedge \hat{y}_i = 0 \\ x_i'' = -\gamma &\iff \hat{x}_i = 0 \wedge \hat{y}_i = 1 \\ x_i'' = 1 + \gamma &\iff \hat{x}_i = 1 \wedge \hat{y}_i = 0 \\ x_i'' = 1 + 2\gamma &\iff \hat{x}_i = 1 \wedge \hat{y}_i = 1 \end{aligned} \tag{5.36}$$

Let $e_{\hat{x}}(x) = gt(x, 1)$. Let:

$$e_{\hat{y}}(x) = or(and(geq(x, \gamma), lt(x, 0)), and(gt(x, 1 + \gamma), leq(x, 1 + 2\gamma))) \tag{5.37}$$

Note that $e_{\hat{x}}(x_i'')$ returns the truth value of $\hat{x}_i$ and $e_{\hat{y}}(x_i'')$ returns the truth value of $\hat{y}_i$. We will also use $e^{-1}(\hat{x}, \hat{y})$ to denote the encoding of $\hat{x}$ and $\hat{y}$.

Definition of *h* Let g be a Boolean formula defined over

$$e_{\hat{x}}(x_1), \dots, e_{\hat{x}}(x_n), e_{\hat{y}}(x_1), \dots, e_{\hat{y}}(x_n)$$

that returns the value of R (using the same technique as cnf_3'). Let inv_F be:

$$inv_F(x) = \max_{i=1,\dots,n} or(open(x_i, -2\gamma, -\gamma), open(x_i, -\gamma, 1+\gamma), open(x_i, 1+\gamma, 1+2\gamma)) \tag{5.38}$$

Let inv_T be:

$$inv_T(x) = \max_{i=1,\dots,n} or(lt(x_i, -2\gamma), gt(x_i, 1 + 2\gamma)) \tag{5.39}$$

We define h as a two-class classifier, where:

$$h_1(x) = or(inv_T(x), and(not(inv_F(x)), g(x))) \tag{5.40}$$

and $h_0(x) = not(h_1(x))$.

Note that:

- If $x_i \in (-\infty, -2\gamma) \cup (1 + 2\gamma, +\infty)$ for some i, the top class is 1;
- Otherwise, if $x_i \in (-2\gamma, -\gamma) \cup (-\gamma, 1 + \gamma) \cup (1 + \gamma, 1 + 2\gamma)$, the top class is 0;
- Otherwise, the top class is 1 if the formula is true and 0 if the formula is false.

Additionally, $\forall x' \in B\left(x_s, \frac{1}{2}\right).h(x') = 0$.

Definition of C We define C as $C(x) = \{0, 1\}$.

Lemma 2.1. $z \in \Sigma_2 3SAT \implies q(z) \in CCA$

Proof. If $z \in \Sigma_2 3SAT$, then there exists a Boolean vector x^* such that $\forall \hat{y}. \neg R(x^*, \hat{y})$.

Then all of the following statements are true:

- $h(x^*) = 0$, since $x^* \in \{0, 1\}^n \subseteq (-\gamma, 1 + \gamma)^n$;
- $h(x^*) \in C(x_s)$, since $C(x_s) = \{0, 1\}$;
- $b(x^*, a_h(x^*)) = -\varepsilon + 2\gamma$, since $x^* \in \{0, 1\}^n$. As a consequence, the inner ball is $B(x^*, 2\gamma)$.

We still need to prove that, $\forall x^{**} \in B(x^*, 2\gamma)$, $h(x'') = h(x')$, which we have already shown to be equal to 0.

Let $x^{**} \in B(x^*, 2\gamma)$. For all $i = 1, \dots, n$, $x_i^{**} \in [x_i^* - 2\gamma, x_i^* + 2\gamma]$. In other words, $x_i^{**} \in [-2\gamma, 2\gamma]$ or $x_i^{**} \in [1 - 2\gamma, 1 + 2\gamma]$, depending on the value of x_i^{**}.

There are two cases:

- $x_j^{**} \notin \{-2\gamma, -\gamma, 1 + \gamma, 1 + 2\gamma\}$ for at least one j:
  - There are two sub-cases:
    - $x_j^* = 0$:
      - Then $x_j^{**} \in (-2\gamma, \gamma) \cup (\gamma, 2\gamma]$;
      - Since this interval is a subset of $(-2\gamma, -\gamma) \cup (-\gamma, 1 + \gamma) \cup (1 + \gamma, 1 + 2\gamma)$, $h(x^{**}) = 0$;
    - $x_j^* = 1$:
      - Then $x_j^{**} \in [1 - 2\gamma, 1 + \gamma) \cup (1 + \gamma, 1 + 2\gamma)$;
      - Since this interval is a subset of $(-2\gamma, -\gamma) \cup (-\gamma, 1 + \gamma) \cup (1 + \gamma, 1 + 2\gamma)$, $h(x^{**}) = 0$;
- $x_j^{**} \in \{-2\gamma, -\gamma, 1 + \gamma, 1 + 2\gamma\}$ for all j:

- Then the top class of h is either 0 or 1, depending on the truth value of $R(\hat{x}, \hat{y})$;
- Since the original formula is false for all choices of $\hat{y}$ (and thus all possible encodings of $\hat{x}$ and $\hat{y}$), $h(x^{**}) = 0$.

Therefore, we proved that all conditions are satisfied, which means that $q(z) \in CCA$.

□

Lemma 2.2. $q(z) \in CCA \implies z \in \Sigma_2 3SAT$

Proof. Since $q(z) \in CCA$, there exists a $x^* \in B\left(x_s, \frac{1}{2}\right)$ such that $h(x^*) \in C(x_s)$ and $\forall x'' \in B(x^*, \varepsilon + b(x^*, a_h(x^*)).h(x'') = h(x')$. We will prove that x^* is a solution to $\Sigma_2 3SAT$.

We first prove by contradiction that $x^* \in \{0,1\}^n$.

Suppose that $x_i^* \in (0,1)$ for some i. Then $b(x^*, a_h(x^*)) = -\varepsilon + \delta$, which means that the inner ball is $B(x^*, \delta)$. Let x^{**} be defined as follows:

$$x_j^{**} = \begin{cases} \delta & j = i \\ x_j^* & \text{otherwise} \end{cases} \tag{5.41}$$

Note that $x^{**} \in B(x^*, \delta)$. $h(x^{**}) = 1$, since $x_i^* = \delta \in (1 + 2\gamma, +\infty)$. However, $h(x^*) = 0$, since $x^* \in [0,1]^n$. This implies that there exists a $x^{**} \in B(x^*, \varepsilon + b(x', a_h(x^*))$ such that $h(x^*) \neq h(x^{**})$, which contradicts the hypothesis that $\forall x'' \in B(x^*, \varepsilon + b(x^*, a_h(x^*)).h(x'') = h(x^*)$. Therefore, $x^* \in \{0,1\}^n$.

Since $x^* \in \{0,1\}^n$, $h(x^*) = 0$ and $b(x^*, a_h(x^*)) = -\varepsilon + 2\gamma$, which means that the inner ball is $B(x^*, 2\gamma)$. This ball thus contains the encodings $e^{-1}(x^*, \hat{y})$ for all possible choices of $\hat{y}$.

Consider a generic $\hat{y}$. Then $e^{-1}(x^*, \hat{y}) \in \{1 - 2\gamma, 1 - \gamma, 1 + \gamma, 1 + 2\gamma\}^n$, which means that $h(e^{-1}(x^*, \hat{y})) = 1$ iff the formula is true. Since we know that $h(x^*) = 0$ and since $\forall x'' \in B(x^*, 2\gamma).h(x'') = h(x')$, we can conclude that $h(e^{-1}(x^*, \hat{y})) = 0$ and thus $R(x^*, \hat{y})$ is false.

In other words, $R(x^*, \hat{y})$ is false for all choices of $\hat{y}$, which means that x^* is a solution to Equation (5.31) and thus that $z \in \Sigma_2 3SAT$.

□

Since:

- $q(z)$ can be computed in polynomial time;
- $z \in \Sigma_2 3SAT \implies q(z) \in CCA$;
- $q(z) \in CCA \implies z \in \Sigma_2 3SAT$;

we can conclude that $\Sigma_2 3SAT \leq CCA$.

5.4 Finding a Locally Robust Model is Σ_2^P-Complete

Finally, we shift our attention towards the parameterized version of *L-ROB*, which involves finding the parameters for a classifier such that the resulting model is locally robust.

5.4.1 Definition of *PL-ROB*

Let *PL-ROB* be the set of tuples $\langle x, \varepsilon, f_\theta, v \rangle$ such that:

$$\exists \theta' \in v(f). \forall x' \in B(x, \varepsilon). f_{\theta'}(x') = f_{\theta'}(x) \tag{5.42}$$

where:

- $x \in X$;
- X is a polynomial-bounded space;
- $v(f)$ is the set of valid parameter vectors for f;
- $\theta' \in v(f)$ can be checked in polynomial time w.r.t. the size of the tuple.

Theorem 3. *PL-ROB is Σ_2^P-complete.*

5.4.2 *PL-ROB* $\in \Sigma_2^P$

Similarly to the proof for CCA, we can prove that $PL\text{-}ROB \in \Sigma_2^P$ by showing that there exists a problem $A \in P$ and a polynomial p such that $\forall \Gamma = \langle x, \varepsilon, f_\theta, v \rangle$:

$$\Gamma \in PL\text{-}ROB \iff \exists y.|y| \leq p(|\Gamma|) \wedge (\forall z.(|z| \leq p(|\Gamma|) \implies \langle \Gamma, y, z \rangle \in A)) \tag{5.43}$$

This can be proven by setting $y = \theta'$, $z = x'$ and A as the set of triplets $\langle \Gamma, \theta', x' \rangle$ such that all of the following are true:

- $\theta' \in v(f)$;
- $\|x - x'\|_\infty \leq \varepsilon$;
- $f_\theta(x) = f_\theta(x')$.

Since all properties can be checked in polynomial time, $A \in P$ and thus $PL\text{-}ROB \in \Sigma_2^P$.

5.4.3 *PL-ROB* is Σ_2^P-Hard

We will prove that $PL\text{-}ROB$ is Σ_2^P-hard by showing that $\Sigma_2 3SAT \leq PL\text{-}ROB$.

Let $n_{\hat{x}}$ be the length of $\hat{x}$ and let $n_{\hat{y}}$ be the length of $\hat{y}$.

Given a set z of 3CNF clauses, we construct the following query $q(z)$ for $PL\text{-}ROB$:

$$q(z) = \langle x_s, \frac{1}{2}, f_\theta, v \rangle \tag{5.44}$$

where $x_s = \left(\frac{1}{2}, \ldots, \frac{1}{2}\right)$ is a vector with $n_{\hat{y}}$ elements and $v(f) = \{0, 1\}^{n_{\hat{x}}}$. Note that $\theta' \in v(f)$ can be checked in polynomial time w.r.t. the size of the tuple.

Truth Values We will encode the truth values of $\hat{x}$ using θ', while we will encode the truth values of $\hat{y}$ using x' through the same technique mentioned in Section 5.2.3.

Definition of f_θ We define f_θ as follows:

- $f_{\theta,1}(x) = and(not(isx_s(x)), cnf_3''(x))$, where cnf_3'' is defined over θ and $bin(x)$ using the same technique mentioned in Section 5.2.3;
- $f_{\theta,0}(x) = not(f_{\theta,1}(x))$.

Note that $f_\theta(x_s) = 0$ for all choices of θ. Additionally, f_θ is designed such that:

$$\forall x' \in B\left(x_s, \frac{1}{2}\right) \setminus \{x_s\}.\forall \theta' \in v(f).(f_{\theta'}(x') = 1 \iff R(\theta', bin(x')))) \tag{5.45}$$

Lemma 3.1. $z \in \Sigma_2 3SAT \implies q(z) \in PL\text{-}ROB$

Proof. Since $z \in \Sigma_2 3SAT$, there exists a Boolean vector x^* such that $\forall \hat{y}.\neg R(x^*, \hat{y})$.

Then both of the following statements are true:

- $x^* \in v(f)$, since $x^* \in \{0,1\}^{n_{\hat{x}}}$;
- $\forall x' \in B(x_s, \varepsilon).f_{x^*}(x') = 0$, since $f_{x^*}(x') = 1 \iff R(x^*, bin(x'))$;

Therefore, x^* is a valid solution for Equation (5.42) and thus $q(z) \in PL\text{-}ROB$.

□

Lemma 3.2. $q(z) \in PL\text{-}ROB \implies z \in \Sigma_2 3SAT$

Proof. Since $q(z) \in PL\text{-}ROB$, there exists a θ^* such that:

$$\theta^* \in v(f) \wedge \forall x' \in B(x_s, \varepsilon).f_{\theta^*}(x') = f_{\theta^*}(x_s) \tag{5.46}$$

Note that $\theta^* \in \{0,1\}^{n_{\hat{x}}}$, since $\theta^* \in v(f)$. Moreover, $\forall \hat{y}.\neg R(\theta^*, \hat{y})$, since $bin(\hat{y}) = \hat{y}$ and $f_{\theta^*}(\hat{y}) = 1 \iff R(\theta^*, \hat{y})$.

Therefore, θ^* is a valid solution for Equation (5.31), which implies that $z \in co\forall\exists 3SAT$.

□

Since:

- $q(z)$ can be computed in polynomial time;
- $z \in \Sigma_2 3SAT \implies q(z) \in PL\text{-}ROB$;
- $q(z) \in CCA \implies z \in co\forall\exists PL\text{-}ROB$.

We can conclude that $\Sigma_2 3SAT \leq PL\text{-}ROB$.

5.4.4 Corollaries

Corollary 3.1 (Finding a Globally Robust Model is $\boldsymbol{\Sigma_2^P}$-Complete)**.** *Let PG-ROB be the set of tuples $\langle \chi, \varepsilon, f_\theta, v \rangle$ such that:*

$$\exists \theta' \in v(f). \forall x \in X. \left(\chi(x) \implies (\forall x' \in B(x, \varepsilon). f_{\theta'}(x') = f_{\theta'}(x))\right) \tag{5.47}$$

where:

- *$\theta' \in v(f)$ can be checked in polynomial time w.r.t. the size of the tuple;*
- *X is a polynomial-bounded space;*
- *$\chi(x)$ can be verified in polynomial time w.r.t. the size of the tuple.*

Then PG-ROB is Σ_2^P-complete.

Proof. $PG\text{-}ROB \in \Sigma_2^P$ is a consequence of the Kuratowski-Tarski Theorem (using a language A defined in the same way as in Section 5.4.2). Since $PL\text{-}ROB$ is a special case of $PG\text{-}ROB$ where $\chi(x)$ for only one x, $PL\text{-}ROB \leq PG\text{-}ROB$ and thus $PG\text{-}ROB$ is Σ_2^P-hard. □

Corollary 3.2 (Training a Defense with Constrained Loss is $\boldsymbol{\Sigma_2^P}$-Complete)**.** *Let D-DEF be the set of tuples $\langle \chi, \mathcal{L}, \kappa, \varepsilon, f_\theta, v \rangle$ such that:*

$$\exists \theta' \in v(f). \mathcal{L}(f, \theta') \leq \kappa \wedge (\forall x \in X. \left(\chi(x) \implies (\forall x' \in B(x, \varepsilon). f_{\theta'}(x') = f_{\theta'}(x))\right)) \tag{5.48}$$

where:

- *$\mathcal{L}(f, \theta)$ is the loss of f with parameter vector θ;*

- *$\mathcal{L}(f, \theta') \leq \kappa$ can be verified in polynomial time w.r.t. the size of the tuple;*
- *$\theta' \in v(f)$ can be verified in polynomial time w.r.t. the size of the tuple;*
- *X is a polynomial-bounded space;*
- *$\chi(x)$ can be verified in polynomial time w.r.t. the size of the tuple.*

Then D-DEF is Σ_2^P-complete.

Proof. *D-DEF* $\in \Sigma_2^P$ is a consequence of the Kuratowski-Tarski Theorem (using a language A defined in the same way as in Section 5.4.2). Since *D-DEF* is a special case of *D-DEF* where $\kappa = 0$ and $\mathcal{L}(f, \theta) = 0$, *PG-ROB* $\leq$ *D-DEF* and thus *D-DEF* is Σ_2^P-hard. □

Corollary 3.3 (Training a Defense with Optimal Loss is $\mathbf{\Sigma_2^P}$-Hard)**.** *Let O-DEF be the following optimization problem:*

$$\arg\min_{\theta' \in v(f)} \mathcal{L}(f, \theta') \text{ s.t. } \forall x \in X. \, (\chi(x) \implies (\forall x' \in B(x, \varepsilon). f_{\theta'}(x') = f_{\theta'}(x))) \tag{5.49}$$

where:

- *$\mathcal{L}(f, \theta)$ is the loss of f with parameter vector θ;*
- *$\mathcal{L}(f, \theta') \leq k$ can be verified in polynomial time w.r.t. the size of the tuple;*
- *$\theta' \in v(f)$ can be checked in polynomial time w.r.t. the size of the tuple;*
- *X is a polynomial-bounded space;*
- *$\chi(x)$ can be verified in polynomial time w.r.t. the size of the tuple.*

Then O-DEF is Σ_2^P-hard.

Proof. Since *D-DEF* is the decision version of *O-DEF* and since *D-DEF* is Σ_2^P-hard, *O-DEF* is Σ_2^P-hard. □

Chapter 6

Experimental Evaluation

We now evaluate the effectiveness of CA as a metadefense. Specifically, we test whether $d(x, x_h)$, where x_h is an adversarial example found by a heuristic attack, is close to the true decision boundary distance (i.e. $d^*(x)$). Specifically, we test whether pools of heuristic attacks approximate $d^*(x)$ in a predictable manner. The underlying rationale is that different adversarial attacks should be able to cover for their reciprocal blind spots, providing a more reliable estimate.

One key limitation of our evaluation is that studying the estimation consistency requires sampling from a chosen distribution (in our case the MNIST [42] and CIFAR10 [40] datasets), thus violating Guideline 1. Therefore, studying the behavior of adversarial attacks on other distributions (as well as with other types of defended models) represents an important topic for future work.

6.1 Experimental Setup

6.1.1 General Information

All our code is written in Python + PyTorch [61], with the exception of the MIPVerify interface, which is written in Julia.

We randomly selected ∼2.3k samples each from the test set of two datasets, MNIST and CIFAR10, sampling uniformly across each ground truth label. The first 250 samples of the test set of each dataset were used for hyperparameter tuning and were thus not considered in our analysis.

6.1.2 Models

We used three architectures per dataset (named A, B and C), each trained in three settings, namely standard training, PGD adversarial training [47] and PGD adversarial training with ReLU loss and pruning [97] (from now on referred to as ReLU training), for a total of nine configurations per dataset. All models were trained using Adam [38] and dataset augmentation.

When performing adversarial training, following [47] we used the final adversarial example found by the Projected Gradient Descent attack, instead of the closest. To maximize uniformity, we used for each configuration the same training and pruning hyperparameters (when applicable), which we report in Table 6.1.

Since our analysis requires computing exact decision boundary distances, and since size and depth both have a strong negative impact on solver times, we used small and shallow networks with parameters between ∼2k and ∼80k. We performed a manual hyperparameter and architecture search to find a suitable compromise between accuracy and exact attack convergence. The process required approximately 4 months.

Overall, the natural accuracies for standard training are significantly below the state of the art (89.63% - 95.87% on MNIST and 47.85% - 55.81% on CIFAR10). Adversarial training also had a negative effect on natural accuracies (84.54% - 94.24% on MNIST and 45.19% - 51.35% on CIFAR10), similarly to ReLU training (83.69% - 93.57% on MNIST and 32.27% - 37.33% on CIFAR10).

We report the chosen architectures in Tables 6.2 and 6.3, while Table 6.4 outlines their accuracies and parameter counts.

Table 6.1: Training and pruning hyperparameters.

Parameter Name	**Value**	
	MNIST	**CIFAR10**
Common Hyperparameters		
Epochs	425	
Learning Rate	1e-4	
Batch Size	32	128
Adam β	(0.9, 0.999)	
Flip %	50%	
Translation Ratio	0.1	
Rotation (deg.)	15°	
Adversarial Hyperparameters (Adversarial and ReLU only)		
Attack	PGD	
Attack #Iterations	200	
Attack Learning Rate	0.1	
Adversarial Ratio	1	
ε	0.05	2/255
ReLU Hyperparameters (ReLU only)		
L1 Regularization Coeff.	2e-5	1e-5
RS Loss Coeff.	1.2e-4	1e-3
Weight Pruning Threshold	1e-3	
ReLU Pruning Threshold	90%	

Table 6.2: MNIST Architectures.

(a) MNIST A

Input
Flatten
Linear (in = 784, out = 100)
ReLU
Linear (in = 100, out = 10)
Output

(b) MNIST B

Input
Conv2D (in = 1, out = 4, 5x5 kernel, stride = 3, padding = 0)
ReLU
Flatten
Linear (in = 256, out = 10)
Output

(c) MNIST C

Input
Conv2D (in = 1, out = 8, 5x5 kernel, stride = 4, padding = 0)
ReLU
Flatten
Linear (in = 288, out = 10)
Output

Table 6.3: CIFAR10 architectures.

(a) CIFAR10 A

Input
Conv2D (in = 3, out = 8, 3x3 kernel, stride = 2, padding = 0)
ReLU
Flatten
Linear (in = 1800, out = 10)
Output

(b) CIFAR10 B

Input
Conv2D (in = 3, out = 20, 5x5 kernel, stride = 4, padding = 0)
ReLU
Flatten
Linear (in = 980, out = 10)
Output

(c) CIFAR10 C

Input
Conv2D (in = 3, out = 8, 5x5 kernel, stride = 4, padding = 0)
ReLU
Conv2D (in = 8, out = 8, 3x3 kernel, stride = 2, padding = 0)
ReLU
Flatten
Linear (in = 72, out = 10)
Output

Table 6.4: Parameter counts and accuracies of trained models.

Architecture	**#Parameters**	**Training**	**Accuracy**
MNIST A	79510	Standard	95.87%
		Adversarial	94.24%
		ReLU	93.57%
MNIST B	2674	Standard	89.63%
		Adversarial	84.54%
		ReLU	83.69%
MNIST C	3098	Standard	90.71%
		Adversarial	87.35%
		ReLU	85.67%
CIFAR10 A	18234	Standard	53.98%
		Adversarial	50.77%
		ReLU	32.85%
CIFAR10 B	11330	Standard	55.81%
		Adversarial	51.35%
		ReLU	37.33%
CIFAR10 C	1922	Standard	47.85%
		Adversarial	45.19%
		ReLU	32.27%

6.1.3 Heuristic Attacks

We first ran a pool of heuristic adversarial attacks on each example. Specifically, we used the Basic Iterative Method [41], the Brendel & Bethge attack [8], the Carlini & Wagner attack [14], Deepfool [51], the Fast Gradient Sign Method [25] and the Projected Gradient Descent attack [47], as well as simply adding uniform noise to the input. Consistently with [4] and [91], we set $d(x_1, x_2) = \|x_1 - x_2\|_\infty$, where $\| \cdot \|_\infty$ is the L^∞ norm. For each example, we considered the closest feasible adversarial example found by any attack in the pool.

For the Basic Iterative Method (BIM), the Fast Gradient Sign Method (FGSM) and the Projected Gradient Descent (PGD) attack, we used the implementations provided by the AdverTorch library [19]. For the Brendel & Bethge (B&B) attack and the Deepfool (DF) attack, we used the implementations provided by the Foolbox Native library [66]. The Carlini & Wagner and the uniform noise attacks were instead implemented by the authors. For attacks that take ε as parameter (i.e. BIM, Carlini & Wagner, Deepfool, FGSM and PGD), we first performed an initial search with a decaying value of ε, followed by a binary search..

In order to pick the attack parameters, we performed an extensive manual search and obtained the "strong" parameter set, which prioritizes finding adversarial examples at the expense of computational time. The process took approximately 3 months. We then modified the strong set in order to obtain the "balanced" parameter set. We report the parameters of both sets (as well as the parameters of the binary and ε decay searches) in Tables 6.5 and 6.6.

6.1.4 MIPVerify

We then ran the exact solver-based attack MIPVerify [83], which is able to find the closest adversarial example to a given input. We used the implementation provided by the Julia library MIPVerify.jl in conjunction with Gurobi [29].

Table 6.5: Parameters of heuristic attacks.

Attack	**Parameter Name**	**MNIST**		**CIFAR10**	
		Strong	**Balanced**	**Strong**	**Balanced**
BIM	Initial Search Factor	0.75			
	Initial Search Steps	30			
	Binary Search Steps	20			
	#Iterations	2k	200	5k	200
	Learning Rate	1e-3	1e-2	1e-5	1e-3
Brendel & Bethge	Initial Attack	Blended Noise			
	Overshoot	1.1			
	LR Decay	0.75			
	LR Decay Every n Steps	50			
	#Iterations	5k	200	5k	200
	Learning Rate	1e-3	1e-3	1e-5	1e-3
	Momentum	0.8			
Carlini & Wagner	Minimum τ	1e-5			
	Initial τ	1			
	τ Factor	0.95	0.9	0.99	0.9
	Initial Const	1e-5			
	Const Factor	2			
	Maximum Const	20			
	Reduce Const	False			
	Warm Start	True			
	Abort Early	True			
	Learning Rate	1e-2	1e-2	1e-5	1e-4
	Max Iterations	1k	100	5k	100
	τ Check Every n Steps	1			
	Const Check Every n Steps	5			
	Iter. Check Every n Steps	Disabled			

Table 6.6: Parameters of heuristic attacks (cont.).

Attack	**Parameter Name**	**MNIST**		**CIFAR10**	
		Strong	**Balanced**	**Strong**	**Balanced**
Deepfool	#Iterations	5k			
	Candidates	10			
	Overshoot	1e-5			
	Loss	Logits			
FGSM	Initial Search Factor	0.75			
	Initial Search Steps	30			
	Binary Search Steps	20			
PGD	Initial Search Factor	0.75			
	Initial Search Steps	30			
	Binary Search Steps	20			
	#Iterations	5k	200	5k	200
	Learning Rate	1e-4	1e-3	1e-4	1e-3
	Random Initialization	True			
Uniform Noise	Initial Search Factor	0.75			
	Initial Search Steps	30			
	Binary Search Steps	20			
	Runs	8k	200	8k	200

Since MIPVerify can be sped up by providing a distance upper bound, we used the same pool of adversarial attack utilized throughout the paper. For CIFAR10 we used the strong parameter set, while for MNIST we used the strong parameter set with some differences (reported in Table 6.7). Since numerical issues might cause the distance upper bound computed by the heuristic attacks to be slightly different from the one computed by MIPVerify, we ran a series of *exploratory runs*, each with a different correction factor (1.05, 1.25, 1.5, 2), and picked the first factor that caused MIPVerify to find a feasible (but not necessarily optimal) solution. If the solution was not optimal, we then performed a *main run* with a higher computational budget. We provide the parameters of MIPVerify in Table 6.8. We also report in Table 6.9 the percentage of tight bounds for each combination.

We removed the examples for which MIPVerify crashed in at least one setting, obtaining 2241 MNIST examples and 2269 CIFAR10 examples. We also excluded from our analysis all adversarial examples for which MIPVerify did not find optimal bounds (atol $= 1e - 5$, rtol $= 1e - 10$). These examples represent on average 4.38% of MNIST examples and 1.56% of CIFAR10 examples.

Table 6.7: Parameter set used to initialize MIPVerify for MNIST. All other parameters are identical to the strong MNIST attack parameter set.

Attack Name	**Parameter Name**	**Value**
BIM	#Iterations	5k
	Learning Rate	1e-5
Brendel & Bethge	Learning Rate	1e-3
Carlini & Wagner	Tau Factor	0.99
	Learning Rate	1e-3
	#Iterations	5k

Table 6.8: Parameters of MIPVerify.

Parameter Name	**Value**	
	Exploration	**Main**
Absolute Tolerance	1e-5	
Relative Tolerance	1e-10	
Threads	1	
Timeout (s)	120	7200
Tightening Absolute Tolerance	1e-4	
Tightening Relative Tolerance	1e-10	
Tightening Timeout (s)	20	240
Tightening Threads	1	

6.1.5 Hardware

MIPVerify and the strong attack pool were run on the CINECA Galileo100 HPC cluster. Each node of the cluster has 384 GB of RAM and features two Intel CascadeLake 8260 CPUs, each with 24 cores and a clock frequency of 2.4GHz. The entire process (including test runs) required ∼45k core-hours.

The balanced attack pool was run on a single machine with an AMD Ryzen 5 1600X six-core 3.6 GHz processor, 16 GB of RAM and an NVIDIA GTX 1060 6 GB GPU. The process took approximately 8 hours.

Table 6.9: MIPVerify bound tightness statistics.

Architecture	**Training**	**% Tight**
MNIST A	Standard	99.46%
	Adversarial	96.61%
	ReLU	96.52%
MNIST B	Standard	99.73%
	Adversarial	91.70%
	ReLU	98.66%
MNIST C	Standard	91.34%
	Adversarial	89.29%
	ReLU	97.23%
CIFAR10 A	Standard	97.40%
	Adversarial	96.34%
	ReLU	100.00%
CIFAR10 B	Standard	98.55%
	Adversarial	97.31%
	ReLU	100.00%
CIFAR10 C	Standard	99.34%
	Adversarial	97.05%
	ReLU	100.00%

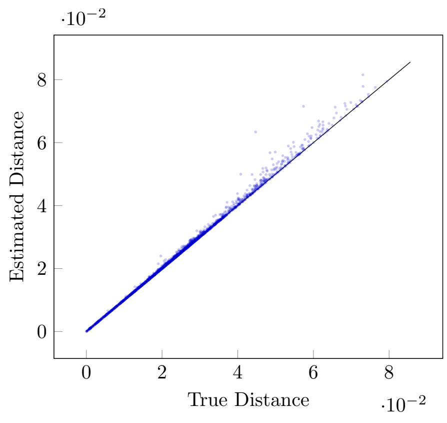
6.2 Approximation Consistency

Across all settings, the mean distance found by the strong attack pool is 4.09±2.02% higher for MNIST and 2.21±1.16% higher for CIFAR10 than the one found by MIPVerify. For 79.81±15.70% of the MNIST instances and 98.40±1.63% of the CIFAR10 ones, the absolute difference is less than 1/255, which is the minimum distance in 8-bit image formats. The balanced attack pool performs similarly, finding distances that are on average 4.65±2.16% higher for MNIST and 2.04±1.13% higher for CIFAR10. The difference is below 1/255 for 77.78±16.08% of MNIST examples and 98.74±1.13% of CIFAR10 examples. We plot the distances found by the attack pools in relation to the true decision boundary distance in Figures 6.1 to 6.6.

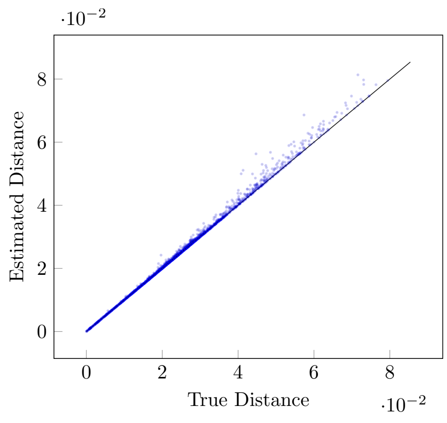
For all datasets, architectures and training techniques there appears to be a strong, linear, correlation between the distance of the output of the heuristic attacks and the true decision boundary distance. This observation suggests that even a simple linear model for the buffer function b may lead to good results. Therefore, we calibrate a buffer using least-squares fitting on each combination. Tables 6.10 to 6.13 detail the performance of both attack sets on every combination.

For the strong parameter set, we find that the average R^2 across all settings is 0.992±0.004 for MNIST and 0.997±0.003 for CIFAR10. The balanced parameter set performs similarly, achieving an R^2 of 0.990±0.006 for MNIST and 0.998±0.002 for CIFAR10. From these results, we conjecture that increasing the computational budget of heuristic attacks does not necessarily improve predictability, although further tests would be needed to confirm such a claim.

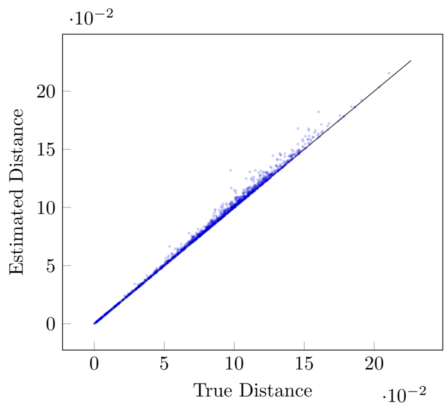
Additionally, both attack pools perform more consistently on standard training compared to adversarial and ReLU training, as shown in Tables 6.14 to 6.17. This suggests that adversarial training and ReLU might have a negative effect on the ability of an attack to correctly evaluate the robustness of a model.



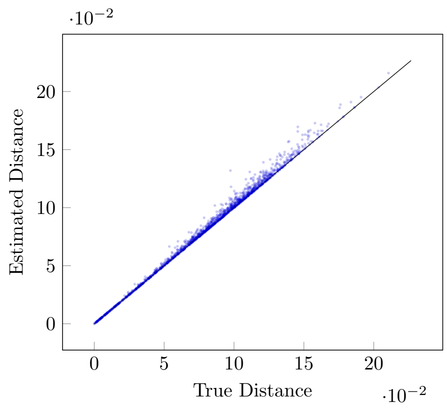
(a) MNIST A Standard Strong



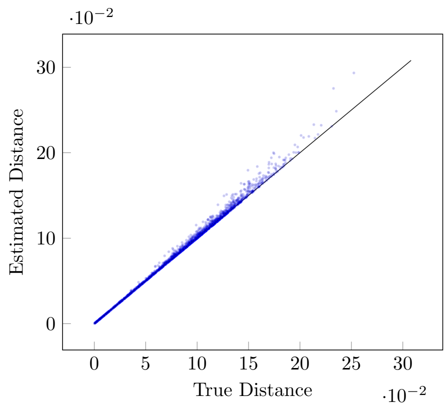
(b) MNIST A Standard Balanced



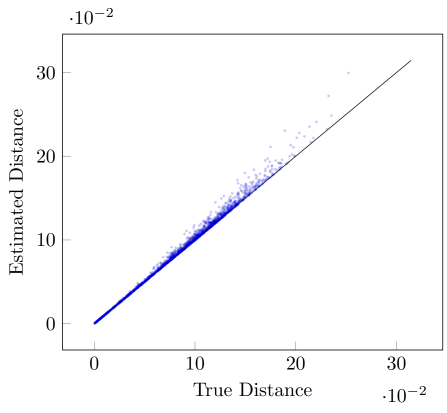
(c) MNIST A Adversarial Strong



(d) MNIST A Adversarial Balanced

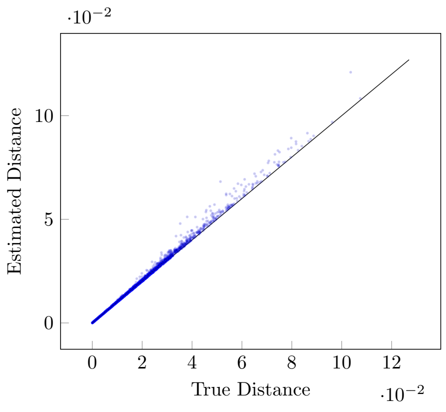


(e) MNIST A ReLU Strong

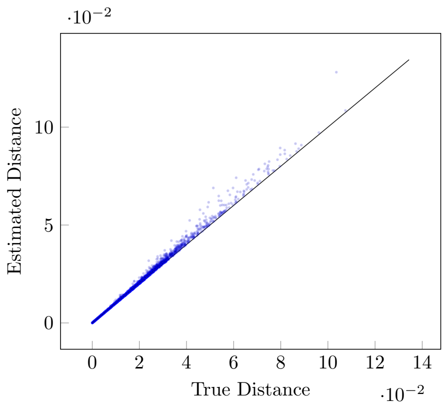


(f) MNIST A ReLU Balanced

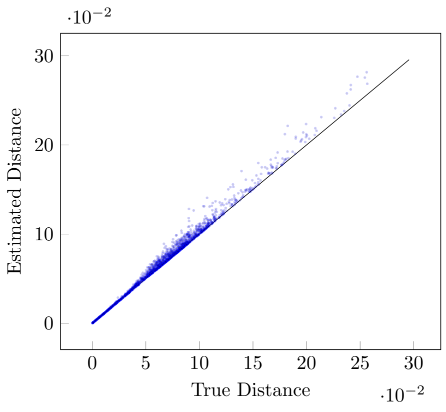
Figure 6.1: Decision boundary distances found by the attack pools compared to those found by MIPVerify on MNIST A. The black line represents the theoretical optimum.



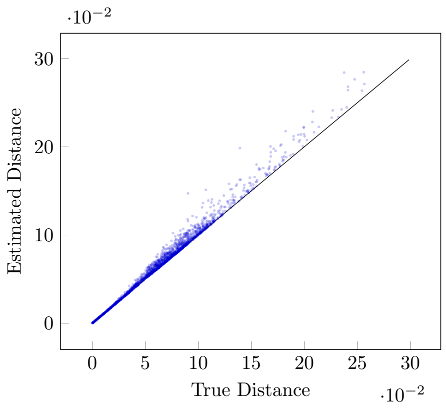
(a) MNIST B Standard Strong



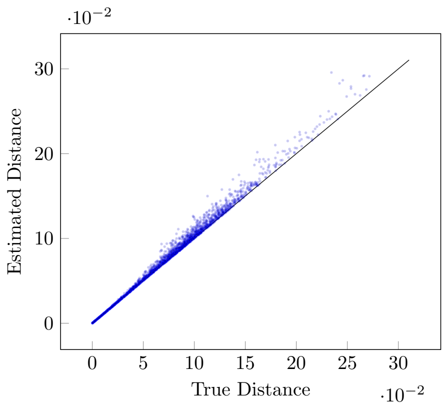
(b) MNIST B Standard Balanced



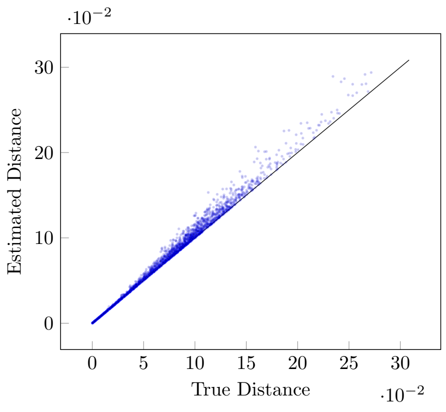
(c) MNIST B Adversarial Strong



(d) MNIST B Adversarial Balanced

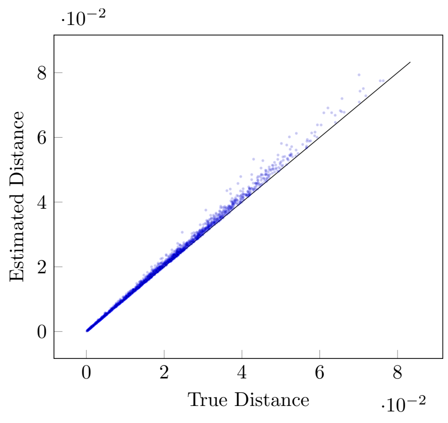


(e) MNIST B ReLU Strong

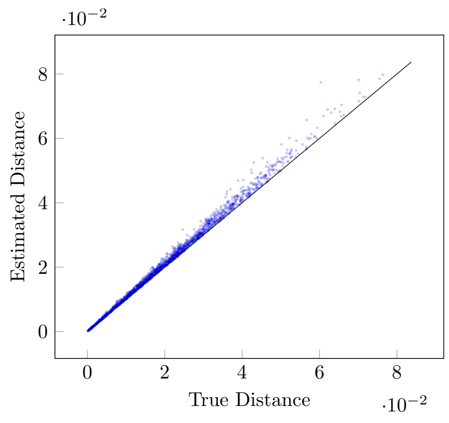


(f) MNIST B ReLU Balanced

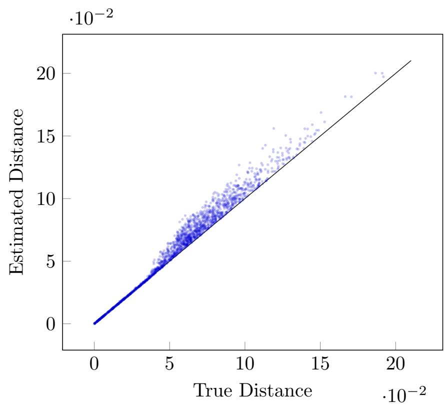
Figure 6.2: Decision boundary distances found by the attack pools compared to those found by MIPVerify on MNIST B. The black line represents the theoretical optimum.



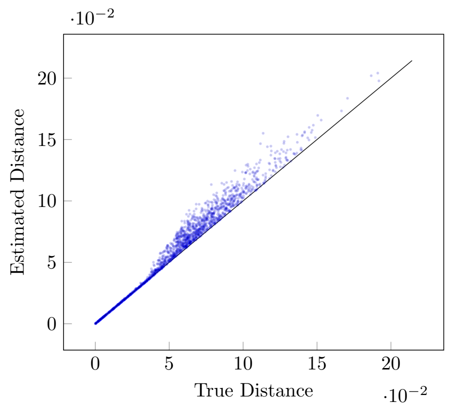
(a) MNIST C Standard Strong



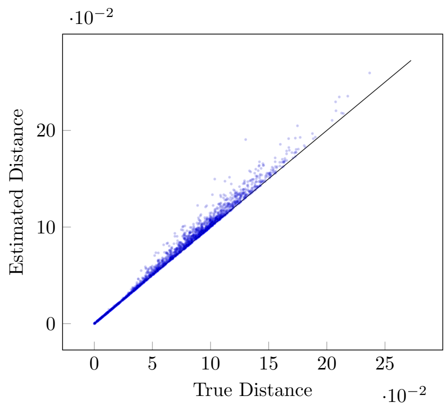
(b) MNIST C Standard Balanced



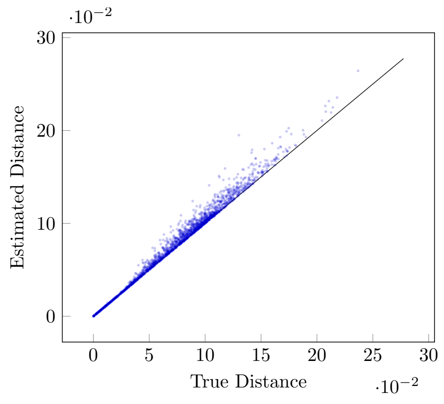
(c) MNIST C Adversarial Strong



(d) MNIST C Adversarial Balanced

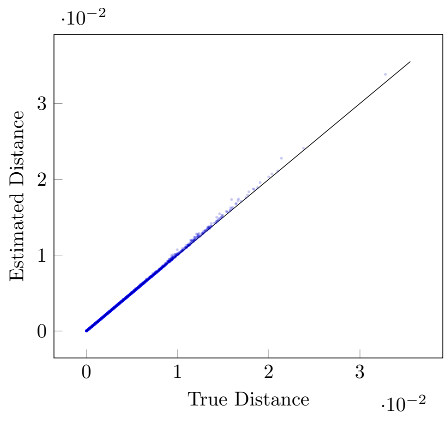


(e) MNIST C ReLU Strong

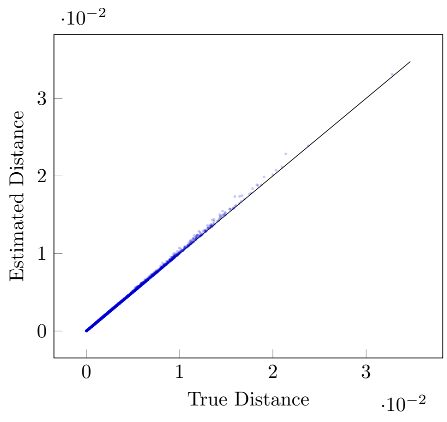


(f) MNIST C ReLU Balanced

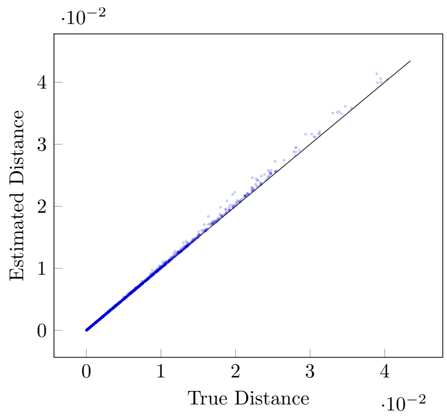
Figure 6.3: Decision boundary distances found by the attack pools compared to those found by MIPVerify on MNIST C. The black line represents the theoretical optimum.



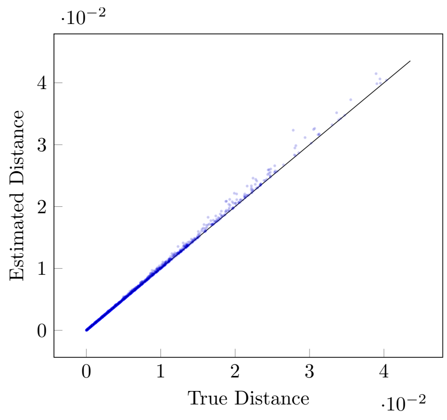
(a) CIFAR10 A Standard Strong



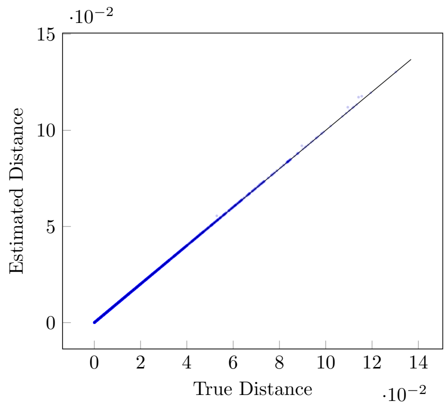
(b) CIFAR10 A Standard Balanced



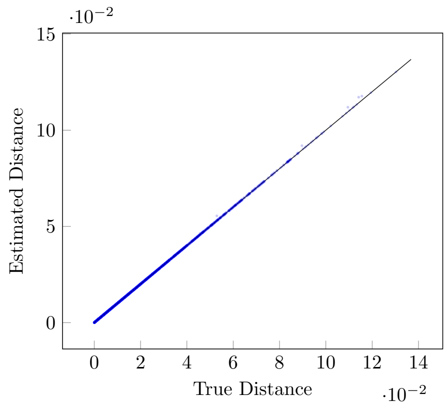
(c) CIFAR10 A Adversarial Strong



(d) CIFAR10 A Adversarial Balanced

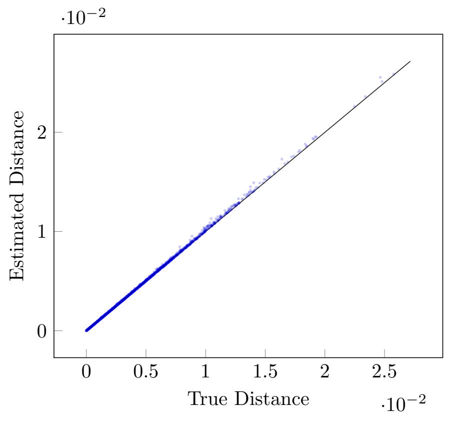


(e) CIFAR10 A ReLU Strong

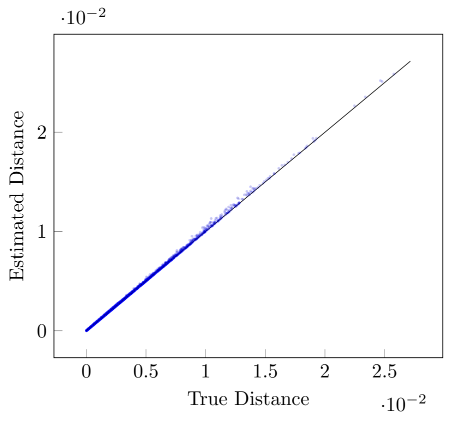


(f) CIFAR10 A ReLU Balanced

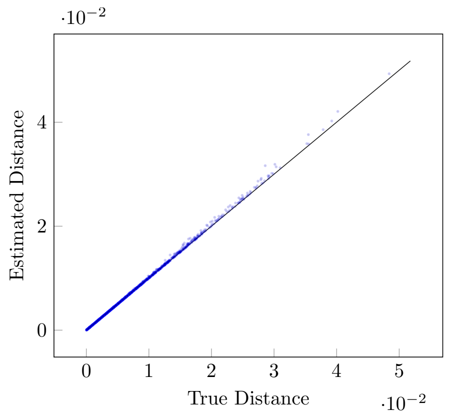
Figure 6.4: Decision boundary distances found by the attack pools compared to those found by MIPVerify on CIFAR10 A. The black line represents the theoretical optimum.



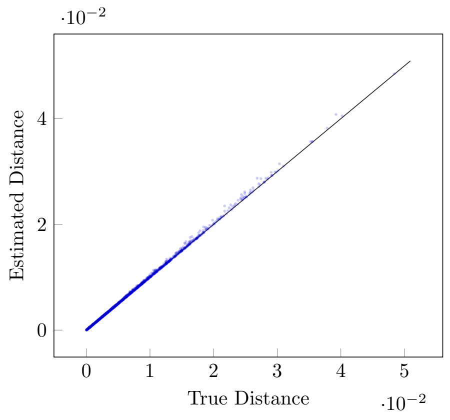
(a) CIFAR10 B Standard Strong



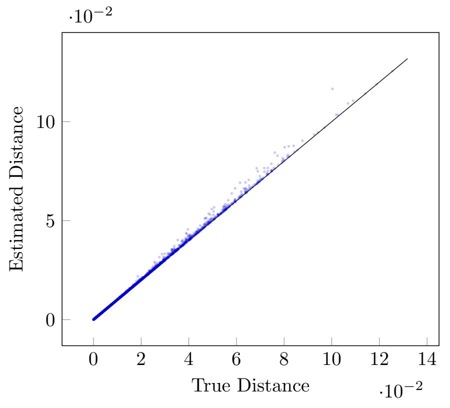
(b) CIFAR10 B Standard Balanced



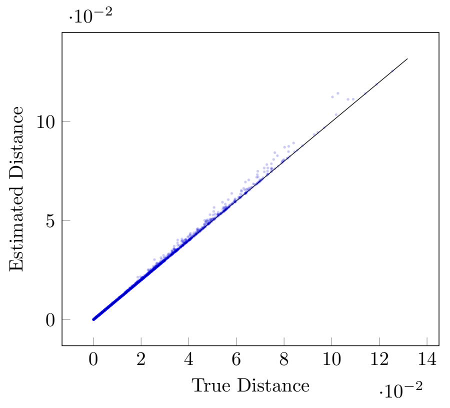
(c) CIFAR10 B Adversarial Strong



(d) CIFAR10 B Adversarial Balanced

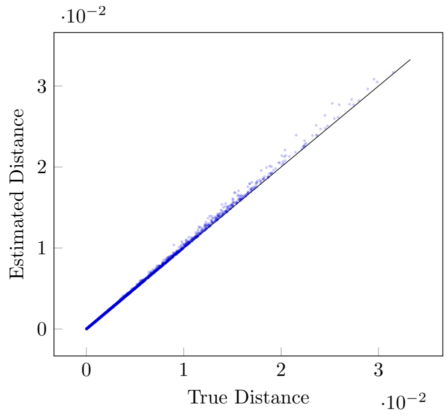


(e) CIFAR10 B ReLU Strong

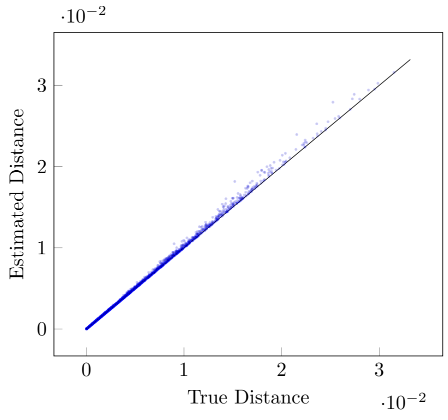


(f) CIFAR10 B ReLU Balanced

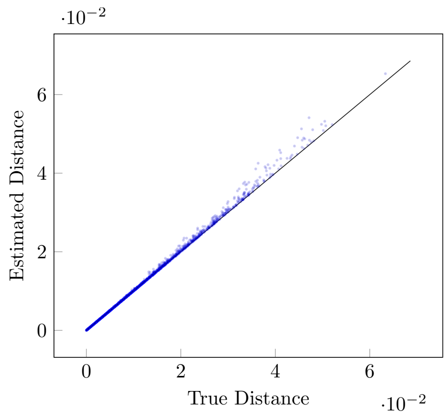
Figure 6.5: Decision boundary distances found by the attack pools compared to those found by MIPVerify on CIFAR10 B. The black line represents the theoretical optimum.



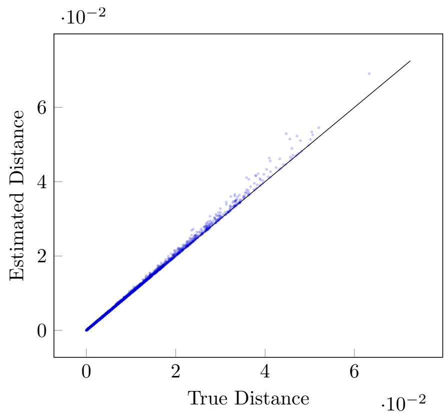
(a) CIFAR10 C Standard Strong



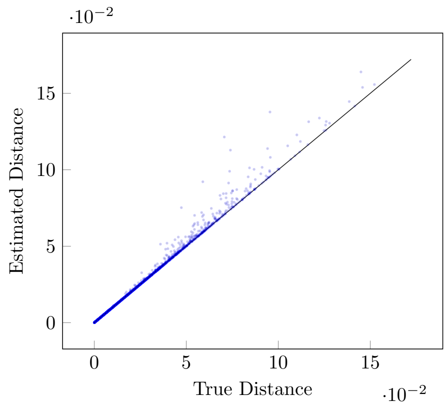
(b) CIFAR10 C Standard Balanced



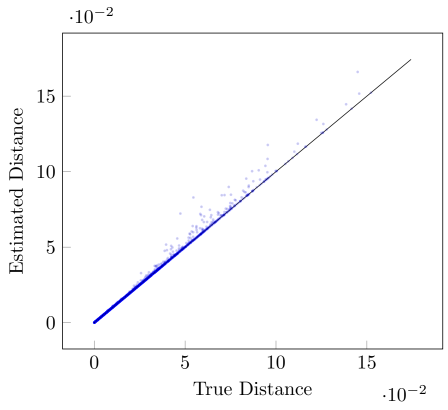
(c) CIFAR10 C Adversarial Strong



(d) CIFAR10 C Adversarial Balanced



(e) CIFAR10 C ReLU Strong



(f) CIFAR10 C ReLU Balanced

Figure 6.6: Decision boundary distances found by the attack pools compared to those found by MIPVerify on CIFAR10 C. The black line represents the theoretical optimum.

Table 6.10: Performance of the strong attack set on MNIST.

Architecture	**Training**	**Success Rate**	**Difference**	**% Below 1/255**	**R^2**
MNIST A	Standard	100.00%	1.51%	98.16%	0.996
	Adversarial	100.00%	2.48%	81.43%	0.994
	ReLU	100.00%	2.14%	84.33%	0.995
MNIST B	Standard	100.00%	3.38%	97.36%	0.995
	Adversarial	100.00%	4.34%	75.09%	0.991
	ReLU	100.00%	4.80%	68.02%	0.992
MNIST C	Standard	100.00%	4.52%	96.92%	0.996
	Adversarial	100.00%	8.76%	48.78%	0.981
	ReLU	100.00%	4.84%	68.24%	0.988

Table 6.11: Performance of the balanced attack set on MNIST.

Architecture	**Training**	**Success Rate**	**Difference**	**% Below 1/255**	**R^2**
MNIST A	Standard	100.00%	1.68%	97.94%	0.995
	Adversarial	100.00%	2.87%	77.64%	0.993
	ReLU	100.00%	2.55%	80.86%	0.993
MNIST B	Standard	100.00%	4.09%	96.55%	0.995
	Adversarial	100.00%	4.90%	72.60%	0.988
	ReLU	100.00%	5.53%	62.96%	0.989
MNIST C	Standard	100.00%	5.43%	96.04%	0.995
	Adversarial	100.00%	9.50%	48.43%	0.977
	ReLU	100.00%	5.28%	66.96%	0.986

Table 6.12: Performance of the strong attack set on CIFAR10.

Architecture	**Training**	**Success Rate**	**Difference**	**% Below 1/255**	**R^2**
CIFAR10 A	Standard	100.00%	1.62%	100.00%	0.999
	Adversarial	100.00%	4.42%	95.88%	0.995
	ReLU	100.00%	0.26%	100.00%	1.000
CIFAR10 B	Standard	100.00%	1.44%	100.00%	0.999
	Adversarial	100.00%	3.17%	97.69%	0.997
	ReLU	100.00%	1.38%	98.81%	0.999
CIFAR10 C	Standard	100.00%	2.11%	100.00%	0.999
	Adversarial	100.00%	3.10%	97.14%	0.996
	ReLU	100.00%	2.35%	96.12%	0.990

Table 6.13: Performance of the balanced attack set on CIFAR10.

Architecture	**Training**	**Success Rate**	**Difference**	**% Below 1/255**	**R^2**
CIFAR10 A	Standard	100.00%	1.71%	100.00%	0.999
	Adversarial	100.00%	4.18%	96.57%	0.995
	ReLU	100.00%	0.18%	100.00%	1.000
CIFAR10 B	Standard	100.00%	1.53%	100.00%	0.999
	Adversarial	100.00%	2.92%	98.46%	0.996
	ReLU	100.00%	1.19%	98.94%	0.999
CIFAR10 C	Standard	100.00%	2.06%	100.00%	0.999
	Adversarial	100.00%	3.12%	97.28%	0.996
	ReLU	100.00%	1.45%	97.44%	0.995

Table 6.14: MNIST strong pool performance by training technique.

Training	**Difference**	**$< 1/255$**	**$\mathbf{R^2}$**
Standard	3.14±1.24%	97.48±0.51%	0.996±0.000
Adversarial	5.19±2.63%	68.43±14.14%	0.989±0.006
ReLU	3.93±1.26%	73.53±7.63%	0.991±0.003

Table 6.15: MNIST balanced pool performance by training technique.

Training	**Difference**	**$< 1/255$**	**$\mathbf{R^2}$**
Standard	3.73±1.55%	96.84±0.80%	0.995±0.000
Adversarial	5.76±2.77%	66.22±12.75%	0.986±0.007
ReLU	4.45±1.35%	70.26±7.67%	0.989±0.003

Table 6.16: CIFAR10 strong pool performance by training technique.

Training	**Difference**	**$< 1/255$**	**$\mathbf{R^2}$**
Standard	1.72±0.29%	100.00±0.00%	0.999±0.000
Adversarial	3.56±0.61%	96.90±0.76%	0.996±0.001
ReLU	1.33±0.85%	98.31±1.62%	0.996±0.005

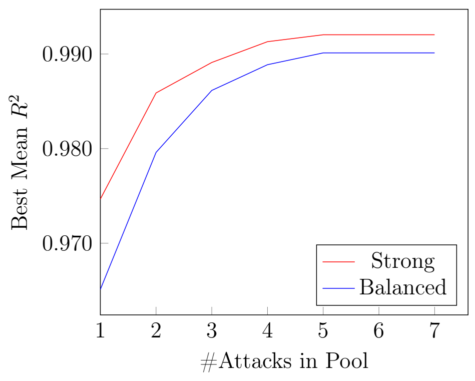
Table 6.17: CIFAR10 balanced pool performance by training technique.

Training	**Difference**	**$< 1/255$**	**$\mathbf{R^2}$**
Standard	1.76±0.22%	100.00±0.00%	0.999±0.000
Adversarial	3.41±0.55%	97.43±0.78%	0.996±0.000
ReLU	0.94±0.55%	98.80±1.05%	0.998±0.002

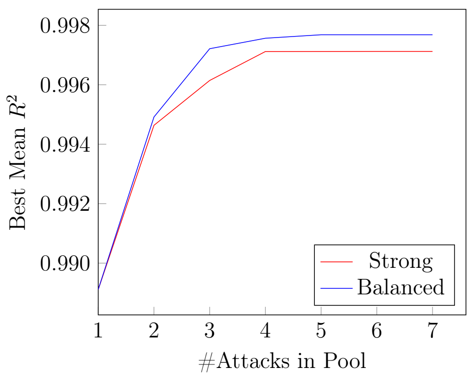
6.3 Attack Pool Ablation Study

Due to the nontrivial computational requirements of running several attacks on the same input, we now study whether it is possible to drop some attacks from the pool without compromising its predictability. Specifically, we consider all possible pools of size n (with a success rate of 100%) and pick the one with the highest average R^2 value over all architectures and training techniques. As show in Figure 6.7, adding attacks *does* increase predictability, although with diminishing returns. For example, the pool composed of the Basic Iterative Method, the Brendel & Bethge Attack and the Carlini & Wagner attack achieves on its own a R^2 value of 0.988 ± 0.004 for MNIST+strong, 0.986 ± 0.005 for MNIST+balanced, 0.935 ± 0.048 for CIFAR10+strong and 0.993 ± 0.003 for CIFAR10+balanced. Moreover, dropping both the Fast Gradient Sign Method and uniform noise leads to negligible ($\ll 0.001$) absolute variations in the mean R^2. These findings suggest that, as far as consistency is concerned, the choice of attacks represents a more important factor than the number of attacks in a pool.

We outline the best attack pools by size in Tables 6.18 to 6.21. Additionally, we report the performance of pools composed of individual attacks in Tables 6.22 to 6.25. Finally, we detail the performance of dropping a specific attack in Tables 6.26 to 6.29.



(a) MNIST



(b) CIFAR10

Figure 6.7: Best mean R^2 value in relation to the number of attacks in the pool.

Table 6.18: Best pools of a given size by success rate and R^2 for MNIST strong.

n	**Attacks**	**Success Rate**	**Difference**	**$< 1/255$**	**R^2**
1	PGD	100.00±0.00%	10.98±4.41%	51.83±27.78%	0.975±0.010
2	C&W, PGD	100.00±0.00%	7.99±3.31%	60.68±25.43%	0.986±0.005
3	B&B, C&W, PGD	100.00±0.00%	4.71±1.97%	77.97±15.52%	0.989±0.004
4	B&B, C&W, DF, PGD	100.00±0.00%	4.36±2.03%	79.02±15.62%	0.991±0.005
5	No FGSM, Uniform	100.00±0.00%	4.09±2.02%	79.81±15.70%	0.992±0.005
6	No Uniform	100.00±0.00%	4.09±2.02%	79.81±15.70%	0.992±0.005
7	All	100.00±0.00%	4.09±2.02%	79.81±15.70%	0.992±0.005

Table 6.19: Best pools of a given size by success rate and R^2 for MNIST balanced.

n	Attacks	Success Rate	Difference	$< 1/255$	$\mathbf{R^2}$
1	BIM	100.00±0.00%	11.72±4.18%	50.92±26.43%	0.965±0.010
2	BIM, B&B	100.00±0.00%	6.11±2.28%	73.23±15.90%	0.980±0.007
3	BIM, B&B, C&W	100.00±0.00%	5.29±2.06%	75.72±16.10%	0.986±0.005
4	BIM, B&B, C&W, DF	100.00±0.00%	4.85±2.10%	77.33±15.85%	0.989±0.005
5	No FGSM, Uniform	100.00±0.00%	4.65±2.16%	77.78±16.08%	0.990±0.006
6	No Uniform	100.00±0.00%	4.65±2.16%	77.78±16.08%	0.990±0.006
7	All	100.00±0.00%	4.65±2.16%	77.78±16.08%	0.990±0.006

Table 6.20: Best pools of a given size by success rate and R^2 for CIFAR10 strong.

n	Attacks	Success Rate	Difference	$< 1/255$	$\mathbf{R^2}$
1	DF	100.00±0.00%	6.11±3.49%	95.06±4.81%	0.989±0.011
2	DF, PGD	100.00±0.00%	4.71±2.37%	96.32±3.56%	0.995±0.007
3	C&W, DF, PGD	100.00±0.00%	2.54±1.30%	98.17±2.00%	0.996±0.006
4	B&B, C&W, DF, PGD	100.00±0.00%	2.21±1.16%	98.40±1.63%	0.997±0.003
5	No FGSM, Uniform	100.00±0.00%	2.21±1.16%	98.40±1.63%	0.997±0.003
6	No Uniform	100.00±0.00%	2.21±1.16%	98.40±1.63%	0.997±0.003
7	All	100.00±0.00%	2.21±1.16%	98.40±1.63%	0.997±0.003

Table 6.21: Best pools of a given size by success rate and R^2 for CIFAR10 balanced.

n	**Attacks**	**Success Rate**	**Difference**	**< 1/255**	**R^2**
1	DF	100.00±0.00%	6.11±3.49%	95.06±4.81%	0.989±0.011
2	B&B, DF	100.00±0.00%	2.52±1.51%	98.23±1.81%	0.995±0.004
3	BIM, B&B, DF	100.00±0.00%	2.21±1.25%	98.53±1.52%	0.997±0.002
4	BIM, B&B, C&W, DF	100.00±0.00%	2.06±1.16%	98.73±1.32%	0.998±0.002
5	No FGSM, Uniform	100.00±0.00%	2.04±1.13%	98.74±1.29%	0.998±0.002
6	No FGSM	100.00±0.00%	2.04±1.13%	98.74±1.29%	0.998±0.002
7	All	100.00±0.00%	2.04±1.13%	98.74±1.29%	0.998±0.002

Table 6.22: Performance of individual attacks for MNIST strong.

Attack	**Success Rate**	**Difference**	**< 1/255**	**R^2**
BIM	100.00±0.00%	10.90±4.42%	53.57±28.07%	0.966±0.012
B&B	99.99±0.01%	18.50±7.09%	58.78±9.91%	0.812±0.044
C&W	100.00±0.00%	17.52±2.74%	48.02±21.28%	0.910±0.024
Deepfool	100.00±0.00%	21.59±7.73%	44.15±20.02%	0.923±0.027
FGSM	99.72±0.51%	44.43±15.76%	28.20±17.30%	0.761±0.132
PGD	100.00±0.00%	10.98±4.41%	51.83±27.78%	0.975±0.010
Uniform	99.52±0.91%	414.47±140.54%	0.82±0.55%	0.623±0.138

Table 6.23: Performance of individual attacks for MNIST balanced.

Attack	**Success Rate**	**Difference**	**$<$ 1/255**	**R^2**
BIM	100.00±0.00%	11.72±4.18%	50.92±26.43%	0.965±0.010
B&B	99.99±0.03%	18.65±7.29%	58.43±9.61%	0.812±0.039
C&W	100.00±0.00%	22.55±3.83%	38.95±22.49%	0.904±0.025
Deepfool	100.00±0.00%	21.59±7.73%	44.15±20.02%	0.923±0.027
FGSM	99.72±0.51%	44.43±15.76%	28.20±17.30%	0.761±0.132
PGD	100.00±0.00%	16.23±6.59%	48.08±28.88%	0.905±0.070
Uniform	98.66±1.90%	521.61±181.40%	0.57±0.38%	0.484±0.122

Table 6.24: Performance of individual attacks for CIFAR10 strong.

Attack	**Success Rate**	**Difference**	**$<$ 1/255**	**R^2**
BIM	91.96±7.40%	19.97±5.95%	80.32±12.97%	0.934±0.041
B&B	100.00±0.00%	508.66±196.37%	42.74±7.85%	0.174±0.074
C&W	99.98±0.02%	10.67±3.64%	90.09±5.51%	0.926±0.030
Deepfool	100.00±0.00%	6.11±3.49%	95.06±4.81%	0.989±0.011
FGSM	100.00±0.00%	31.80±11.12%	69.20±17.72%	0.847±0.123
PGD	100.00±0.00%	19.36±5.99%	77.23±15.89%	0.952±0.027
Uniform	99.99±0.02%	1206.79±277.68%	2.48±0.88%	0.910±0.044

Table 6.25: Performance of individual attacks for CIFAR10 balanced.

Attack	**Success Rate**	**Difference**	**$< 1/255$**	**$\mathbf{R^2}$**
BIM	100.00±0.00%	19.23±5.92%	77.33±15.89%	0.954±0.025
B&B	100.00±0.00%	50.64±52.17%	81.20±10.68%	0.615±0.349
C&W	99.89±0.09%	17.44±4.01%	84.82±8.51%	0.923±0.026
Deepfool	100.00±0.00%	6.11±3.49%	95.06±4.81%	0.989±0.011
FGSM	100.00±0.00%	31.80±11.12%	69.20±17.72%	0.847±0.123
PGD	100.00±0.00%	20.18±6.56%	76.97±16.07%	0.947±0.031
Uniform	99.85±0.26%	1617.74±390.50%	1.80±0.67%	0.853±0.068

Table 6.26: Performance of pools without a specific attack for MNIST strong.

Dropped Attack	**Success Rate**	**Difference**	**$< 1/255$**	**$\mathbf{R^2}$**
None	100.00±0.00%	4.09±2.02%	79.81±15.70%	0.992±0.005
BIM	100.00±0.00%	4.35±2.03%	79.02±15.62%	0.991±0.005
B&B	100.00±0.00%	6.76±3.31%	64.46±25.01%	0.990±0.005
C&W	100.00±0.00%	4.65±2.20%	77.70±16.02%	0.989±0.006
Deepfool	100.00±0.00%	4.33±1.97%	79.04±15.75%	0.990±0.004
FGSM	100.00±0.00%	4.09±2.02%	79.81±15.70%	0.992±0.005
PGD	100.00±0.00%	4.26±1.99%	79.36±15.59%	0.991±0.004
Uniform	100.00±0.00%	4.09±2.02%	79.81±15.70%	0.992±0.005

Table 6.27: Performance of pools without a specific attack for MNIST balanced.

Dropped Attack	**Success Rate**	**Difference**	**$< 1/255$**	**$\mathbf{R^2}$**
None	100.00±0.00%	4.65±2.16%	77.78±16.08%	0.990±0.006
BIM	100.00±0.00%	5.13±2.27%	76.14±15.98%	0.988±0.007
B&B	100.00±0.00%	7.93±3.69%	60.79±25.99%	0.987±0.006
C&W	100.00±0.00%	4.93±2.22%	77.05±15.96%	0.988±0.006
Deepfool	100.00±0.00%	5.03±2.14%	76.34±16.36%	0.988±0.005
FGSM	100.00±0.00%	4.65±2.16%	77.78±16.08%	0.990±0.006
PGD	100.00±0.00%	4.85±2.10%	77.33±15.85%	0.989±0.005
Uniform	100.00±0.00%	4.65±2.16%	77.78±16.08%	0.990±0.006

Table 6.28: Performance of pools without a specific attack for CIFAR10 strong.

Dropped Attack	**Success Rate**	**Difference**	**$< 1/255$**	**$\mathbf{R^2}$**
None	100.00±0.00%	2.21±1.16%	98.40±1.63%	0.997±0.003
BIM	100.00±0.00%	2.21±1.16%	98.40±1.63%	0.997±0.003
B&B	100.00±0.00%	2.54±1.30%	98.17±2.00%	0.996±0.006
C&W	100.00±0.00%	3.83±2.06%	96.84±3.12%	0.996±0.004
Deepfool	100.00±0.00%	4.02±1.19%	95.65±3.10%	0.992±0.005
FGSM	100.00±0.00%	2.21±1.16%	98.40±1.63%	0.997±0.003
PGD	100.00±0.00%	2.50±1.48%	98.11±1.93%	0.995±0.005
Uniform	100.00±0.00%	2.21±1.16%	98.40±1.63%	0.997±0.003

Table 6.29: Performance of pools without a specific attack for CIFAR10 balanced.

Dropped Attack	**Success Rate**	**Difference**	**$< 1/255$**	**R^2**
None	100.00±0.00%	2.04±1.13%	98.74±1.29%	0.998±0.002
BIM	100.00±0.00%	2.07±1.15%	98.72±1.31%	0.998±0.002
B&B	100.00±0.00%	4.08±1.95%	97.26±2.70%	0.996±0.006
C&W	100.00±0.00%	2.18±1.22%	98.54±1.50%	0.997±0.002
Deepfool	100.00±0.00%	4.00±0.99%	95.89±3.13%	0.993±0.003
FGSM	100.00±0.00%	2.04±1.13%	98.74±1.29%	0.998±0.002
PGD	100.00±0.00%	2.06±1.16%	98.73±1.32%	0.998±0.002
Uniform	100.00±0.00%	2.04±1.13%	98.74±1.29%	0.998±0.002

6.4 Quantile-Based Calibration

We test quantile-based calibration using 5-fold cross-validation on each configuration. Specifically, we calibrate the model using 1%, 50% and 99% as quantiles. Tables 6.30 to 6.33 provide a comparison between the expected quantile and the average true quantile of each configuration on the validation folds.

Additionally, we build quantile-calibrated detectors of robust inputs (i.e. inputs with a decision boundary distance that is greater than ε) and plot in Figures 6.8 to 6.13 their mean F_1 score in relation to the choice of ε. For most choices of ε, 50th-percentile (i.e. median) calibration achieves the best results, suggesting that median calibration might be more robust (possibly due to its lower sensitivity to outliers). Moreover, the non-monotonicity of the F_1 scores indicates that picking higher values of ε (thus using a more restrictive definition of "robust") does not necessarily improve the quality of the detector.

Table 6.30: Expected vs true quantile for MNIST strong with 5-fold cross-validation.

Architecture	**Training**	**Exp. Quantile**	**True Quantile**
A	Standard	1.00%	0.99±1.02%
		50.00%	49.93±2.35%
		99.00%	95.60±3.77%
	Adversarial	1.00%	1.11±0.53%
		50.00%	50.25±1.58%
		99.00%	89.84±6.42%
	ReLU	1.00%	1.11±0.45%
		50.00%	50.02±1.72%
		99.00%	91.95±5.64%
B	Standard	1.00%	1.07±0.48%
		50.00%	49.80±0.76%
		99.00%	97.76±0.71%
	Adversarial	1.00%	1.22±1.01%
		50.00%	49.88±4.63%
		99.00%	98.10±0.36%
	ReLU	1.00%	1.04±0.77%
		50.00%	49.98±3.17%
		99.00%	97.69±1.41%
C	Standard	1.00%	1.07±0.37%
		50.00%	50.17±1.64%
		99.00%	98.73±0.42%
	Adversarial	1.00%	1.05±0.29%
		50.00%	49.87±3.58%
		99.00%	99.00±0.47%
	ReLU	1.00%	1.06±0.67%
		50.00%	50.02±1.85%
		99.00%	93.99±3.51%

Table 6.31: Expected vs true quantile for MNIST balanced with 5-fold cross-validation.

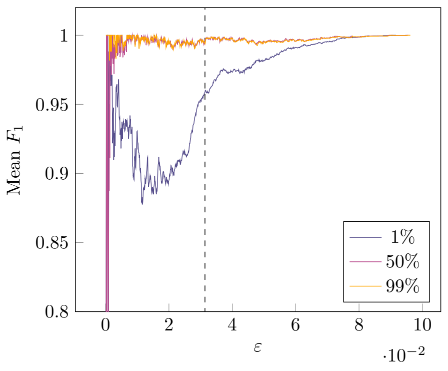
Architecture	Training	Exp. Quantile	True Quantile
A	Standard	1.00%	1.30±0.79%
		50.00%	49.98±3.10%
		99.00%	93.99±2.59%
	Adversarial	1.00%	0.97±0.40%
		50.00%	50.12±1.14%
		99.00%	90.44±1.90%
	ReLU	1.00%	1.02±0.31%
		50.00%	50.02±1.05%
		99.00%	95.10±2.82%
B	Standard	1.00%	1.03±0.36%
		50.00%	49.98±0.70%
		99.00%	98.88±0.45%
	Adversarial	1.00%	1.17±0.97%
		50.00%	50.17±4.54%
		99.00%	98.69±0.59%
	ReLU	1.00%	1.04±0.49%
		50.00%	50.34±2.49%
		99.00%	98.73±0.53%
C	Standard	1.00%	1.07±0.33%
		50.00%	49.98±0.91%
		99.00%	98.88±0.55%
	Adversarial	1.00%	1.10±0.37%
		50.00%	50.12±4.15%
		99.00%	99.00±0.35%
	ReLU	1.00%	1.06±0.67%
		50.00%	50.12±2.67%
		99.00%	98.62±0.50%

Table 6.32: Expected vs true quantile for CIFAR10 strong with 5-fold cross-validation.

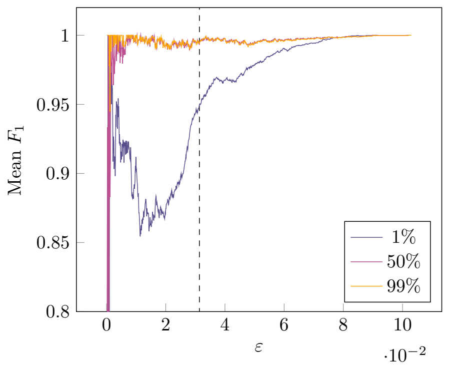
Architecture	Training	Exp. Quantile	True Quantile
A	Standard	1.00%	1.09±0.86%
		50.00%	50.09±1.84%
		99.00%	98.82±0.63%
	Adversarial	1.00%	1.05±0.23%
		50.00%	49.86±3.59%
		99.00%	98.90±0.62%
	ReLU	1.00%	0.97±0.41%
		50.00%	49.93±3.42%
		99.00%	97.66±1.35%
B	Standard	1.00%	0.98±0.18%
		50.00%	49.91±1.18%
		99.00%	98.84±0.56%
	Adversarial	1.00%	0.91±0.48%
		50.00%	50.00±3.58%
		99.00%	98.69±0.72%
	ReLU	1.00%	1.10±0.72%
		50.00%	49.98±2.21%
		99.00%	98.85±0.61%
C	Standard	1.00%	0.93±0.60%
		50.00%	50.00±1.86%
		99.00%	98.71±0.71%
	Adversarial	1.00%	1.09±0.17%
		50.00%	50.14±2.63%
		99.00%	98.27±0.81%
	ReLU	1.00%	1.01±0.62%
		50.00%	50.02±2.09%
		99.00%	96.17±2.40%

Table 6.33: Expected vs true quantile for CIFAR10 balanced with 5-fold cross-validation.

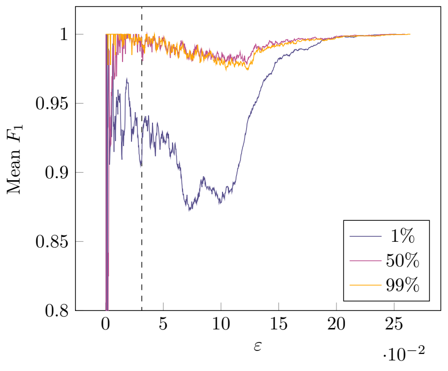
Architecture	Training	Exp. Quantile	True Quantile
A	Standard	1.00%	0.95±0.61%
		50.00%	50.32±2.38%
		99.00%	98.87±0.59%
	Adversarial	1.00%	1.05±0.23%
		50.00%	50.23±2.65%
		99.00%	98.81±0.96%
	ReLU	1.00%	4.14±5.32%
		50.00%	50.37±1.02%
		99.00%	94.62±2.87%
B	Standard	1.00%	1.07±0.46%
		50.00%	49.91±2.78%
		99.00%	98.93±0.73%
	Adversarial	1.00%	1.13±0.57%
		50.00%	50.18±2.05%
		99.00%	98.82±0.71%
	ReLU	1.00%	1.23±0.38%
		50.00%	50.11±0.38%
		99.00%	98.77±0.51%
C	Standard	1.00%	0.98±0.50%
		50.00%	50.09±2.21%
		99.00%	98.85±0.43%
	Adversarial	1.00%	1.09±0.26%
		50.00%	49.96±2.72%
		99.00%	98.86±0.32%
	ReLU	1.00%	1.01±0.36%
		50.00%	49.93±1.60%
		99.00%	97.93±0.63%



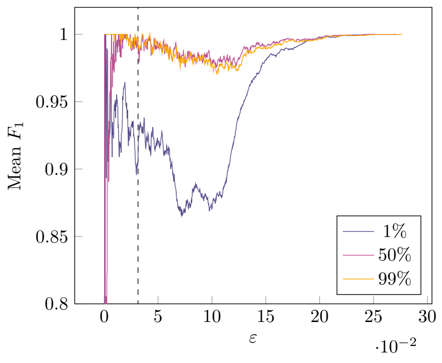
(a) MNIST A Standard Strong



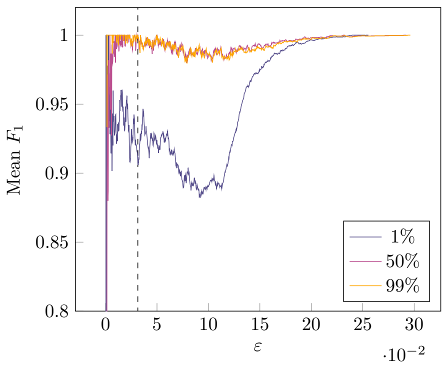
(b) MNIST A Standard Balanced



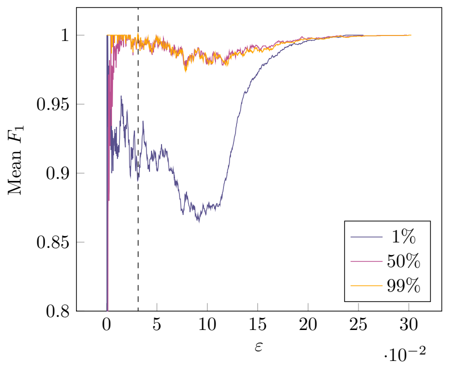
(c) MNIST A Adversarial Strong



(d) MNIST A Adversarial Balanced

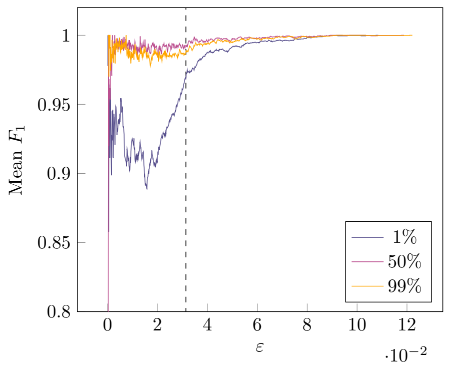


(e) MNIST A ReLU Strong

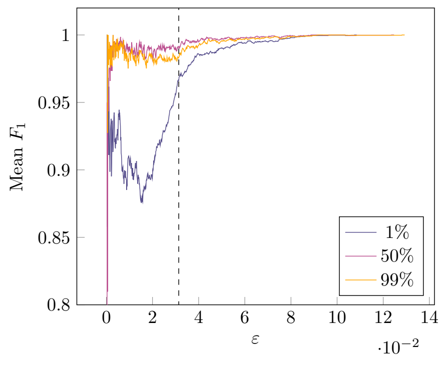


(f) MNIST A ReLU Balanced

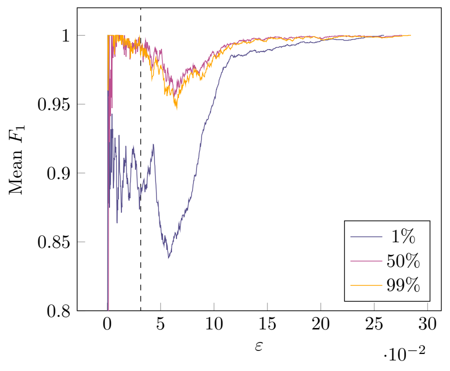
Figure 6.8: F_1 scores in relation to ε for MNIST A for each considered percentile. For ease of visualization, we set the graph cutoff at $F_1 = 0.8$. We also mark 8/255 (a common choice for ε) with a dotted line.



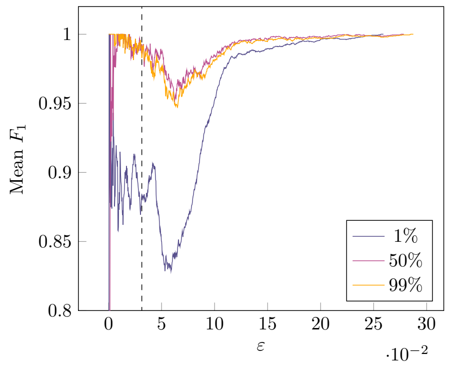
(a) MNIST B Standard Strong



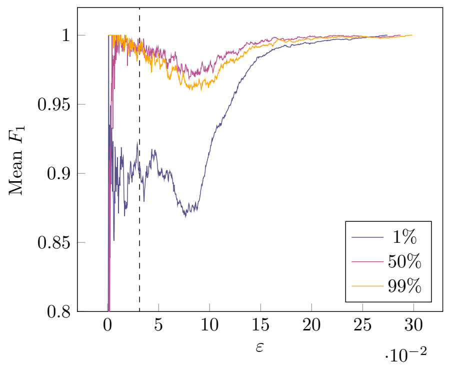
(b) MNIST B Standard Balanced



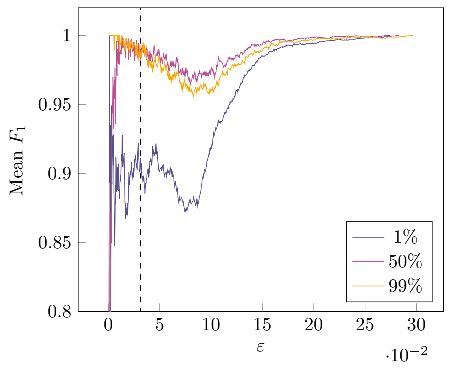
(c) MNIST B Adversarial Strong



(d) MNIST B Adversarial Balanced

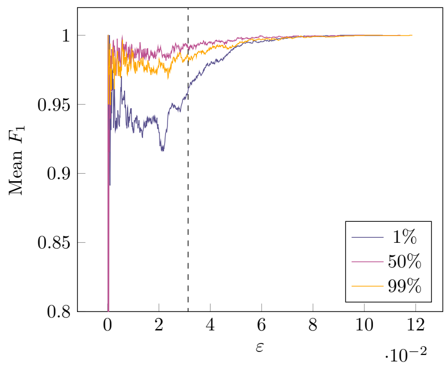


(e) MNIST B ReLU Strong

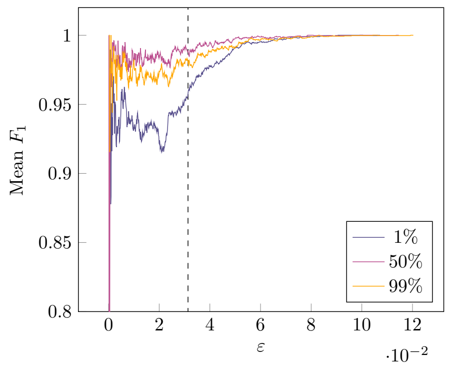


(f) MNIST B ReLU Balanced

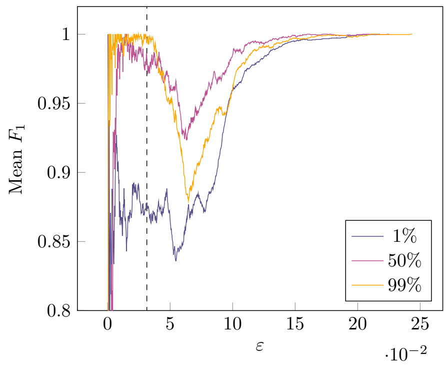
Figure 6.9: F_1 scores in relation to ε for MNIST B for each considered percentile. For ease of visualization, we set the graph cutoff at $F_1 = 0.8$. We also mark 8/255 (a common choice for ε) with a dotted line.



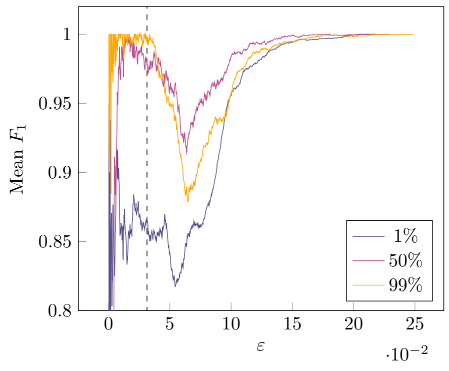
(a) MNIST C Standard Strong



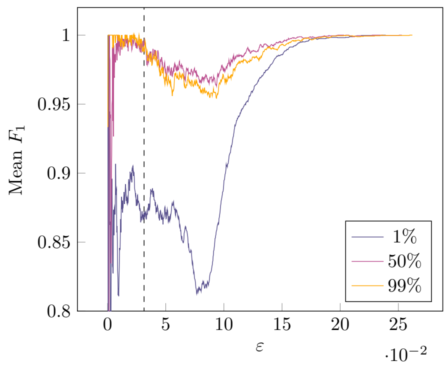
(b) MNIST C Standard Balanced



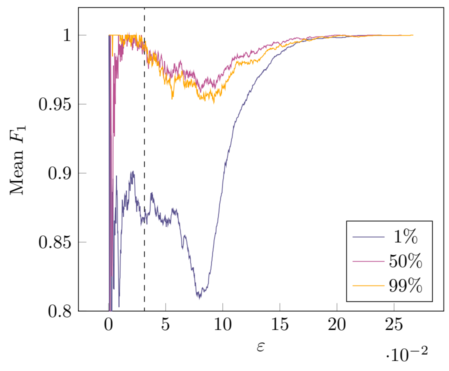
(c) MNIST C Adversarial Strong



(d) MNIST C Adversarial Balanced

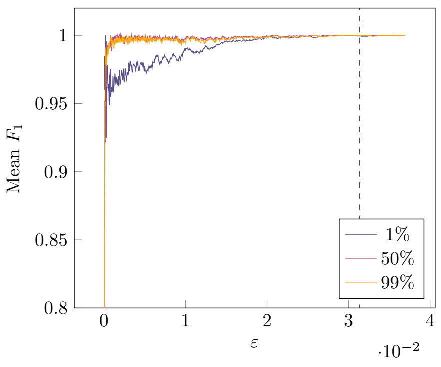


(e) MNIST C ReLU Strong

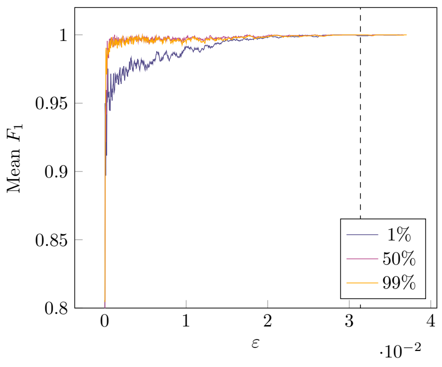


(f) MNIST C ReLU Balanced

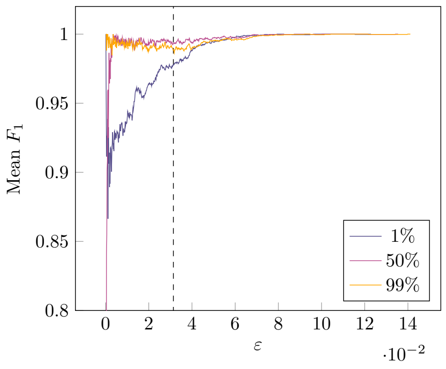
Figure 6.10: F_1 scores in relation to ε for MNIST C for each considered percentile. For ease of visualization, we set the graph cutoff at $F_1 = 0.8$. We also mark 8/255 (a common choice for ε) with a dotted line.



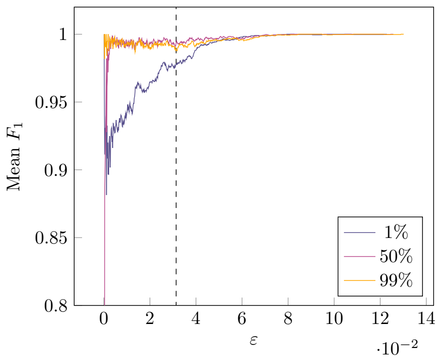
(a) CIFAR10 A Standard Strong



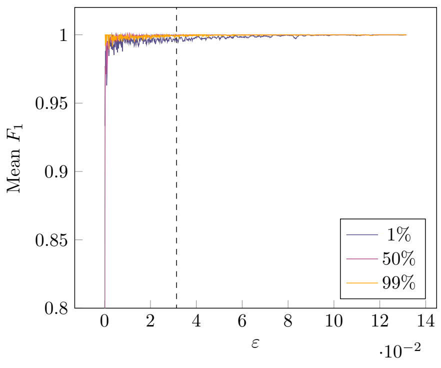
(b) CIFAR10 A Standard Balanced



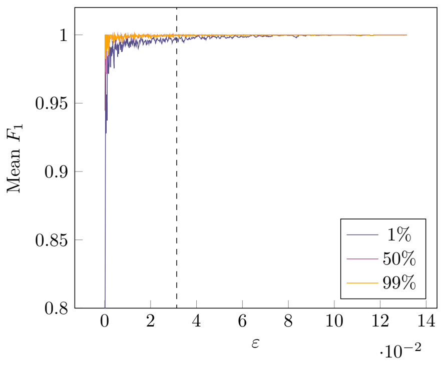
(c) CIFAR10 A Adversarial Strong



(d) CIFAR10 A Adversarial Balanced

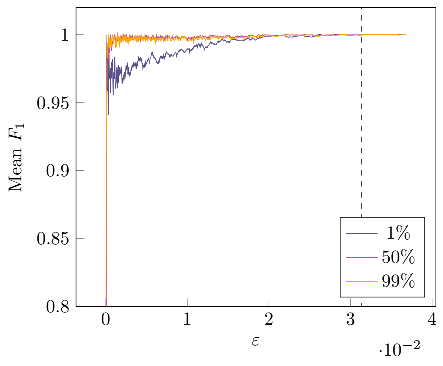


(e) CIFAR10 A ReLU Strong

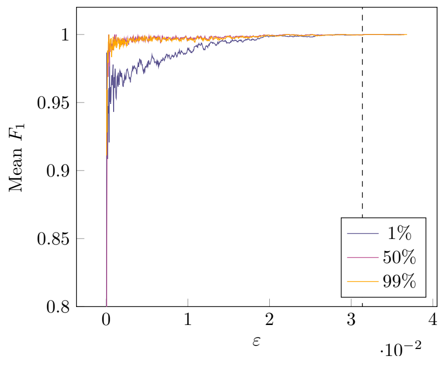


(f) CIFAR10 A ReLU Balanced

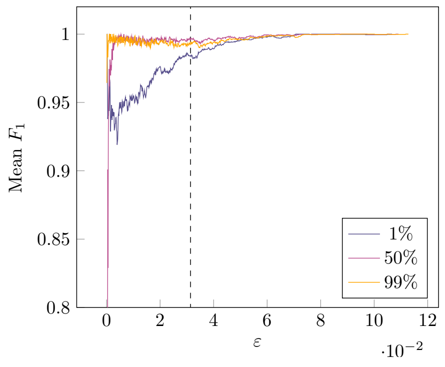
Figure 6.11: F_1 scores in relation to ε for CIFAR10 A for each considered percentile. For ease of visualization, we set the graph cutoff at $F_1 = 0.8$. We also mark 8/255 (a common choice for ε) with a dotted line.



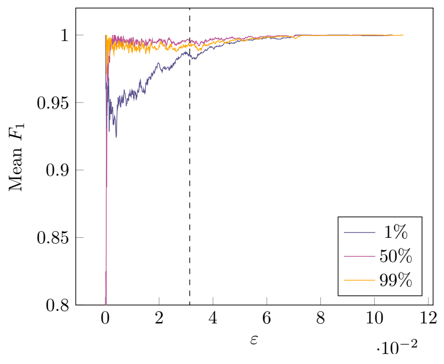
(a) CIFAR10 B Standard Strong



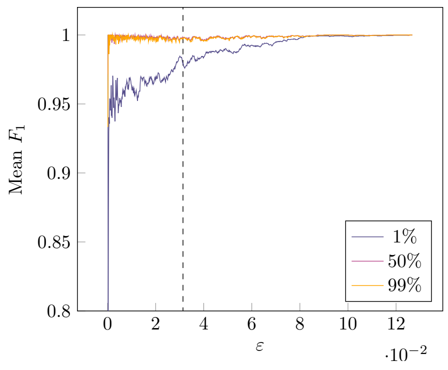
(b) CIFAR10 B Standard Balanced



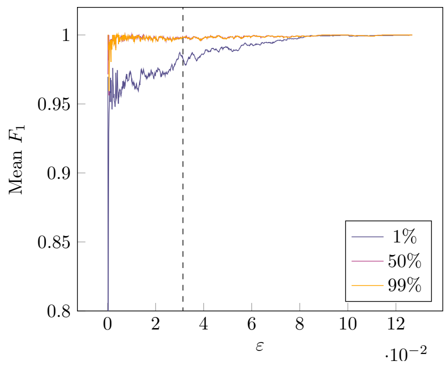
(c) CIFAR10 B Adversarial Strong



(d) CIFAR10 B Adversarial Balanced

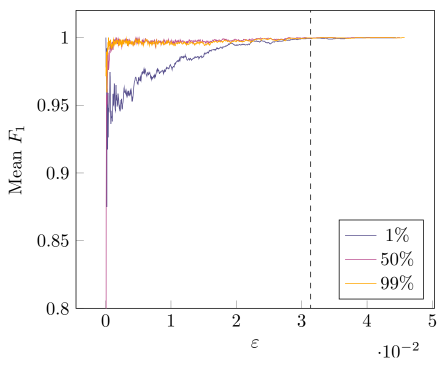


(e) CIFAR10 B ReLU Strong

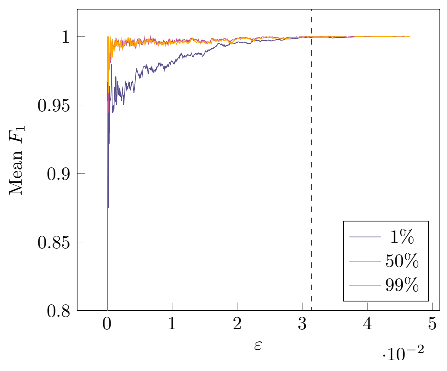


(f) CIFAR10 B ReLU Balanced

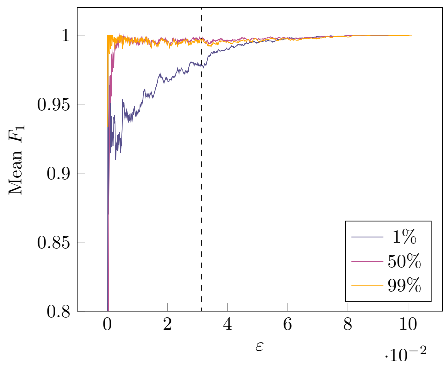
Figure 6.12: F_1 scores in relation to ε for CIFAR10 B for each considered percentile. For ease of visualization, we set the graph cutoff at $F_1 = 0.8$. We also mark 8/255 (a common choice for ε) with a dotted line.



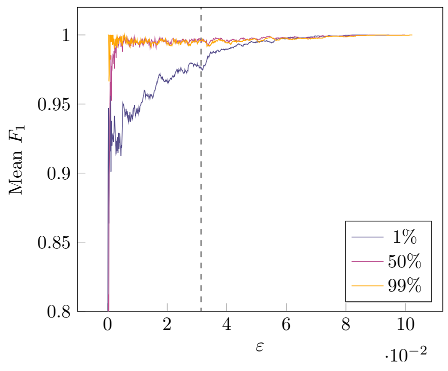
(a) CIFAR10 C Standard Strong



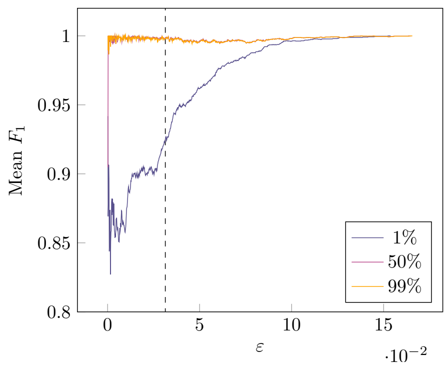
(b) CIFAR10 C Standard Balanced



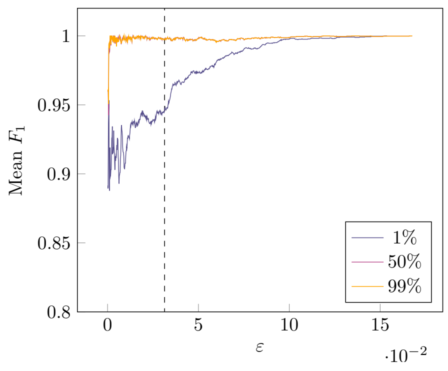
(c) CIFAR10 C Adversarial Strong



(d) CIFAR10 C Adversarial Balanced



(e) CIFAR10 C ReLU Strong



(f) CIFAR10 C ReLU Balanced

Figure 6.13: F_1 scores in relation to ε for CIFAR10 C for each considered percentile. For ease of visualization, we set the graph cutoff at $F_1 = 0.8$. We also mark $8/255$ (a common choice for ε) with a dotted line.

6.5 UG100 Dataset

We collect all the adversarial examples found by both MIPVerify and the heuristic attacks into a new dataset, which we name UG100. UG100 can be used as a benchmark for new adversarial attacks, since we can compare its performance with both the theoretical optimum and heuristic attack pools. Another potential application involves studying the factors that affect whether a certain attack over-estimates the decision boundary distance. Examples include the true distance from the decision boundary and the landscape of the attack loss. Finally, UG100 can be used to perform small-scale adversarial training against the worst-case adversarial examples.

We report the composition of the UG100 dataset by ground truth label in Table 6.34.

Table 6.34: Ground truth labels of the UG100 dataset.

(a) MNIST

Ground Truth	**Count**	**%**
0	219	9.77%
1	228	10.17%
2	225	10.04%
3	225	10.04%
4	225	10.04%
5	220	9.82%
6	227	10.13%
7	221	9.86%
8	225	10.04%
9	226	10.08%

(b) CIFAR10

Ground Truth	**Count**	**%**
Airplane	228	10.05%
Automobile	227	10.00%
Bird	228	10.05%
Cat	228	10.05%
Deer	226	9.96%
Dog	227	10.00%
Frog	227	10.00%
Horse	227	10.00%
Ship	225	9.92%
Truck	226	9.96%

Chapter 7

Conclusion

We studied six key asymmetries that undermine adversarial defenses and formulated four guidelines to minimize their effect. We then proved that while attacking a model is NP-complete, defending it is Σ_2^P-complete. From these observations, we formulated Counter-Attack, an asymmetry-free metadefense that can detect defense failures. We provided both a theoretical and an empirical analysis, showing that as long as the employed attacks are reliable, CA can correctly identify non-robust points. We also proved that running the exact formulation of CA is NP-complete and fooling the exact formulation of CA is Σ_2^P, providing a strong formal argument in a favor of the computational robustness of CA.

One key aspect of CA is that it relies on a very general property of adversarial attacks, i.e. the way they approximate the distance from the decision boundary. This makes the approach "future-proof": as the understanding of robustness improves and more reliable attacks are developed, CA becomes inherently stronger. Moreover, the fact that CA relies on adversarial attacks suggests that designing robust models is not at odds with training models that are easy to attack. Therefore, the development of these "attack-friendly" models represents a promising research direction. Our work also showcases the potential impact of developing adversarial attacks that, even without being stronger than state-of-the-art attacks, provide guaranteed approximation factors.

Due to its independence from the specific characteristics of the defended model, CA can also be applied to non-ML tools (e.g. signature-based malware detectors). We also believe that it is possible to extend CA beyond classification.

Overall, Counter-Attack represents an important step towards viewing adversarial attacks not as a threat to the robustness of models, but rather as a fundamental tool to build more robust models.

Acknowledgments

We acknowledge the CINECA award under the ISCRA initiative, for the availability of high performance computing resources and support.

We also thank Rebecca Montanari and Andrea Borghesi for their advice and support.

Bibliography

[1] Morteza Ali Ahmadi, Rouhollah Dianat, and Hossein Amirkhani. An adversarial attack detection method in deep neural networks based on re-attacking approach. *Multimedia Tools and Applications*, 80(7):10985–11014, 2021.

[2] Naveed Akhtar, Ajmal Mian, Navid Kardan, and Mubarak Shah. Advances in adversarial attacks and defenses in computer vision: A survey. *IEEE Access*, 9:155161–155196, 2021.

[3] Anish Athalye and Nicholas Carlini. On the robustness of the CVPR 2018 white-box adversarial example defenses. *arXiv preprint arXiv:1804.03286*, 2018.

[4] 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, 2018.

[5] Mitali Bafna, Jack Murtagh, and Nikhil Vyas. Thwarting adversarial examples: An l_0-robust sparse fourier transform. *Advances in Neural Information Processing Systems*, 31, 2018.

[6] Arjun Nitin Bhagoji, Daniel Cullina, and Prateek Mittal. Dimensionality reduction as a defense against evasion attacks on machine learning classifiers. *arXiv preprint arXiv:1704.02654*, 2:1, 2017.

[7] 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.

[8] Wieland Brendel, Jonas Rauber, Matthias Kümmerer, Ivan Ustyuzhaninov, and Matthias Bethge. Accurate, reliable and fast robustness evaluation. *Advances in Neural Information Processing Systems*, 32, 2019.

[9] Tom B Brown, Dandelion Mané, Aurko Roy, Martín Abadi, and Justin Gilmer. Adversarial patch. *arXiv preprint arXiv:1712.09665*, 2017.

[10] Nicholas Carlini, Guy Katz, Clark Barrett, and David L Dill. Provably minimally-distorted adversarial examples. *arXiv preprint arXiv:1709.10207*, 2017.

[11] Nicholas Carlini and David Wagner. Defensive distillation is not robust to adversarial examples. *arXiv preprint arXiv:1607.04311*, 2016.

[12] 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, 2017.

[13] Nicholas Carlini and David Wagner. Magnet and "efficient defenses against adversarial attacks" are not robust to adversarial examples. *arXiv preprint arXiv:1711.08478*, 2017.

[14] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 39–57. IEEE, 2017.

[15] 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.

[16] Francesco Croce, Sven Gowal, Thomas Brunner, Evan Shelhamer, Matthias Hein, and Taylan Cemgil. Evaluating the adversarial robustness of adaptive test-time defenses. *arXiv preprint arXiv:2202.13711*, 2022.

[17] Chen Dan, Yuting Wei, and Pradeep Ravikumar. Sharp statistical guaratees for adversarially robust gaussian classification. In *International Conference on Machine Learning*, pages 2345–2355. PMLR, 2020.

[18] Guneet S Dhillon, Kamyar Azizzadenesheli, Zachary C Lipton, Jeremy Bernstein, Jean Kossaifi, Aran Khanna, and Anima Anandkumar. Stochastic activation pruning for robust adversarial defense. In *International Conference on Learning Representations*, 2018.

[19] Gavin Weiguang Ding, Luyu Wang, and Xiaomeng Jin. AdverTorch v0.1: An adversarial robustness toolbox based on pytorch. *arXiv preprint arXiv:1902.07623*, 2019.

[20] Edgar Dobriban, Hamed Hassani, David Hong, and Alexander Robey. Provable tradeoffs in adversarially robust classification. *arXiv preprint arXiv:2006.05161*, 2020.

[21] Tommaso Dreossi, Shromona Ghosh, Alberto Sangiovanni-Vincentelli, and Sanjit A Seshia. A formalization of robustness for deep neural networks. *arXiv preprint arXiv:1903.10033*, 2019.

[22] Krishnamurthy Dvijotham, Robert Stanforth, Sven Gowal, Timothy A Mann, and Pushmeet Kohli. A dual approach to scalable verification of deep networks. In *UAI*, volume 1, page 3, 2018.

[23] Noureen Fatima, Ali Shariq Imran, Zenun Kastrati, Sher Muhammad Daudpota, Abdullah Soomro, and Sarang Shaikh. A systematic literature review on text generation using deep neural network models. *IEEE Access*, 2022.

[24] Reuben Feinman, Ryan R Curtin, Saurabh Shintre, and Andrew B Gardner. Detecting adversarial samples from artifacts. *arXiv preprint arXiv:1703.00410*, 2017.

[25] Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*, 2015.

[26] Sorin Grigorescu, Bogdan Trasnea, Tiberiu Cocias, and Gigel Macesanu. A survey of deep learning techniques for autonomous driving. *Journal of Field Robotics*, 37(3):362–386, 2020.

[27] Kathrin Grosse, Praveen Manoharan, Nicolas Papernot, Michael Backes, and Patrick McDaniel. On the (statistical) detection of adversarial examples. *arXiv preprint arXiv:1702.06280*, 2017.

[28] Chuan Guo, Mayank Rana, Moustapha Cisse, and Laurens Van Der Maaten. Countering adversarial images using input transformations. In *International Conference on Learning Representations*, 2018.

[29] Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2022.

[30] Husheng Han, Kaidi Xu, Xing Hu, Xiaobing Chen, Ling Liang, Zidong Du, Qi Guo, Yanzhi Wang, and Yunji Chen. Scalecert: Scalable certified defense against adversarial patches with sparse superficial layers. *Advances in Neural Information Processing Systems*, 34, 2021.

[31] Warren He, James Wei, Xinyun Chen, Nicholas Carlini, and Dawn Song. Adversarial example defenses: ensembles of weak defenses are not strong. In *Proceedings of the 11th USENIX Conference on Offensive Technologies*, pages 15–15, 2017.

[32] Matthias Hein and Maksym Andriushchenko. Formal guarantees on the robustness of a classifier against adversarial manipulation. *Advances in Neural Information Processing Systems*, 30, 2017.

[33] Dan Hendrycks and Kevin Gimpel. Early methods for detecting adversarial images. In *International Conference on Learning Representations (Workshop Track)*, 2017.

[34] Hossein Hosseini, Sreeram Kannan, and Radha Poovendran. Are odds really odd? bypassing statistical detection of adversarial examples. *arXiv preprint arXiv:1907.12138*, 2019.

[35] Shengyuan Hu, Tao Yu, Chuan Guo, Wei-Lun Chao, and Kilian Q Weinberger. A new defense against adversarial images: Turning a weakness into a strength. *Advances in Neural Information Processing Systems*, 32, 2019.

[36] Guy Katz, Clark Barrett, David Dill, Kyle Julian, and Mykel Kochenderfer. Reluplex: An efficient smt solver for verifying deep neural networks. *arXiv preprint arXiv:1702.01135*, 2017.

[37] Asifullah Khan, Anabia Sohail, Umme Zahoora, and Aqsa Saeed Qureshi. A survey of the recent architectures of deep convolutional neural networks. *Artificial intelligence review*, 53(8):5455–5516, 2020.

[38] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[39] Roger Koenker and Kevin F Hallock. Quantile regression. *Journal of economic perspectives*, 15(4):143–156, 2001.

[40] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. Technical report, 2009.

[41] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial machine learning at scale. 2017.

[42] Yann LeCun, Corinna Cortes, and Christopher J.C. Burges. The MNIST database of handwritten digits. `http://yann.lecun.com/exdb/mnist/`, 1998.

[43] Klas Leino, Zifan Wang, and Matt Fredrikson. Globally-robust neural networks. In *International Conference on Machine Learning*, pages 6212–6222. PMLR, 2021.

[44] Xin Li and Fuxin Li. Adversarial examples detection in deep networks with convolutional filter statistics. In *Proceedings of the IEEE international conference on computer vision*, pages 5764–5772, 2017.

[45] Yingzhen Li, John Bradshaw, and Yash Sharma. Are generative classifiers more robust to adversarial attacks? In *International Conference on Machine Learning*, pages 3804–3814. PMLR, 2019.

[46] 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. In *International Conference on Learning Representations*, 2018.

[47] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*, 2018.

[48] Dongyu Meng and Hao Chen. Magnet: a two-pronged defense against adversarial examples. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 135–147, 2017.

[49] Jan Hendrik Metzen, Tim Genewein, Volker Fischer, and Bastian Bischoff. On detecting adversarial perturbations. In *International Conference on Learning Representations*, 2017.

[50] Yifei Min, Lin Chen, and Amin Karbasi. The curious case of adversarially robust models: More data can help, double descend, or hurt generalization. In *Uncertainty in Artificial Intelligence*, pages 129–139. PMLR, 2021.

[51] 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.

[52] Taesik Na, Jong Hwan Ko, and Saibal Mukhopadhyay. Cascade adversarial machine learning regularized with a unified embedding. In *International Conference on Learning Representations*, 2018.

[53] Nina Narodytska and Shiva Prasad Kasiviswanathan. Simple black-box adversarial perturbations for deep networks. *arXiv preprint arXiv:1612.06299*, 2016.

[54] Ambar Pal and René Vidal. A game theoretic analysis of additive adversarial attacks and defenses. *Advances in Neural Information Processing Systems*, 33:1345–1355, 2020.

[55] Tianyu Pang, Kun Xu, Yinpeng Dong, Chao Du, Ning Chen, and Jun Zhu. Rethinking softmax cross-entropy loss for adversarial robustness. *arXiv preprint arXiv:1905.10626*, 2019.

[56] Tianyu Pang, Kun Xu, Chao Du, Ning Chen, and Jun Zhu. Improving adversarial robustness via promoting ensemble diversity. In *International Conference on Machine Learning*, pages 4970–4979. PMLR, 2019.

[57] Tianyu Pang, Kun Xu, and Jun Zhu. Mixup inference: Better exploiting mixup to defend adversarial attacks. In *International Conference on Learning Representations*, 2020.

[58] Nicolas Papernot and Patrick McDaniel. On the effectiveness of defensive distillation. *arXiv preprint arXiv:1607.05113*, 2016.

[59] 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.

[60] 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, 2016.

[61] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[62] Rafael Pinot, Florian Yger, Cédric Gouy-Pailler, and Jamal Atif. A unified view on differential privacy and robustness to adversarial examples. *arXiv preprint arXiv:1906.07982*, 2019.

[63] Aaditya Prakash, Nick Moran, Solomon Garber, Antonella DiLillo, and James Storer. Deflecting adversarial attacks with pixel deflection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8571–8580, 2018.

[64] Muni Sreenivas Pydi and Varun Jog. The many faces of adversarial risk. *Advances in Neural Information Processing Systems*, 34, 2021.

[65] Aditi Raghunathan, Jacob Steinhardt, and Percy Liang. Certified

defenses against adversarial examples. *arXiv preprint arXiv:1801.09344*, 2018.

[66] 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.

[67] Ambrish Rawat, Martin Wistuba, and Maria-Irina Nicolae. Adversarial phenomenon in the eyes of bayesian deep learning. *arXiv preprint arXiv:1711.08244*, 2017.

[68] Clémence Réda, Emilie Kaufmann, and Andrée Delahaye-Duriez. Machine learning applications in drug development. *Computational and structural biotechnology journal*, 18:241–252, 2020.

[69] Jie Ren, Die Zhang, Yisen Wang, Lu Chen, Zhanpeng Zhou, Yiting Chen, Xu Cheng, Xin Wang, Meng Zhou, Jie Shi, et al. A unified game-theoretic interpretation of adversarial robustness. *arXiv preprint arXiv:2111.03536*, 2021.

[70] Eitan Richardson and Yair Weiss. A bayes-optimal view on adversarial examples. *Journal of Machine Learning Research*, 22(221):1–28, 2021.

[71] Kevin Roth, Yannic Kilcher, and Thomas Hofmann. The odds are odd: A statistical test for detecting adversarial examples. In *International Conference on Machine Learning*, pages 5498–5507. PMLR, 2019.

[72] Pouya Samangouei, Maya Kabkab, and Rama Chellappa. Defense-gan: Protecting classifiers against adversarial attacks using generative models. In *International Conference on Learning Representations*, 2018.

[73] Sanchari Sen, Balaraman Ravindran, and Anand Raghunathan. Empir: Ensembles of mixed precision deep networks for increased robustness against adversarial attacks. *arXiv preprint arXiv:2004.10162*, 2020.

[74] Shiwei Shen, Guoqing Jin, Ke Gao, and Yongdong Zhang. Ape-gan: Adversarial perturbation elimination with gan. *arXiv preprint arXiv:1707.05474*, 2017.

[75] Samuel Henrique Silva and Peyman Najafirad. Opportunities and challenges in deep learning adversarial robustness: A survey. *arXiv preprint arXiv:2007.00753*, 2020.

[76] Gagandeep Singh, Timon Gehr, Matthew Mirman, Markus Püschel, and Martin Vechev. Fast and effective robustness certification. *Advances in Neural Information Processing Systems*, 31, 2018.

[77] Aman Sinha, Hongseok Namkoong, Riccardo Volpi, and John Duchi. Certifying some distributional robustness with principled adversarial training. In *International Conference on Learning Representations*, 2018.

[78] Yang Song, Taesup Kim, Sebastian Nowozin, Stefano Ermon, and Nate Kushman. Pixeldefend: Leveraging generative models to understand and defend against adversarial examples. In *International Conference on Learning Representations*, 2018.

[79] Larry J Stockmeyer. The polynomial-time hierarchy. *Theoretical Computer Science*, 3(1):1–22, 1976.

[80] Jiawei Su, Danilo Vasconcellos Vargas, and Kouichi Sakurai. One pixel attack for fooling deep neural networks. *IEEE Transactions on Evolutionary Computation*, 23(5):828–841, 2019.

[81] 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.

[82] Rajkumar Theagarajan, Ming Chen, Bir Bhanu, and Jing Zhang. Shield-nets: Defending against adversarial attacks using probabilistic adver-

sarial robustness. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6988–6996, 2019.

[83] Vincent Tjeng, Kai Y Xiao, and Russ Tedrake. Evaluating robustness of neural networks with mixed integer programming. In *International Conference on Learning Representations*, 2019.

[84] Florian Tramèr, Jens Behrmann, Nicholas Carlini, Nicolas Papernot, and Jörn-Henrik Jacobsen. Fundamental tradeoffs between invariance and sensitivity to adversarial perturbations. In *International Conference on Machine Learning*, pages 9561–9571. PMLR, 2020.

[85] Florian Tramer, Nicholas Carlini, Wieland Brendel, and Aleksander Madry. On adaptive attacks to adversarial example defenses. *Advances in Neural Information Processing Systems*, 33:1633–1645, 2020.

[86] 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.

[87] Gunjan Verma and Ananthram Swami. Error correcting output codes improve probability estimation and adversarial robustness of deep neural networks. *Advances in Neural Information Processing Systems*, 32, 2019.

[88] Guillaume Vidot, Paul Viallard, Amaury Habrard, and Emilie Morvant. A pac-bayes analysis of adversarial robustness. *arXiv preprint arXiv:2102.11069*, 2021.

[89] Haotao Wang, Tianlong Chen, Shupeng Gui, TingKuei Hu, Ji Liu, and Zhangyang Wang. Once-for-all adversarial training: In-situ tradeoff between robustness and accuracy for free. *Advances in Neural Information Processing Systems*, 33:7449–7461, 2020.

[90] Siyue Wang, Xiao Wang, Pu Zhao, Wujie Wen, David Kaeli, Peter Chin, and Xue Lin. Defensive dropout for hardening deep neural networks

under adversarial attacks. In *Proceedings of the International Conference on Computer-Aided Design*, pages 1–8, 2018.

[91] 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, 2018.

[92] 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. In *International Conference on Learning Representations*, 2018.

[93] Boxi Wu, Heng Pan, Li Shen, Jindong Gu, Shuai Zhao, Zhifeng Li, Deng Cai, Xiaofei He, and Wei Liu. Attacking adversarial attacks as a defense. *arXiv preprint arXiv:2106.04938*, 2021.

[94] Yuhang Wu, Sunpreet S Arora, Yanhong Wu, and Hao Yang. Beating attackers at their own games: Adversarial example detection using adversarial gradient directions. *arXiv preprint arXiv:2012.15386*, 2020.

[95] Zuxuan Wu, Ser-Nam Lim, Larry S Davis, and Tom Goldstein. Making an invisibility cloak: Real world adversarial attacks on object detectors. In *European Conference on Computer Vision*, pages 1–17. Springer, 2020.

[96] Chang Xiao, Peilin Zhong, and Changxi Zheng. Enhancing adversarial defense by k-winners-take-all. *arXiv preprint arXiv:1905.10510*, 2019.

[97] Kai Y Xiao, Vincent Tjeng, Nur Muhammad Shafiullah, and Aleksander Madry. Training for faster adversarial robustness verification via inducing relu stability. In *International Conference on Learning Representations*, 2019.

[98] Cihang Xie, Jianyu Wang, Zhishuai Zhang, Zhou Ren, and Alan Yuille. Mitigating adversarial effects through randomization. In *International Conference on Learning Representations*, 2018.

[99] Han Xu, Xiaorui Liu, Yaxin Li, Anil Jain, and Jiliang Tang. To be robust or to be fair: Towards fairness in adversarial training. In *International Conference on Machine Learning*, pages 11492–11501. PMLR, 2021.

[100] Yao-Yuan Yang, Cyrus Rashtchian, Hongyang Zhang, Russ R Salakhutdinov, and Kamalika Chaudhuri. A closer look at accuracy vs. robustness. In *Advances in Neural Information Processing Systems*, volume 33, pages 8588–8601, 2020.

[101] Yuzhe Yang, Guo Zhang, Dina Katabi, and Zhi Xu. Me-net: Towards effective adversarial robustness with matrix estimation. In *International Conference on Machine Learning*, pages 7025–7034. PMLR, 2019.

[102] Xuwang Yin, Soheil Kolouri, and Gustavo K Rohde. Adversarial example detection and classification with asymmetrical adversarial training. *arXiv preprint arXiv:1905.11475*, 2019.

[103] Valentina Zantedeschi, Maria-Irina Nicolae, and Ambrish Rawat. Efficient defenses against adversarial attacks. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pages 39–49, 2017.

[104] Haimin Zhang and Min Xu. Towards counteracting adversarial perturbations to resist adversarial examples. 2020.

[105] 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, 2019.

[106] 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, 2018.

[107] Jingfeng Zhang, Xilie Xu, Bo Han, Gang Niu, Lizhen Cui, Masashi Sugiyama, and Mohan Kankanhalli. Attacks which do not kill training make adversarial learning stronger. In *International Conference on Machine Learning*, pages 11278–11287. PMLR, 2020.

[108] Yan Zhou, Murat Kantarcioglu, and Bowei Xi. A survey of game theoretic approach for adversarial machine learning. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 9(3):e1259, 2019.