universität wien

MASTERARBEIT | MASTER'S THESIS

Titel | Title

Adversarial Examples in Machine Learning:
Training and Certified Robustness

verfasst von | submitted by

Jonas Schuhmann

angestrebter akademischer Grad | in partial fulfilment of the requirements for the degree of

Master of Science (MSc)

Wien | Vienna, 2024

Studienkennzahl lt. Studienblatt	Degree programme code as it appears on the student record sheet:	UA 066 821
Studienrichtung lt. Studienblatt	Degree programme as it appears on the student record sheet:	Masterstudium Mathematik
Betreut von	Supervisor:	Assoz. Prof. Dr. Philipp Christian Petersen M.Sc.

Abstract

Diese Arbeit untersucht die mathematischen Grundlagen zur Generierung sogenannter Adversarial Examples und entwickelt Methoden zum Schutz von Modellen des maschinellen Lernens gegen solche gezielten Störungen. Das Konzept, bekannt als Adversarial Risk, beschreibt die Robustheit eines Modells als Min-Max-Optimierungsproblem. Adversarial Training wird als primäre Abwehrstrategie analysiert und experimentell bewertet. Um die Robustheit weiter zu verbessern, wird eine Schranke für das Adversarial Risk eingeführt. Basierend auf diesen theoretischen Erkenntnissen wird die TRADES-Methode als eine optimierte Variante des Adversarial Trainings vorgestellt. Darüber hinaus wird eine neue Klasse von Adversarial Examples eingeführt, die als Regular Adversarial Examples bezeichnet wird. Es wird ein Algorithmus entwickelt und experimentell getestet, der die Abwesenheit dieser Regular Adversarial Examples innerhalb eines bestimmten Störbereichs für neuronale Netzwerke bestätigt. Die Ergebnisse zeigen, dass dieser Ansatz moderate Robustheitsgarantien bietet, insbesondere für einfachere Netzwerkarchitekturen.

Abstract

This work explores the mathematical foundations for generating adversarial examples and develops methods to safeguard machine learning models against such targeted perturbations. Central to this is the concept of adversarial risk, which allows the robustness of a model to be formulated as a Min-Max problem. Adversarial training is analyzed and experimentally evaluated as the primary defense strategy. To further enhance robustness, a bound for adversarial risk is introduced. Based on these theoretical insights, the TRADES method is presented as an optimized variant of adversarial training. Additionally, we introduce a new class of adversarial examples, referred to as regular adversarial examples. We also develop and experimentally test an algorithm designed to certify the absence of these regular adversarial examples within a specified perturbation range for neural networks. The results demonstrate that this approach provides moderate robustness guarantees, particularly for simpler network architectures.

Acknowledgement

First and foremost, I would like to express my deepest gratitude to my supervisor, Philipp Petersen, for providing me with such a fascinating topic and for his continuous support throughout my thesis. I am also deeply grateful to my family, whose support has allowed me to pursue my studies. Lastly, I wish to extend my thanks to the friends I've made during my studies, whose presence made this experience both memorable and enriching.

Contents

1 Introduction

Machine learning, particularly deep learning, has gained massive popularity due to its ability to deliver groundbreaking results across a wide range of fields. From achieving near-human accuracy in image classification [12], to advancements in speech recognition [11], and the emergence of large language models like ChatGPT, machine learning has transformed various domains.

In many machine learning tasks, it is very important to have models that are highly secure, reliable and robust, as they are often used in safety-critical environments such as autonomous vehicles or financial fraud detection. However, as Szegedy et al. [29] showed, it is easy to trick a model that was trained to very high accuracy into misclassifying by manipulating the input with extremely small perturbations.
In image classification, these so-called adversarial examples have perturbations so small that the difference between the original and the perturbed image can't be detected by the human eye (see Figure 1.1). This phenomenon is not limited to computer vision

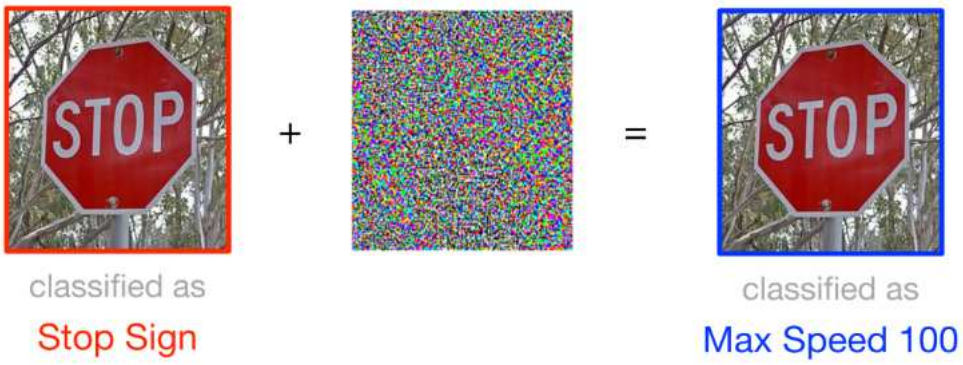

Figure 1.1: Adversarial example for a traffic sign (picture by [33])

alone [31]. Adversarial examples can also be created to fool text classification models [7] and to mislead graph neural networks [37]. When a model is attacked with an adversarial example, the incorrect prediction is often made with very high confidence [10]. What is more concerning is the fact that an adversarial image designed to fool one model can also fool another model, even if the other model has a different architecture or was trained on an entirely different dataset [36]. These so-called black-box attacks, where the attacker has no knowledge of the model's parameters, pose a significant threat because they allow an attacker to mislead a classifier without needing any information about

the target model. Right now it is not completely clear, why these adversarial examples even emerge. There is also some criticism [28] of the common definition of adversarial examples, as it fails to consider the potential for label change in the perturbed sample. For instance, in some images, even a small perturbation can be sufficient to cause a label change.

Because of the threat of adversarial examples, it is important to develop defences against and create robust models. One of the main tools to develop robust models is the so-called adversarial training [16]. Instead of the normal training procedure, in each training iteration, the training set is replaced with adversarial examples. With this tool, it is partly possible to create robust models.

In this thesis, Chapter 1 will provide a rigorous introduction to adversarial examples from a mathematical perspective. In Section 1.4 we will explore methods for effectively generating these adversarial examples. These so-called adversarial attacks are crucial for the subsequent chapter, where they will be used to develop models that are robust to such examples.

Next, we present in Chapter 2 a risk framework for adversarial examples, where we introduce the so-called adversarial risk instead of the normal risk. With the adversarial risk we can formulate the robustness of a model to adversarial examples as a min-max problem. Then in Section 2.2 we will discuss the so-called adversarial training, a key method for addressing this problem and a primary defense against adversarial attacks. Afterwards, in Section 2.3 we will conduct experiments to evaluate both the effectiveness of these adversarial attacks and the robustness of models trained with adversarial training.

In the subsequent Chapter 3 we will aim to present an alternative version of adversarial training, which achieves even greater robustness than the standard one. To do that, we will introduce surrogate loss functions in Section 3.1 and explore associated risk bounds. We will then apply this theory in Section 3.2 to bound the adversarial risk. Using this risk bound, we will present in Section 3.3 adversarial training through TRADES, which has been shown to provide greater robustness compared than standard adversarial training.

In the final Chapter 4, we will introduce an alternative version of adversarial examples that helps to address label changes caused by the perturbation of the images. We will demonstrate in Sections 4.1 and 4.2 how these so-called regular adversarial examples can emerge and discuss their implications for model robustness. Following this, we will present in Section 4.3 an algorithm for certifying robustness, enabling us to verify the non-existence of these regular adversarial examples within a given perturbation range for binary neural networks. In the last section 4.4, we will show through experiments that this algorithm provides moderate robustness guarantees, but mainly for shallow neural

networks.

1.1. Notation

In this thesis, we will always consider the label space Y to be discrete and finite. Specifically, we define:

$$Y = \{a_1, a_2, \ldots, a_n\} \subset \mathbb{Z},$$

where $n \in \mathbb{N}$ represents the number of possible labels. Furthermore, we will generally consider the input space X to be a subset of the Euclidean space $\mathbb{R}^d$ for some $d \in \mathbb{N}$. Except for Section 3.1, we will always assume that $X = \mathbb{R}^d$.

We define a hypothesis set $\mathcal{H} \subset \{X \to Y\}$, where each element of $\mathcal{H}$ is a function that maps inputs X to labels Y. The elements of $\mathcal{H}$ are called classifiers or hypotheses. For simplicity, we often restrict ourselves to a subset of hypotheses that can be described by a set of parameters. Let $\Theta \subset \mathbb{R}^d$ denote the parameter set, where d is the number of parameters. Then, similar to Fix et al. [8], we assume that there exists a function

$$h : \Theta \times X \longrightarrow Y$$

such that a subset of hypotheses $h \in \mathcal{H}$ can be represented in a parameterized form. Specifically, for each parameter $\theta \in \Theta$, we define a parameterized function $h_\theta : X \to Y$ by:

$$h_\theta(x) := h(\theta, x).$$

This allows us to define the following set:

$$\mathcal{H}_\Theta = \{h_\theta \mid \theta \in \Theta\} \subset \mathcal{H},$$

which we call a parametric hypothesis space. Each element $h_\theta \in \mathcal{H}_\Theta$ is called a parameterized hypothesis, or simply a parameterization. In practice, for example, when using neural networks with a fixed architecture, the vector $\theta \in \Theta$ describes how the specific weights and biases are chosen.

1.2. Neural Networks

Neural networks are among the most successful and influential models in machine learning, having achieved groundbreaking results across various domains. To discuss their robustness later, we start with a brief introduction to their fundamental concepts. Inspired by the structure and function of the human brain, a neural network consists

of units called neurons. The idea was introduced by McCulloch and Pitts [17], who described each neuron as computing a function $\nu : \mathbb{R}^d \longrightarrow \mathbb{R}$ given by:

$$\nu(x) = \sigma(w^T x + b),$$

where w denotes the weight vector, b is the bias term, and $\sigma : \mathbb{R} \longrightarrow \mathbb{R}$ is the activation function. In a neural network, neurons are organized into layers. The output of neurons from one layer serves as input for neurons in the next layer, creating a network structure. Formally, a neural network is defined as:

Definition 1.2.1. Let $L \in \mathbb{N}$, $d_0, \ldots, d_{L+1} \in \mathbb{N}$, and let $\sigma : \mathbb{R} \longrightarrow \mathbb{R}$. A function $\Phi : \mathbb{R}^{d_0} \longrightarrow \mathbb{R}^{d_{L+1}}$ is called a neural network if there exist matrices $W^{(\ell)} \in \mathbb{R}^{d_{\ell+1} \times d_\ell}$ and vectors $b^{(\ell)} \in \mathbb{R}^{d_{\ell+1}}$, $\ell = 0, \ldots, L$, such that with

$$\begin{aligned} x^{(0)} &:= x \\ x^{(\ell)} &:= \sigma\left(W^{(\ell-1)} x^{(\ell-1)} + b^{(\ell-1)}\right) \quad \text{for} \quad \ell \in \{1, \ldots, L\} \\ x^{(L+1)} &:= W^{(L)} x^{(L)} + b^{(L)} \end{aligned}$$

it holds

$$\Phi(x) = x^{(L+1)} \quad \text{for all } x \in \mathbb{R}^{d_0}.$$

We call L the depth, $d_{\max} = \max_{\ell=1,\ldots,L} d_\ell$ the width, σ the activation function, and $(\sigma; d_0, \ldots, d_{L+1})$ the architecture of the neural network Φ. Moreover, $W^{(\ell)} \in \mathbb{R}^{d_{\ell+1} \times d_\ell}$ are the weight matrices and $b^{(\ell)} \in \mathbb{R}^{d_{\ell+1}}$ are the bias vectors of Φ for $\ell = 0, \ldots, L$.

Neural networks with a depth of one are called to as shallow networks, while those with a depth greater than one are known as deep networks. Commonly used activation functions include:

Definition 1.2.2. The Rectified Linear Unit (ReLU) activation function is defined as:

$$\sigma_{\mathrm{ReLU}}(x) = \max(0, x).$$

Definition 1.2.3. The Sigmoid activation function is defined as

$$\sigma_{\mathrm{sig}}(x) = \frac{1}{1 + e^{-x}}.$$

In practice, a neural network is trained by minimizing the empirical risk (see Definition 2.1.2). This process involves adjusting the network's weights to minimize the loss function. Backpropagation [24], a widely used algorithm, allows the efficient computation of the gradient of the loss function with respect to the weights by applying the chain rule. This gradient information is then used by optimization methods, such as stochastic gradient descent, to iteratively update the weights and improve the network's performance. For further details, we refer to [9, 19].

1.3. Definition of Adversarial Examples

First, we want to define adversarial examples in a mathematically rigorous way:

Definition 1.3.1. Let $\mathbb{R}^d$ be the input and Y be the label space. Further, let D be a distribution on $\mathbb{R}^d \times Y$ and let $h \in \mathcal{H}$ be a classifier. For $(x, y) \in \mathbb{R}^d \times Y$ drawn from the distribution D, and a set $P \subset \mathbb{R}^d$ of allowed perturbations, we call $\tilde{x} \in (x + P)$ an adversarial example of x, if

$$h(\tilde{x}) \neq y.$$

Note that $(x + P)$ corresponds to the set $\{x + v \mid v \in P\}$. In the context of image classification, P is often chosen to represent the set of images that are perceived as similar. Typically, this is modeled by considering balls in the ℓ^p-norm around the points. This means, for a maximal perturbation $\epsilon > 0$ and $p \in \mathbb{N} \cup \{\infty\}$, where $d \in \mathbb{N}$, we define P as

$$B_\epsilon^p(0) := \{z \in \mathbb{R}^d : ||z||_p \leq \epsilon\}.$$

Although $p = \infty$ is commonly used, focusing solely on a specific norm has its limitations, as we will discuss later.

Note that if $0 \in P$ and we have a sample $(x, y) \in \mathbb{R}^d \times Y$ drawn from the distribution D which is not correctly classified by h (i.e. we have $h(x) \neq y$), then x is an adversarial example of itself. This will be one of the reasons for defining adversarial examples in a different way later.

1.4. Adversarial Attacks

In the following section, we will introduce the concept of adversarial attacks, which are used to effectively generate adversarial examples. To begin, it is important to distinguish between two different settings under which an adversary may attack a given classifier. The following definitions are based on [34]:

- **White-Box Attack:**

  In a white-box setting, the adversary has full access to all information about the target model, including its architecture, parameters, gradients, and other internal details. This allows the adversary to carefully craft adversarial examples by using the complete knowledge of the model.

- **Black-Box Attack:** In a black-box setting, the adversary does not have access to the internal configuration of the classifier. Instead, they can only input data and observe the corresponding outputs. Typically, adversaries generate multiple queries to analyze the model's input-output behavior, allowing them to identify and exploit weaknesses without direct knowledge of the model's internal workings.

In this thesis, we will focus exclusively on White-Box attacks. This is not a significant limitation due to the transferability property of adversarial examples. This phenomenon,

first observed by [29], refers to the ability of an adversarial example designed to fool a specific classifier to often successfully mislead another, different classifier as well. Papernot et al. [20] demonstrated a practical method for exploiting this transferability: they first train a substitute classifier to mimic the target model using a synthetic dataset generated by the adversary, which is labeled according to the target model's outputs. Adversarial examples are then generated using the substitute classifier, and these examples are effective in fooling the target model in most cases.

Now, to discuss the generation of adversarial examples, consider a deterministic setting where $x \in \mathbb{R}^n$ represents an input with a corresponding label $y \in \{a_1, \dots, a_m\} \subset \mathbb{Z}$, where $m \in \mathbb{N}$. Assume that x is drawn according to a probability distribution over $\mathbb{R}^d$, and that a classifier $h \in \mathcal{H}$ correctly classifies x, i.e., $h(x) = y$.

Our goal is to find a perturbed version $\tilde{x}$ such that $h(\tilde{x}) \neq y$. One common approach to achieve this is to maximize the loss function, thereby finding an adversarial example that forces the classifier to make a false prediction. This leads to the following optimization problem:

$$\max_{\tilde{x} \in \mathbb{R}^d} L(h(\tilde{x}), y). \tag{1.1}$$

However, we do not want to allow $\tilde{x}$ to be arbitrarily far from the original input x. To ensure that $\tilde{x}$ remains similar to x, we restrict $\tilde{x}$ to a neighbourhood around x. Additionally, to guarantee that the solution in Problem (1.1) exists, we assume that L and h are continuous and choose a compact set $P \subset \mathbb{R}^d$. This ensures the existence of $x' \in (x + P)$ such that:

$$x' = \arg \max_{\tilde{x} \in (x+P)} L(h(\tilde{x}), y). \tag{1.2}$$

In the following, we will set $P = B_\epsilon^p(0)$, where $B_\epsilon^p(0)$ denotes the ℓ^p-ball around 0 with radius ϵ. If the loss function L and the classifier h are differentiable, we can apply gradient-based optimization methods to solve this problem. To proceed, let $p \in \mathbb{N} \cup \{\infty\}$, $f : \mathbb{R}^n \to \mathbb{R}$ be a differentiable function, and $x \in \mathbb{R}^n$ be a point such that $\nabla f(x) \neq 0$. We define a normalized steepest descent direction as:

$$x_{\text{nsd}}^p = \arg\min\{\nabla f(x)^\top v \mid \|v\|_{\ell^p} = 1\}.$$

Note that the normalized steepest descent direction is not necessarily unique. If there are multiple directions that minimize the expression, we select one arbitrarily. The idea of choosing the normalized steepest descent direction becomes clear when we consider the first-order Taylor approximation of $f(x + v)$ around x:

$$f(x + v) \approx f(x) + \nabla f(x)^T v.$$

Our normalized steepest descent direction is thus a step of norm 1 that maximizes the decrease in the linear approximation of f. It is clear that this direction depends significantly on the choice of norm.

Lemma 1.4.1. *Let $f : \mathbb{R}^d \longrightarrow \mathbb{R}$ be differentiable and $x \in \mathbb{R}^d$ such that $\nabla f(x) \neq 0$, then for the normalized steepest descent directions it holds:*

1. *If $p = 1$, then $x^1_{nsd} = -sign\left(\frac{\partial f(x)}{\partial x_i}\right) e_i$, where $i \in \arg\max_{j \in [d]} |(\nabla f(x))_j|$ and e_i denotes the i-th coordinate vector.*
2. *If $p = 2$, then $x^2_{nsd} = -\frac{\nabla f(x)}{\|\nabla f(x)\|_2}$.*
3. *If $p = \infty$, then $x^\infty_{nsd} = -sign(\nabla f(x))$.*

Proof. The proof can be found in the book from Boyd and Vandenberghe [2]. □

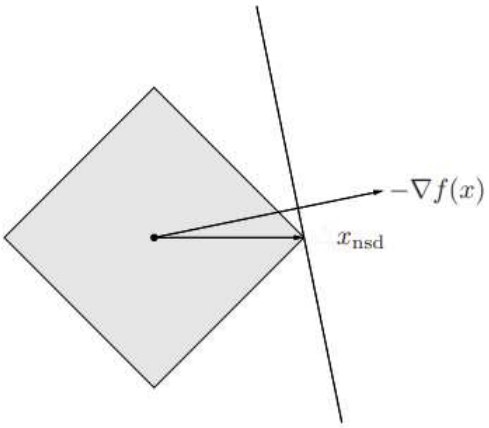

Figure 1.2: Normalized steepest descent direction for the ℓ^1-norm [2].

In the case of the ℓ^1-norm the normalized steepest descent direction can always be chosen in the direction of a standard basis vector. For example, in the Figure 1.2 we have $x^1_{nsd} = e_1$. We can now introduce the following optimization method, which aims to find a minimizer using a fixed step size:

Algorithm 1 Steepest Descent Method

Require: differentiable f, step size $\gamma > 0$, starting point x, maximal number of iterations $T \in \mathbb{N}$, norm $p \in \mathbb{N} \cup \{\infty\}$

1: $x_0 = x$
2: **for** $k = 0$ to $T - 1$ **do**
3: compute $(x_k)^p_{nsd}$
4: $x_{k+1} = x_k + \gamma(x_k)^p_{nsd}$
5: **end for**

For $p = 2$, this algorithm corresponds to normalized gradient descent. To find the maximizer of the loss function, we modify Step 4 by replacing the '+' with a '−'.

For $p = \infty$, we can define a single-step method to solve Equation (1.2). Specifically, the goal is to find a potential adversarial example $\tilde{x}$ within the ϵ-ball around x under the ℓ^∞-norm:

Algorithm 2 Fast Gradient Sign Method [10]

Require: $(x, y) \in X \times Y$, differentiable classifier h, differentiable loss function L, maximal perturbation $\epsilon > 0$

1: $\tilde{x} = x + \epsilon \text{ sign}(\nabla_x L(h(x), y))$

This single-step method with step size ϵ ensures that the perturbed point $\tilde{x}$ will be within $B_\epsilon^\infty(x)$. Known as the Fast Gradient Sign Method (FGSM) [10], it was one of the earliest techniques developed for generating adversarial examples. While the FGSM does not always produce successful adversarial examples, it frequently generates effective adversarial inputs for different values of ϵ. Its single-step nature makes FGSM highly efficient, although it is generally less powerful than other methods, which we will discuss next. In addition, the Fast Gradient Sign Method suffers from the problem of label leakage, an issue that we will explain in detail in our experiments.

It is clear that we can use Algorithm 1 to create a more powerful iterative attack than FGSM. However, since the solution of the problem is restricted to the ball, we first need to introduce a projection onto it. We define $\Pi_{x,\epsilon}^p : \mathbb{R}^n \to \mathbb{R}^n$ as follows:

$$\Pi_{x,\epsilon}^p(x') := \arg \min_{z \in B_\epsilon^p(x)} ||z - x'||_p$$

for $p \in \mathbb{N} \cup \{\infty\}$. For example in the case of $p = \infty$ we get:

$$(\Pi_{x,\epsilon}^\infty(x'))_i = \begin{cases} x_i' & \text{if } |x_i' - x_i| \leq \epsilon \\ \epsilon \text{ sign}(x_i') & \text{if } |x_i' - x_i| > \epsilon \end{cases} \tag{1.3}$$

for all $i \in \{1, \ldots, n\}$. With this projection defined, we can now formulate an iterative method for arbitrary $p \in \mathbb{N} \cup \{\infty\}$:

Algorithm 3 Projected Gradient Descent Attack [16]

Require: $(x, y) \in X \times Y$, differentiable classifier h, differentiable loss function L, maximal perturbation $\epsilon > 0$, step size $\gamma > 0$, norm $p \in \mathbb{N} \cup \{\infty\}$, maximal iterations $K \in \mathbb{N}$

1: $x_0 = x$
2: **for** $k = 0$ to $K - 1$ **do**
3: compute $(x_k)_{nsd}^p$
4: $z_k = x_k - \gamma (x_k)_{nsd}^p$
5: $x_{k+1} = \Pi_{x,\epsilon}^p(z_k)$
6: **end for**
7: $\tilde{x} = x_K$

Here, $(x_k)^p_{\text{nsd}} = \arg\min\{\nabla L(h(x_k))^T v : \|v\|_p = 1\}$, and we recall Lemma 1.4.1 for the specific solutions with respect to the different ℓ^p-norms. This method is known as the Projected Gradient Descent Attack (PGD). Although it's not exactly the traditional projected gradient descent, we will use this terminology as it is widely used in the literature. However, for $p = 2$ it is similar to the standard projected gradient descent with a normalized gradient. For norms other than $p = 2$, the optimization method varies slightly.

The Projected Gradient Descent Attack, introduced by Madry et al. [16], is one of the most powerful adversarial methods known in the literature. Due to its iterative nature, it significantly outperforms the FGSM but is also computationally more expensive.

Several other methods for generating adversarial examples include the Jacobian-based Saliency Map Attack (JSMA) [21], which iteratively manipulates the most influential pixel of an image to fool the model. Additionally, the Carlini and Wagner (CW) attack ([5] and the DeepFool attack [18] are known for their effectiveness, although the first is computationally very expensive. Since these attacks are all gradient-based, we would also like to mention an attack that is not: the so-called Decision Boundary Attack [3], which finds adversarial examples by iteratively adjusting an input along the decision boundary without using the gradient.

No single attack with a specific norm can reliably fool all models across all images. According to Schott et al. [26], different attacks and norms may be optimal for different samples and models. This suggests that the effectiveness of an adversarial attack can depend significantly on both the characteristics of the model and the specific sample being targeted.

In addition to finding adversarial examples that fool the model in some way, it is also possible to create targeted adversarial examples [29]. In this approach, we aim for a specific target label $\tilde{y} \neq y$, looking for an adversarial example $\tilde{x} \in X$ such that $h(x) = y$ and $h(\tilde{x}) = \tilde{y}$. For instance, the adversarial example of the stop sign shown in Figure 1.1 illustrates such a targeted attack.

2 Adversarial Training

Now that we understand how to create adversarial examples, it is crucial to explore methods for defending against them. One of the most important defense techniques is adversarial training, which was first introduced by Goodfellow et al. [10]. The concept behind adversarial training is simple: in each training iteration, the training set is replaced with adversarial examples generated from attacks on the model. It has been shown by Madry et al. [16] that adversarial training can be formulated as a min-max problem. To derive this formulation, we first need to introduce a risk framework for adversarial examples.

2.1. Adversarial Risk

First, we introduce the standard risk, which is crucial in the field of machine learning:

Definition 2.1.1. Let $\mathbb{R}^d$ be the input and Y be the label space. Additionally, let D be a distribution on $\mathbb{R}^d \times Y$ and let $L : Y \times Y \to \mathbb{R}$ be a loss function. For a classifier $h \in \mathcal{H}$, we call

$$R(h) = \mathbb{E}_{(x,y)\sim D}[L(h(x), y)]$$

the risk of h.

Since the distribution D is typically unknown, the risk cannot be computed directly. However, by drawing a sample of $m \in \mathbb{N}$ observations, which are i.i.d. drawn from D, we can estimate the risk using the so-called empirical risk:

Definition 2.1.2. Let $m \in \mathbb{N}$, $L : Y \times Y \to \mathbb{R}$ be a loss function, and $S = \{(x_i, y_i)\}_{i=1}^m \in (\mathbb{R}^d \times Y)^m$ be a sample. For a classifier $h \in \mathcal{H}$ we define the empirical risk of h as:

$$\widehat{R}(h) = \frac{1}{m} \sum_{i=1}^{m} L(h(x_i), y_i).$$

First, note that the following holds:

$$\mathbb{E}_{S\sim D^m}\left[\widehat{R}(h)\right] = R(h).$$

This indicates that the empirical risk is an unbiased estimator of the true risk. Additionally, according to the weak law of large numbers, the sample mean of an i.i.d. sample from a random variable converges in probability to its expected value. This motivates the approach of finding a good classifier by minimizing the empirical risk.

Definition 2.1.3. Let $\mathcal{H}$ be a hypothesis set. Then we call $h^* \in \mathcal{H}$ such that

$$\widehat{R}(h^*) = \min_{h \in \mathcal{H}} \widehat{R}(h) \tag{2.1}$$

an empirical risk minimizer.

Note that an empirical risk minimizer may not always exist, and if it does, it may not be unique. As an alternative to the traditional risk, we can also consider the adversarial risk. Instead of evaluating the loss at each sample directly, we focus on the worst-case loss within a specified region around each sample point. Similar to the definition of adversarial examples, we introduce a set of allowed perturbations $P \subset X$. To ensure that the maximum exists, we assume that P is compact and restrict our hypothesis set to consist only of continuous functions.

Definition 2.1.4. Let $\mathbb{R}^d$ be the input and Y be the label space. Further, let D be a distribution on $\mathbb{R}^d \times Y$, $L : Y \times Y \to \mathbb{R}$ a continuous loss function, and let the set of allowed perturbations $P \subset \mathbb{R}^d$ be compact. For a classifier $h \in \mathcal{H}$, we define the adversarial risk of h as:

$$R_{\text{adv}}(h) = \mathbb{E}_{(x,y)\sim D}\left[\max_{\delta \in P} L(h(x+\delta), y)\right].$$

The inner term of the adversarial risk corresponds to Equation (1.2), where we aimed to create an adversarial example.
Now we can also formulate an empirical version of the adversarial risk.

Definition 2.1.5. Let $m \in \mathbb{N}$, let $L : Y \times Y \to \mathbb{R}$ be a continuous loss function, $P \subset \mathbb{R}^d$ a compact set and let $S = \{(x_i, y_i)\}_{i=1}^m \subset (\mathbb{R}^d \times Y)^m$ be a sample. For $h \in \mathcal{H}$, we define

$$\widehat{R}_{\text{adv}}(h) = \frac{1}{m}\sum_{i=1}^{m} \max_{\delta \in P} L(h(x_i+\delta), y_i).$$

and call it the empirical adversarial risk of h.

The next step is to find a solution that minimizes the empirical adversarial risk in order to obtain a more robust classifier:

Definition 2.1.6. Let $\mathcal{H}$ be a hypothesis set. Then we call $h^* \in \mathcal{H}$ such that

$$\widehat{R}_{\text{adv}}(h^*) = \min_{h \in \mathcal{H}} \widehat{R}_{\text{adv}}(h) \tag{2.2}$$

an adversarial empirical risk minimizer.

To find such an adversarial empirical risk minimizer, we need to consider the setting where we have a parameterized hypothesis as defined in Section 1.1.

Definition 2.1.7. Let $m \in \mathbb{N}$, let $L : Y \times Y \to \mathbb{R}$ be a continuous loss function, $P \subset \mathbb{R}^d$ a compact set and let $S = \{(x_i, y_i)\}_{i=1}^m \subset (\mathbb{R}^d \times Y)^m$ be a sample. For a compact parameter set Θ, we define the problem

$$\min_{\theta \in \Theta} \frac{1}{m} \sum_{i=1}^{m} \max_{\delta \in P} L(h_\theta(x_i + \delta), y_i). \tag{2.3}$$

This problem of finding such a minimizer has a nice interpretation. Finding the maximum in the inner term represents the attacker's perspective of identifying the worst-case perturbation to fool a given classifier. Conversely, the outer minimization problem aims to find a model that is robust to such worst-case perturbations. Thus, by finding an adversarial empirical risk minimizer, we obtain a robust classifier.

However, this task is quite challenging due to the non-convex nature of the outer minimization problem and the inner non-concave maximization problem. Assuming that both our classifier and the loss function are continuously differentiable, it seems reasonable to approach the outer with a gradient based optimization method. To do this, we first need to compute the gradient with respect to the inner problem. Considering a single sample $x \in \mathbb{R}^d$ and $y \in Y$, we have:

$$\nabla_\theta \max_{\delta \in P} \underbrace{L(h_\theta(x + \delta), y)}_{=:g(\theta,\delta)}. \tag{2.4}$$

It is important to note that this gradient does not generally exist. In the adversarial training literature, it has become common practice to compute the gradient function at a maximizer of the inner problem. This procedure is theoretically supported by Madry et al. [16], but it was recently disproved by Latorre et al. [14], who showed that the classical Theorem of Danskin was used incorrectly.

In the following, we will state Danskin's Theorem and provide a counterexample of its application to the gradient (2.4). To do so, we need to define the one-sided directional derivative of a function, since the error lies in the incorrect definition of this derivative.

Definition 2.1.8. Let $\phi : \mathbb{R}^d \to \mathbb{R}$. For a nonzero vector $\gamma \in \mathbb{R}^d$, the one-sided directional derivative of ϕ in the direction γ at the point θ is defined as the one-sided limit:

$$D_\gamma \phi(\theta) := \lim_{t \to 0^+} \frac{\phi(\theta + t\gamma) - \phi(\theta)}{t\|\gamma\|_2}. \tag{2.5}$$

In the paper by Madry et al. [16], it was assumed that the limit in Danskin's Theorem was taken as a two-sided limit. However, Danskin's Theorem actually considers the one-sided directional derivative as follows:

Theorem 2.1.9 (Danskin [6]). *Let P be a compact topological space, and let $g : \mathbb{R}^d \times P \longrightarrow \mathbb{R}$ be a continuous function such that $g(\cdot, \delta)$ is differentiable for all $\delta \in P$, and*

$\nabla_\theta g(\theta, \delta)$ is continuous on $\mathbb{R}^d \times P$. Define

$$\phi(\theta) := \max_{\delta \in P} g(\theta, \delta), \quad P^*(\theta) := \arg\max_{\delta \in P} g(\theta, \delta).$$

Let $\gamma \in \mathbb{R}^d$ with $\|\gamma\|_2 = 1$ be an arbitrary unit vector. The one sided directional derivative $D_\gamma \phi(\theta)$ of ϕ in the direction γ at the point θ exists and is given by the formula

$$D_\gamma \phi(\theta) = \max_{\delta \in P^*(\theta)} \langle \gamma, \nabla_\theta g(\theta, \delta) \rangle.$$

All we know from Danskin's theorem is that the one-sided directional derivative exists. However, it is generally not true that $-D_\gamma \phi(\theta) = D_{-\gamma} \phi(\theta)$. This was mistakenly assumed in the following corollary, which is therefore incorrect:

Corollar 2.1.10. *[16] Let $\delta^* \in P^*(\theta)$. If $-\nabla_\theta g(\theta, \delta^*) \neq 0$, then it is a descent direction for ϕ at θ.*

To disprove this statement, we will use the counterexample provided by Latorre et al. [14].

Example 2.1.11. Let $P := [0, 1]$ and let $u, v \in \mathbb{R}^2$ be unit vectors such that $-1 < \langle u, v \rangle < 0$. That is, u and v form an obtuse angle. Let

$$g(\theta, \delta) = \delta \langle \theta, u \rangle + (1 - \delta) \langle \theta, v \rangle + \delta(\delta - 1).$$

Then there exists an $\theta \in P$ such that $-\nabla_\theta g(\theta, \delta^*)$ is for all $\delta^* \in P^*(\theta)$ an ascent direction and there exists an $\gamma \in \mathbb{R}^d$, which is a descent direction.

Proof. It is clear, that all conditions of Theorem 2.1.9 are satisfied. We take $\theta = 0$ and get $P^*(0) = \arg\max_{\delta \in [0,1]} \delta(\delta - 1) = \{0, 1\}$. For $\delta = 0$, we have $\nabla_\theta g(\theta, 0) = \nabla_\theta h(\theta, v) = v$ and at $\delta = 1$ we get $\nabla_\theta g(\theta, 1) = \nabla_\theta h(\theta, u) = u$.
Now we compute the value of the directional derivatives in the negative direction of such vectors. According to Danskin's Theorem, we have

$$D_{-v} \phi(0) = \max_{\delta \in \{0,1\}} \langle -v, \nabla_\theta g(\theta, \delta) \rangle = \max(\langle -v, v \rangle, \langle -v, u \rangle) \geq -\langle v, u \rangle > 0$$

where $-\langle v, u \rangle > 0$ holds by construction. Analogously, we have

$$D_{-u} \phi(0) = \max_{\delta \in \{0,1\}} \langle -u, \nabla_\theta g(\theta, \delta) \rangle = \max(\langle -u, u \rangle, \langle -u, v \rangle) \geq -\langle u, v \rangle > 0.$$

That means that all such directions are ascent directions. However, for the direction $\gamma = -(u + v)$, we have

$$D_\gamma \phi(\theta) \propto \max_{\delta \in \{0,1\}} \langle -(u + v), \nabla_\theta g(\theta, \delta) \rangle = \max(\langle -u - v, u \rangle, \langle -u - v, v \rangle) = -1 - \langle u, v \rangle < 0$$

where the last inequality holds because $-1 < \langle u, v \rangle < 0$. Hence, $-(u + v)$ is a descent direction. □

In summary, even if we successfully find the maximum in the inner problem, moving in the direction of $-\nabla_\theta g(\theta, \delta^*)$ might still result in an increase in adversarial risk. According to Danskin's Theorem, to find a true descent direction, we must consider all possible directions. Specifically, we need to solve the following optimization problem to determine the steepest descent direction:

$$\min_{\gamma:\|\gamma\|_2=1} \max_{\delta \in P^\star(\theta)} \langle \gamma, \nabla_\theta g(\theta, \delta) \rangle.$$

However, for a potentially infinite set $P^\star(\theta)$, this problem is extremely challenging and is rarely used in practice.

2.2. Adversarial Training

Given the immense success of adversarial training in practice, we will follow the approach proposed by Madry et al. [16]. Therefore, we will assume that the gradient (2.4) exists for all $\theta \in \Theta$ and for all samples $(x, y) \in S$. With this strong assumption, Corollary 2.1.10 holds. Specifically, for a sample set S consisting of a single point $(x, y) \in \mathbb{R}^d \times Y$, where

$$x^* = x + \arg\max_{\delta \in P} L(h_\theta(x + \delta), y)$$

the gradient

$$-\nabla_\theta L(h_\theta(x^*), y)$$

is a descent direction for $\widehat{R}_{\text{adv}}(h_\theta)$ if it is non-zero.

To find a potential adversarial example x^*, we can use the Projected Gradient Descent (PGD) attack (Algorithm 3). This leads to the following algorithm, which aims to produce a classifier that is robust to adversarial examples.

Algorithm 4 Adversarial Training [16]

Require: samples $S = \{(x^i, y^i)\}_{i=1}^m$, initial model parameters $\theta_0 \in \Theta$, maximal perturbation $\epsilon > 0$, PGD step size $\gamma > 0$, step size $\eta > 0$, maximal iterations PGD $K \in \mathbb{N}$, maximal iterations training $T \in \mathbb{N}$, norm $p \in \mathbb{N} \cup \{\infty\}$

1: **for** $t = 0$ to $T - 1$ **do**
2: **for** $i = 1$ to m **do**
3: Generate potential adversarial examples x_i^* with PGD attack 3
4: **end for**
5: $\theta_{t+1} = \theta_t - \eta \frac{1}{m} \sum_i^m \nabla_\theta L(h_{\theta_t}(x_i'), y_i)$
6: **end for**

In practice, the ℓ^∞-norm is commonly used for adversarial training. Note that in Step 3, we cannot be certain that we have created an adversarial example. This is because the PGD attack does not guarantee that an adversarial example will always be found.

We can also reformulate this algorithm to process in batches, which eliminates the need to compute the entire gradient in Step 5 during each iteration.

Additionally, in Step 3, we can replace the process of generating potential adversarial examples with alternative attack methods. For instance, instead of using PGD, we could use FGSM, JSMA, or DeepFool to create the adversarial examples. However, a model trained with attack A may not be robust against attacks of type B. For example, a model trained using FGSM may not perform well when attacked with PGD, as we will demonstrate later. Schott et al. [26] also assume that adversarial training can overfit to the chosen norm and may not be robust to attacks using other norms.

Adversarial training is known to be computationally expensive [32]. The number of gradient computations required in each training iteration scales with $\mathcal{O}(mK)$, where m is the size of the training set and K denotes the number of iterations used for the PGD attack. In contrast, standard training requires only $\mathcal{O}(m)$ gradient computations per iteration, making adversarial training approximately K times slower.

To address this issue, Shafahi et al. [27] introduced "free adversarial training". The core idea is to improve efficiency by combining the computation of the ascent direction for adversarial perturbations with the descent computation for updating network parameters. This is achieved by processing data in batches and integrating these computations into a single backward pass, rather than performing separate forward and backward passes for each adversarial perturbation within a batch. As a result, free adversarial training is up to 10 times faster than standard adversarial training while maintaining comparable robustness.

Another significant point is that the effectiveness of the algorithm depends strongly on the allowed perturbation ϵ. An attack with a slightly larger perturbation than the one used during adversarial training can significantly degrade the accuracy of the model, as we will show in the next section.

2.3. Experimental Results

In the following experiments, we aim to evaluate the effectiveness of the various adversarial attacks and the robustness properties of adversarial training. Note that we will only focus on white box attacks. This means that the adversary will always have direct access to the model. The code for the experiments is available at https://github.com/j-schuhmann/Adversarial-Examples-in-Machine-Learning-Training-and-Certified-Robustness.

For the experiments, we will use the MNIST dataset, which consists of handwritten digits from 0 to 9. The set includes a training set of 60,000 images and a test set of 10,000 images. Due to computational constraints, we will only use a subset of 10,000 images from the training set for our experiments. For the classification, we will use a

simple convolutional neural network. The architecture includes two convolutional layers, with 32 and 64 filters. Each of the convolutional layers is followed by a 2x2 max pooling layer, which aims to reduce the spatial dimensions. After that a fully connected layer of size 1024 is applied. When trained normally, our network achieves an accuracy of $98,4\%$ on the test set.

First, we will evaluate the effectiveness of the FGSM (Algorithm 2) and PGD (Algorithm 3) attacks on our normally trained model. To assess these attacks, we will apply various perturbations in the ℓ^∞-norm. The PGD attack will be tested with 10 iterations, using a step size of $2.5 \cdot \epsilon/10$. This configuration ensures that the attack can reach the boundary of B_ϵ^∞ and make several iterations at that boundary. The plot below illustrates the accuracy of the trained model when tested on images perturbed by the FGSM and PGD attacks.

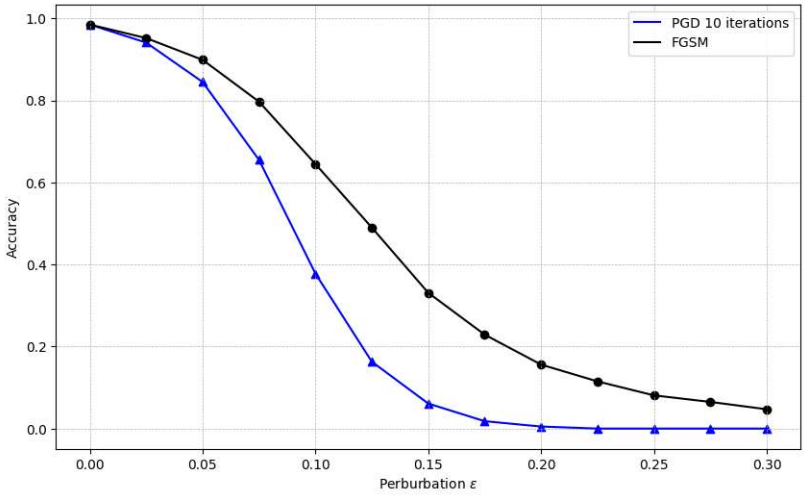

Figure 2.1: ℓ^∞-adversarial attacks on the normal trained model

We can see that even a small perturbation of 0.1 drastically reduces the model's accuracy when attacked with PGD. As already assumed, the PGD attack significantly outperforms FGSM.

For adversarial training (Algorithm 4) with FGSM, we train our net against an adversary using a perturbation of 0.3 in the ℓ^∞-norm. We then evaluate the performance of the adversarially trained model both on the unperturbed test set images and on different perturbed test images. These perturbed examples are created using FGSM and PGD with $\epsilon = 0.3$ in the ℓ^∞-norm. The following table presents the respective accuracies, including results with different numbers of iterations for the PGD attack:

Method	Iterations	Accuracy
Natural	-	96,1%
FGSM	-	77,3%
PGD	10	0,0%
PGD	40	0,0%

Table 2.1: Performance of the with FGSM adversarially trained model against different adversaries for a perturbation of 0.3 in the ℓ^∞-norm

When adversarially training the model with FGSM adversarial examples, a significant issue becomes immediately apparent: the model lacks robustness against PGD attacks. While it shows some resilience against FGSM attacks, its robustness against PGD attacks is no better than that of the naturally trained model. We also note that the so-called label leaking effect was not observed in our case. This effect was first observed by Kurakin et al. [13] and describes a phenomenon where FGSM generates a very restricted set of adversarial examples that a model can overfit. According to the authors, the model appears to learn the simple and predictable transformations introduced by the one-step method. We believe that the absence of the label leaking effect in our case can be attributed to our relatively small dataset, which contains only 10,000 images—insufficient for the model to learn this phenomenon. In conclusion, adversarially training a model using FGSM perturbed images is generally not recommended.

We now proceed to adversarially train our network using PGD-perturbed images. This training involves 10 iterations of PGD with a step size of $2.5 \cdot 0.3/10$, and similar to the previous experiment, uses a perturbation of 0.3 in the ℓ^∞-norm. On our test set, this PGD adversarially trained model achieves an accuracy of $96,2\%$. We again evaluate the model against different perturbed examples and obtain the following results:

Method	Iterations	Accuracy
Natural	-	96.2%
FGSM	-	86.3%
PGD	10	79.3%
PGD	40	78.3%
PGD	100	78.1%

Table 2.2: Performance of the with PGD adversarially trained model against different adversaries for a perturbation of 0.3 in the ℓ^∞-norm

We observe that our model is relatively robust against PGD and FGSM attacks. However, there remains a discrepancy of more than 10% in performance compared to the unperturbed images. This discrepancy is probably due to the relatively small training set of 10,000 images. In the original paper by Madry et al. [16], this issue does not occur, as they used the full training set of 60000 images. We also note that increasing the number of PGD iterations does not significantly affect the accuracy.

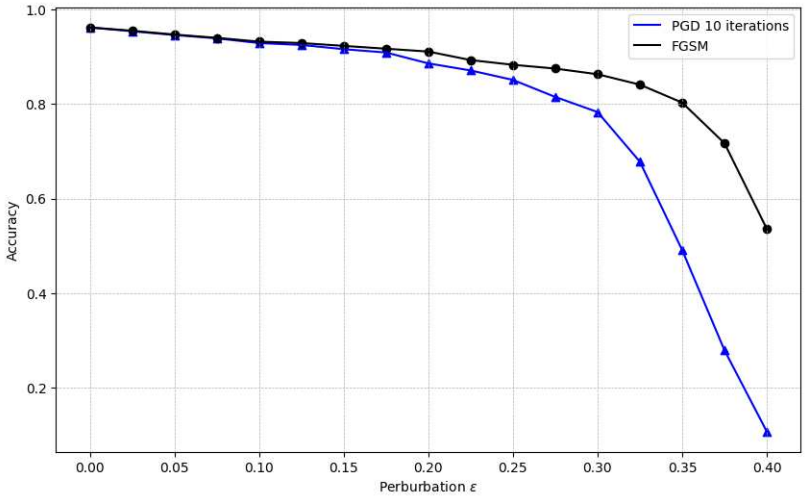

Figure 2.2: ℓ^∞-adversarial attacks on the with $\epsilon = 0.3$ in ℓ^∞-norm PGD adversarially trained model

Furthermore we observe that our adversarially trained model achieves only 96.2% accuracy on the unperturbed test images, compared to 98.4% for the naturally trained model. This phenomenon, where adversarially trained models perform worse on the test set than naturally trained models, is well-documented. The authors in [30] even suggest that there may be an inherent trade-off between standard accuracy and adversarial robustness of a model.

We will now evaluate the robustness of our PGD adversarially trained model against various ϵ, i.e. different perturbations. Figure 2.2 illustrates that the model remains robust with only a small drop in accuracy when the perturbation ϵ is at or below 0.3. However, once the perturbation exceeds 0.3, the model's performance declines sharply. This poses a significant concern, as an adversary could exploit this vulnerability by attacking the adversarially trained model with a perturbation of $\epsilon' = \epsilon + 0.05$ (e.g., in the MNIST dataset), making a successful attack highly probable.

Now we want to test what happens when we attack our with the ℓ^∞-norm adversarilly trained model with a different norm. We will use the PGD attack again, but attack this time with the ℓ^2-norm. The step size will be set to $2.5 \cdot \epsilon/10$ and we will run 10 iterations.

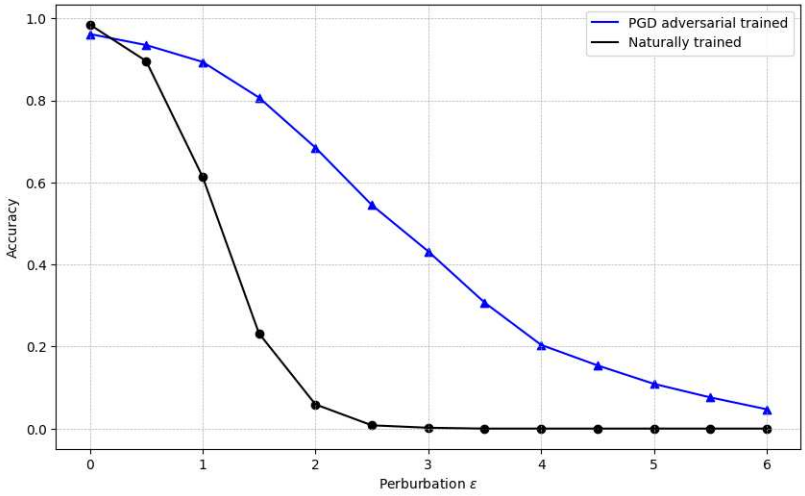

Figure 2.3: ℓ^2-adversarial attacks on the natural trained and the with $\epsilon = 0.3$ in ℓ^∞-norm PGD adversarially trained model

In Figure 2.3, we observe that the PGD adversarially trained model appears to be relatively robust against ℓ^2-PGD attacks, but it does not show a sharp drop in accuracy like the ℓ^∞-attacks do once a certain threshold is crossed. Furthermore, according to Li et al. [15] and Schott et al. [25], the PGD ℓ^2-attack tends to overestimate the ℓ^2-robustness of the model. By using the Decision Boundary Attack from Brendel et al. [3], it has been demonstrated that the ℓ^∞-adversarially trained model shows only limited robustness against ℓ^2-attacks. However this weakness is not observed when the ℓ^∞-PGD adversarially trained model faces ℓ^∞-attacks.

We also aim to investigate the computational time required for adversarial training. In PGD adversarial training, each image undergoes an iterative calculation during every epoch, resulting in significantly longer training times compared to standard training. All experiments were conducted on an Intel(R) Core(TM) i5-7200U CPU @ 2.50GHz. The following table presents the training times for standard training, FGSM adversarial training, and PGD adversarial training.

Method	Time
Natural	9m 29.3s
FGSM	91m 58.4s
PGD	983m 22.8s

Table 2.3: Computational time for standard training, FGSM adversarial training, and PGD adversarial training

Adversarial training is significantly more time-consuming than natural training. For example, using 10 iterations of PGD can increase computational time by a factor of ap-

proximately 100. Using the FGSM for adversarial training reduces this factor to about 10, but the training time remains significantly high.

These longer training times are consistent with the results of other researchers. For example, Madry et al. [16] reported that adversarially training a model on the CIFAR-10 dataset took over four days on a Titan X GPU, even with only 7 iterations of PGD.

3 Risk Bounds

In the following chapter, we will introduce surrogate loss functions and investigate the corresponding risk bounds. We will use this theory to establish bounds on the adversarial risk. Subsequently, we will present adversarial training using TRADES, which has been shown to be more robust than standard adversarial training.

3.1. Risk Bounds via Surrogate Loss Functions

This section, including its theorems and proofs, closely follows the work of Bartlett et al. [1], which established a general relationship between the risk using the 0-1 loss and the risk using any nonnegative surrogate loss function. In the following section, we will assume that $X \subset \mathbb{R}^d$.

We will focus on the binary classification task, where we have a distribution over $X \times \{-1, 1\}$, with X representing the input space. We also define $\text{sign}(0) = -1$. We will use the 0-1 loss along with a function $\Phi : X \longrightarrow \mathbb{R}$, which is assumed to be measurable. In this setup, our classifier $h : X \to \{-1, 1\}$ is defined as $h(x) = \text{sign}(\Phi(x))$.

Note that, in the case of the 0-1 loss, the risk is euqal to the probability of misclassification, i.e.,

$$R(h) = \mathbb{E}_{(x,y)\sim D}[\mathbb{1}(h(x) \neq y)] = \mathbb{P}_{(x,y)\sim D}(h(x) \neq y).$$

We define the function $\eta : X \longrightarrow [0, 1]$ with

$$\eta(x) = \mathbb{P}_{(x,y)\sim D}(y = 1|x) \quad \text{for almost all } x.$$

Note that $\eta(x)$ is only defined almost everywhere w.r.t. the marginal distribution of x.

Definition 3.1.1. Given a distribution D over $X \times Y$, the Bayes error R^* is defined as the infimum of the errors achieved by measurable functions $h : X \to Y$:

$$R^* = \inf_{h \text{ measurable}} R(h).$$

A hypothesis h with $R(h) = R^*$ is called a Bayes classifier.

Then for our binary classifiaction task, we get the following Bayes classifier:

Proposition 3.1.2. *For all classifiers $h : X \longrightarrow \{1, -1\}$ the function*

$$h_{bayes}(x) := \begin{cases} 1 & \text{if } \eta(x) > \frac{1}{2} \\ -1 & \text{otherwise} \end{cases}$$

satiesfies $R(h_{bayes}) \leq R(h)$.

Proof. Fix $x \in X$ and take any arbitrary function h. Using the identity $\mathbb{1}(h(x) = -1) = 1 - \mathbb{1}(h(x) = 1)$, we obtain:

$$\begin{aligned}
\mathbb{P}_{(x,y)\sim D}(y \neq h(x)|x) &= 1 - \mathbb{P}_{(x,y)\sim D}(y = h(x)|x) \\
&= 1 - \left[\mathbb{P}_{(x,y)\sim D}(y = 1, h(x) = 1|x) + \mathbb{P}_{(x,y)\sim D}(y = -1, h(x) = -1|x)\right] \\
&= 1 - \left[\mathbb{E}_{(x,y)\sim D}[\mathbb{1}(y = 1)\mathbb{1}(h(x) = 1)|x] + \mathbb{E}_{(x,y)\sim D}[\mathbb{1}(y = -1)\mathbb{1}(h(x) = -1)|x]\right] \\
&= 1 - \left[\mathbb{1}(h(x) = 1)\mathbb{E}_{(x,y)\sim D}[\mathbb{1}(y = 1)|x] + \mathbb{1}(h(x) = -1)\mathbb{E}_{(x,y)\sim D}[\mathbb{1}(y = -1)|x]\right] \\
&= 1 - \left[\mathbb{1}(h(x) = 1)\mathbb{P}_{(x,y)\sim D}(y = 1|x) + \mathbb{1}(h(x) = -1)\mathbb{P}_{(x,y)\sim D}(y = -1|x)\right] \\
&= 1 - \left[\mathbb{1}(h(x) = 1)\eta(x) + \mathbb{1}(h(x) = -1)(1 - \eta(x))\right].
\end{aligned}$$

From this, we derive:

$$\begin{aligned}
&\mathbb{P}_{(x,y)\sim D}(y \neq h(x)|x) - \mathbb{P}_{(x,y)\sim D}(y \neq h_{\text{bayes}}(x)|x) \\
&= \eta(x)\left[\mathbb{1}(h_{\text{bayes}}(x) = 1) - \mathbb{1}(h(x) = 1)\right] + (1 - \eta(x))\left[\mathbb{1}(h_{\text{bayes}}(x) = -1) - \mathbb{1}(h(x) = -1)\right] \\
&= \eta(x)\left[\mathbb{1}(h_{\text{bayes}}(x) = 1) - \mathbb{1}(h(x) = 1)\right] - (1 - \eta(x))\left[\mathbb{1}(h_{\text{bayes}}(x) = 1) - \mathbb{1}(h(x) = 1)\right] \\
&= (2\eta(x) - 1)\left[\mathbb{1}(h_{\text{bayes}}(x) = 1) - \mathbb{1}(h(x) = 1)\right].
\end{aligned}$$

Next, we examine the different cases. When x is such that $\eta(x) > \frac{1}{2}$, we have:

$$\underbrace{(2\eta(x) - 1)}_{>0} \underbrace{\left(\underbrace{\mathbb{1}(h_{\text{bayes}}(x) = 1)}_{=1} - \underbrace{\mathbb{1}(h(x) = 1)}_{=1 \text{ or } =0}\right)}_{\geq 0}.$$

When x is such that $\eta(x) \leq \frac{1}{2}$, we obtain:

$$\underbrace{(2\eta(x) - 1)}_{\leq 0} \underbrace{\left(\underbrace{\mathbb{1}(h_{\text{bayes}}(x) = 1)}_{=0} - \underbrace{\mathbb{1}(h(x) = 1)}_{=1 \text{ or } =0}\right)}_{\leq 0}.$$

Thus, we conclude that:

$$\mathbb{P}_{(x,y)\sim D}(y \neq h(x)|x) - \mathbb{P}_{(x,y)\sim D}(y \neq h_{\text{bayes}}(x)|x) \geq 0.$$

Taking the expectation with respect to x, we receive:

$$R(h) - R(h_{\text{bayes}}) \geq 0.$$

□

Furthermore as $\text{sign}(0) = -1$ we can also write the Bayes classifier as

$$h_{\text{bayes}}(x) = \text{sign}(\eta(x) - 1/2). \tag{3.1}$$

Minimizing the 0-1 loss function directly is often challenging due to its non-convex nature. To overcome this difficulty, we use a surrogate loss function. A surrogate loss function serves as an alternative to the 0-1 loss, replacing it with a function that is easier to optimize. Typically, surrogate loss functions are chosen to be convex, which simplifies the optimization. In the following, let $\varphi : \mathbb{R} \longrightarrow [0, \infty)$ denote our chosen surrogate loss function. We will now define the risk associated with this surrogate loss function:

Definition 3.1.3. For a surrogate loss function $\varphi : \mathbb{R} \longrightarrow [0, \infty)$ and a function Φ we name

$$R_\varphi(\Phi) = \mathbb{E}_{(x,y)\sim D}(\varphi(y\Phi(x))$$

the φ-risk of Φ. We also define

$$R^*_\varphi = \inf_{\Phi \text{ measurable}} R_\varphi(\Phi)$$

as the optimal φ-risk.

Now we observe that

$$\mathbb{E}_{(x,y)\sim D}[\varphi(y\Phi(x)] = \mathbb{E}_{x\sim D}[\eta(x)\varphi(\Phi(x)) + (1 - \eta(x))\varphi(-\Phi(x)].$$

Our goal is to investigate the φ-risk of Φ at a specific point $x \in X$. It is clear that:

$$\mathbb{E}_{(x,y)\sim D}[\varphi(y\Phi(x)|x] = \eta(x)\varphi(\Phi(x)) + (1 - \eta(x))\varphi(-\Phi(x)).$$

To minimize the φ-risk of Φ at a specific point x, we should choose $\Phi(x)$ to pointwise optimize the right-hand side. For a fixed x, where $\eta = \eta(x)$ and $\alpha = \Phi(x)$ are also fixed, we define

$$H_\varphi(\eta) = \inf_{\alpha\in\mathbb{R}} \{\eta\varphi(\alpha) + (1 - \eta)\varphi(-\alpha)\}$$

as the optimal conditional φ-risk. We also define the inner part as

$$C_{\varphi,\eta} = \eta\varphi(\alpha) + (1 - \eta)\varphi(-\alpha).$$

Now, the optimal φ-risk can be written as

$$R^*_\varphi = \mathbb{E}_{x\sim D}[H_\varphi(\eta(x))]. \tag{3.2}$$

To illustrate, consider the following domain $X = \{x_0\}$. In this case, minimizing the ϕ-risk involves selecting a function Φ that minimizes the value $C_{\varphi,\eta(x_0)}(\Phi(x_0))$.
We want a classifier that has the same sign as the Bayes classifier defined in Proposition 3.1.2. That is, for some fixed x, with $\alpha := \Phi(x)$ and $\eta := \eta(x)$, we require

$$\text{sign}(\alpha) = \text{sign}(\Phi(x)) = h_{\text{bayes}}(x) = \text{sign}(\eta(x) - 1/2) = \text{sign}(\eta - 1/2),$$

which simply means that we want the following condition to be satisfied:

$$\alpha(\eta - 1/2) > 0.$$

With this, we define

$$H_\varphi^-(\eta) = \inf_{\substack{\alpha\in\mathbb{R} \\ \alpha(\eta-1/2)\leq 0}} C_{\varphi,\eta}(\alpha) = \inf_{\substack{\alpha\in\mathbb{R} \\ \alpha(\eta-1/2)\leq 0}} \{\eta\varphi(\alpha) + (1-\eta)\varphi(-\alpha)\}.$$

Thus, for a fixed point x, this represents the optimal value of the conditional φ-risk under the constraint that the sign of $\alpha = \Phi(x)$ does not match the sign of the Bayes classifier $(\eta - 1/2) = (\eta(x) - 1/2)$. Using this, we can formulate a basic condition on our surrogate loss function φ. With this we want to force a potential minimizer to have the correct sign:

Definition 3.1.4. A function $\varphi : \mathbb{R} \longrightarrow [0, \infty)$ is classification-calibrated if, for all $\eta \neq \frac{1}{2}$,

$$H_\varphi^-(\eta) > H_\varphi(\eta).$$

Consider the domain $X = \{x_0\}$ once again. We already know that minimizing the ϕ-risk involves selecting a function Φ that minimizes the value $C_{\varphi,\eta(x_0)}(\Phi(x_0))$. With a classification-calibrated surrogate loss, this minimization also requires that if $\Phi(x_0)$ does not align with the prediction of the Bayes classifier, it results in a strictly larger φ-risk.

We now introduce a transformation of our surrogate loss function φ as follows:

Definition 3.1.5. Given a loss function $\varphi : \mathbb{R} \longrightarrow [0, \infty)$, we define $\psi : [-1, 1] \longrightarrow [0, \infty)$, called the ψ-transform, by

$$\psi = \tilde{\psi}^{**},$$

where

$$\tilde{\psi}(\theta) = H_\varphi^-\left(\frac{1+\theta}{2}\right) - H_\varphi\left(\frac{1+\theta}{2}\right)$$

and $\tilde{\psi}^{**}$ denotes the Fenchel-Legendre biconjugate of ψ. For a function g, the biconjugate g^{**} is the unique function satisfying

$$\operatorname{Epi} g^{**} = \overline{\operatorname{conv}} \operatorname{Epi} g,$$

where $\operatorname{Epi} g = \{(r, s) : g(r) \leq s\}$ is the epigraph of g and $\overline{\operatorname{conv}}$ denotes the closed convex hull.

Note that g is convex if and only if $\operatorname{Epi} g$ is a convex set. Before we can state the main theorem of this section, we need to establish a few results regarding this transformation.

Lemma 3.1.6. *For any nonnegative loss function φ, the functions H_φ, H_φ^-, and ψ have the following properties:*

1. *H_φ and H_φ^- are symmetric about $1/2$, and ψ is symmetric about 0. For all $\eta \in [0,1]$,*
$$H_\varphi(\eta) = H_\varphi(1-\eta), \quad H_\varphi^-(\eta) = H_\varphi^-(1-\eta),$$
and
$$\psi(\eta) = \psi(-\eta).$$

2. *H_φ is concave, and for $0 \le \eta \le 1$, it satisfies*
$$H_\varphi(\eta) \le H_\varphi(1/2) = H_\varphi^-(1/2).$$

3. *H_φ^- is concave on $[0,1/2]$, and for all $\eta \in [0,1]$, it satisfies*
$$H_\varphi^-(\eta) \ge H_\varphi(\eta).$$

4. *H_φ and H_φ^- are continuous on $[0,1]$.*

5. *ψ and $\tilde{\psi}$ are continuous on $[-1,1]$.*

6. *ψ is nonnegative and minimal at 0.*

7. *$\psi(0) = 0$.*

8. *$\psi(\theta) > 0$ for all $\theta \in (0,1]$ if and only if φ is classification-calibrated.*

9. *If φ is classification-calibrated, then ψ is strictly increasing on $[0,1]$.*

Proof. 1) This follows directly from the definitions of H_φ and H_φ^-.

2) Since H_φ is defined as the infimum over concave functions of η, it is itself concave. Therefore,
$$H_\varphi\left(\frac{1}{2}\right) = H_\varphi\left(\frac{1}{2}\eta + \frac{1}{2}(1-\eta)\right) \ge \frac{1}{2}H_\varphi(\eta) + \frac{1}{2}H_\varphi(1-\eta) = H(\eta).$$
Furthermore, if $\eta = \frac{1}{2}$, then for any $\alpha \in \mathbb{R}$, we have $\alpha(\eta - \frac{1}{2}) \le 0$. This implies that $H_\varphi\left(\frac{1}{2}\right) = H_\varphi^-\left(\frac{1}{2}\right)$.

3) The concavity of H_φ^- is also evident, as it is the infimum over concave functions. From the definition, we know that $H_\varphi^-(\eta) \ge H_\varphi(\eta)$ for all $\eta \in [0,1]$.

4) Because concave functions are continuous on the relative interior of their domains, we know that H_φ is continuous on $(0,1)$. To prove continuity on $[0,1]$, it suffices to show that H_φ is left-continuous at 1, given the symmetry of H_φ. Applying Theorem 5.0.1 from Rockafellar [23] and noting that $[0,1]$ is locally simplicial, we conclude that the convex function $-H_\varphi$ is upper semicontinuous at 1, which implies that H_φ is lower semicontinuous at 1.

To demonstrate upper semicontinuity, fix $\epsilon > 0$. By the definition of H_φ, there exists an α_ϵ such that $\varphi(\alpha_\epsilon) \leq H_\varphi(1) + \epsilon/2$. Then, for all $\eta \in \left[1 - \frac{\epsilon}{2\varphi(-\alpha_\epsilon)}, 1\right]$, we have

$$H_\varphi(\eta) \leq C_{\varphi,\eta}(\alpha_\epsilon) = \eta\varphi(\alpha_\epsilon) + \underbrace{(1-\eta)}_{\leq \epsilon/(2\varphi(-\alpha_\epsilon))} \varphi(-\alpha_\epsilon) \leq H_\varphi(1) + \epsilon.$$

Since this holds for all $\epsilon > 0$, we get $\limsup_{\eta \to 1} H_\varphi(\eta) \leq H(1)$, establishing upper semicontinuity. Hence, H_φ is continuous on $[0,1]$. Using the same argument, we can also show the continuity of H_φ^- on $[0,1/2]$ and $[1/2,1]$, implying that H_φ^- is continuous on $[0,1]$.

5) From Part 4, the continuity of $\tilde{\psi}$ is clear. Since ψ is closed (i.e., its epigraph is closed) and convex, it must be lower semicontinuous. Applying Theorem 5.0.1 again, the claim follows.

6) From Parts 2 and 3, it is immediately clear that $\tilde{\psi}$ is nonnegative and minimal at 0. Furthermore, since $\operatorname{epi}(\psi)$ is the convex hull of $\operatorname{epi}(\tilde{\psi})$, ψ must also be nonnegative and minimal at 0.

7) This follows directly from Part 2.

8) Suppose that φ is classification-calibrated, meaning that for all $\theta \in (0,1]$, $\tilde{\psi}(\theta) > 0$. Assume $(\theta, 0) \in \operatorname{epi}(\psi)$; since the point is a convex combination of points in $\operatorname{epi}(\tilde{\psi})$, θ must be equal to 0. Therefore, for all $\theta \in (0,1]$, points in $\operatorname{epi}(\psi)$ of the form (θ, c) must have $c > 0$, and the closure of Epi $\tilde{\psi}$ (because $\tilde{\psi}$ is continuous see Remark 3.1.7) implies $\psi(\theta) > 0$. Conversely, if φ is not classification-calibrated, then there exists a $\theta > 0$ such that $\tilde{\psi}(\theta) = 0$, which implies $\psi(\theta) = 0$.

9) Consider $\psi : [0,1] \to \mathbb{R}$. Since the epigraph of ψ is convex, ψ itself must be convex. Parts 7 and 8 establish that $\psi(0) = 0$ and $\psi(\theta) > 0$ for all $\theta \in (0,1]$. If $x > y \in [0,1]$, there exists $\lambda \in [0,1)$ such that $y = \lambda x$. Thus,

$$\psi(y) = \psi(\lambda x) = \psi(\lambda x + (1-\lambda)\cdot 0) \leq \lambda\psi(x) + (1-\lambda)\psi(0) = \lambda \underbrace{\psi(x)}_{>0} < \psi(x).$$

□

Since the non-negativity was established in Part 6), it follows that ψ is well defined.

Remark 3.1.7. The convex hull of a function g is defined as

$$\operatorname{conv} g = \inf\{s : (x,s) \in \operatorname{conv}(\operatorname{epi} g)\}.$$

It can be shown [23] that, if g is lower semicontinuous, then Epi g is a closed set. This implies that

$$g^{**} = \operatorname{conv} g.$$

Since $\tilde{\psi}$ is continuous by Lemma 3.1.6, it follows that

$$\psi = \operatorname{conv} \tilde{\psi}.$$

With the result from the previous lemma, we are now ready to prove the main theorem.

Theorem 3.1.8. *For any nonnegative loss function $\varphi : \mathbb{R} \longrightarrow [0, \infty)$, any function $\Phi : X \longrightarrow \mathbb{R}$, and any distribution D on $X \times \{-1, 1\}$, it holds for the classifier $h(x) = \operatorname{sign}(\Phi(x))$:*

1. $$\psi(R(h) - R^*) \leq R_\varphi(\Phi) - R_\varphi^*.$$

2. *If φ is classification-calibrated*

$$R(h) - R^* \leq \psi^{-1}(R_\varphi(\Phi) - R_\varphi^*).$$

Proof. 1) First we consider the Bayes classifier like in (3.1) and notice:

$$\begin{aligned} R(h) - R^* &= R(h) - R(h_{\text{bayes}}) \\ &= \mathbb{E}_{x\sim D}\left(\mathbb{1}\left[\operatorname{sign}(\Phi(x)) \neq \operatorname{sign}(\eta(x) - 1/2)\right] |2\eta(x) - 1|\right). \end{aligned}$$

Now by applying Jensen's inequality, because ψ is by definition convex, and using from Lemma 3.1.6 that $\psi(0) = 0$, we get

$$\begin{aligned} \psi(R(h) - R^*) &= \psi(\mathbb{E}_{x\sim D}\left(\mathbb{1}\left[\operatorname{sign}(\Phi(x)) \neq \operatorname{sign}(\eta(x) - 1/2)\right] |2\eta(x) - 1|\right) \\ &\leq \mathbb{E}_{x\sim D}\left(\psi(\mathbb{1}\left[\operatorname{sign}(\Phi(x)) \neq \operatorname{sign}(\eta(x) - 1/2)\right] |2\eta(x) - 1|)\right) \\ &= \mathbb{E}_{x\sim D}\left(\mathbb{1}\left[\operatorname{sign}(\Phi(x)) \neq \operatorname{sign}(\eta(x) - 1/2)\right] \psi(|2\eta(x) - 1|)\right). \end{aligned}$$

Furthermore by using $\psi(\theta) \leq \tilde{\psi}(\theta)$ for all $\theta \in [-1, 1]$ we get:

$$\begin{aligned} \psi(R(h) - R^*) &\leq \mathbb{E}_{x\sim D}\left(\mathbb{1}\left[\operatorname{sign}(\Phi(x)) \neq \operatorname{sign}(\eta(x) - 1/2)\right] \tilde{\psi}(|2\eta(x) - 1|)\right) \\ &= \mathbb{E}_{x\sim D}\left(\mathbb{1}\left[\operatorname{sign}(\Phi(x)) \neq \operatorname{sign}(\eta(x) - 1/2)\right] \left(H_\varphi^-(\eta(x)) - H_\varphi(\eta(x))\right)\right) \\ &= \mathbb{E}_{x\sim D}\left(\mathbb{1}\left[\operatorname{sign}(\Phi(x)) \neq \operatorname{sign}(\eta(x) - 1/2)\right] \left(\inf_{\substack{\alpha\in\mathbb{R} \\ \alpha(\eta-1/2)\leq 0}} C_{\varphi,\eta(x)}(\alpha) - H_\varphi(\eta(x))\right)\right) \\ &\leq \mathbb{E}_{x\sim D}\left(C_{\varphi,\eta(x)}(\Phi(x)) - H_\varphi(\eta(x))\right) \\ &= R_\varphi(\Phi) - R_\varphi^*. \end{aligned}$$

2) Lemma 3.1.6 Part 9) tells us that ψ is strictly increasing, so it has to be invertible and the claim follows.

□

Next, we will demonstrate that the bound given in Theorem 3.1.8 Part 1) is indeed tight.

Theorem 3.1.9. *Suppose that $|X| \geq 2$. For any nonnegative loss function φ, any $\epsilon > 0$, any $\theta \in [0,1]$, there exists a probability distribution D over $X \times \{-1,1\}$ and a function $\Phi : X \longrightarrow \mathbb{R}$ with a classifier $h(x) = sign(\Phi(x))$ such that*

$$R(h) - R^* = \theta$$

and

$$\psi(\theta) \leq R_\varphi(\Phi) - R_\varphi^* \leq \psi(\theta) + \epsilon.$$

Proof. The first inequality follows directly from Theorem 3.1.8. For the other one, we first pick a $\epsilon > 0$ and $\theta \in [0,1]$. We know that $\operatorname{epi}\psi = \overline{\operatorname{conv}}(\operatorname{epi}\tilde{\psi})$, so we can choose $\gamma, \alpha_1, \alpha_2 \in [0,1]$, for which $\theta = \gamma\alpha_1 + (1-\gamma)\alpha_2$ and $\psi(\theta) \geq \gamma\tilde{\psi}(\alpha_1) + (1-\gamma)\tilde{\psi}(\alpha_2)$. Now we choose distinct $x_1, x_2 \in X$ and define $\mathbb{P}_{x\sim D}$ such that $\mathbb{P}_{x\sim D}(\{x_1\}) = \gamma$, $\mathbb{P}_{x\sim D}(\{x_2\}) = 1-\gamma$, $\eta(x_1) = (1+\alpha_1)/2$ and $\eta(x_2) = (1+\alpha_2)/2$. Further from the definition of H_φ^-, we can choose clearly a function $\Phi : X \longrightarrow \mathbb{R}$ such that, $\Phi(x_1) \leq 0$, $\Phi(x_2) \leq 0$, $C_{\varphi,\eta(x_1)}(\Phi(x_1)) \leq H_\varphi^-(\eta(x_1)) + \epsilon/2$ and $C_{\varphi,\eta(x_2)}(\Phi(x_2)) \leq H_\varphi^-(\eta(x_2)) + \epsilon/2$. To see that, just notice that $\Phi(x_1)(2\eta(x_1) - 1) = \Phi(x_1)\alpha_1 \leq 0$. With that and (3.2) we get:

$$\begin{aligned}
R_\varphi(\Phi) - R_\varphi^* &= \mathbb{E}_{(x,y)\sim D}(\varphi(y\Phi(x)) - \inf_g \mathbb{E}_{(x,y)\sim D}(\varphi(yg(x)) \\
&= \mathbb{E}_{x\sim D}\left[C_{\varphi,\eta(x)}(\Phi(x)\right] - \mathbb{E}_{x\sim D}\left[H_\varphi(\eta(x)\right] \\
&= \gamma\left(C_{\varphi,\eta(x_1)}(\Phi(x_1)) - H_\varphi(\eta(x_1))\right) + (1-\gamma)\left(C_{\varphi,\eta(x_2)}(\Phi(x_2)) - H_\varphi(\eta(x_2))\right) \\
&\leq \gamma\left(H_\varphi^-(\eta(x_1)) - H_\varphi(\eta(x_1))\right) + (1-\gamma)\left(H_\varphi^-(\eta(x_2)) - H_\varphi(\eta(x_2))\right) + \epsilon \\
&= \gamma\tilde{\psi}(\alpha_1) + (1-\gamma)\tilde{\psi}(\alpha_2) + \epsilon \\
&\leq \psi(\theta) + \epsilon.
\end{aligned}$$

Now we notice that $\operatorname{sign}(\Phi(x_1)) = \operatorname{sign}(\Phi(x_2)) = -1$, and $\eta(x_1), \eta(x_2) \geq 1/2$, which tells us:

$$\begin{aligned}
R(h) - R^* &= R(h) - R(h_{\text{bayes}}) \\
&= \mathbb{E}_{x\sim D}[|2\eta(x) - 1|] \\
&= \gamma(2\eta(x_1) - 1) + (1-\gamma)(2\eta(x_2) - 1) \\
&= \theta.
\end{aligned}$$

□

The question of whether a loss function is classification-calibrated or not is very important for the use of Theorem 3.1.8. It would also be very useful to know exactly what form the ψ-transformation takes. Therefore we state the following theorem for convex surrogate loss functions:

Theorem 3.1.10. *1. Let φ be a convex loss function. Then φ is classification-calibrated if and only if it is differentiable at 0 and $\varphi'(0) < 0$.*

2. If φ is convex and classification-calibrated, then

$$\psi(\theta) = \varphi(0) - H\left(\frac{1+\theta}{2}\right).$$

Proof. For the proof we refer to [1]. □

In their paper, they also prove that the following loss functions are classification-calibrated and and provide the corresponding ψ-transform.

Loss	$\boldsymbol{\varphi(\alpha)}$	$\boldsymbol{\psi(\theta)}$
Hinge	$\max\{1-\alpha, 0\}$	θ
Squared hinge	$(\max\{1-\alpha, 0\})^2$	θ^2
Sigmoid	$1 - \tanh(\alpha)$	θ
Exponential	$\exp(-\alpha)$	$1 - \sqrt{1-\theta^2}$

Table 3.1: Various classification-calibrated loss functions and their corresponding ψ-transforms

3.2. Adversarial Risk Bound

The following chapter is based on of the work of Zhang et al. [35], which extends the concepts introduced by Bartlett et al. [1]—discussed in the previous section— to adversarial robustness. With our understanding of surrogate losses, we can now apply them to the adversarial risk together with the 0-1 loss. This approach will enable us to establish a bound for the adversarial risk.

We again consider a measurable function $\Phi : \mathbb{R}^d \longrightarrow \mathbb{R}$ and classifiers of the form $h(x) = \text{sign}(\Phi(x))$. Therefore we can write:

$$R_{\text{adv}} = \mathbb{E}_{(x,y)\sim D}\left[\mathbb{1}(\exists x' \in B_\epsilon^p(x) : \Phi(x')y \leq 0)\right].$$

First, we define the decision boundary of a function:

Definition 3.2.1. For a function $\Phi : \mathbb{R}^d \longrightarrow \mathbb{R}$ we denote by

$$DB(\Phi) = \{x \in \mathbb{R}^d | f(x) = 0\}$$

the decision boundary of it.

With this, we can also define:

Definition 3.2.2. For a function $\Phi : \mathbb{R}^d \longrightarrow \mathbb{R}$ and $\epsilon > 0$ we denote by

$$U(\Phi, \epsilon) = \{x \in \mathbb{R}^d | \exists x' \in B_\epsilon^p(x) \text{ such that } \Phi(x)\Phi(x') \leq 0\}$$

the neighborhood of the decision boundary.

As for all $x \in DB(\Phi)$ we have $\Phi(x) = 0$, we know that $DB(\Phi) \subset U(\Phi, \epsilon)$ for all $\epsilon > 0$. We now introduce another form of risk, which measures the performance around the decision boundary.

Definition 3.2.3. For a classifier of the form $h(x) = \text{sign}(\Phi(x))$ and $\epsilon > 0$ the boundary error is defined as

$$R_{\text{bdy}}(h) = \mathbb{E}_{(x,y)\sim D}\left[\mathbb{1}(x \in U(\Phi, \epsilon) | \Phi(x)y > 0)\right].$$

With this boundary error, we can easily decompose the adversarial risk.

Proposition 3.2.4. *For a classifier h, the following holds:*

$$R_{adv}(h) = R(h) + R_{bdy}(h). \tag{3.3}$$

Proof. Assume we have a x with $\mathbb{1}(\exists x' \in B_\epsilon^p(x) : \Phi(x')y \leq 0) = 1$. Then we have either $\Phi(x)y \leq 0$ or $\Phi(x)y > 0$. The first case directly implies that $\mathbb{1}(\Phi(x)y \leq 0) = 1$ and $\mathbb{1}(\Phi(x)y > 0) = 0$. In the second case, we must have an $x' \in B_\epsilon^p(x)$ with $x' \neq x$ such that $\Phi(x')y \leq 0$. But that is equivalent to $\Phi(x')\Phi(x) \leq 0$. This implies that $x \in U(\Phi, \epsilon)$ and $\mathbb{1}(\Phi(x)y > 0) = 1$. □

Now we can use the last section to derive:

Theorem 3.2.5. *For any nonnegative classification-calibrated loss function $\varphi : \mathbb{R} \longrightarrow [0, \infty)$, any classifier of the form $h(x) = \text{sign}(\Phi(x))$, any distribution D on $\mathbb{R}^d \times \{-1, +1\}$ and any $\epsilon > 0$, it holds:*

1. $$R_{adv}(h) - R^* \leq \psi^{-1}(R_\varphi(\Phi) - R_\varphi^*) + \mathbb{P}_{(x,y)\sim D}\left[x \in U(\Phi, \epsilon) : \Phi(x)y > 0\right].$$

2. *If we assume further that, $\varphi(0) \geq 1$ and φ and Φ are continuous, then for every $\lambda > 0$, it holds:*

$$R_{adv}(\Phi, \epsilon) - R^* \leq \psi^{-1}(R_\varphi(\Phi) - R_\varphi^*) + \mathbb{E}_{(x,y)\sim D}\left[\max_{x' \in B_\epsilon^p(x)} \varphi\left(\Phi(x')\Phi(x)/\lambda\right)\right]. \tag{3.4}$$

Proof. 1) First we observe that:

$$R_{\text{bdy}}(\Phi, \epsilon) = \mathbb{E}_{(x,y)\sim D}\left[\mathbb{1}(x \in U(\Phi, \epsilon) : \Phi(x)y > 0)\right] = \mathbb{P}_{(x,y)\sim D}\left[x \in U(\Phi, \epsilon) : \Phi(x)y > 0\right].$$

Using the decomposition of the adversarial risk given in Proposition 3.2.4 and Theorem 3.1.8, we obtain the first inequality by:

$$R_{\text{adv}}(\Phi, \epsilon) - R^* = R(\Phi) - R^* + R_{\text{bdy}}(\Phi, \epsilon) \leq \psi^{-1}(R_\varphi(\Phi) - R_\varphi^*) + R_{\text{bdy}}(\Phi, \epsilon).$$

2) For the second inequality we simply use $\varphi(0) \geq 1$ and get

$$\begin{aligned}
\mathbb{P}_{(x,y)\sim D}\left[x \in U(\Phi,\epsilon) : \Phi(x)y > 0\right] &\leq \mathbb{P}_{(x,y)\sim D}\left[x \in U(\Phi,\epsilon)\right] \\
&= \mathbb{E}_{(x,y)\sim D}\left[\mathbb{1}(x \in U(\Phi,\epsilon)\right] \\
&= \mathbb{E}_{(x,y)\sim D}\left[\max_{x' \in B_\epsilon^p(x)} \mathbb{1}(\Phi(x') \neq \Phi(x))\right] \\
&= \mathbb{E}_{(x,y)\sim D}\left[\max_{x' \in B_\epsilon^p(x)} \mathbb{1}(\Phi(x')\Phi(x)/\lambda < 0)\right] \\
&\leq \mathbb{E}_{(x,y)\sim D}\left[\max_{x' \in B_\epsilon^p(x)} \varphi(\Phi(x')\Phi(x)/\lambda)\right].
\end{aligned}$$

Where the max in the last equation exists, as $B_\epsilon^p(x)$ is compact and φ and Φ are continuous. To see the last inequality, we first observe that if $\Phi(x')\Phi(x)\lambda \geq 0$ for all $x' \in B_\epsilon^p(x)$ we have, since φ is nonnegative:

$$\max_{x' \in B_\epsilon^p(x)} \mathbb{1}(\Phi(x')\Phi(x)\lambda < 0) = 0 \leq \max_{x' \in B_\epsilon^p(x)} \varphi(\Phi(x')\Phi(x)\lambda).$$

On the other hand, if $\Phi(x')\Phi(x)/\lambda < 0$ for an $x' \in B_\epsilon^p(x)$, we know that $x \in U(\Phi,\epsilon)$. Therefore there exists an $x'' \in B_\epsilon^p(x)$ with $\Phi(x'') = 0$ and as we know further, that $\varphi(0) \geq 1$ we finally receive:

$$\max_{x' \in B_\epsilon^p(x)} \mathbb{1}(\Phi(x')\Phi(x)\lambda < 0) = 1 \leq \varphi(0) = \varphi(\Phi(x'')\Phi(x)/\lambda) \leq \max_{x' \in B_\epsilon^p(x)} \varphi(\Phi(x')\Phi(x)/\lambda).$$

□

Note that ψ is the ψ-transform of φ, which we defined in Definition 3.1.5. The parameter λ will act as an important hyperparamter for an algorithmic defence.

3.3. Adversarial Training by TRADES

In the following, we present an alternative version of adversarial training as proposed by Zhang et al. [35]. The goal is to develop an algorithm that achieves even greater robustness than the adversarial training method outlined in Algorithm 4. To accomplish this, we first apply Theorem 3.2.5, which requires minimizing the following expression:

$$\min_{\Phi \in \mathcal{H}} \left(\psi^{-1}(R_\varphi(\Phi) - R_\varphi^*) + \mathbb{E}\left[\max_{x' \in B_\epsilon^p(x)} \varphi\left(\Phi(x')\Phi(x)/\lambda\right)\right]\right) \tag{3.5}$$

to minimize $R_{\text{adv}}(h) - R^*$. For simplicity, we replace ψ^{-1} with the identity function. Additionally, we rely on the parameter λ to approximately capture the effect of ψ^{-1}. Therefore (3.5) simplifies to:

$$\min_{\Phi \in \mathcal{H}} \mathbb{E}\left[\varphi(\Phi(x)y) + \max_{x' \in B_\epsilon^p(x)} \varphi\left(\Phi(x')\Phi(x)/\lambda\right)\right]. \tag{3.6}$$

Problem (3.6) should describe a trade-off between the φ-risk R_φ and the adversarial risk R_{adv}. The term $\varphi(\Phi(x)y)$ corresponds to optimizing for high classification accuracy, while the second term serves as a regularization mechanism to produce adversarially robust classifiers.

Now, again as in Definition 2.3 with the empirical adversarial risk, we make the approach to estimate Equation (3.6) empirically. In the following we will just work with adversarial examples in the ℓ^∞-norm, i.e. $p = \infty$. However, the following can easily be extended to all $p \in \mathbb{N}$. We will assume that we draw a sample of $m \in \mathbb{N}$ observations, which are i.i.d. drawn from the distribution D. Then we get for a sample $S = \{(x_i, y_i)\}_{i=1}^m \subset (\mathbb{R}^d \times \{-1, 1\})$ and a compact parameter set Θ:

$$\min_{\theta \in \Theta} \frac{1}{m} \sum_{i=1}^{m} \left[\varphi(\Phi_\theta(x_i)y_i) + \max_{x' \in B_\epsilon^p(x_i)} \varphi\left(\Phi_\theta(x')\Phi_\theta(x_i)/\lambda\right) \right]. \tag{3.7}$$

In the following, we will assume that φ and Φ are differentiable. As in the standard adversarial training described in Section 2.2, the authors [35] assume that for Equation (3.7), the following expression is a descent direction:

$$-\frac{1}{m} \sum_{i=1}^{m} \nabla_\theta \left[\varphi(\Phi_\theta(x_i)y_i) + \varphi\left(\Phi_\theta(x_i^*)\Phi_\theta(x_i)/\lambda\right) \right],$$

where

$$x_i^* = \arg \max_{x' \in B_\epsilon^p(x_i)} \varphi\left(\Phi_\theta(x')\Phi_\theta(x_i)\right) \tag{3.8}$$

for all $i \in \{1 \dots, m\}$. However, it is important to note that there is currently no theoretical justification for assuming that this is actually a descent direction. Despite this, the practical success of this approach has been observed, and we will proceed with this assumption.

To find the potential adversarial examples x_i^* in (3.8), we will use the PGD attack (Algorithm 3) with the ℓ^∞-norm. By setting $x_0' = x_i$ the normalized steepest descent direction in the first iteration of PGD is then according to Lemma 1.4.1, the following:

$$-\text{sign}(\nabla_{x_0'} \varphi(\left(\Phi_\theta(x_0')\Phi_\theta(x_i)\right)$$

where $i \in \{1, \dots, m\}$. With this we can therefore define the following algorithm named TRADES (TRadeoff-inspired Adversarial DEfense via Surrogate-loss minimization):

Algorithm 5 Adversarial Training by TRADES [35]

Require: samples $S = \{(x^i, y^i)\}_{i=1}^m$, initial model parameters $\theta^0 \in \Theta$, maximal perturbation $\epsilon > 0$, PGD step size $\gamma > 0$, step size $\eta > 0$, maximal iterations PGD $K \in \mathbb{N}$, maximal iterations training $T \in \mathbb{N}$, tuning parameter $\lambda > 0$

1: **for** $t = 0$ to $T - 1$ **do**
2: **for** $i = 1$ to m **do**
3: $x_i' = x_i$
4: **for** $k = 1$ to K **do**
5: $x_i' = \Pi_{x_i,\epsilon}^{\infty}\left(\gamma \text{ sign}\left[\nabla_{x_i'}\varphi((\Phi_\theta(x_i')\Phi_\theta(x_i))\right] + x_i'\right)$
6: **end for**
7: **end for**
8: $\theta_{t+1} = \theta_t - \eta\frac{1}{m}\sum_{i=1}^m \nabla_\theta\left[\varphi(\Phi_{\theta_t}(x_i)y_i) + \varphi\left(\Phi_{\theta_t}(x_i')\Phi_{\theta_t}(x_i)/\lambda\right)\right]$
9: **end for**

This algorithm aims to minimize the objective in Equation (3.7), with the goal of developing a robust and accurate binary classifier. Originally, Zhang et al. [35] presented Algorithm 5 in a different way by extending Problem (3.7) to multi-class classification and replacing φ with the cross-entropy loss. However, this extension is mainly supported by heuristic arguments. Therefore, we will only consider the formulation of Problem (3.7) in its binary classification context.

In numerical experiments, Zhang et al. [35] showed that their proposed method indeed slightly outperforms the standard adversarial training (Algorithm 4) in terms of robustness to various adversarial attacks. They also showed that the bound (3.4) in Theorem 3.2.5 is tight for the MNIST dataset.

4 Regular Adversarial Examples and Certified Robustness

In the previous context, when discussing adversarial examples as defined in 1.3.1, we assumed that these are label-invariant. This means that the specific point retains its true label despite perturbing it. However, in practical scenarios such as image classification, this assumption may not always hold. Therefore, similar to the approach in [22], we will introduce a modified definition of adversarial examples and explore how they can emerge. Additionally, we will develop a method to certify the nonexistence of these newly defined adversarial examples within a certain neighborhood and evaluate the effectiveness of this method through experiments.

4.1. Regular Adversarial Examples

Before we introduce the new definition of adversarial examples, we first need to clarify the main issue with the adversarial examples defined in 1.3.1. To illustrate this, consider again the setting of the experiments in Section 2.3. We take the ℓ^2-PGD attack on the normal trained model. The perturbation is set to $\epsilon = 1$ and we use 40 iterations with a step size of 0.01 on every image. The following are images of the unperturbed images and below the corresponding perturbed ones.

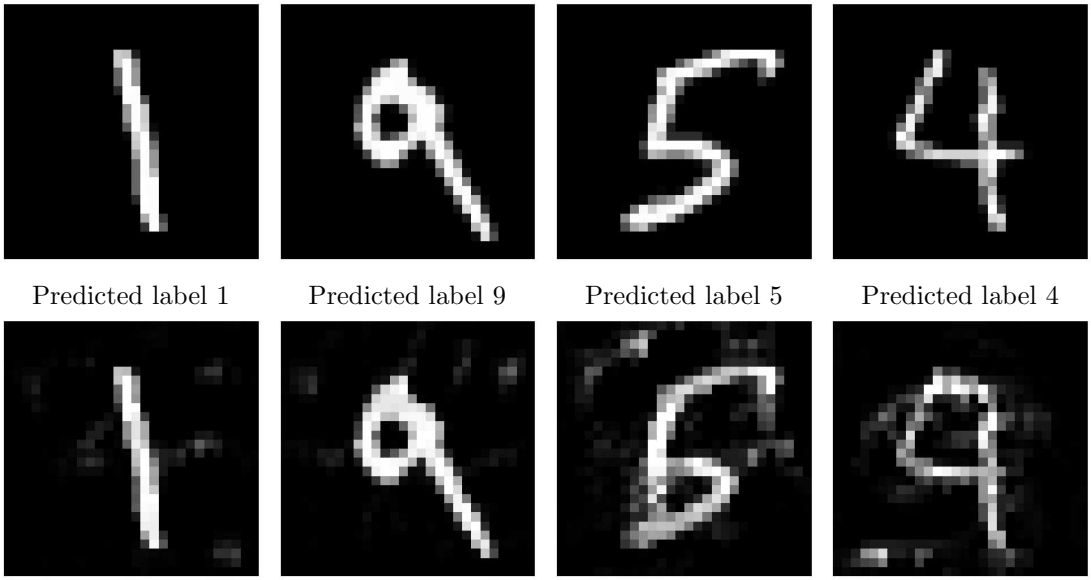

Figure 4.1: In the first row the unperturbed images and in the second the respective perturbed one are displayed. The adversarial examples were created with a maximal perturbation of $\epsilon = 1$ in the ℓ_2-norm. In the caption of every picture the prediction of the model is displayed.

The images on the left show a successful attack using Projected Gradient Descent (PGD). This means that although the attack did not change the true label of the image, it managed to fool the model. On the other hand, the two images on the right side of the second row could be labeled as 6 and 9 by a human observer. This indicates that the PGD attack was not label-invariant in this case and has changed the true labels of the images. This is a problem because, as stated in Definition 1.3.1, where we defined adversarial examples, all four images in Figure 4.1 would be considered as adversarial examples.

To address this problem Petersen and Zech [22] introduced a new version of adversarial examples, which we refer to as regular adversarial examples. They achieved this by defining a so-called ground-truth classifier. The following theoretical observations closely follow their work. To establish this concept, consider a data distribution D on $\mathbb{R}^d \times Y$, where Y is an arbitrary label space. The feature support of the distribution is defined as:

$$D_x := \{x \in \mathbb{R}^d : \exists y \text{ s.t. } (x, y) \in \operatorname{supp}(D)\}.$$

With this we can define the ground-truth classifier:

$$g : \mathbb{R}^d \longrightarrow Y,$$

which satisfies

$$g(x) = \arg\max_{y \in Y} \mathbb{P}(y|x) \quad \text{for almost all } x \in D_x. \tag{4.1}$$

Note that, since the conditional distribution in this case is defined only almost everywhere with respect to the marginal distribution of x, we can only assume that (4.1) holds for almost all $x \in D_x$. Furthermore, the definition of the ground-truth classifier implies that for every x in the feature support of the distribution, the ground-truth classifier acts as a Bayes classifier. However, g can also classify points that lie outside the feature support of the distribution.

This classifier aims to model the human ability to classify a given image. It is clear, that this approach is not perfect, as different people may classify the same image differently. However, it does provide a theoretical framework to address the issue of label changes when an image is perturbed.

Furthermore, we could also extend the function g to output an additional value $c \in \mathbb{Z} \setminus Y$, which would correspond to non-relevant or nonsensical input data x. However, for simplicity, we won't consider this here. Now we can define our different version of adversarial examples:

Definition 4.1.1. Let $g : \mathbb{R}^d \longrightarrow Y$ be the ground-truth classifier, $h : \mathbb{R}^d \longrightarrow Y$ a classifier, $\epsilon > 0$ the maximal perturbation and $p \in \mathbb{N} \cup \{\infty\}$. For $x \in \mathbb{R}^d$ we call $x' \in B_\epsilon^p(x)$ a regular adversarial example of x, if and only if

1. $g(x) = g(x')$,
2. $h(x) = g(x)$ and $h(x') \neq g(x')$.

So a regular adversarial example is an input that, although its label remains unchanged according to the ground-truth classifier g, can deceive the classifier h into making an incorrect prediction. To make the differences between the two definitions of adversarial examples clear, we give a quick example:

Example 4.1.2. Consider again the setting in Figure 4.1 and denote the images in the first row by x_1, x_2, x_3 and x_4. These images, together with their corresponding labels y, are sampled from the unknown distribution D. Specifically, we have: $(x_1, 1), (x_2, 9)$, $(x_3, 5), (x_4, 4) \in \text{supp}(D)$. We assume that a Bayes classifier would classify these images identically. Hence, we can define our ground-truth classifier $g : \mathbb{R}^d \longrightarrow \{0, \ldots, 9\}$ for these images as follows: $g(x_1) = 1, g(x_2) = 0, g(x_3) = 5$ and $g(x_4) = 4$. We set $\epsilon = 1$, $p = 2$ and our classifier h is just the normal trained one from the experiments in Section 2.3. We denote the associated perturbed images in the second row by $x_1^{\text{adv}}, x_2^{\text{adv}}, x_3^{\text{adv}}, x_4^{\text{adv}}$. We assume, that these perturbed images are not elements of the feature support D_x, which allows us to also define g on the perturbed images as well: $g(x_1^{\text{adv}}) = 1, g(x_2^{\text{adv}}) = 9, g(x_3^{\text{adv}}) = 6$ and $g(x_4^{\text{adv}}) = 9$. This models the predictions a human might make when classifying these images. It is immediately clear that x_1^{adv} is a regular adversarial example, as $x_1^{\text{adv}} \in B_1^2(x_1)$, $g(x_1^{\text{adv}}) = g(x_1) = 1$ and we have

$$4 = h(x_1^{\text{adv}}) \neq g(x_1^{\text{adv}}) = 1.$$

The same argument applies to x_2^{adv}. The other two images are not regular adversarial examples, because PGD has changed the label according to the ground-truth classifier g. That is, we have:

$$6 = g(x_3^{\text{adv}}) \neq g(x_3) = 5 \quad \text{and} \quad 4 = g(x_4^{\text{adv}}) \neq g(x_4) = 9.$$

Note that by Definition 1.3.1 all four perturbed images are adversarial examples, which shows that the new definition effectively handles potential label changes.

Clearly, regular adversarial examples depend heavily on the definition of the ground-truth classifier. In the previous example one could have simply defined g to be constant 0 for all $x \in \mathbb{R}^d \setminus D_x$. Under this definition, none of the perturbed images in Figure 4.1 would qualify as a regular adversarial example.

In the following, we will restrict ourselves to binary classification, where $Y = \{-1, 1\}$. Additionally, we set $\text{sign}(0) = -1$. Next, we will examine the different situations under which regular adversarial examples can occur. To uniquely define the ground truth classifier for all $x \in D_x$, we will assume that it holds either

$$\mathbb{P}[y = 1|x] < \mathbb{P}[y = -1|x] \quad \text{or} \quad \mathbb{P}[y = 1|x] > \mathbb{P}[y = -1|x]$$

for all $x \in D_x$. That means that when we have a Bayes classifier h we get:

$$h(x) = g(x) \quad \text{for all } x \in D_x.$$

Furthermore, we say that the distribution exhausts the domain if $D_x = \mathbb{R}^d$. Given this, we can distinguish between the following four cases:

1. **Bayes classifier and exhaustive distribution:** If h is a Bayes classifier and $D_x = \mathbb{R}^d$, then every $x \in \mathbb{R}^d$ is classified the same by h and the ground-truth classifier g. Therefore, no regular adversarial examples can emerge.

2. **Bayes classifier and non-exhaustive distribution:** If h is a Bayes classifier and $D_x \neq \mathbb{R}^d$, then h coincides with g for all $x \in D_x$. However, on $\mathbb{R}^d/D_x$ the two classifiers don't have to coincide, which means that regular adversarial examples can exist.

3. **Not a Bayes classifier and exhaustive distribution:** If h is not a Bayes classifier and $D_x = \mathbb{R}^d$, then there exists an $x' \in R^d$ with $h(x') \neq g(x')$. Therefore, if there exists a $x \in B_\epsilon(x')$ with $g(x) = h(x)$, then x' is an regular adversarial example of x. However, if we have a classifier with $h(x) = -g(x)$ for all $x \in \mathbb{R}^d$, then no regular adversarial example emerges. Hence, regular adversarial examples can exist in this case.

4. **Not a Bayes classifier and non-exhaustive distribution:** In this case, anything is possible. One can indeed construct a classifier h, such that no regular adversarial examples exist, but for the most classifiers, one can find regular adversarial examples.

Now, we aim to provide examples for the cases 2 and 3. To do this, we first prove a statement that ensures the emergence of adversarial examples for affine classifiers under certain conditions.

Proposition 4.1.3. *Let $w, \overline{w} \in \mathbb{R}^d$ be nonzero. For $x \in \mathbb{R}^d$, let $h(x) = sign(w^T x)$ be a classifier and let $g(x) = sign(\overline{w}^T x)$ be the ground-truth classifier. For every $x \in \mathbb{R}^d$ with $h(x)g(x) = 1$, $|w^T x| \neq 0$, and all $\delta \in (0, |w^T x|)$ such that*

$$\frac{|\overline{w}^T x|}{\|\overline{w}\|_p} > \frac{\delta + |w^T x|}{\|w\|_p} \frac{|w^T \overline{w}|}{\|\overline{w}\|_p \|w\|_p} \tag{4.2}$$

it holds that

$$x' = x - h(x) \frac{\delta + |w^T x|}{\|w\|_p^2} w$$

is a regular adversarial example of x with $x' \in B_\epsilon^p(x)$, where $\epsilon = \frac{\delta + |w^T x|}{\|w\|_p}$.

Proof. First, we show that x' lies within $x' \in B_\epsilon^p(x)$. We have:

$$\|x - x'\|_p = \left\| \frac{\delta + |w^T x|}{\|w\|_p^2} w \right\|_p = \frac{\delta + |w^T x|}{\|w\|_p} = \epsilon.$$

Thus, $x' \in B_\epsilon^p(x)$. Now we show that $g(x)g(x') = 1$, i.e. that $(\overline{w}^T x)(\overline{w}^T x')$ is positive. By plugging in x', we get:

$$\overline{w}^T x \left(\overline{w}^T x - h(x) \frac{\delta + |w^T x|}{\|w\|_p^2} \overline{w}^T w \right) = |\overline{w}^T x|^2 - |\overline{w}^T x| \frac{\delta + |w^T x|}{\|w\|_p^2} \overline{w}^T w \tag{4.3}$$

$$\geq |\overline{w}^T x|^2 - |\overline{w}^T x| \frac{\delta + |w^T x|}{\|w\|_p^2} |\overline{w}^T w|, \tag{4.4}$$

where the equality holds because $h(x) = g(x) = \text{sign}(w^T x)$ due to the assumption. Dividing the right-hand side of (4.3) by $|w^T x| \|w\|_p$, which is positive by the assumption, we obtain

$$\frac{|\overline{w}^T x|}{\|\overline{w}\|_p} - \frac{\delta + |w^T x|}{\|w\|_p} \frac{|\overline{w}^T w|}{\|w\|_p \|\overline{w}\|_p}. \tag{4.5}$$

The term (4.5) is positive thanks to (4.2). Further we have:

$$\begin{aligned}(w^T x)(w^T x') &= |w^T x|^2 - w^T x h(x) \frac{\delta + |w^T x|}{\|w\|_p^2} w^T w \\ &= |w^T x|^2 - |w^T x|(\delta + |w^T x|) < 0.\end{aligned}$$

Therefore, $h(x) \neq h(x')$ and the proof is complete. □

Note that the inequality $|w^T\overline{w}| \leq \|w\|\|\overline{w}\|$ always holds. Therefore, (4.2) implies that a regular adversarial example will always exist if

$$\frac{|\overline{w}^T x|}{\|\overline{w}\|_p} > \frac{|w^T x|}{\|w\|_p} > 0. \tag{4.6}$$

First, we present an example of the emergence of regular adversarial examples in Case 2, i.e. we have a Bayes classifier and a non-exhaustive distribution:

Example 4.1.4. Assume that the ground-truth classifier $g : \mathbb{R}^d \longrightarrow \{-1, 1\}$ is given by $g(x) = \text{sign}(\overline{w}^T x)$ for $\overline{w} \in \mathbb{R}^d$, with $\|\overline{w}\|_p = 1$. Further let D be the uniform distribution on

$$\{(\lambda w, g(\lambda w)) \mid \lambda \in [-1, 1] \setminus \{0\}\} \subseteq \mathbb{R}^d \times \{-1, 1\}.$$

Then the feature support equals

$$D_x = \{\lambda w \mid \lambda \in [-1, 1] \setminus \{0\}\} \subseteq \text{span}\{w\}.$$

Next, fix $\alpha \in (0, 1)$ and set $w := \alpha\overline{w} + (1 - \alpha)v$ for some $v \in \overline{w}^{\perp}$ with $\|v\|_p = 1$, so that $\|w\|_p = 1$. We let $h(x) := \text{sign}(w^T x)$. We now show that every $x \in D_x$ satisfies the assumptions of Proposition 4.1.3, and therefore admits a regular adversarial example. Fix $x \in D_x$, then we get:

$$h(x) = h(\lambda\overline{w}) = \text{sign}(\alpha\lambda\overline{w}^T\overline{w} + (1 - \alpha)\lambda v^T\overline{w}) = \text{sign}(\lambda\overline{w}^T\overline{w}) = g(x).$$

Therefore, $h(x) = g(x)$ for all $x \in D_x$ and h is a Bayes classifier. We also observe, that $|w^T x| \leq \alpha|\overline{w}^T x|$ for all $x \in D_x$. Furthermore, for every $\delta > 0$ it holds that

$$\epsilon := \frac{\delta + |w^\top x|}{\|w\|_p} \leq \delta + \alpha.$$

Thus, Theorem 4.1.3 implies that for every $x \in D_x$ with $|w^T x| \neq 0$, there exists a $\delta < |w^T x|$ such that a regular adversarial example exists in $B^p_{\delta+\alpha}(x)$.

Now, let's turn to an example for the Case 3, where we have a classifier that is not a Bayes classifier and the distribution is exhaustive.

Example 4.1.5. We again assume that the ground-truth classifier $g : \mathbb{R}^d \longrightarrow \{-1, 1\}$ is given by $g(x) = \text{sign}(\overline{w}^T x)$ for $\overline{w} \in \mathbb{R}^d$, with $\|\overline{w}\|_p = 1$. We define the decision boundary of g as $\text{DB}_g = \{x \in \mathbb{R}^d | \overline{w}^T x = 0\}$. Then we let D_x be a distribution on $\mathbb{R}^d$ with a positive Lebesgue density everywhere outside the decision boundary. Now, we define D as the joint distribution of $(X, g(X))$, where $X \sim D_x$. We now define our classifier h with the following considerations: We assume $w \neq \overline{w}$, otherwise, h would simply be a Bayes classifier. Additionally, we exclude the case $w = -\overline{w}$, as this would result in h making incorrect predictions for all $x \in \mathbb{R}^d$. Finally, let w such that $\|w\|_p = 1$ and define the classifier as $h(x) = \text{sign}(w^T x)$. Given that $D_x = \mathbb{R}^d$, the distribution is exhaustive.

Furthermore, since the decision boundaries DB_g and DB_h do not coincide, we observe that:

$$\operatorname{dist}\left[h^{-1}(-1) \cap g^{-1}(1), h^{-1}(1) \cap g^{-1}(1)\right] = 0$$

and

$$\operatorname{dist}\left[h^{-1}(1) \cap g^{-1}(-1), h^{-1}(-1) \cap g^{-1}(-1)\right] = 0.$$

As a result, for every $\epsilon > 0$, there is a positive probability of observing an x that has a regular adversarial example in $B_\epsilon^p(x)$.

4.2. Robustness

Now that we have a basic understanding of how regular adversarial examples emerge, we aim to investigate how to make our classifiers robust against them. In the following, we will consider classifiers of the form $h(x) = \operatorname{sign}(\Phi(x))$, where $\Phi : \mathbb{R}^d \longrightarrow \mathbb{R}$. We will now show that, under a Lipschitz condition on Φ, it is possible to establish a robustness statement for regular adversarial examples.

Proposition 4.2.1. *Let $\Phi : \mathbb{R}^d \longrightarrow \mathbb{R}$ be C-Lipschitz, with respect to a ℓ^p-norm with $p \in \mathbb{N} \cup \{\infty\}$, $C > 0$ and further $s > 0$. Let $h(x) = \operatorname{sign}(\Phi(x))$ be a classifier and $g : \mathbb{R}^d \longrightarrow \{-1, 1\}$ be a ground-truth classifier. Moreover, let $x \in \mathbb{R}^d$ be such that*

$$\Phi(x)g(x) \geq s.$$

Then, the point x has no regular adversarial example inside of $B_{s/C}^p(x)$.

Proof. Let $x \in \mathbb{R}^d$ satisfy $\Phi(x)g(x) \geq s$ and take an arbitrary $x' \in B_{s/C}^p(x)$. Because Φ is C-Lipschitz, we get:

$$|\Phi(x) - \Phi(x')| \leq C||x - x'||_p \leq s.$$

As we also know that $|\Phi(x)| = \Phi(x)g(x) \geq s$, we conclude that $\Phi(x')$ has the same sign as $\Phi(x')$, which shows that x' cannot be a regular adversarial example of x. □

One issue with this statement is that assuming a specific Lipschitz constant for the classifier may overly restrict its capabilities. However, for neural networks with the Relu activation function (see Definition 1.2.2), we can show that under a specific assumption on the training set, there exists a network that interpolates the training data and does not allow regular adversarial examples of a certain perturbation for the training set.

Theorem 4.2.2. *Let $m \in \mathbb{N}$, let $g : \mathbb{R}^d \to \{-1, 1\}$ be a ground-truth classifier, and let $\{(x_i, g(x_i))\}_{i=1}^m \subset (\mathbb{R}^d \times \{-1, 1\})^m$. Furthermore, we assume that,*

$$\sup_{i \neq j} \frac{|g(x_i) - g(x_j)|}{||x_i - x_j||_1} =: \tilde{M} > 0.$$

Then there exists a ReLU neural network Φ *with* $depth(\Phi) = \mathcal{O}(\log(m))$ *and* $width(\Phi) = \mathcal{O}(dm)$ *such that for all* $i = 1, \dots, m$

$$sign(\Phi(x_i)) = g(x_i)$$

and for $\epsilon = 1/\tilde{M}$ *we have that* x_i *has no regular adversarial example inside of* $B_\epsilon^1(x_i)$.

Proof. We see directly that g satisfies the conditions of Theorem 5.0.2, so we receive a ReLU neural network Φ which interpolates the data, is $\tilde{M}$- Lipschitz and further fulfills the size conditions. Moreover, we get:

$$\Phi(x_i)g(x_i) = |\Phi(x_i)| = |y_i| = 1.$$

So by applying Proposition 4.2.1 we can finish the proof. □

Note that this theorem applies only to regular adversarial examples that are bounded by the ℓ^1-norm. One problem with the above statements is that, in the case of deep neural networks, global Lipschitz bounds can grow exponentially with the network's depth. Therefore, we will present a theorem that uses only the local Lipschitz property to make a robustness statement for ℓ^∞-regular adversarial examples.

Theorem 4.2.3. *Let* $h : \mathbb{R}^d \longrightarrow \{-1, 1\}$ *be a classifier of the form* $h(x) = sign(\Phi(x))$ *and* $g : \mathbb{R}^d \longrightarrow \{-1, 1\}$ *be the ground-truth classifier. Let* $x \in \mathbb{R}^d$ *and define:*

$$\alpha = \max_{R>0} \min \left\{ \Phi(x)g(x) \Big/ \sup_{\substack{y \in B_R^\infty(x) \\ y \neq x}} \frac{|\Phi(y) - \Phi(x)|}{||x - y||_\infty}, R \right\}. \tag{4.7}$$

We set the minimum to R, if a division by 0 appears in (4.7). *Then for* $\epsilon < \alpha$ *there are no regular adversarial examples of* x *in* $B_\epsilon^\infty(x)$.

Proof. Let $x \in \mathbb{R}^d$ be arbitrary and assume, towards a contradiction, that for $0 < \epsilon < \alpha$ satisfying (4.7), there exists an regular adversarial example x' of x with $x' \in B_\epsilon^\infty(x)$. The supremum in Equation (4.7) implies that Φ is constant on a ball of radius R around x. In particular, for $x' \in B_\epsilon^\infty(x) \subset B_R^\infty(x)$ it holds that $h(x) = h(x')$ and therefore x' cannot be a regular adversarial example. Now we assume that the supremum in (4.7) is not equal to zero. By the equation, it holds that

$$\delta < \Phi(x)g(x) \Big/ \sup_{\substack{y \in B_R^\infty(x) \\ y \neq x}} \frac{|\Phi(y) - \Phi(x)|}{\|x - y\|_\infty}. \tag{4.8}$$

Furthermore we get:

$$\begin{aligned} |\Phi(x') - \Phi(x)| &\leq \sup_{\substack{y \in B_R^\infty(x) \\ y \neq x}} \frac{|\Phi(y) - \Phi(x)|}{\|x - y\|_\infty} \|x - x'\|_\infty \\ &\leq \sup_{\substack{y \in B_R^\infty(x) \\ y \neq x}} \frac{|\Phi(y) - \Phi(x)|}{\|x - y\|_\infty} \epsilon > \Phi(x)g(x), \end{aligned}$$

where the last inequality follows out of (4.8). In the end, we receive:

$$\begin{aligned} g(x)\Phi(x') &= g(x)\Phi(x') + g(x)(\Phi(x') - \Phi(x)) \\ &\geq g(x)\Phi(x) - |\Phi(x') - \Phi(x)| > 0. \end{aligned}$$

Therefore, x' cannot be a regular adversarial example of x. □

Since the supremum in (4.7) is bounded by the Lipschitz constant on $B_R^\infty(x)$, the theorem depends only on the local Lipschitz constant of Φ.

4.3. Certified Robustness

One problem remains: the computation of (4.7) could be quite challenging. Therefore, we will present a simple algorithm that provides a method for neural networks to verify the non-existence of regular adversarial examples with a specific perturbation. In the following, we will only focus on regular adversarial examples in the ℓ^∞-norm. We will need the following definition:

Definition 4.3.1. Let $\Phi : \mathbb{R}^d \longrightarrow \mathbb{R}$ be a continuous function, then for a perturbation $\epsilon > 0$ and $x \in \mathbb{R}^d$ we define

$$z^{\epsilon,\max} := \max_{y \in B_\epsilon^\infty(x)} \Phi(y) \tag{4.9}$$

and

$$z^{\epsilon,\min} := \min_{y \in B_\epsilon^\infty(x)} \Phi(y). \tag{4.10}$$

With that we get:

Proposition 4.3.2. *Let $h : \mathbb{R}^d \longrightarrow \{-1, 1\}$ be a classifier of the form $h(x) = sign(\Phi(x))$ and $g : \mathbb{R}^d \longrightarrow \{-1, 1\}$ be a ground-truth classifier. Furthermore, let $x \in \mathbb{R}^d$ satisfy $h(x) = g(x)$. Then x has no regular adversarial examples in $B_\epsilon^\infty(x)$ if $z^{\epsilon,\max} z^{\epsilon,\min} > 0$.*

Proof. Since $z^{\epsilon,\max} z^{\epsilon,\min} > 0$ implies, that all points in $B_\epsilon^\infty(x)$ are classified the same, the claim follows. □

Interestingly, we can also make a similar statement not only for regular adversarial examples but also for adversarial examples defined according to Definition 1.3.1.

Proposition 4.3.3. *Let $h : \mathbb{R}^d \longrightarrow \{-1, 1\}$ be a classifier of the form $h(x) = sign(\Phi(x))$. Let $(x, y) \in \mathbb{R}^d \times \{-1, 1\}$ be drawn from the distribution D and assume that $h(x) = y$. Then:*

1. *If $z^{\epsilon,\max} z^{\epsilon,\min} > 0$, the point x has no adversarial examples in $B_\epsilon^\infty(x)$.*
2. *If $z^{\epsilon,\max} z^{\epsilon,\min} < 0$, there exists a point $x' \in B_\epsilon^\infty(x)$, which is an adversarial example of x.*

Proof. The proof of Part 1 is identical to the one in Proposition 4.3.2. For Part 2, the condition $z^{\epsilon,\max} z^{\epsilon,\min} < 0$ implies that both signs $+1$ and -1, must occur as outputs of h. Therefore, there exists an $x' \in B_\epsilon^\infty(x)$ such that $h(x) \neq h(x')$, which proves the statement. □

So, now we need to find a method to efficiently compute $z^{\epsilon,\max}$ and $z^{\epsilon,\min}$. Fortunately, if Φ is a neural network, it is quite simple to obtain upper and lower bounds for these values. Let $|A|$ denote the matrix, where each entry is the absolute value of the corresponding entry in matrix A. Then we define

$$A^+ := \frac{|A| + A}{2} \quad \text{and} \quad A^- := \frac{|A| - A}{2}.$$

We are now ready to present the algorithm that enables us to establish certified robustness guarantees:

Algorithm 6 Certified Robustness Algorithm [22]

Input: Weight matrices $W^{(\ell)} \in \mathbb{R}^{d_{\ell+1} \times d_\ell}$ and bias vectors $b^{(\ell)} \in \mathbb{R}^{d_{\ell+1}}$ for $\ell = 0, \ldots, L$ with $d_{L+1} = 1$, activation function σ, input vector $x \in \mathbb{R}^{d_0}$, perturbation $\epsilon > 0$
Output: $\Phi(x)$ and bounds for $z^{\epsilon,\max}$ and $z^{\epsilon,\min}$

$x^{(0)} = x$
$\epsilon^{(0),\text{up}} = \epsilon \mathbf{1} \in \mathbb{R}^{d_0}$
$\epsilon^{(0),\text{low}} = \epsilon \mathbf{1} \in \mathbb{R}^{d_0}$
for $\ell = 0$ to $L - 1$ **do**
    $x^{(\ell+1)} = \sigma(W^{(\ell)} x^{(\ell)} + b^{(\ell)})$
    $\epsilon^{(\ell+1),\text{up}} = \sigma(W^{(\ell)} x^{(\ell)} + (W^{(\ell)})^+ \epsilon^{(\ell),\text{up}} + (W^{(\ell)})^- \epsilon^{(\ell),\text{low}} + b^{(\ell)}) - x^{(\ell+1)}$
    $\epsilon^{(\ell+1),\text{low}} = x^{(\ell+1)} - \sigma(W^{(\ell)} x^{(\ell)} - (W^{(\ell)})^+ \epsilon^{(\ell),\text{low}} - (W^{(\ell)})^- \epsilon^{(\ell),\text{up}} + b^{(\ell)})$
end for
$x^{(L+1)} = W^{(L)} x^{(L)} + b^{(L)}$
$\epsilon^{(L+1),\text{up}} = (W^{(L)})^+ \epsilon^{(L),\text{up}} + (W^{(L)})^- \epsilon^{(L),\text{low}}$
$\epsilon^{(L+1),\text{low}} = (W^{(L)})^+ \epsilon^{(L),\text{low}} + (W^{(L)})^- \epsilon^{(L),\text{up}}$
return $x^{(L+1)}$, $x^{(L+1)} + \epsilon^{(L+1),\text{up}}$, $x^{(L+1)} - \epsilon^{(L+1),\text{low}}$

The algorithm is computationally very efficient, as its complexity is comparable to that of a normal forward pass through Φ. However, it is not yet clear whether the algorithm produces meaningful results. Thus, similar to Petersen and Zech [22], we make the following observations:

Proposition 4.3.4. *Let Φ be a neural network with weight matrices $W^{(\ell)} \in \mathbb{R}^{d_{\ell+1} \times d_\ell}$, bias vectors $b^{(\ell)} \in \mathbb{R}^{d_{\ell+1}}$ for $\ell = 0, \ldots, L$, and a monotonically increasing activation function σ. Let $x \in \mathbb{R}^d$, then the output of Algorithm 6 satisfies*

$$x^{L+1} + \epsilon^{(L+1),up} \geq z^{\epsilon,\max} \quad \textit{and} \quad x^{L+1} - \epsilon^{(L+1),low} \leq z^{\epsilon,\min}. \tag{4.11}$$

Proof. Take $y, x \in \mathbb{R}^d$ with $y \in B_\epsilon^\infty(x)$ and let $y^{(\ell)}, x^{(\ell)}$ for $\ell = 0, \dots, L+1$ be as in Algorithm 6 applied to y, x, respectively. Also let $\epsilon^{\ell,\text{up}}, \epsilon^{\ell,\text{low}}$ for $\ell = 0, \dots, L+1$ be as in Algorithm 6 applied to x. Now we will show by induction over $\ell = 0, \dots, L+1$ that

$$y^{(\ell)} - x^{(\ell)} \leq \epsilon^{\ell,\text{up}} \quad \text{and} \quad x^{(\ell)} - y^{(\ell)} \leq \epsilon^{\ell,\text{low}}, \tag{4.12}$$

where the inequalities are understood entry-wise for vectors. As $\Phi(y) = y^{L+1}$ this directly implies:

$$\Phi(y) \leq x^{L+1} + \epsilon^{(L+1),\text{up}} \quad \text{and} \quad \Phi(y) \geq x^{L+1} - \epsilon^{(L+1),\text{low}}.$$

Since y was arbitrary this then proves the result. The case $\ell = 0$ is clear, as we assumed that $y \in B_\epsilon^\infty(x)$. Now, we assume that the statement was shown for $\ell < L$. We get:

$$\begin{aligned} y^{(\ell+1)} - x^{(\ell+1)} - \epsilon^{(\ell+1),\text{up}} &= \sigma(W^{(\ell)} y^{(\ell)} + b^{(\ell)}) \\ &\quad - \sigma\left(W^{(\ell)} x^{(\ell)} + (W^{(\ell)})^+ \epsilon^{(\ell),\text{up}} + (W^{(\ell)})^- \epsilon^{(\ell),\text{low}} + b^{(\ell)}\right). \end{aligned}$$

Looking at the last term and considering the monotonicity of the activation function σ, we see that

$$y^{(\ell+1)} - x^{(\ell+1)} \leq \epsilon^{\ell+1,\text{up}}$$

holds, if

$$W^{(\ell)} y^{(\ell)} \leq W^{(\ell)} x^{(\ell)} + (W^{(\ell)})^+ \epsilon^{(\ell),\text{up}} + (W^{(\ell)})^- \epsilon^{(\ell),\text{low}}. \tag{4.13}$$

To show that Equation (4.13) is indeed true, we observe that

$$\begin{aligned} W^{(\ell)}(y^{(\ell)} - x^{(\ell)}) &= (W^{(\ell)})^+ (y^{(\ell)} - x^{(\ell)}) - (W^{(\ell)})^- (y^{(\ell)} - x^{(\ell)}) \\ &= (W^{(\ell)})^+ (y^{(\ell)} - x^{(\ell)}) + (W^{(\ell)})^- (x^{(\ell)} - y^{(\ell)}) \\ &\leq (W^{(\ell)})^+ \epsilon^{(\ell),\text{up}} + (W^{(\ell)})^- \epsilon^{(\ell),\text{low}}, \end{aligned}$$

where we used the induction hypothesis for the last inequality. This shows the first estimate in (4.12).
Like in the first part, we have

$$\begin{aligned} x^{(\ell+1)} - y^{(\ell+1)} \leq \epsilon^{(\ell+1),\text{low}} &= \sigma\Big(W^{(\ell)} x^{(\ell)} - (W^{(\ell)})^+ \epsilon^{(\ell),\text{low}} \\ &\quad - (W^{(\ell)})^- \epsilon^{(\ell),\text{up}} + b^{(\ell)}\Big) - \sigma(W^{(\ell)} y^{(\ell)} + b^{(\ell)}). \end{aligned}$$

Again using the monotonicity of σ

$$x^{(\ell+1)} - y^{(\ell+1)} \leq \epsilon^{\ell+1,\text{low}}$$

is true, if

$$W^{(\ell)} y^{(\ell)} \geq W^{(\ell)} x^{(\ell)} - (W^{(\ell)})^+ \epsilon^{(\ell),\text{low}} - (W^{(\ell)})^- \epsilon^{(\ell),\text{up}}. \tag{4.14}$$

holds. To prove (4.14), we can conclude with the induction assumption that

$$\begin{aligned} W^{(\ell)}(x^{(\ell)} - y^{(\ell)}) &= (W^{(\ell)})^+(x^{(\ell)} - y^{(\ell)}) - (W^{(\ell)})^-(x^{(\ell)} - y^{(\ell)}) \\ &= (W^{(\ell)})^+(x^{(\ell)} - y^{(\ell)}) + (W^{(\ell)})^-(y^{(\ell)} - x^{(\ell)}) \\ &\leq (W^{(\ell)})^+\epsilon^{(\ell),\text{low}} + (W^{(\ell)})^-\epsilon^{(\ell),\text{up}}, \end{aligned}$$

is valid. Therefore, 4.12 holds for all $\ell \leq L$. Showing the case $\ell = L + 1$ follows by the same argument, but replacing σ by the identity. □

Now we aim to solve the following maximization problem for $x \in \mathbb{R}^d$:

$$\begin{aligned} &\max \quad \epsilon \\ \text{s.t.} \quad &z^{\epsilon,\max} z^{\epsilon,\min} > 0 \\ &\epsilon > 0. \end{aligned} \tag{4.15}$$

According, to Proposition 4.3.4, we can find a upper bound for $z^{\epsilon,\max}$ and a lower bound for $z^{\epsilon,\min}$. Therefore, we will instead solve:

$$\begin{aligned} &\max \quad \epsilon \\ \text{s.t.} \quad &(x^{(L+1)} + \epsilon^{(L+1),\text{up}})(x^{(L+1)} - \epsilon^{(L+1),\text{low}}) > 0 \\ &\epsilon > 0, \end{aligned} \tag{4.16}$$

where we used the outputs of Algorithm 6. For $x \in \mathbb{R}^d$, we define:

$$f_x^{\text{up}}(\epsilon) := x^{(L+1)} + \epsilon^{(L+1),\text{up}} \quad \text{and} \quad f_x^{\text{low}}(\epsilon) := x^{(L+1)} - \epsilon^{(L+1),\text{low}}.$$

Note that $f_x^{\text{low}}(\epsilon) \leq f_x^{\text{up}}(\epsilon)$ for all $\epsilon > 0$. Therefore, (4.16) simplifies to:

$$\begin{aligned} &\max \quad \epsilon \\ \text{s.t.} \quad &f_x^{\text{up}}(\epsilon) < 0 \quad \text{or} \quad f_x^{\text{low}}(\epsilon) > 0 \\ &\epsilon > 0. \end{aligned} \tag{4.17}$$

One issue with (4.17) are the strictly inequalities. Even if the feasible set is nonempty, an optimal solution may not exist. Therefore, we will consider instead the following problem:

$$\begin{aligned} &\max \quad \epsilon \\ \text{s.t.} \quad &f_x^{\text{up}}(\epsilon) \leq -\delta \quad \text{or} \quad f_x^{\text{low}}(\epsilon) \geq \delta \\ &\epsilon \geq \delta. \end{aligned} \tag{4.18}$$

where $\delta > 0$. The feasible set of this problem is defined as:

$$X = \left\{ \epsilon \in \mathbb{R} \;\middle|\; \begin{aligned} &\epsilon \geq \delta, \\ &f_x^{\text{up}}(\epsilon) \leq -\delta \quad \text{or} \quad f_x^{\text{low}}(\epsilon) \geq \delta \end{aligned} \right\}.$$

Now we will show, that when the feasible set of (4.18) is nonempty than a unique optimal solution does exist.

Proposition 4.3.5. *Let Φ be neural network with a continuous and monotonically increasing activation function σ. Let $h : \mathbb{R}^d \longrightarrow \{-1, 1\}$ be a classifier of the form $h(x) = sign(\Phi(x))$ and let $x \in \mathbb{R}^d$. Assume there exists $\tilde{y}_1, \tilde{y}_2 \in \mathbb{R}^d$ such that $h(\tilde{y}_1) = 1$ and $h(\tilde{y}_2) = -1$. If the feasible set of Problem (4.18) is nonempty, then there exists a unique optimal solution $\epsilon_x^{\max}$.*

Proof. Let $\tilde{\epsilon} \in X$, meaning $\tilde{\epsilon} \geq \delta$ and $f_x^{\text{up}}(\tilde{\epsilon}) \leq -\delta$ or $f_x^{\text{low}}(\tilde{\epsilon}) \geq \delta$. First, consider the case $f_x^{\text{up}}(\tilde{\epsilon}) \leq -\delta$. We will show that there exists an $\epsilon' \geq \delta$ such that $f_x^{\text{up}}(\epsilon') > 0$. Assume towards a contradiction, that $f_x^{\text{up}}(\epsilon) \leq 0$ for all $\epsilon \geq \delta$. By Proposition 4.3.4, it follows that

$$\Phi(y) \leq f_x^{\text{up}}(\epsilon) \leq 0 \quad \text{for all } \epsilon \geq \delta, y \in B_\epsilon^\infty(x).$$

This implies that $\Phi(y) \leq 0$ for all $y \in \mathbb{R}^d$. However, by assumption, there exists a $\tilde{y}_1$ such that $h(\tilde{y}_1) = 1$. Therefore, we have:

$$-1 = \text{sign}(\Phi(\tilde{y}_1)) = h(\tilde{y}_1) = 1,$$

which is a contradiction. Hence, there exists an ϵ' with $f_x^{\text{up}}(\epsilon') > 0$. Since σ is continuous the same must hold for f_x^{up}. Further we know that $-\delta \in [f_x^{\text{up}}(\tilde{\epsilon}), f_x^{\text{up}}(\epsilon')]$, so the intermediate value Theorem ensures the existence of a ϵ_δ with $f_x^{\text{up}}(\epsilon_\delta) = -\delta$. Now consider the set:

$$S_\delta = \{\epsilon \in \mathbb{R} \mid \tilde{\epsilon} \leq \epsilon \leq \epsilon' \text{ and } f_x^{\text{up}}(\epsilon) = -\delta\}.$$

From our previous discussion, we know that $\epsilon_\delta \in S_\delta$, so the set S_δ is non-empty. Moreover, due to the continuity of f_x^{up}, the set S_δ is closed, and since it is also bounded by $\tilde{\epsilon}$ and ϵ', it is compact. Given that every compact subset of $\mathbb{R}$ has a unique maximum element, the set S_δ must have a unique maximum, which we denote as $\epsilon_x^{\max}$. This $\epsilon_x^{\max}$ is then the unique solution to Problem (4.18).

Next, consider the case $f_x^{\text{low}}(\tilde{\epsilon}) \geq \delta$. We will show that there exists an $\epsilon' \geq \delta$ such that $f_x^{\text{low}}(\epsilon) \leq 0$. Assume for contradiction, that $f_x^{\text{low}}(\epsilon) > 0$ for all $\epsilon \geq \delta$. Again, using Proposition 4.3.4, we have:

$$\Phi(y) \geq f_x^{\text{low}}(\epsilon) > 0 \quad \text{for all } \epsilon \geq \delta, y \in B_\epsilon^\infty(x).$$

Thus $\Phi(y) > 0$ for all $y \in \mathbb{R}^d$. However, by assumption, there exists an $\tilde{y}_2 \in \mathbb{R}^d$ such that $h(\tilde{y}_2) = 1$. Therefore, we have:

$$1 = \text{sign}(\Phi(\tilde{y}_2)) = h(\tilde{y}_2) = -1,$$

which is a contradiction. This shows, that there exists an ϵ' with $f_x^{\text{low}}(\epsilon') < 0$. Because $\delta \in [f_x^{\text{low}}(\tilde{\epsilon}), f_x^{\text{low}}(\epsilon')]$ and f_x^{low} is continuous the intermediate value Theorem provides a ϵ_δ such that $f_x^{\text{low}}(\epsilon_\delta) = \delta$. We now consider:

$$S_\delta = \{\epsilon \in \mathbb{R} \mid \tilde{\epsilon} \leq \epsilon \leq \epsilon' \text{ and } f_x^{\text{low}}(\epsilon) = \delta\}.$$

From our previous discussion, it follows that $\epsilon_\delta \in S_\delta$, demonstrating that S_δ is not empty. Additionally, the continuity of f_x^{low} ensures that S_δ is a compact set.
Therefore there exists a unique maximum element in S_δ, which we denote by $\epsilon_x^{\max}$. This element, $\epsilon_x^{\max}$, serves as the unique solution to Problem (4.18). □

We further note:

Remark 4.3.6. According to Proposition 4.3.2 and Proposition 4.3.3, for every x and the corresponding $\epsilon_x^{\max}$, there are no regular adversarial examples and no adversarial examples of x in $B^\infty_{\epsilon_{\max}}(x)$.

4.4. Experimental Results

In this section, we want to evaluate the recently introduced Algorithm 6 and thus aim to solve (4.18). We will use the MNIST dataset for this evaluation. However, since the algorithm is limited to binary classification, we will modify the dataset accordingly: images with labels $0, 1, 2, 3, 4$ will be assigned a label of 1, while those with labels $5, 6, 7, 8, 9$ will be assigned a label of -1. Due to computational constraints, we will only use a subset of 1000 images for training and another 1000 images for testing. The code for the experiments is available at https://github.com/j-schuhmann/Adversarial-Examples-in-Machine-Learning-Training-and-Certified-Robustness.

For the binary classification task, we will use a simple neural network. We will experiment with four different models, each with a different number of layers, which, according to the algorithm, has a significant effect on the calculation of the $\epsilon_x^{\max}$. The models are structured as follows:

- **Model 1**: Consists of a single fully connected layer with 16 units.
- **Model 2**: Includes two fully connected layers, with 128 units in the first layer and 16 units in the second.
- **Model 3**: Consists of three fully connected layers, with 1024 units in the first layer, 512 units in the second, and 128 units in the third.
- **Model 4**: Uses four fully connected layers with 2048, 1024, 512, and 128 units in the respective layers.

Each model is designed to end with a single output, as required by the algorithm. The prediction of each model is determined by the sign of the output value. After training, the models achieved the following accuracies on the test dataset:

- **Model 1**: 81.9%
- **Model 2**: 84.0%
- **Model 3**: 84.5%

- **Model 4**: 84.8%

The shallow neural network clearly has the lowest accuracy, while the network with the most layers achieves the highest accuracy. Overall, the accuracy across the four models is not very high, which may be attributed to the limited training dataset of only 1000 images.

We now consider the maximization Problem (4.18). The conditions of Proposition 4.3.5 are satisfied because the sigmoid activation function is continuous and monotonically increasing, and our dataset includes both labels. Therefore, Proposition 4.3.5 guarantees that, for every sample x in our test set with a nonempty feasible set, there exists a unique optimal solution to (4.18), with δ set to machine precision.

To find the $\epsilon_x^{\max}$, we compute the root ϵ_x^0 of f_x^{low} or f_x^{up}, then subtract a value of $\gamma = 10^{-7}$ from it. This ensures that $f_x^{\text{up}}(\epsilon_x^0 - \gamma) \leq -\delta$ or $f_x^{\text{low}}(\epsilon_x^0 - \gamma) \geq \delta$. The value of γ was determined experimentally. For $\tilde{\gamma} \leq 10^{-8}$ there were instances such that $f_x^{\text{low}}(\epsilon_x^0 - \tilde{\gamma}) < \delta$. The roots ϵ_x^0 were calculated using Brent's method [4], a root finding method that combines root bracketing, bisection, and inverse quadratic interpolation. For this method, the bracketing interval was set to $[0, 1]$, with the maximum number of iterations limited to 50. This approach allows us to approximately solve the maximization Problem (4.18), and find an approximation for $\epsilon_x^{\max}$.

We will calculate the $\epsilon_x^{\max}$ values for each model and each image in the test set where all four models classify correctly. This step is essential to ensure that the conditions in Proposition 4.3.2 hold, as otherwise there could be cases where $g(x) \neq h(x)$. Next, we will present boxplots of these $\epsilon_x^{\max}$ values for each model. Due to the significant differences in the magnitude of the $\epsilon_x^{\max}$ across the models, we will create two separate plots to effectively show the variations:

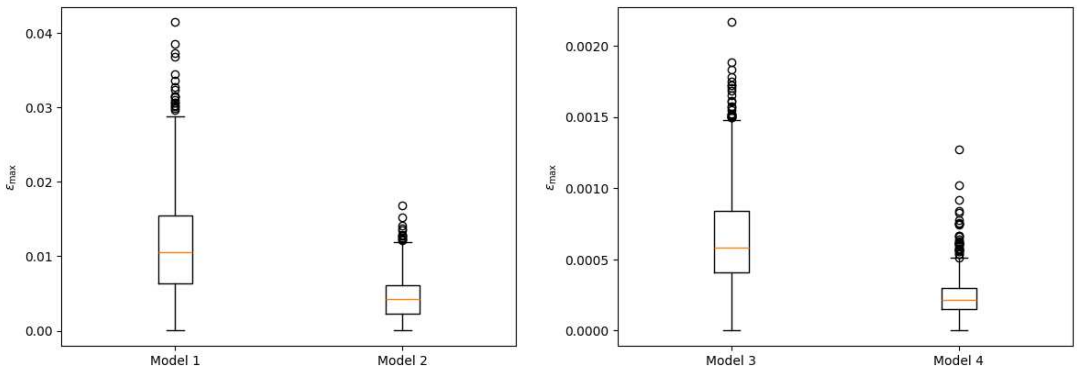

Figure 4.2: Boxplots of the $\epsilon_x^{\max}$ values of all four models. The orange line represents the median of the distributions.

The plot reveals significant variances of the $\epsilon_x^{\max}$ values across the different trained

models. The means of the $\epsilon_x^{\max}$ for each model are as follows:

- **Model 1**: 0.01139
- **Model 2**: 0.00447
- **Model 3**: 0.00065
- **Model 4**: 0.00024

These results suggest a potential correlation between the depth of the neural network and the magnitude of $\epsilon_x^{\max}$. Specifically, deeper models tend to exhibit smaller $\epsilon_x^{\max}$ values. We conclude that the effectiveness of Algorithm 6 reduces significantly with deeper neural networks. For MNIST images, where the typical perturbation for ℓ^∞-attacks is $\epsilon = 0.3$, deeper models provide only limited information about robustness. In contrast, Model 1, which has only one layer, yields more meaningful bounds for certain images in the test set. However, even in this case, the average of the $\epsilon_x^{\max}$ is more than 25 times smaller than the typical perturbation used in attacks.

We will only focus on Model 1 in the upcoming experiments, as it provides the most relevant bounds. The following plot shows the histogram of the $\epsilon_x^{\max}$ values for this model.

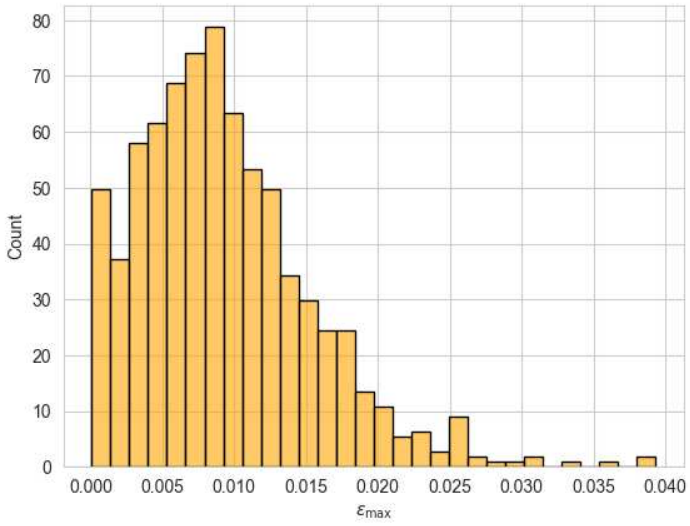

Figure 4.3: Histogram of the $\epsilon_x^{\max}$ values from Model 1, using only the correctly classified images from the test set.

We observe that relevant bounds are obtained only for a subset of the correctly classified images of the test set. In particular, less than 40% of the $\epsilon_x^{\max}$ values are greater than 0.01, but also more than 95% are greater than 0.001. One can interpret this also from another point of view: We know that for a correctly classified image x with its

corresponding $\epsilon_x^{\max}$, there is no regular adversarial example of x in $B^{\infty}_{\epsilon_x^{\max}}(x)$. This implies that, for a given a image x, our model is robust against attacks with any perturbation $\epsilon' \leq \epsilon_x^{\max}$. Therefore, given an attack perturbation ϵ' we can, as in [15], determine the proportion of robust examples in the test set, which serves as a lower bound for the accuracy.

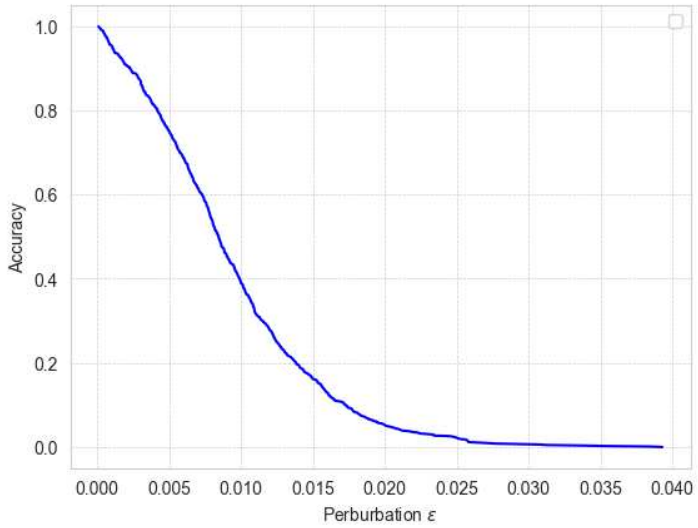

Figure 4.4: Lower bounds of the accuracy of Model 1 against increasing ℓ^{∞}-attacks.

Figure 4.4 can be interpreted as follows: For example, it shows that Model 1 achieves at least an accuracy of 39% against any ℓ^{∞}-attack, as long as the attack perturbation is less than 0.01. Note that this analysis only considers images that were correctly classified.

We now empirically test our method by attacking each correctly classified test image, with its computed $\epsilon_x^{\max}$. As in Section 2.3, we use the PGD attack (Algorithm 3). Although the PGD attack was not introduced for regular adversarial examples, it remains usable here. Suppose we want to check if the correctly classified test image x has a regular adversarial example in $B^{\infty}_{\epsilon_x^{\max}}(x)$. The PGD attack aims to find a $x^{\text{adv}} \in B^{\infty}_{\epsilon_x^{\max}}(x)$ such that $h(x^{\text{adv}}) \neq h(x)$. If no such x^{adv} is found, we conclude that for every perturbed $x' \in B^{\infty}_{\epsilon_x^{\max}}(x)$, created using PGD, one of the following must be true:

$$g(x) \neq g(x') \quad \text{or} \quad g(x) = g(x') = h(x) = h(x').$$

According to Definition 4.1.1, the image x' is therefore no regular adversarial example.

For the PGD attack, we assign a unique perturbation and step size to each image. The perturbation is set to $\epsilon = \epsilon_x^{\max}$ and we use 40 iterations with a step step size of $2\epsilon/40$. This configuration ensures that the attack can reach the boundary of B^{∞}_{ϵ} and make several iterations at that boundary. We observed that with these settings, the PGD attack was unable to find an adversarial example and therefore also no regular adversarial

example. This, therefore, also shows empirically that our Algorithm 6 works successfully.

We also evaluate the computational time required to determine $\epsilon_x^{\max}$ for a given image across various models. All experiments are conducted on an Intel(R) Core(TM) i5-7200U CPU @ 2.50GHz, yielding the following results:

Different Models	Model 1	Model 2	Model 3	Model 4
Time	0.13862	0.16655	0.63330	0.87864

Table 4.1: Average computational time in seconds for the different models to compute the $\epsilon_x^{\max}$ value for a single image.

As expected, the computational time is shortest for the shallow Model 1. However, even for this model, the time required to compute a single $\epsilon_x^{\max}$ is not negligible. This could be problematic for larger datasets. It would be worth exploring whether using an alternative root-finding method, other than Brent's method, could significantly reduce the computational time for solving Problem (4.18).

Additionally, we investigated whether there is a relationship between loss/accuracy and $\epsilon_x^{\max}$, but no clear correlation was found. Further experiments were carried out to see if adversarial training had any effect on magnitude of the $\epsilon_x^{\max}$, but these experiments also did not reveal a clear relationship.

5 Appendix

Theorem 5.0.1 ([23], Thm. 10.2). *Let f be a convex function on $\mathbb{R}^n$, an let S be any locally simplicial subset of $\operatorname{dom} f$. Then f is upper semi-continuous relative to S. In particular, if f is lower-semicontinuous, then f is continuous relative to S. We say that S is locally simplicial, if for all $x \in S$ there exist a finite collection of simplicies $S_1, \dots, S_m$ such that, for some neighborhood U of x,*

$$U \cap (S_1 \cup \cdots \cup S_m) = U \cap S.$$

Theorem 5.0.2 ([22], Thm. 9.6). *Let $m, d \in \mathbb{N}$, $\Omega \subset \mathbb{R}^d$, $f : \Omega \longrightarrow \mathbb{R}$ and let $x_1 \dots, x_m \in \Omega$, $y_1, \dots, y_m \in \mathbb{R}$ satisfy*

$$f(x_i) = y_i \quad \text{for all} \quad i = 1, \dots, m$$

and

$$Lip(f) = \sup_{x \neq y} \frac{|f(x) - f(y)|}{||x - y||_1} \geq \sup_{i \neq j} \frac{|y_i - y_j|}{||x_i - x_j||_1} =: \tilde{M} > 0.$$

Further, let $M \geq \tilde{M}$, then there exists a ReLU neural network Φ, which is C-Lipschitz (with respect to the $||.||_1$ norm), where $C \leq M$. Further Φ interpolates the data, depth$(\Phi) = O(\log(m))$, width$(\Phi) = O(dm)$ and all weights of Φ are bounded in absolute value by $\max\{M, \max\{|y_i| : i = 1, \dots, m\}\}$.

Bibliography

[1] Bartlett, P. L., Jordan, M. I. and McAuliffe, J. D. [2006], 'Convexity, classification, and risk bounds', *Journal of the American Statistical Association* **101**(473), 138–156.

[2] Boyd, S. and Vandenberghe, L. [2004], *Convex Optimization*, Cambridge University Press.

[3] Brendel, W., Rauber, J. and Bethge, M. [2018], 'Decision-based adversarial attacks: Reliable attacks against black-box machine learning models'.
URL: *https://arxiv.org/abs/1712.04248*

[4] Brent, R. P. [1973], *Algorithms for Minimization Without Derivatives*, Prentice-Hall, Englewood Cliffs, NJ.

[5] Carlini, N. and Wagner, D. A. [2016], 'Towards evaluating the robustness of neural networks', *CoRR* **abs/1608.04644**.
URL: *http://arxiv.org/abs/1608.04644*

[6] Danskin, J. M. [1966], 'The theory of max-min, with applications', *SIAM Journal on Applied Mathematics* **14**(4), 641–664.
URL: *https://doi.org/10.1137/0114053*

[7] Ebrahimi, J., Rao, A., Lowd, D. and Dou, D. [2018], HotFlip: White-box adversarial examples for text classification, *in* 'Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)', pp. 31–36.

[8] Fix, J., Frezza-Buet, H. and Geist, M. [2024], *Machine Learning*.
URL: *https://frezza.pages.centralesupelec.fr/teachml2/Poly/Poly-ML-SDImetz.pdf*

[9] Goodfellow, I., Bengio, Y. and Courville, A. [2016], *Deep Learning*, MIT Press. `http://www.deeplearningbook.org`.

[10] Goodfellow, I. J., Shlens, J. and Szegedy, C. [2015], Explaining and harnessing adversarial examples, *in* Y. Bengio and Y. LeCun, eds, 'Proceedings of the 3rd International Conference on Learning Representations (ICLR)', San Diego, CA, USA.

[11] Hinton, G., Deng, L., Yu, D., Dahl, G., Mohamed, A.-r., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Kingsbury, B. et al. [2012], 'Deep neural networks for acoustic modeling in speech recognition', *IEEE Signal Processing Magazine* **29**.

[12] Krizhevsky, A., Sutskever, I. and Hinton, G. E. [2012], Imagenet classification with deep convolutional neural networks, *in* F. Pereira, C. Burges, L. Bottou and K. Weinberger, eds, 'Advances in Neural Information Processing Systems', Vol. 25, Curran Associates, Inc.

[13] Kurakin, A., Goodfellow, I. J. and Bengio, S. [2017], Adversarial machine learning at scale, *in* 'International Conference on Learning Representations (ICLR)'.

[14] Latorre, F., Krawczuk, I., Dadi, L. T., Pethick, T. and Cevher, V. [2023], Finding actual descent directions for adversarial training, *in* 'Proceedings of the Eleventh International Conference on Learning Representations (ICLR)'.

[15] Li, B., Chen, C., Wang, W. and Carin, L. [2019], 'Certified adversarial robustness with additive noise'.
URL: *https://arxiv.org/abs/1809.03113*

[16] Madry, A., Makelov, A., Schmidt, L., Tsipras, D. and Vladu, A. [2018], 'Towards deep learning models resistant to adversarial attacks', *International Conference on Learning Representations* .

[17] McCulloch, W. S. and Pitts, W. [1943], 'A logical calculus of the ideas immanent in nervous activity', *The Bulletin of Mathematical Biophysics* **5**(4), 115–133.

[18] Moosavi-Dezfooli, S., Fawzi, A. and Frossard, P. [2015], 'Deepfool: a simple and accurate method to fool deep neural networks', *CoRR* **abs/1511.04599**.
URL: *http://arxiv.org/abs/1511.04599*

[19] Nielsen, M. A. [2018], 'Neural networks and deep learning'.
URL: *http://neuralnetworksanddeeplearning.com/*

[20] Papernot, N., McDaniel, P., Goodfellow, I., Jha, S., Celik, Z. B. and Swami, A. [2017], 'Practical black-box attacks against machine learning'.
URL: *https://arxiv.org/abs/1602.02697*

[21] Papernot, N., McDaniel, P., Jha, S., Fredrikson, M., Celik, Z. B. and Swami, A. [2016], The limitations of deep learning in adversarial settings, *in* '2016 IEEE European Symposium on Security and Privacy (EuroSP)', pp. 372–387.

[22] Petersen, P. and Zech, J. [2024], 'Mathematical theory of deep learning'.
URL: *https://arxiv.org/abs/2407.18384*

[23] Rockafellar, R. [1970], Convex analysis, Princeton University Press, Princeton, NJ.

[24] Rumelhart, D. E., Hinton, G. E. and Williams, R. J. [1986], 'Learning representations by back-propagating errors', *nature* **323**(6088), 533–536.

[25] Schott, L., Rauber, J., Bethge, M. and Brendel, W. [2018], 'Towards the first adversarially robust neural network model on mnist'.
URL: *https://arxiv.org/abs/1805.09190*

[26] Schott, L., Rauber, J., Brendel, W. and Bethge, M. [2018], 'Robust perception through analysis by synthesis', *CoRR* **abs/1805.09190**.
URL: *http://arxiv.org/abs/1805.09190*

[27] Shafahi, A., Najibi, M., Ghiasi, A., Xu, Z., Dickerson, J., Studer, C., Davis, L. S., Taylor, G. and Goldstein, T. [2019], 'Adversarial training for free!'.
URL: *https://arxiv.org/abs/1904.12843*

[28] Stutz, D., Hein, M. and Schiele, B. [2018], 'Disentangling adversarial robustness and generalization', *CoRR* **abs/1812.00740**.
URL: *http://arxiv.org/abs/1812.00740*

[29] Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I. J. and Fergus, R. [2014], Intriguing properties of neural networks, *in* Y. Bengio and Y. LeCun, eds, '2nd International Conference on Learning Representations, ICLR 2014, Conference Track Proceedings', Banff, AB, Canada.
URL: *http://arxiv.org/abs/1312.6199*

[30] Tsipras, D., Santurkar, S., Engstrom, L., Turner, A. and Madry, A. [2019], 'Robustness may be at odds with accuracy'.
URL: *https://arxiv.org/abs/1805.12152*

[31] Tu, J., Ren, M., Manivasagam, S., Liang, M., Yang, B., Du, R., Cheng, F. and Urtasun, R. [2020], Physically realizable adversarial examples for lidar object detection, *in* 'Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition', pp. 13716–13725.

[32] Wong, E., Rice, L. and Kolter, J. Z. [2020], 'Fast is better than free: Revisiting adversarial training'.
URL: *https://arxiv.org/abs/2001.03994*

[33] Wu, X. and Chen, J. [2019], Robust attribution regularization, *in* 'ALTACognita Workshop'.
URL: *https://www.altacognita.com/robust-attribution/*

[34] Xu, H., Ma, Y., Liu, H., Deb, D., Liu, H., Tang, J. and Jain, A. K. [2019], 'Adversarial attacks and defenses in images, graphs and text: A review'.
URL: *https://arxiv.org/abs/1909.08072*

[35] Zhang, H., Yu, Y., Jiao, J., Xing, E. P., Ghaoui, L. E. and Jordan, M. I. [2019], 'Theoretically principled trade-off between robustness and accuracy', *CoRR* **abs/1901.08573**.
URL: *http://arxiv.org/abs/1901.08573*

[36] Zhou, W., Hou, X., Chen, Y., Tang, M., Huang, X., Gan, X. and Yang, Y. [2018], Transferable adversarial perturbations, *in* 'Proceedings of the European Conference on Computer Vision (ECCV)'.

[37] Zügner, D., Akbarnejad, A. and Günnemann, S. [2018], Adversarial attacks on neural networks for graph data, *in* 'Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery amp; Data Mining', KDD '18, ACM. **URL:** *http://dx.doi.org/10.1145/3219819.3220078*