Towards deep neural networks robust to adversarial examples

Matyasko, Alexander

2020

Matyasko, A. (2020). Towards deep neural networks robust to adversarial examples. Doctoral thesis, Nanyang Technological University, Singapore.

https://hdl.handle.net/10356/143316

https://doi.org/10.32657/10356/143316

Towards Deep Neural Networks Robust to Adversarial Examples

Alexander Matyasko

School of Electrical & Electronic Engineering

A thesis submitted to the Nanyang Technological University
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy

2020

Statement of Originality

I hereby certify that the work embodied in this thesis is the result of original research, is free of plagiarised materials, and has not been submitted for a higher degree to any other University or Institution.

July 30, 2020

.....

Date

.....

Alexander Matyasko

Supervisor Declaration Statement

I have reviewed the content and presentation style of this thesis and declare it is free of plagiarism and of sufficient grammatical clarity to be examined. To the best of my knowledge, the research and writing are those of the candidate except as acknowledged in the Author Attribution Statement. I confirm that the investigations were conducted in accord with the ethics policies and integrity standards of Nanyang Technological University and that the research data are presented honestly and without prejudice.

July 30, 2020

Date

L.P.Chau

Prof. Lap Pui Chau

Authorship Attribution Statement

This thesis contains material from papers accepted at conferences in which I am listed as an author.

Chapter 3 is published as Alexander Matyasko, Lap Pui Chau. Margin maximization for robust classification using deep learning. In International Joint Conference on Neural Networks (IJCNN). 300–307 (2017). DOI: 10.1109/IJCNN.2017.7965869.

The contributions of the co-authors are as follows:

- I proposed the idea, designed the experiments, and prepared the manuscript.
- The manuscript was revised by Lap Pui Chau.

Chapter 4 is published as Alexander Matyasko, Lap Paui Chau, Improved Network Robustness with Adversary Critic. In Conference on Neural Information Processing Systems (NeurIPS). 10578–10587 (2018).

The contributions of the co-authors are as follows:

- I proposed the idea, designed the experiments, and prepared the manuscript.
- The manuscript was revised by Lap Pui Chau.

Chapter 5 is submitted and undergoing review in IEEE Transactions on Neural Networks and Learning Systems as PDPGD: Primal-Dual Proximal Gradient Descent Adversarial Attack.

The contributions of the co-authors are as follows:

- I proposed the idea, designed the experiments, and prepared the manuscript.
- The manuscript was revised by Lap Pui Chau.

July 30, 2020
.....
Date

.....
Alexander Matyasko

Acknowledgements

I want to express my biggest gratitude to my advisor, Associate Professor Lap-Pui Chau, who has provided support and gave me freedom for my research. When I arrived in Singapore, he welcomed and guided me throughout my studies. Without a doubt, he set an example of what a real researcher should be.

I want to thank my colleagues in Singapore. Without your guidance and support, it will be so much more challenging to keep my focus during Ph.D. Special thanks go to Jie Chen for his insightful suggestions and guidance during my Ph.D. life. Special thanks go to Cheen-Hau Tan, whose working tenacity set an example for me. Special thanks go to Junhui Hou for many discussions that we had. I also want to thank Yi Wang, Yun Ni, Xiaoxi Ma, Shengyu Nan, and Zhen-Peng Bian for their help during my studies. Besides that, I want to thank the technical staff of Nanyang Technological University. In particular, special thanks go to Teng Kwee Ng.

I want to thank my friends, who made my stay so much more memorable and joyful. When I came to Singapore, I did not know anything or anyone here. Years later, this city grew onto me. I had many fond memories and got even more friends here. Naturally, Singapore became my second home.

I want to thank my parents for their unwavering support and encouragement through the difficult times I had in my study. Many times I had difficulties, but they always found words to give me their support even thousands of miles apart. I also want to thank personally to my partner, Syimah, who helped me to get organized, especially towards the end of my Ph.D.

Finally, I would like to thank Nanyang Technological University for providing a generous scholarship and state-of-the-art facilities for my study.

"Anything that can go wrong will go wrong."

—Murphy's law

To my dear mom and grandma Anna

Abstract

Deep neural networks have achieved success in a variety of applications in the last decade, surpassing human performance in complex arrays of tasks, e.g. identification of objects, interpretation of natural language, and text-to-speech transcription. Nowadays, deep learning has become the dominant approach for any problem where learning from data is necessary. If the data is the "nail", then deep learning is the "hammer". Nevertheless, state-of-the-art deep neural networks are prone to small perturbations in the input data. For example, recent experiments have shown that adding adversarial noise to inputs creates images that are optically indistinguishable from the original data, but the neural network misclassifies it with high confidence. A similar vulnerability has been identified in the task of natural language understanding where targeted adversarial perturbation, which changes only a few words, can change the semantic meaning of the whole sentence. Likewise, speech recording can be altered, such that it transcribes to any desired sentence, but remains over 99 percent similar to the original soundtrack. These adversarially crafted modifications to the input, so-called *adversarial examples*, are neural network "blind spots" and is the main subject of this dissertation.

Human perception is remarkably robust to different variations and various distortions in the input. We can reliably perceive and accurately navigate the world around us without even realizing the difficulty of the task. Likewise, a machine learning system should be robust to changes in the environment. If some input distortion changes the prediction of the model, the disturbance should be semantically meaningful to a human observer. However, intriguingly, an input adversarial distortion for deep neural networks can be constructed, such that original and modified inputs are perceptually indistinguishable to the human observer. The lack of model robustness to imperceptible perturbations in the input is counter-intuitive and undesirable and puts into question the ability of deep neural networks to generalize to the unseen data. Additionally, the existence of *adversarial examples* reduces model interpretability and restricts applications of deep learning models in safety and security-critical environments, e.g. face verification, self-driving cars.

In this Ph.D. thesis, we outline the problem of adversarial examples and show several partial solutions to it. The existence of adversarial examples has spurred significant interest in deep learning research. The research on adversarial examples can be broadly divided into research on attacks and research on defenses. We make original contributions to both fields of research. First of all, we establish a connection between classifier margin and its robustness. We generalize a support vector machine (SVM) margin maximization objective to deep neural networks. We also prove that our formulation is equivalent to robust optimization. In the subsequent work, we suggest that ideally, adversarial examples for the robust classifier should be indistinguishable from the regular data. Unlike approaches based on robust optimization, we do not require that an input noise is label non-changing. We formulate a problem of learning robust classifier in the framework of generative adversarial networks (GAN), where an auxiliary network, or an adversary discriminator, is trained to distinguish regular and adversarial data. Then, a robust classifier is trained to classify the original inputs correctly and to fool the discriminator with its adversarial examples. Finally, accurately estimating the model's robustness is a challenging task. Existing attack methods require multiple restarts or do not explicitly minimize the norm of the perturbation. To address the above limitations, we propose a primal-dual proximal gradient attack algorithm. Our attack is fast and accurate. We directly solve the attacker's problem for any $\mathcal{L}_p$-norm for which the proximal operator can be computed in the closed-form. We hope that our attack will be considered as a benchmark for a future comparison between different defense methods.

In summary, we present two defenses and one white-box attack in this thesis. Future efforts should address the robustness of deep neural networks to unrestricted adversarial examples, provide strong theoretical guarantees on the model's performances, and verify model robustness for a comprehensive comparison of various defenses.

Contents

List of Figures

List of Tables

Symbols and Acronyms

Symbols

$\mathcal{R}^n$	the n-dimensional Euclidean space
$\| \cdot \|_p$	the l_p-norm of a vector
∇f	the gradient vector
$\mathcal{C}^k$	the function with continuous partial derivatives up to k orders
Π_C	the projection operator on the set of constraints C
$H(\cdot)$	Hessian, the second-order of derivatives
$KL(\cdot)$	Kullback–Leibler divergence function

Acronyms

NN	Neural Network
DNN	Deep Neural Network
ConvNet/CNN	Convolutional Neural Network
RO	Robust optimization
FGSM	Fast Gradient Sign Method
BIM	Basic Iterative Method
AT	Adversarial Training
VAT	Virtual Adversarial Training
GAN	Generative Adversarial Network
w.r.t.	with respect to
i.i.d.	independent and identically distributed

Chapter 1

Introduction

Human visual perception is surprisingly robust when it comes to changes in object appearance, shape, and pose. We can reliably perceive and accurately navigate the world around us without even realizing the difficulty of the task. An open question in computer vision is: can computers understand the world around us with the same accuracy and reliability as humans do? Creating such algorithms is the main quest in computer vision. Nature's design of the human mind inspires us to develop such algorithms. Deep neural networks (DNNs) have emerged as one universal algorithm in recent years. DNNs mimic information processing in the human brain: DNNs have multiple processing layers that extract a higher-level representation of the data, which is similar to the visual cortex. With recent advances, deep neural networks have surpassed human performance on the image classification task [1], which suggests that the problem of recognizing objects in the images may be solved or closed to being solved. Besides that, deep neural networks found success on other perceptual tasks, such as speech recognition [2] and machine translation [3]. DNNs have been applied to almost every problem. We can say without exaggeration: If the "data" is the nail, then deep neural networks are regarded as the "hammer".

1.1 Motivation

The success of deep neural networks in countless applications is undeniable. Still, DNNs and their properties remain poorly understood. They are often treated

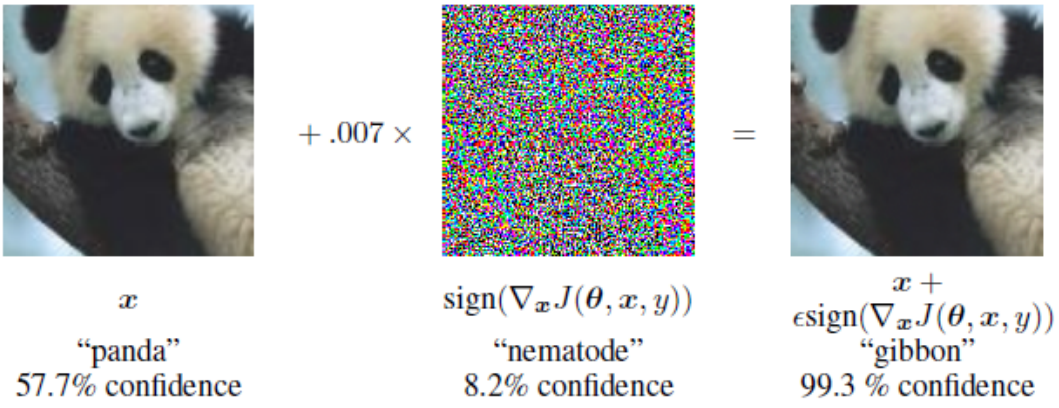


FIGURE 1.1: An adversarial example for GoogleNet network [4] was generated using Fast Gradient Sign method [5]. GoogleNet correctly classified the image on the left as a "panda" class. An adversarial image (shown on the right) was constructed by adding small imperceptible perturbation (shown in the middle) to the original image. The neural network misclassified the adversarial image as a "gibbon". As we can see, the original and adversarial images are visually indistinguishable to the human eye, yet, GoogleNet predicted a different label for the adversarial image.

as a black-box input-output mapping: "If it ain't broke, don't fix it". Recent experiments have shown that state-of-the-art deep neural networks are broken in one intriguing way: they are sensitive to small imperceptible perturbations in the input data [5, 6]. The intriguing result is that the changes required to shift the prediction of DNNs are invisible to a human eye. For example, Figure 1.1 shows an example of the image that was classified correctly as a "panda" but was misclassified as a "gibbon" after adding small perturbation. Yet, the perturbated image still looks like a "panda" to us. These examples, so-called *adversarial examples*, are "blind spots" or optical illusions for deep neural networks [7]. One can argue that such perturbations are unnatural and unlikely to appear in the physical world. However, since the discovery of this baffling property, [8, 9] attacked image classifier in the physical world. Adversarial examples for DNNs are not limited to images. The attacks on DNNs have been applied in other domains. [10] changed a few words in the sentence to flip the sentiment of the review. [11–13] added unnoticeable background noise, so that the audio was transcribed to the desired sentence.

Human visual system is remarkably robust to changes in the object's appearance, shape, and pose. Computer vision systems should be as accurate and reliable as a human visual system. The lack of DNNs robustness to small, invisible perturbations is counter-intuitive, undesirable, and risky. If the model exceeds human-level performance on the task, how is it possible that the same model performs worse

than random on the inputs that are only slightly different? The existence of adversarial examples also puts into question the generalization ability of DNNs: do they learn how to perform the task, or do they learn how to parrot the answers? Additionally, it limits the use of the deep neural networks in security-critical applications where the adversary can potentially exploit such vulnerability. For example, the attacker in [9, 14] rendered a "graffiti" pattern on a road sign, which looks harmless to a human, but this pattern tricked road sign recognition system. Similarly, the attacker in [15] hacked face recognition system. The authors printed a pattern on the eyeglass frame that let the attacker remain undetected or impersonate another person. If AI system is not robust, it is difficult to trust its decisions.

The problem of adversarial examples has spurred tremendous interest in the safety of DNNs [16] and led to interesting developments in deep learning research [5, 17]. Numerous techniques have been proposed to attack DNNs and improve robustness of DNNs to adversarial noise. The field of research in this area can be broadly divided into the research on attacks (Team A) and the research on the defenses (Team B):

- Team A searches for the novel ways to attack DNNs: white-box and black-box attacks; attacks on image classification, detection, image segmentation systems; attacks in other domains, e.g. text, speech.
- Team B develops new methods to safeguard against all existing attacks and potential future attacks: certifiable defenses; adversarial training and robust optimization approaches; detection of adversarial examples.

Two research teams play a cat-and-mouse game or an arms race: Team A introduces a novel attack algorithm that beats all existing defenses. In response, Team B develops a novel defense algorithm that safeguards the model from all existing attacks. This reactive strategy is impractical because we can attack DNNs in multiple ways, and we can not protect DNNs from future attacks. Instead, in this thesis, we strive to develop principal methods to improve robustness of DNNs against all possible attacks. Reducing the sensitivity of deep neural networks to adversarial noise does not only address the security concerns but can also improve the generalization ability of DNNs.

1.2 Contributions and Dissertation Outline

In this Ph.D. thesis, we outline the problem of adversarial examples and show several partial solutions to it. We now provide an overview of the research problems investigated in this dissertation. We make three key research contributions in this work:

- In Chapter 3, we present a novel defense method that applies margin maximization principle to improve the robustness of deep neural networks. Our formulation generalizes well-celebrated support vector machine classifier (SVM). Unlike SVM, decision boundary of DNNs is highly non-linear. We approximate the decision boundary, so that the margin, or the distance to the decision boundary, can be efficiently computed. We maximize margin approximation to improve robustness. We prove that the proposed objective is equivalent to a robust optimization (RO) problem. In experiments on image classification tasks, we verify that the proposed regularization method increases the robustness of DNNs to adversarial noise. Our defense method achieves state-of-the-art robustness results on MNIST and CIFAR10 classification tasks.

- In Chapter 4, we highlight some limitations of the robust optimization approach. Robust optimization (RO) relies on a problem-specific uncertainty that needs to be selected before robust training. For many perceptual problems, it is not clear which perturbations do not change the input label. Instead, we proposed an orthogonal to RO approaches. We asked ourselves a question: what is an ideal solution to the problem of adversarial examples? We argue that, ideally, what confuses neural networks should be confusing to humans. Mathematically, it is equivalent to minimizing the probabilistic distance between the distribution of natural examples and adversarial examples. For a robust classifier, its adversarial examples should be indistinguishable from the natural data. To achieve this, we employ an auxiliary network, or adversary critic, which distinguishes regular and adversarial examples. A robust classifier is then trained to classify the original inputs correctly and to fool the discriminator with its adversarial examples. We show in the experiments that our defense improves robustness both numerically and perceptually. Notably, our defense significantly reduces a perceptual gap

between deep neural networks and time-limited humans. In the experiments with human annotators on Amazon MTurk, we show that when human annotators are asked to label adversarial images, the annotators agree with its adversarial label in 90% of the time.

- In Chapter 5, we present our novel l_p-norm adversarial attack. Evaluating the adversarial robustness of various models has proven to be extremely challenging. Existing adversarial attacks require thousands of iterations and multiple restarts to find the minimal adversarial perturbation [18]. The second type of attacks, fast attacks [5, 17], ignore the norm minimization penalty and solve a simpler perturbation-bounded problem. This chapter introduces an attack that directly solves the original non-convex constrained minimization problem. We interpret optimizing the Lagrangian of the adversarial attack as a two-player game: the first player minimizes the Lagrangian wrt the adversarial noise; the second player maximizes the Lagrangian wrt the regularisation penalty. Our attack algorithm simultaneously optimizes primal and dual variables to find the minimal adversarial perturbation. In addition, for non-smooth l_p-norm minimization, e.g. l_∞, l_1, and l_0-norms, we introduce primal-dual proximal gradient descent attack. Experimental results show that our attack method is significantly stronger than current state-of-the-art attacks on MNIST and CIFAR-10 datasets against unregularized and adversarially trained models. We hope that our attack will be used as a benchmark for a future comparison between different defense methods.

Before diving into technical details, we review machine learning basics and the basics of neural networks in Chapter 2. Then, we introduce a problem of adversarial examples. We highlight the main challenges in reducing the vulnerability of deep neural networks to adversarial noise. We survey recent attack and defense methods and discuss the limitations of existing approaches.

Chapter 2

Background and Literature Review

Human mind allows us to recognize objects and places seamlessly. In their first efforts in the 1960s, researchers attempted to construct programmable rules for extracting useful information from images. However, it soon became clear that the complexity of visual data makes the task of designing extraction rules impossible. The world around us is just too complex to be described with simple rules. The solution is to enable computers to learn directly from the data. Machine learning algorithms allow the computer to learn from experience without being explicitly programmed to perform the task at hand. This chapter provides background information about machine learning in general. We cover empirical risk minimization, discuss regularization methods, and the issue of data overfitting. We describe the basic principles underlying deep neural network models. Then, we introduce the problem of adversarial examples. We present some recent attacks on deep neural networks and defenses against adversarial noise.

2.1 Machine Learning Basics

Artificial intelligence (AI) has captivated our imagination for decades. If developed, it might be the greatest humankind invention. The first attempts to program intelligent machines can be traced back to the 1960s. After the initial optimism, the researchers soon learned that the complexity of the real world makes it infeasible

to program a computer with explicit set of rules for performing the task. Learning in some form is necessary. While the way a machine learns is different from the way a human learns, machine learning has been largely inspired by the principles of continuous improvement from experience in human learning. Machine learning research can thus potentially shed light on the principles that govern human intelligence.

Machine learning is the study of algorithms that enable computers to learn from data. A commonly cited and more formal definition of machine learning is "A computer program is said to learn from experience $\mathbb{E}$ with respect to some class of tasks $\mathbb{T}$ and performance measure $\mathbb{P}$ if its performance at tasks in $\mathbb{T}$, as measured by $\mathbb{P}$, improves with experience $\mathbb{E}$" [19]. We can describe many learning problems in this framework. As an example, let's consider a basic image classification task:

Task $\mathbb{T}$: is to recognize objects in the image.

Performance measure $\mathbb{P}$: is a percentage of correctly classified images.

Experience $\mathbb{E}$: is a dataset of images with the target labels.

Machine learning algorithms can be divided into three categories according to the amount of information available about the target: supervised learning, unsupervised learning, and reinforcement learning. In this work, we consider supervised learning problems, in particular, image classification problems. For supervised learning, a computer is presented with inputs $\mathbf{X}$ and desired outputs $\mathbf{Y}$. The goal is to learn a mapping from the input space to the target label space $f : \mathbb{X} \to \mathbb{Y}$. Input to the image classifier f is an image $\mathbf{x} \in \mathbb{R}^n$, where n is the dimensionality of the image. Target output for classification with k-labels is $y \in \{1, \ldots, k\}$. Output is usually encoded using 1-hot coding vector of length k with 1 at position l if $y = l$ and 0 at all other positions.

Learning or finding a unique mapping $f : \mathbb{X} \to \mathbb{Y}$ is an *ill-posed problem*. To show that, let us consider a problem of learning boolean function $\{0,1\}^d \mapsto \{0,1\}$ with d inputs. There are 2^d possible training examples and 2^{2^d} possible functions. $2^{2^d - N}$ functions remain after observing N examples. To find a unique solution, we need to observe the whole training set of the size 2^d. As we can see, the data by itself is not sufficient to find the solution. An additional set of assumptions about

hypothesis function must be introduced into a machine learning model. The set of assumptions that makes learning possible is called the *inductive bias*.

Machine learning models can be categorized into *parametric* and *non-parametric* models. An example of a *non-parametric* model is the nearest neighbor classifier. In our work, we consider parameter models. A parametric family of models defines a set of functions $\mathcal{F} = \{f_\theta \,|\, \theta \in \Theta\}$, where $f_\theta(x) = f_\mathcal{F}(x;\theta)$ is a mapping from the input space to the target space; $\theta \in \Theta$ are trainable model parameters, e.g. neural network weights. Additionally, the function family $\mathcal{F}$ can depend on non-trainable parameters or *hyperparameters*, e.g. the type and the number of layers in a neural network. A parametric family restricts the class of functions the model can represent. In this way, we provide an inductive bias to the model for learning about task $\mathbb{T}$.

Each member $f_\theta \in \mathcal{F}$ represents a particular instance of a machine learner. The goal of learning is to find the most suitable member f^* of a parametric family $\mathcal{F}$ for a given task $\mathbb{T}$ and a given performance measure $\mathbb{P}$. For parametric models, this is equivalent to finding optimal parameters $\theta^* \in \Theta$. Assume that training data was drawn independently and identically distributed (i.i.d.). If π is data distribution from which data was i.i.d. drawn, we can formalize learning as an optimization problem:

$$f^* = \arg\min_{f \in \mathcal{F}} \mathop{\mathbb{E}}_{(x,y)\sim\pi} \left[\mathcal{L}\left(f(x;\theta), y\right)\right] \tag{2.1}$$

Data distribution π is not available for most problems of interest. Instead, data is sampled from the empirical or training distribution $\mathbb{D}$:

$$\begin{aligned} \hat{f} &= \arg\min_{f \in \mathcal{F}} \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}\left(f(x_i;\theta), y_i\right) \\ &= \arg\min_{f \in \mathcal{F}} \mathop{\mathbb{E}}_{(x,y)\sim\mathbb{D}} \left[\mathcal{L}\left(f(x;\theta), y\right)\right] \end{aligned} \tag{2.2}$$

where m is a number of training examples.

Does the solution of eq. (2.2) convergences to the solution of eq. (2.1)? Fortunately, we can show that under mild conditions on the function family $\mathcal{F}$ empirical estimate $\hat{f}$ convergences to the optimal solution f^* in the limit of the number of examples $\lim_{m\to\infty} \hat{f} = f^*$. This principle is known as *empirical risk minimization principle*. *Empirical risk minimization* is at the foundation of many machine learning algorithms, including neural networks. To find a solution of Equation (2.2), we

can use any optimization technique, e.g. gradient descent, Newton's method, or a method tailored for the problem's structure. Next, we introduce neural networks as one example of a parametric machine learning model, which is a particular focus on this thesis.

2.2 Neural Networks

Artificial Neural network (ANN) is a parametric model, which has been introduced to machine learning over 60 years ago. Their creation and development were loosely inspired by the layered structure of the human brain. Neural networks for pattern recognition do not try to imitate the human mind per se, yet, much of the NN terminology and ideas have been borrowed from neuroscience. Spiking Neural Networks (SNNs) provide a much more accurate approximation of the computations in the human brain [20, 21]. However, SNNs are difficult to train due to the non-differentiability of the spiking activation function.

Neural network computation is represented as a directed graph of connected *processing units* or *neurons*. Each neuron receives an output from the adjacent neurons as an input and performs some transformation of the data, e.g. weighted sum or output thresholding. An output signal of the neuron is passed to the connected neurons. The neurons that perform related computations are organized in layers. For simplicity, we describe neural networks are as a sequence of layer transformations. The neural networks can be categorized into two groups: feedforward neural networks (FNN) and recurrent neural networks (RNN). In this thesis, we study the robustness of differentiable feedforward neural networks.

2.2.1 Feedforward Neural Networks

We start our review with a description of feedforward neural networks. Each *layer* of a feedforward neural network transforms an output from the previous layer and passes the transformed output as an input to the next layer. Let $f^{(l)}(\,\cdot\,; W_l)$ be the transformation function of the layer l where W_l are the layer parameters. Parameters of the neural network are a concatenation of all layer parameters $\mathbf{W} = [W_1, W_2, \ldots, W_L]$ where L represents the number of layers. Then, the neural

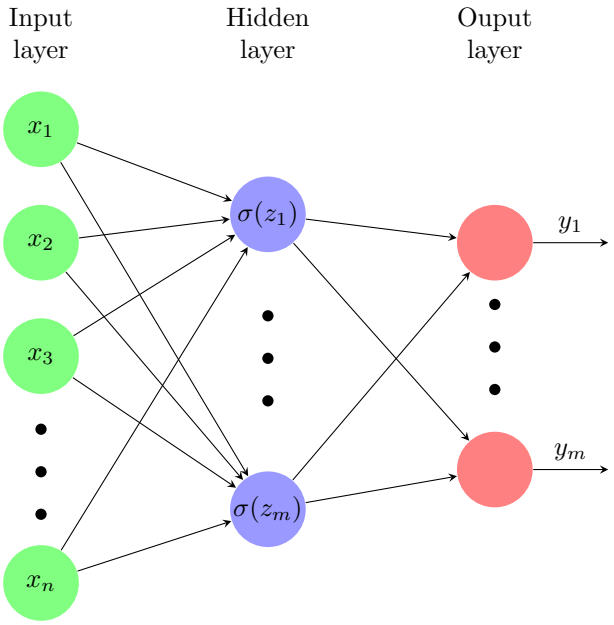


FIGURE 2.1: Feedforward neural network with two fully-connected layers. Each neuron is connected to all neurons in the previous layer, but the neurons in the same layer do not share any connections.

network f performs the following computation:

$$\begin{aligned} f(x;\mathbb{W}) &= \left(f^{(L)} \circ f^{(L-1)} \circ \ldots \circ f^{(1)}\right)(x;\mathbf{W}) \\ &= f^{(L)}\left(\ldots f^{(2)}\left(f^{(1)}\left(x;W_1\right);W_2\right)\ldots;W_L\right) \end{aligned} \tag{2.3}$$

The process of evaluating eq. (2.3) can be interpreted as *forward propagation of information*. Networks with directed acyclic graph structure are commonly called *feedforward*. Networks with cycles are called *recurrent*. In this thesis, we examine feedforward neural networks in detail.

Neural network layers is an active field of research. Layers differ in their pattern of connectivity and transformation function. The most common layer found in neural networks is a fully-connected layer. We show an example of a feedforward neural network with two fully-connected layers in Figure 2.1. Each neuron in a fully-connected layer is connected to all neurons in the previous layer, but neurons within a single layer share no connections. Each neuron in layer computes a weighted

average of the inputs x and applies an activation function $\sigma(\ \cdot\ )$:

$$\begin{aligned} z_j &= \sum_{i=1}^{n} w_{ij} x_i + b_j = w_j^T x + b_j \\ h &= \sigma(z) \end{aligned} \tag{2.4}$$

where w_j is the j-row of weight matrix W, b is the bias vector, and z is the vector of neurons activations. We can express the operation that a fully-connected layer performs using a single matrix-vector product as follows:

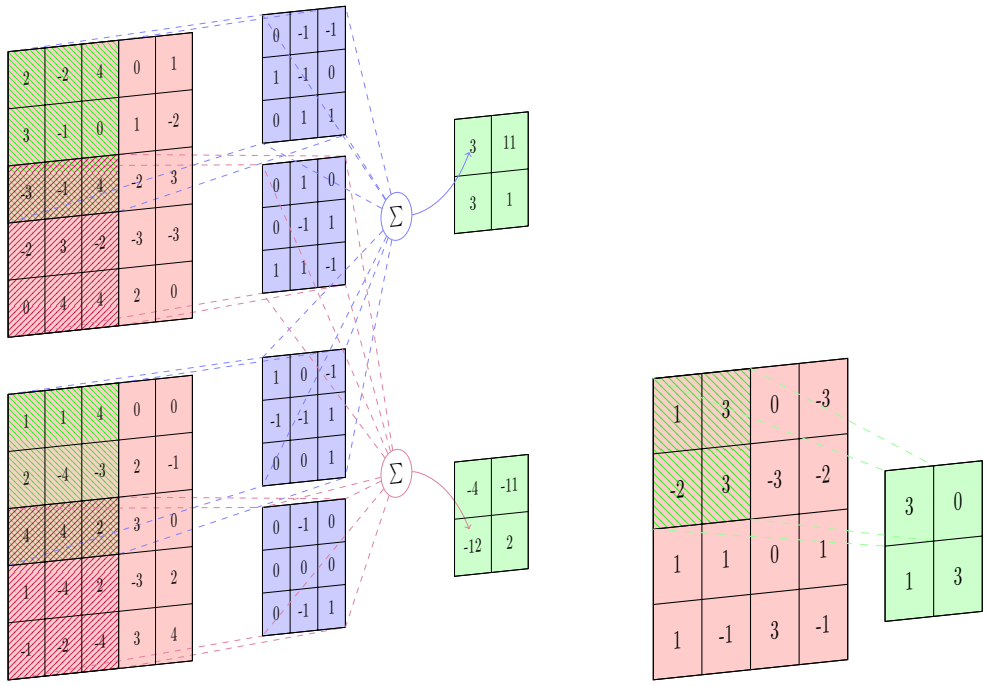
$$h = \sigma(Wx + b) \tag{2.5}$$

A fully-connected layer is a basic block for building a neural network. However, the number of layer parameters limits its applications to vision problems. Consider a fully-connected layer with n inputs and k outputs units. The number of the layer parameters scales as $(n \times k + k)\, L$, where L is the number of neural network layers. It is prohibitively large for image data (where $n,\ k \sim 1000$) and leads to *overfitting* even for simple computer vision tasks. The number of parameters can be reduced by using prior knowledge about the structure of natural images. In the section, we describe convolutional layers that use prior information about images.

2.2.2 Convolutional Neural Networks

Hubel and Wiesel [22] discovered that neurons in the human visual cortex are organized in a complex arrangement of two types of cells: simple and complex cells. Simple cells only respond to the specific edge-like pattern within their receptive field, while complex cells are locally invariant to the pattern's exact location. Discoveries of the simple and complex cells and selective receptive fields in the visual cortex had led to the development of convolutional neural networks [23, 24]. Convolutional neural networks implement the idea of simple and complex cells using convolutional and pooling layers, respectively. We describe the operation that the convolutional layer performs next.

A convolutional layer implements the idea of selective receptive fields found in the simple cells in the mammal visual cortex in the following way:



(A) Convolution operation. (B) Pooling operation.

FIGURE 2.2: Visualization of operations in convolutional neural networks. Figure 2.2a shows operation performed by convolutional layer. The filters F has 3×3 shape. The number of output features maps K is equal to 2. The stride is 2 and no zero padding is used. Each neuron in the output locally connected to the input and the weights are shared for different locations. Figure 2.2b shows operation performed by max-pooling layer.

Local connectivity: Each neuron is connected only to the subset of the neurons in the input layer.

Translation invariance: Weights are shared for different locations in the input because the same object can appear at any location in the image.

The convolutional layer can be conveniently described as a transformation of the input 3D tensor to the output 3D tensor. Let $X \in \mathbb{R}^{W_1 \times H_1 \times D}$ be the input 3D tensor, $W \in \mathbb{R}^{F \times F \times D \times K}$ be the 4D weights tensor where $F \times F$ is the size of filters, and K the number of the output feature maps. Other hyperparameters of the convolutional layer are a stride S and an input padding P. For simplicity, we describe the layer with stride 1 and zero-padding. The output of the layer with the $S = 1$ and $P = 0$ has shape $W_2 \times H_2 \times K$, where $W_2 = W_1 - F$ and $H_2 = H_1 - F$. The output at depth d is the result of the convolution of the d-filter over the input volume:

$$Z_{\cdot,\cdot,d} = X * W_{\cdot,\cdot,\cdot,d} \tag{2.6}$$

The number of parameters of the convolutional layer scales as $F \times F \times D \times K$. An example of the convolutional operation is shown in Figure 2.2a.

Max-pooling layer implements invariance to the specific location of the activation pattern and mimics the complex cells in the visual cortex. Each max-pooling neuron takes a maximum value from the pool region. In this way, the max-pooling layer introduces translational and rotational invariance and reduces the output resolution and the number of parameters and operations for the subsequent layers. An example of max-pooling operation is shown in Figure 2.2b. We have now described all the tools needed to implement a most basic convolutional neural network known as Lenet-5. We show Lenet-5 network architecture in Figure 2.3.

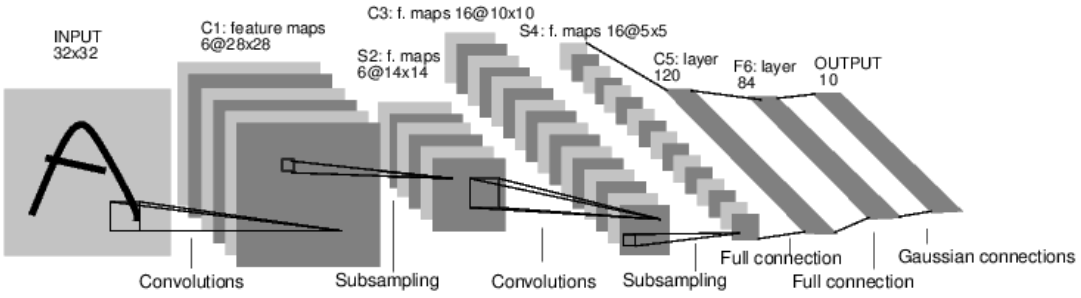


FIGURE 2.3: An example of a convolutional neural network, which is commonly known as Lenet-5 [24].

2.2.3 Network Training

A neural network is a powerful parametric machine learning model. In fact, even a simple 2-layer neural network with sigmoid activation function $f(x) = 1/(1+e^{-x})$ can approximate any continuous function on compact subsets of $\mathbb{R}^n$. This result is known as a universal approximation theorem [25]. Intuitively, it means that neural networks can represent a sufficiently large class of functions and are suitable for a large class of problems. However, the theorem has two caveats. Firstly, it may require an exponential number of units in the hidden layer to represent certain functions. Secondly, it does not tell us how to find optimal parameters of the neural network for the specific task.

To train a neural network, we can use the principle of empirical minimization, which we described in Section 2.1. Recall that in Equation (2.2) we minimize empirical risk on the training dataset $\mathbb{D}$ to find optimal parameters of the classifier. For

neural networks, the choice of the optimizer is limited to the first-order optimization methods due to a large number of parameters. A standard approach for finding the minimum of the loss function is to use gradient descent (GD) method. Parameter update equation for GD optimization is shown below:

$$W = W - \alpha \sum_{i=1}^{N} \nabla \mathcal{L} \left(f \left(x_i; W \right), y_i \right) \tag{2.7}$$

where α is a learning rate hyperparameter. The derivative of the loss function w.r.t. the model parameters in Equation (2.7) can be computed using the back-propagation algorithm [26, 27].

Computing a full gradient over an entire dataset is computationally prohibitive for large datasets. Besides, if some data points contain duplicate information, computing a full gradient is wasteful. For example, consider the dataset that is constructed by repeating an example N times. For large datasets, the gradient is computed over a random subset of examples, which reduces computation. This method is known as minibatch stochastic gradient descent (SGD). Parameter update equation for SGD is shown below:

$$W = W - \alpha \nabla \sum_{i=1}^{B} \mathcal{L} \left(f \left(x_i; W \right), y_i \right)$$

where B is a batch size hyperparameter. One drawback of the mini-batch gradient is a noise variance in the gradient estimate introduced by random sampling. The gradient variance can be reduced by using momentum, which smoothens parameter updates [28, 29].

2.2.4 Deep Neural Networks

Deep neural networks (DNNs) are of special interest in neural networks research. Until recently, however, it was not clear how to train them. The fundamental problem in training deep neural networks is the issue of vanishing and exploding gradients [30]. A repetitive application of the chain rule in the backpropagation algorithm results in the sequence of the layer Jacobian products $\prod_{i=1}^{L} \frac{\partial h^{(i)}}{\partial z^{(i)}}$ where $z^{(i)}$ and $h^{(i)}$ are i-th layer input and output respectively. If the spectral radius of Jacobian matrices is less or bigger than 1, the gradient respectively vanishes

or explodes. If the gradient vanishes, early layers in deep neural networks remain basically untrained or randomly initialized. If the gradient explodes, weight updates are too large, which results in an unstable network. Both problems cause instability in training. Several techniques have been introduced to circumvent the problem of vanishing and exploding gradients.

The most recent re-emergence of interest to neural networks can be attributed to the introduction of new techniques for training deep neural networks, e.g. layer-wise pretraining [31, 32], non-saturating non-linearities [33], regularization methods [34], new architectures [35]. With the latest advances in deep learning, we can now train models with hundreds and even thousands of layers. The depth of neural networks is crucial in the latest success of deep neural networks. DNNs have several theoretical advantages over their shallower counterparts. For example, DNNs can represent certain functions with an exponentially smaller number of parameters compared to shallow neural networks [36, 37]. However, DNNs are notoriously hard to analyze and are often treated as black-box mapping. In the next section, we describe tools used to interpret and analyze predictions of DNNs.

2.2.5 Interpretability of Deep Neural Networks

Neural networks are known to be notoriously hard to analyze and interpret. With the increased depth, DNNs become even less interpretable. In order to understand learned representation, researchers have developed visualization tools to debug deep neural networks. One might visualize weights. For the first layer, weights can be used for visualization purposes. The neural network in the first layer learns to detect simple edges at various orientations, similar to the representation learned by an overcomplete dictionary on natural images [38]. The filters at higher layers, however, combine several lower-level filters. The visualization of the weights in higher layers becomes uninterpretable. For higher layers, the visualization of neuron activations is used to describe what models have learned.

One way to explain what a neural network has learned is to look at what its neurons learned to detect. Instead of searching for a preferred input in the dataset, Erhan et al. [39] constructed a pattern that maximally activates a particular neuron via optimization. Optimization procedure (gradient descent in their experiments) starts at randomly initialized vector and searches for the input that maximally

activates the neuron j at layer l:

$$\begin{aligned} \max_{\mathbf{x}} \quad & \left(h_j^{(l)} \circ h^{(l-1)} \circ \ldots \circ h^{(1)}\right)(\mathbf{x}) \\ \text{s.t.} \quad & \mathbf{x} \in \mathbb{C} \end{aligned} \tag{2.8}$$

where $h_j^{(l)}$ is the activation output of the neuron j in the layer l; $\mathbb{C}$ is the input domain, e.g. $[0, 255]^N$ box constraints for natural images. The images constructed by optimizing Equation (2.8) appear dream-like in structure. Experiments in [39] provided the first evidence to support the conjecture that the higher layers in DNNs represent more abstract visual concepts.

Modern DNNs have millions of neurons. Interpreting individual activations is time-consuming. The scores in the final layer aggregate the activations of all preceding layers in DNNs. To interpret DNNs, Simonyan et al. [40] maximizes the final score for some class y w.r.t. an L_2-regularized image I:

$$\max_{\mathbf{x}} S_y(\mathbf{x}) - \lambda \|\mathbf{x}\|_2 \tag{2.9}$$

where S_y is the classification score for the class y; λ is the parameter for L_2-regularization. For simplicity, we omitted the domain constraint $\mathbf{x} \in \mathbb{C}$. Gradient descent optimization tends to produce unnaturally, dream-like images. Mahendran and Vedaldi [41] integrated a total variation image prior to improve the visualization's interpretability. However, total variation prior significantly increased the cost of inverting representation. [42] introduced a deep generator network (DGN) for synthesizing a preferred input.

Equation (2.9) requires a full optimization of the objective. To improve the speed, the gradient of the output with respect to the input $\frac{\partial S_C}{\partial I}$ can provide an image-specific class saliency visualization [40]. Zeiler and Fergus [43] introduced transposed convolutional layers, also known as deconvolutional layers (DeConv). DeConv layers allow the neural network to reconstruct the input from its activations. Springenberg et al. [44] established a connection between deconvolutional neural

networks and gradient saliency maps. Their method, so-called guided backpropagation, combines standard and deconv backpropagation steps:

$$\text{backpropagation:} \quad \delta_i^{(l)} = \left(h_i^{(l)} > 0\right) \cdot \delta_i^{(l+1)} \tag{2.10}$$

$$\text{deconv backpropagation:} \quad \delta_i^{(l)} = \left(\delta_i^{(l+1)} > 0\right) \cdot \delta_i^{(l+1)} \tag{2.11}$$

$$\text{guided backpropagation:} \quad \delta_i^{(l)} = \left(h_i^{(l)} > 0\right) \cdot \left(\delta_i^{(l+1)} > 0\right) \cdot \delta_i^{(l+1)} \tag{2.12}$$

where $h_i^{(l+1)} = \max\left(h_i^{(l)}, 0\right)$ is the output of rectified linear activation; $\delta^{(l)}$ is the backpropagation error with respect to layer l. As we can in Equation (2.12), guided backpropagation modifies the gradient of ReLU activation function and backpropagates only positive error signals. DNNs interpretability is an active area of research. [45] compared various visualization techniques. [46] questioned if the existing methods provide a better assignment of feature importance than a random assignment. Besides saliency map visualisations for interpretability, other methods distil a complex model, such as DNN, into a more interpretable model [47, 48].

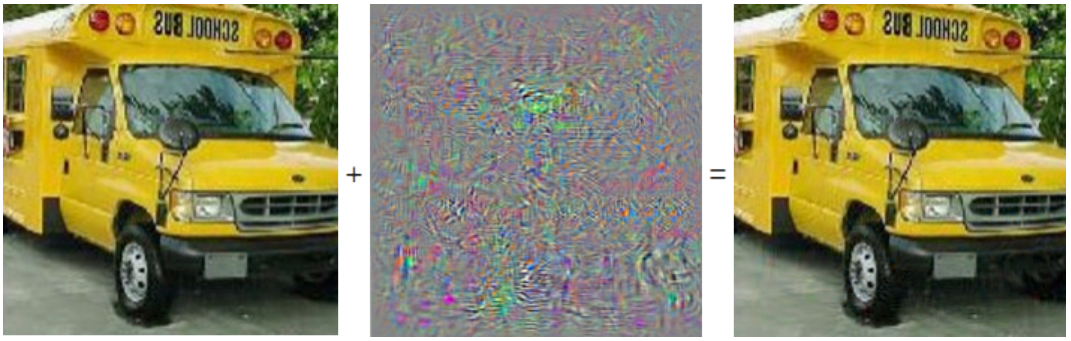


FIGURE 2.4: An adversarial example for GoogleNet network [4] was generated using L-BFGS-B attack [5]. GoogleNet correctly classified the image on the left as a "school bus". An adversarial image (shown on the right) was constructed by adding small imperceptible perturbation (shown in the middle) to the original image. The neural network misclassified the adversarial image as a "ostrich". The original and adversarial images are indistinguishable to the human eye, yet, GoogleNet predicted a different label for the adversarial image.

2.3 Adversarial examples

Deep neural networks (DNNs), which we have reviewed in Section 2.2, have been remarkably successful for a wide range of perceptual problems: image classification [35], object detection [49], speech recognition [2], machine translation [3]. Despite their excellent performance in an extensive range of practical applications, Szegedy et al. [6] found that state-of-the-art deep neural networks, such as GoogleNet, are sensitive to small, imperceptible perturbations in the input. These inputs are commonly known as *adversarial examples*. Adversarial examples are constructed by adding an adversarial noise to the original images. We show an adversarial image for GoogleNet [4] in Figure 2.4. Intriguingly, adversarial images are visually indistinguishable from the original images, yet, DNNs misclassifies adversarial examples with high confidence. This vulnerability of DNNs is not unique to the image recognition problems [12, 50]. For example, the researchers found that DNNs are vulnerable to perturbations in text for language understanding (changing few words in the sentence flips the sentiment of the review) [50] and audio for speech recognition(adding inaudible background noise changes the sentence transcription) [12] tasks.

The lack of robustness to small, imperceptible perturbations is counter-intuitive and undesirable. It reduces our trust in AI systems. Unlike state-of-the-art deep neural networks, human vision is remarkably robust to variations in the input, e.g. changes in the lighting condition or changes in the object shape or pose. For example, if someone wants to conceal his or her identity from the police, the person will have to wear a mask or undergo cosmetic surgery. In comparison, an adversary needs to change only a few pixels in the image to fool a state-of-the-art facial recognition system [51]. Additionally, the existence of adversarial examples poses a serious concern for the application of deep neural networks in safety and security-critical applications [52] where the attacker can abuse such vulnerability. For example, the attacker in [9, 14] generated a “graffiti” sticker on a road sign, which looks harmless, but this pattern tricked road sign recognition system. Similarly, the attacker in [15] hacked face recognition system. The authors printed a pattern on the eyeglass frame that let the attacker remain undetected or even impersonate another person.

The problem of adversarial examples has spurred significant interest in the research of deep neural networks. The field of research in robust deep learning can be broadly divided into the research on attacks [18, 53, 54] and the research on the defenses [5, 17, 55]. Like in an arms race, these two sides compete: today, new defenses are introduced to protect against existing attacks; tomorrow, new attacks are introduced to break new defenses. Since this vulnerability was discovered, dozens of attacks and defenses have been proposed. To no avail, assessing the robustness and training a robust deep neural network remains an open problem. Now, it is even speculated that adversarial examples can fool time-limited human observers [56]. We review adversarial attacks in Section 2.4 and defenses against adversarial attacks in Section 2.5.

Apart from the research on attacks and defenses, explaining the nature of adversarial examples is an important problem. Several hypotheses attempted to explain their properties and existence. Szegedy et al. [6] argued that adversarial examples occupy low probability pockets, or "blind spots", in the input space. The existence of adversarial examples was explained by highly expressive power and nonlinear behavior of deep neural networks. In addition, the authors that adversarial examples generated for one model transfer to the models trained on the same task. [5] showed that adversarial examples could be easily generated by taking a single step in the direction of the input gradient. Additionally, Goodfellow et al. [5] hypothesized that repeated application of ReLU activation $f(x) = \max(0, x)$ at each layer makes deep neural network behave essentially like a linear classifier in high dimensional space. They also showed that adversarial examples exist for linear models, e.g. logistic regression, have a similar weakness.

Instead of analyzing existing state-of-the-art architectures, Duvenaud et al. [57] studied deep Gaussian processes, a type of infinitely-wise deep neural network. Deep Gaussian process can be seen as prior on the composition of functions, such as deep neural networks. Their analysis showed that the NN representation tends to capture fewer degrees of freedom as we increase the number of network layers. Let $f(x)$ be the layer function composition of the deep neural network:

$$f(x) = \left(f^{(L)} \circ f^{(L-1)} \circ \ldots \circ f^{(1)}\right)(x)$$

where $f^{(k)}$ is the k-th layer transformation function. Then, the Jacobian of the deep neural network is a product of Jacobians of all layers:

$$J(x) = J^{(L)} \cdot J^{(L-1)} \dots J^{(2)} \cdot J^{(1)}$$

The product of matrices behaves like the product of eigenvalues. As we increase the depth of a neural network, the largest singular value tends to dominate, thus, essentially reducing the number of degrees of freedom in DNN representation. To fix this pathology in neural networks, they suggested that each layer should depend not only on the output of the previous layer but also on the original input x. Interestingly, input skip-connections suggested in [57] are somewhat similar to the architecture of Residual Neural Networks [35], where each output of the layer block combines the output of the previous with the original input.

Ilyas et al. [58] argued that the lack of robustness is not an inherent property of the model, but it is rather a feature of the data: the data contains non-robust yet useful for classification features. Fawzi et al. [59] introduced a theoretical framework for analyzing classifier robustness. They derived an upper bound for 2-class linear and quadratic classifiers in terms of distinguishability, which intuitively measures the difficulty of the classification task. For a linear classifier, the class distinguishability is defined as the distance between means of two classes. For a quadratic classifier, the class distinguishability is defined as the distance between covariance matrices of the two classes. In the subsequent work [60], the authors established bounds on the robustness of the classifier in terms of the curvature of the its decision boundary. CURE defense [61] minimizes the curvature to improve the robustness.

2.4 Attacks on Deep Neural Networks

In the previous section Section 2.3, we introduced the problem of adversarial examples. A myriad of attacks on DNNs has been proposed since the discovery of adversarial examples. In this section, we attempt to review the landscape of the research on adversarial attacks. For a detailed review of various attacks, interested readers may refer to [62, 63].

Adversarial attacks can be categorized based on the attacker's knowledge about the model (*white-box* vs *black-box* attacks), the attack's specificity (*targeted* vs *non-targeted* attacks), and the perturbation measurement (l_∞-, l_2-, l_1-, and l_0-norm attacks). White-box attacks have full knowledge of the neural network model, including the training data, the architecture, the weights, and the hyperparameters of the model. Black-box attacks have access only to the output of the model, e.g. the final decision or the score. Targeted attacks aim to produce a targeted misclassification, whereas the adversarial label for an untargeted attack can be arbitrary except the original one. Intuitively, a white-box adversary should always be stronger than any black-box adversary because it has complete information about the attack's target model. However, in a special case, when defense obfuscates or shatters the gradients, white-box adversaries tend to overestimate model robustness [64]. Athalye et al. [65] suggested evaluating defense on white-box and black-box adversaries. If the model is more robust to white-box adversaries, then the model might mask the gradients. Next, we review white-box and black-box adversarial attacks. An in-depth review and comparison of various attack methods can be found in [62, 63].

2.4.1 White-box Attacks

To start our review, we formulate a general problem of attacking a deep neural network model in white-box settings. Let $f(\cdot)$ be the mapping from the space of input pixels to the unnormalized predictive distribution on discrete label output space $f : \mathbb{R}^N \to \mathbb{R}^k$, where N is the dimensionality of the input, and k is the number of classes. For a given input image $\mathbf{x}$ with the label y, an adversary aims to find a minimal adversarial perturbation $\mathbf{r}$ wrt some norm $\|\cdot\|$, such that after adding the perturbation to the original image $\mathbf{x}$ it changes the network prediction $\hat{k}(\mathbf{x}+\mathbf{r}) \neq y$. We can formulate the attack's goal as the following optimization problem:

$$\begin{aligned} \min_{\mathbf{r}} \quad & |\mathbf{r}| \\ \text{s.t.} \quad & \hat{k}(\mathbf{x}+\mathbf{r}) \neq y \\ & \mathbf{x}+\mathbf{r} \in \mathbb{C} \end{aligned} \tag{2.13}$$

where $\mathbf{x}$ and $\mathbf{x}+\mathbf{r}$ is the *adversarial* noise and the *adversarial example* respectively; $\mathbb{C}$ is the input domain, e.g. $[0,1]^N$ box constraints for the normalized image. The

problem above is an example of an untargeted adversarial attack. A targeted adversarial attack searches for the perturbation that changes the network prediction to the specific target $\hat{k}(\mathbf{x}+\mathbf{r}) = y_t$.

The optimization problem above is a non-convex constrained norm minimization problem with a non-differentiable error constraint. Solving the problem in Equation (2.13) is a challenging and non-trivial task. A myriad of attack methods has been proposed to solve this optimization problem [5, 6, 8, 17, 18, 53]. Next, we discuss these attack methods in detail.

2.4.1.1 L-BFGS Attack

Szegedy et al. [6] first introduced a targeted gradient-based adversarial attack against DNNs in 2014, which is known as L-BFGS attack. L-BFGS method is the basis of many attack algorithms. The original problem in Equation (2.13) is difficult to solve because the misclassification constraint is non-differentiable. [6] used a surrogate loss function, e.g. hinge loss or cross-entropy, in place of the original non-differentiable error constraint. Then, the authors solved the following unconstrained minimization problem:

$$\begin{aligned} \min_{\mathbf{r}} \quad & |\mathbf{r}| + \lambda \mathcal{L}\left(\mathbf{x}+\mathbf{r}; y\right) \\ \text{s.t.} \quad & \mathbf{x}+\mathbf{r} \in \mathbb{C} \end{aligned} \tag{2.14}$$

where λ is the regularisation penalty for violating of the misclassification constraint; $\mathcal{L}$ is a surrogate loss function which is minimised when $\hat{k}(\mathbf{x}+\mathbf{r}) \neq y$. Interestingly, this optimization problem in Equation (2.14) is similar to the interpretability method in Equation (2.9), which synthesizes a preferred input with a high score for a specific class. For a fixed λ, L-BFGS-B optimization method was used to find a solution of the problem in Equation (2.14). Finally, a line search was performed to find the optimal regularisation weight λ^* such that the perturbation is minimized. [66] suggested to use a bisection search method to find an optimal C. L-BFGS attack produces small imperceptible perturbations and estimates the model robustness accurately. However, L-BFGS attack is impractical against large models because: 1) it uses computationally intensive L-BFGS-B method; 2) it requires multiple restarts for different values of λ, which significantly increases the cost of the attack.

2.4.1.2 Carlini & Wagner Attack

In seminal work, Carlini and Wagner [18] studied the attack's problem in Equation (2.14) and introduced a set of techniques to improve the speed and accuracy of the gradient-based attack. They explored how the choice of the minimizer, the surrogate loss function, and handling of the box constraints affect the attack's optimization. In their experiments, they found that Adam optimizer [29] with multi-class hinge loss and hyperbolic tangent transformation of variables is the best attack on a set of image datasets. This attack, known as C&W, is one of the strongest to-date attacks. The authors also highlighted the importance of using strong attacks when we evaluate and compare defenses based on robustness. In particular, they showed that defensive distillation [18, 67] is not robust to adversarial examples. At the present moment, C&W is the recommended attack for the assessment of DNNs robustness to l_2-norm perturbations [68]. The results for C&W attack should be included in any comparison between various defense methods. However, C&W is impractical for large models. Similar to LBFGS-B attack, it requires full optimization for each value of the regularisation weight λ, which amounts to thousands of iterations in total.

2.4.1.3 Fast Gradient Sign Method

Goodfellow et al. [5] reformulated the original non-convex constrained minimization problem as the problem of minimizing the surrogate loss function subject to the convex l_p-norm perturbation constraint:

$$\begin{aligned} \min_{\mathbf{r}} \quad & \mathcal{L}\left(\mathbf{x}+\mathbf{r}; y\right) \\ \text{s.t.} \quad & \|\mathbf{r}\| \leq \epsilon \\ & \mathbf{x}+\mathbf{r} \in \mathbb{C} \end{aligned} \tag{2.15}$$

The optimization problem in eq. (2.15) can be solved using a first-order approximation of the surrogate loss $\mathcal{L}$ at the current datapoint $\boldsymbol{x}$ as follows:

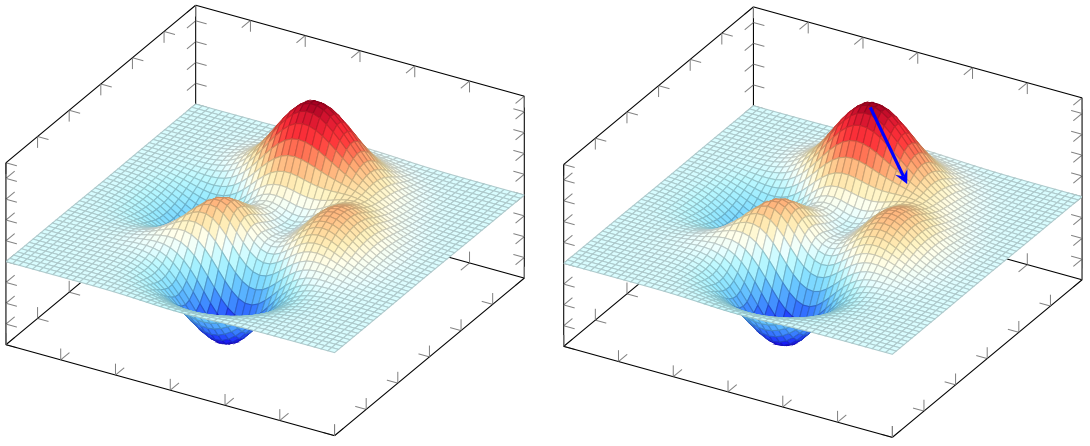
$$\begin{aligned} \min_{\mathbf{r}} \quad & \mathbf{r} \cdot \nabla_{\mathbf{x}} \mathcal{L}(\mathbf{x}; y) \\ \text{s.t.} \quad & \|\mathbf{r}\| \leq C \end{aligned} \tag{2.16}$$

It is easy to show that the solution of the above problem is the C-normalized gradient of the loss. In particular, the adversarial direction for the l_∞-norm constraint is the sign of the loss gradient wrt inputs:

$$\boldsymbol{r} = \epsilon \operatorname{sign}\left(\nabla_{\boldsymbol{x}} \mathcal{L}\left(f\left(\boldsymbol{x}\right); y\right)\right) \tag{2.17}$$

Because of that, this attack method is known as a fast gradient sign method (FGSM). It is interesting to note that FGSM attack is somewhat similar to gradient saliency maps [40], which we discussed in Section 2.2.5. In Figure 2.5, we visually compare L-BFGS and FGSM attacks. FGSM is imprecise at evaluating the model robustness, but it is extremely fast in practice. Due to its speed, FGSM can be used to improve the robustness of the model [5] without significantly increasing the training time. We review the defense based on FGSM attack in Section 2.5.1.

[8, 17] proposed an iterative version of FGSM attack, which is known as projected gradient descent (PGD). PGD iteratively takes a finer step in the direction of the adversarial target. Madry et al. [17] argued that PGD is an optimal first-order adversary. PGD attack is a recommended starting attack for measuring the robustness to l_∞-norm distortions [68]. PGD attack is a simple, fast, and accurate attack, but it does not explicitly minimize the l_p-norm of the perturbation. To



(A) (B)

FIGURE 2.5: A comparison between L-BFGS attack with linear search [6] and fast gradient sign method (FGSM) [5]. L-BFGS attack [6] runs optimization until the convergence (fig. 2.5a). FGSM [5] takes a single step in the direction of the input gradient (fig. 2.5b). Compared to L-BFGS-B method, FGSM requires only a single gradient evaluation. As a result, FGSM is extremely fast but quite inaccurate at measuring the robustness.

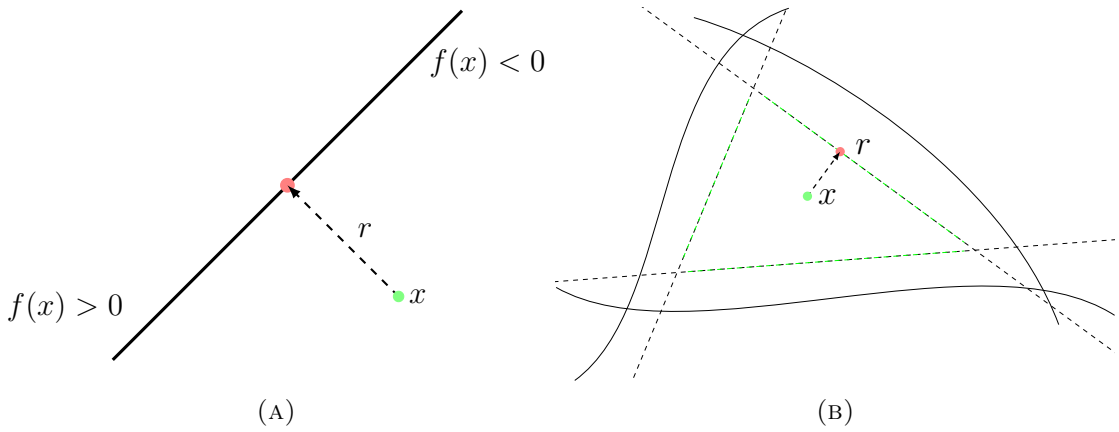
evaluate robustness at N distinct thresholds ϵ, PGD attack needs to be restarted N times which linearly increases the cost of the attack.

2.4.1.4 DeepFool

DeepFool [53] finds the closest class boundary and takes a step in the direction of the closest decision boundary. The optimization process stops as soon as adversarial perturbation is found. For a 2-class affine classifier in Figure 2.6a, the minimal distance to the decision hyperplane can be computed exactly as follows: $\boldsymbol{r}(\boldsymbol{x}) = -\left(f(\boldsymbol{x})/\|\mathbf{w}\|_2^2\right)\mathbf{w}$, where $\mathbf{w}$ is the weight vector. For an n-class nonlinear classifier in Figure 2.6b, Moosavi-Dezfooli et al. [53] approximated the current prediction region of DNNs using polyhedron. The distance to the class j decision boundary for the input $\boldsymbol{x}$ with the label y can be approximated as follows:

$$\boldsymbol{r}_j(\boldsymbol{x}, y) = \frac{|f_y(\boldsymbol{x}) - f_j(\boldsymbol{x})|}{\|\nabla f_y(\boldsymbol{x}) - \nabla f_j(\boldsymbol{x})\|}(\nabla f_y(\boldsymbol{x}) - \nabla f_j(\boldsymbol{x})) \tag{2.18}$$

where $\nabla f_j(\boldsymbol{x})$ is the gradient of the unnormalized output for the class j at the input $\boldsymbol{x}$. The authors computed the distance to the decision boundary for all classes $j \neq y$, in total $(k-1)$ calculations. Then, the minimal adversarial perturbation is the smallest distance among all classes: $r = \arg\min_{j \neq y} r_y(\boldsymbol{x}, y)$. They applied above approximation iteratively until the adversarial perturbation was generated.



(A) (B)

FIGURE 2.6: A minimal adversarial perturbation for a 2-class affine classifier and an n-class nonlinear classifier. The minimal adversarial perturbation for a 2-class affine classifier is the distance to the decision hyperplane (fig. 2.6a). The region of the current prediction for a n-class nonlinear classifier, such as DNNs, can be approximated using polyhedron. Then, the adversarial perturbation for an n-class nonlinear classifier is the shortest distance to the facets of the polyhedron(fig. 2.6b).

DeepFool is a relatively fast and accurate method. However, DeepFool has several limitations: 1) it does not explicitly handle box-constraints on the input $\boldsymbol{x} + \boldsymbol{r} \in [0, 1]^m$; 2) it does not explicitly minimize the norm of the perturbation. The optimization process stops as soon as the adversarial perturbation is found. [54] introduced Fast Adaptive Boundary Attack (FAB), which addresses the limitations of DeepFool attack. FBA solves the box-constrained l_1-, l_2, and l_∞-norm projection on the approximated decision hyperplane exactly. In addition, the authors introduced a biased backward step, which reduces the perturbation norm after each step. SparseFool (SF) [69] proposes a geometry inspired l_1-norm attack, which uses DeepFool attack [53] as subprocedure to estimate the local curvature of the decision boundary. They developed an efficient algorithm to compute l_1-norm projection of the perturbation on the decision boundary subject to image box-constraints.

2.4.1.5 Feature Adversary

Another way to understand the adversarial properties of deep neural networks is to consider the distance in the input space between a source image $\boldsymbol{x}_s$ and a guide image $\boldsymbol{x}_g$ at the layer l. Sabour et al. [70] proposed to minimize representation

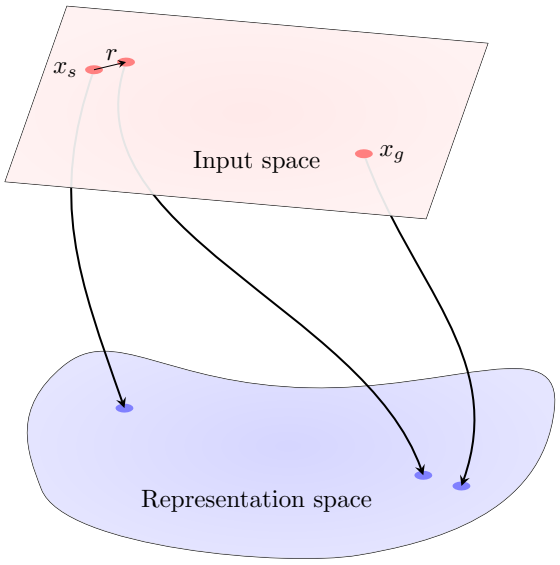


FIGURE 2.7: A diagram of Feature Adversary Attack. Feature Adversary [70] searches for the perturbation $\boldsymbol{r}$ such that the distance between the perturbed source $\boldsymbol{x}_s + \boldsymbol{r}$ and the guide $\boldsymbol{x}_g$ images in the representation space at the layer l is minimized.

distance between source and guide image by changing source image:

$$\begin{aligned} \min_{\mathbf{r}} \quad & \|\phi_k(\mathbf{x}_s + \mathbf{r}) - \phi_k(\mathbf{x}_g)\|_2^2 \\ \text{s.t.} \quad & \|\mathbf{r}\|_\infty \leq \epsilon \end{aligned} \tag{2.19}$$

where ϕ_k is the representation of layer k; ϵ is the perturbation budget. Figure 2.7 shows an intuition behind feature adversary method. Interestingly, this attack is somewhat similar to neural artistic style transfer [71], where the goal is to change the source image, so it has a style similar to the guide image.

2.4.1.6 Other attacks

Decoupling direction and norm l_2-attack (DDN) [72] adjusts the radius of l_2-norm ball used for the perturbation constraint in Equation (2.15). If the perturbation is adversarial/not adversarial, the radius of l_2-norm projection ball can be decreased/increased respectively. [73] extended basic iterative method by integrating the momentum into the attack process. Distributionally adversarial attack (DAA) [74] considers the task of finding an adversarial data distribution for which the generalization risk increases maximally.

So far in this section, we discussed l_∞- and l_2-norm attacks. l_∞- and l_2-norm adversarial attacks produce perturbations that change a large number of pixels in the image. Several attack methods directly minimize non-differentiable l_1-norm and non-convex l_0-quasinorm. Elastic-net attack (EAD) [75] minimizes l_1-norm of the perturbation using fast iterative shrinkage-thresholding algorithm (FISTA) [76]. They suggested that l_1-attacks may complement the analysis of the DNNs robustness. EAD attack is currently a preferred l_1-norm attack [68]. [77] introduced a variant of PGD attack [8, 17] for l_1-norm — sparse l_1 descent attack (SLIDE) attack. At each iteration of their attack, they take a step in the direction of the q-th percentile of the loss gradient. After each step, the perturbation is normalized using l_1-norm projection algorithm [78]. [52] proposed a targeted adversarial attack, known as Jacobian saliency map attack (JSMA), which minimizes l_0-norm perturbation constrain. JSMA uses the Jacobian matrix to select and modify a pair of the most salient pixels in the image. This process is repeated until the adversarial perturbation is found. [51] found that it is possible to change the model's prediction by modifying a single pixel. To generate adversarial examples, they applied a differential evolution (DE) algorithm on the population of vector coordinates that modify a single pixel in the image.

2.4.2 Provable Verification of DNNs

Attacks [5, 6, 18] that we discussed earlier provide an upper bound on the model robustness to adversarial noise. Provable verification methods return a lower bound or a certificate, proving that the model is robust to all perturbations with the specified adversarial budget. Provable verification methods return a lower bound on the model performance subject to adversarial noise, which can prove that the model is robust to all perturbations with the specified budget. [79] observed that for a network with linear rectified units the adversarial perturbation could be found by solving a constrained system. The authors approximated the intractable constrained problem with a tractable linear program. [80] designed a verification algorithm based on Satisfiability module theory (SMT), called Reluplex, which can be used to provably verify the properties of DNNs with linear rectified units. [81] derived the rigorous bounds on the outputs of DNNs using interval arithmetic. With recent advances, provable verification methods are able to scale to moderately sized DNNs, yet, safety verification of large DNNs remains challenging.

2.4.3 Black-box Attacks

A black-box adversary has limited knowledge about the model, e.g. the prediction scores or the outputs of the model. Attacks under the black-box threat model are more difficult to perform than white-box settings because we do not know the model's gradient. Black-box, or gradient-free attacks, could be divided into two categories: gradient estimation and transfer-based attacks. [82, 83] estimate the gradient of the black-box model, which may require many queries for accurate approximation. Transfer-based attacks [64] rely on the fact that adversarial examples transfer between models. However, the efficiency of the transfer-based attacks largely depends on the quality of the substitute network.

[64] introduced a practical black-box adversarial attack based on the property that adversarial examples often transfer between models. They trained a substitute model on the model's task. Then, adversarial examples generated for the substitute model can be used to attack the target model. Brendel et al. [84] introduced a decision-based attack which estimates the decision boundary using rejection sampling. Starting at some adversarial image, they randomly draw a random perturbation from the candidate distribution and minimize the distance to the original image. [82] proposed to sample a random perturbation from an orthonormal basis of discrete cosine transform (DCT), which significantly improves query-efficiency of the decision-based attack.

Gradient-based attacks should be almost always stronger than gradient-free attacks. However, gradient masking [64] can fool gradient-based attacks and give a false sense of security [65]. If the defense obfuscates the gradients, gradient-free attacks often perform better than white-box attacks. [68] suggested that defenses should be tested on both white-box and black-box adversaries. If the model is more robust to white-box adversaries, then the model masks the gradients.

2.4.4 Adversarial Attacks in the Physical World

Initially, the threat of adversarial examples was considered frivolous for real-world applications. [6] argued that adversarial noise occupies low probability "pockets" in the input space that act like a "blind spots" to DNNs. Because of that, it was not clear if adversarial examples are physically realizable. The problem of

adversarial noise was not considered to be a serious security issue for real-world systems. However, since then, several methods have been introduced that are effective in producing physically feasible and imperceptible adversarial disturbances. [8] showed that it is possible to generate printable adversarial images that are consistently misclassified when shown to the camera of a cell phone. [15] generated a special pattern that misleads face recognition and face identification systems when printed on the specs frame. [9, 14] introduced a road sign attack. They generated a special sticker that fooled an object detection system when the sticker was placed on the road sign. [85] argued that adversarial examples are not a concern for objection detection in autonomous vehicles due to the changing physical conditions of a moving car. However, the findings in [9, 14, 86] suggest that it is not the case.

2.5 Defenses against Adversarial Examples

The disparity between the error on the test data and the error on the adversarial examples raises several questions regarding the limitations and weaknesses of the existing deep learning models. In this section, we review some recent defense algorithms for improving DNN robustness to adversarial noise. In particular, we cover robust optimization and adversarial (re)training, adversarial detection, and adversarial denoising.

2.5.1 Robust Optimization and Adversarial (Re)training

We start our review by recalling a mathematical formulation for the robust multiclass classification. Robust optimization (RO) seeks a solution robust to the worst-case perturbations $\boldsymbol{r}$ of the input $\boldsymbol{x}$:

$$\min_{\mathbf{W}} \sum_{i=1}^{N} \max_{\boldsymbol{r}_i \in \mathcal{U}_i} \mathcal{L}(f(\boldsymbol{x}_i + \boldsymbol{r}_i; \mathbf{W}); y_i) \tag{2.20}$$

where $\mathcal{L}$ is the training loss, e.g. cross-entropy; $\mathbf{W}$ is the classifier parameters; $\boldsymbol{r}_i$ is the arbitrary (even adversarial) perturbation of the input $\boldsymbol{x}_i$; $\mathcal{U}_i$ is the uncertainty set, e.g. l_p-norm ϵ-ball $\mathcal{U}_i = \{\boldsymbol{r}_i : \|\boldsymbol{r}_i\|_p \leq \epsilon\}$. Prior information about the task can be used to select a problem-specific uncertainty set $\mathcal{U}$. For example, small

changes of each pixel in the natural image are unlikely to result in the change of the underlying label.

Robust optimization or robust classification in Equation (2.20) is a principled and well-studied approach to improve a classifier's robustness to the input perturbations. Several standard regularization methods in machine learning are equivalent to the robust optimization, e.g. l_1 lasso regression [87] and l_2 support vector machine [88]. Likewise, the researchers developed several RO-inspired techniques to improve the robustness of DNNs. Next, we discuss one such technique — adversarial training [5, 17, 89].

The minimax robust optimization problem in Equation (2.20) is intractable for highly nonlinear deep neural networks. Madry et al. [17] solved the robust minimax problem using alternating optimization. First, the authors fixed the parameters $\mathbf{W}$ and computed the worst-case input perturbation $\boldsymbol{r}$ as follows:

$$\arg\max_{\boldsymbol{r}_i \in \mathcal{U}_i} \mathcal{L}\left(f\left(\boldsymbol{x}_i + \boldsymbol{r}_i; \mathbf{W}\right); y_i\right) \tag{2.21}$$

After that, the parameters $\mathbf{W}$ are updated to minimize the loss $\mathcal{L}$ on the adversarial example $(\boldsymbol{x}_i + \boldsymbol{r}, y_i)$. We can see that the problem in Equation (2.21) is exactly equivalent to the norm-constrained attack in Equation (2.15) when the uncertainty set $\mathcal{U}$ is an ϵ-ball.

Adversarial training, as it was introduced in [5], is a variant of robust training presented above. AT trains neural network on the mixture of original and adversarial images:

$$\min_{\mathbf{W}} \sum_{i=1}^{N} \mathcal{L}(f(\boldsymbol{x}_i; \mathbf{W}), y_i) + \lambda \mathcal{L}(f(\boldsymbol{x}_i + \boldsymbol{r}_i; \mathbf{W}), y_i) \tag{2.22}$$

where $\boldsymbol{r}_i$ is the adversarial perturbation, which was generated using Fast Gradient Sign method (FGSM). We show a diagram of adversarial training in Figure 2.8.

Other attack methods can be used to solve Equation (2.21). In fact, a stronger attack will generally result in a more robust classifier. Madry et al. [17] experimentally argued that Projected Gradient Descent (PGD) adversary is the optimal first-order optimizer of Equation (2.21). Adversarial training with PGD attack [17] resulted in the model that is significantly more robust that the models trained with weaker FGSM attack [5]. Ensemble Adversarial Training (EAT) [90] uses an ensemble of the models to generate adversarial examples.

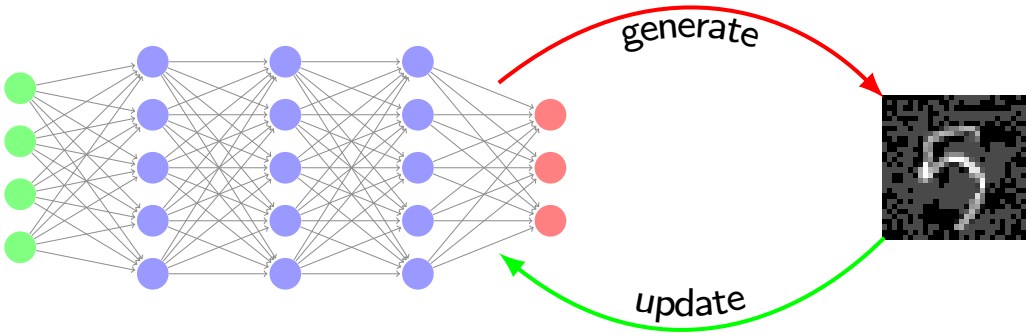


FIGURE 2.8: A diagram of Adversarial (Re)training defense (AT) [5, 17]. Adversarial examples are generated on the fly using FGSM attack. The generated adversarial examples are used to augment the training dataset. Then, the model is trained on a mixture of clean and adversarial images.

Miyato et al. [91] proposed virtual adversarial training (VAT). VAT smoothens a decision boundary in the vicinity of the training examples:

$$\min_{\mathbf{W}} \sum_{i=1}^{N} \mathcal{L}\left(f\left(\boldsymbol{x}_i; \mathbf{W}\right), y_i\right) + \mathrm{KL}\left(p(\boldsymbol{x}_i) \| p(\boldsymbol{x}_i + \boldsymbol{r}_i)\right) \tag{2.23}$$

where $p(\boldsymbol{x})$ is the predictive distribution of the model $f(\,\cdot\,; \mathbf{W})$ on the input $\boldsymbol{x}$; $\mathrm{KL}(p\|q)$ is Kullback–Leibler divergence between the distributions p and q. The first term in Equation (2.23) is a standard supervised loss. The second term in Equation (2.23) encourages local manifold smoothness within the vicinity of training points. Adversarial perturbation $\boldsymbol{r}_i$ in Equation (2.23) was computed by solving the following optimization problem:

$$\boldsymbol{r}_i = \underset{\|\boldsymbol{r}\|_2 \leq \epsilon}{\arg\min} \{-\mathrm{KL}\left(p(y|\boldsymbol{x}_i) \,\|p\left(y|\boldsymbol{x}_i + \boldsymbol{r}\right)\} \tag{2.24}$$

The authors used the power iteration method to find the solution of Equation (2.24). Unlike AT method, VAT does not require a label during training; this technique was also successfully applied for the task of semi-supervised classification. The authors of TRADES defense [92] theoretically characterize the trade-off between accuracy and robustness for classification problems. Inspired by the theoretical analysis, they proposed to minimize classification-calibrated loss (e.g. cross-entropy) and robustness regularization term (e.g. KL-divergence, which is similar to VAT).

2.5.2 Adversarial Detection

Detecting adversarial examples is an alternative way to mitigate the problem of adversarial examples at test time. If the detector finds an adversarial input, an autonomous operation can be stopped, and human intervention can be requested. [93] proposed to train a detector network on the hidden layer's representation of the guarded model. [94] adopted a Bayesian interpretation of Dropout to extract confidence intervals during testing. Then, the optimal threshold was selected to distinguish natural images from adversarial. [95] introduced an adaptive noise reduction technique. To detect adversarial examples, they compared the prediction of DNNs on original and denoised inputs. [96] proposed a reverse cross-entropy loss for training DNNs, which improves the detection of out-of-distribution and adversarial examples. Nonetheless, Carlini and Wagner [97] extensively studied and demonstrated the limitations of the detection-based methods. Detection-aware attacks can break detection-based defenses in both white-box and black-box settings.

2.5.3 Adversarial Denoising

Adversarial noise can be removed from the image via denoising. [98] showed that its possible to remove some adversarial noise with a small mean filter. Features squeezing [99] reduces the search space available to an adversary, which makes the attack more difficult. [100] extended contractive autoencoder [101] regularization to deep neural networks. Contractive regularization minimizes the Frobenius norm of the Jacobian matrix, which measures the output's sensitivity to the changes in the input. Reducing the sensitivity to the changes in the input improved the model's robustness. PixelDefend [102] used PixelCNN prior [103] to reconstruct the original image from its adversarial example. [104] studied the robustness of DNNs when the input image was preprocessed using JPEG compression, total variation minimization, and image quilting. These simple input transformations were shown to be effective in countering adversarial examples. [105] proposed a defense method by randomization at inference time, e.g. random resizing and random padding. [106] stochastically combined several weak denoising defenses, e.g. JPEG compression, non-local means denoising. Feature denoising [107] introduced special network blocks that denoise hidden features using non-local means.

2.6 Conclusion

This chapter presents background information about machine learning and deep neural networks. We attempt to review the landscape of the research on adversarial examples. We introduce the problem of adversarial examples. We discuss current theoretical explanations of why DNNs are vulnerable to adversarial noise. We review recent attacks against deep neural networks and recent methods to safeguard these models from adversarial attacks. However, we are unable to cover and discuss the growing body of research in this area. For a detailed overview of the research on adversarial examples, interested readers may refer to [62, 63].

Chapter 3

Deep Margin Maximization

As we have argued in detail in the previous chapter, the lack of model robustness to input disturbances is counterintuitive and undesirable. It also limits the use of deep neural networks in safety and security-critical applications. Current state-of-the-art approaches to improving model robustness, such as adversarial (re)training (AT), augment training dataset with *adversarial examples* to improve robustness. However, adversarial (re)training defense relies on the assumption that data disturbance does not change the underlying label of the input. Additionally, when using data augmentation, the model may fail to anticipate changes in an adversary. In this chapter, we describe our work in which we propose to maximize a geometric margin of the classifier to improve robustness. Intuitively, a large margin classifier is robust to all perturbations with norm less than the margin. Based on this idea, we introduce a novel margin maximization objective for deep neural networks. We theoretically show that the proposed objective is equivalent to the robust optimization problem for a neural network. Our work seamlessly generalizes SVM margin objective to deep neural networks. In the experiments, we extensively verify the effectiveness of the proposed margin maximization objective to improve neural network robustness and reduce overfitting on *MNIST* and *CIFAR-10* datasets.

3.1 Introduction

Adversarial noise is a minimal input perturbation that changes classifier prediction. *Adversarial examples* are the corrupted inputs which are visually similar to

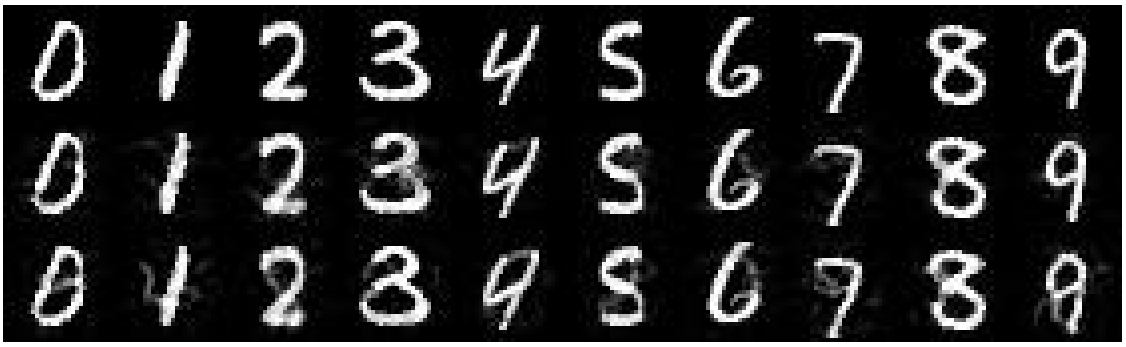


FIGURE 3.1: Adversarial examples were generated using DeepFool atttack [53] for naturally trained and regularized convolutional neural network Lenet-5 [24] on MNIST dataset. The first row displays the original images. The second row displays adversarial images for the naturally trained model. The third row shows adversarial images for the model trained with our regularization method. After our regularization method was applied, we can observe that adversarial images are distinguishable in a visually meaningful way from the original images.

the original input images but for which classifier produces incorrect label prediction (see fig. 3.1). Designing classifiers robust to worst-case perturbation has several potential advantages. Firstly, an input to classifier can contain measurement errors or can be maliciously corrupted in adversarial settings by the attacker [52]. Secondly, worst-case analysis using robustness provides generalization bounds alternative to the traditional uniform convergence bounds used in statistical learning theory [108]. Lastly, the local stability of the prediction is important for many applications, e.g. image search, near-duplicate detection, video-frames classification [109].

The topic of adversarial examples has spurred significant interest in deep learning research. Current approaches for improving network robustness against *adversarial noise* can be viewed as a form of data regularization using adversarial examples [5, 53, 91]. Data augmentation is conceptually simple, but it has several limitations. Firstly, an adversarial distortion should be label non-changing. If the perturbation is too large, it can alter the true class label. Additionally, generating an adversarial noise is time-consuming, and approximate methods are used in [5, 91]. Secondly and most importantly, data regularization methods [5] discard information about the dependency between the model parameters and an adversarial noise. The model fails to anticipate changes in the adversary. As a result, the regularized neural network tends to overfit the fooling method used for data augmentation [6, 53]. Regularized neural network classifies correctly seen adversarial examples, but it remains vulnerable to the newly generated adversarial examples.

In this chapter, we propose a novel margin maximization objective for deep neural networks. Our margin formulation for binary classification minimizes the norm of the Jacobian, which is analogous to [101] but offers a novel interpretation of the contractive penalty. For multiclass problems, we minimize the maximum norm of the difference between gradient w.r.t. inputs of the correct and incorrect prediction. We establish a connection between a margin of a classifier and its robustness. We prove that the introduced margin maximization objective and the robust optimization are equivalent for deep neural networks. Thus, the proposed objective theoretically guarantees the robustness of the neural network. Our formulation generalizes support vector machine (SVM) margin maximization [110]. In a special case of deep linear networks, our objective reduces to a soft-margin SVM. We note that using SVM objective with deep neural networks was considered before [111]. However, the formulation used in [111] considered margin maximization in the feature space of the final layer which does not guarantee robustness in the input space [88]. To our best knowledge, this work is the first one to consider margin maximization in the input space for deep neural networks.

To summarize, our main contributions are as follows:

- We introduce a novel margin maximization objective for deep neural networks. Our formulation generalizes linear SVM margin objective to nonlinear classifiers. We theoretically show that the proposed margin maximization objective is equivalent to the robust optimization problem for a neural network.
- We experimentally verify the effectiveness of the proposed approach on MNIST and CIFAR-10 datasets and show that adversary-aware model optimization significantly improves the robustness of regularized networks and decreases overfitting. Additionally, we observe that adversarial images generated for the network regularized with our defense are distinguishable in a visually meaningful way from the original images.

3.2 Related work

Fooling deep neural networks Szegedy [6] considered maximizing the likelihood of incorrect prediction with norm constraint on the input perturbation. Box-constrained L-BFGS-B optimization with line search was used to find a minimal adversarial perturbation. LFGS-B optimization is time-consuming, thus impractical as a method for generating adversarial noise during training. Goodfellow [5] proposed to use a first-order approximation of the loss function to find a minimal adversarial noise. The authors showed that the scaled sign of the loss function gradient w.r.t. inputs is an adversarial direction for l_∞-norm perturbation constraint. This method, known as fast gradient sign method (FGSM), is imprecise, but it has dramatically improved the speed of generating adversarial noise. DeepFool [53] iteratively estimates adversarial noise by taking a step in the direction of the closest decision boundary until the prediction is changed. Bastani [79] observed that for a network with linear rectified units adversarial perturbation could be found by solving a linearly constrained program. Sabour [70] proposed to generate an adversarial noise such that the source image has a similar hidden layer representation to the guided image. Their experiments demonstrated that the hidden layer representation is also not robust to the adversarial noise. Nguyen [7] proposed an evolutionary algorithm to generate synthetic images that are classified by the neural network with high confidence.

Improving deep neural networks robustness Training on a mixture of clean and adversarial examples was considered as a way to regularize model in [6]. Computationally intensive L-BFGS-S optimization in [6] limited an application of adversarial noise for data augmentation during training. Goodfellow [5] introduced adversarial training (AT), which uses a weighted combination of the loss on clean and adversarial examples during training. Adversarial noise was generated using less accurate fast gradient sign method [5], which allowed efficient data regularization during model training. Layer-wise contractive penalty [101] was considered to improve robustness of deep neural networks in [100]. Miyato [91] proposed virtual adversarial training (VAT) using local distributional smoothness (LDS) regularization. LDS objective is defined as a KL-divergence between prediction distribution on clean images and prediction distribution on adversarially corrupted images. Notably, VAT training can be applied to semi-supervised learning. Improving robustness to image preprocessing (e.g. compression, cropping) was considered

in [109]. Network distillation [112] as a defense against adversarial noise was proposed in [113]. Double Backpropagation [114] is also related to our work. Double backpropagation minimizes the sum of squared input gradients and is similar to the contractive penalty [101].

Robustness and generalization Robust classification and optimization under input uncertainty have been extensively studied in machine learning [115]. For several classifiers, their regularized formulation was shown to be equivalent to the robust optimization problem. The most notable example is a regularized support vector machine [110]. Intuitively, SVM's margin maximization objective guarantees the robustness of the prediction to any adversarial perturbation with the norm less than the optimal margin. This connection between SVM margin maximization and robust optimization was formally established in [88]. Xu [88] proved that soft-margin SVM is equivalent to the robust optimization problem. In addition, the authors showed that a similar result holds for kernelized SVM with locally smooth kernels. Generalized algorithmic robustness was introduced in [108] and was used to derive and improve generalization bounds. Our work generalizes SVM margin maximization objective to nonlinear classifiers and provides similar theoretical guarantees. We note that [111] trained a deep neural network with SVM margin objective. However, standard SVM formulation does not guarantee robustness in the input space for highly nonlinear DNNs.

Several works examined why neural networks are unstable to adversarial noise. Szegedy [6] argued that adversarial examples occupy low-probability pockets in an input space of zero measure. Contrary to that, Goodfellow et al. [5] demonstrated that adversarial noise could be generated by taking a single step in the direction of gradient w.r.t. inputs. Thus, adversarial examples form linear subspaces in the input space. Further experiments in [66] verified that adversarial examples occupy large regions in the pixel space. Fawzi [60] studied the relationship between robustness to random and adversarial noise. Their theoretical analysis showed that in high dimensions, given a small curvature of the classifier decision boundary, robustness to the random noise can be guaranteed even when the network is vulnerable to the adversarial noise.

3.3 Margin Maximization and Robustness

Before introducing margin maximization objective for deep neural networks, we first recall formulation of linear SVM [110] and review some results for the robustness of regularized SVM [88].

Let $f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b$ be a binary classifier where $\mathbf{w} \in \mathbb{R}^N$ and $b \in \mathbb{R}$ are the classifier parameters. Given the input $\mathbf{x}$, the prediction $\hat{y}$ is $\text{sign}(f(\mathbf{x}))$. Following standard SVM terminology, we define *functional margin* $\tilde{\gamma}$ and *geometric margin* γ for a given training example $(\mathbf{x}, y)$ as:

$$\tilde{\gamma} = y(\mathbf{w}^T\mathbf{x} + b) \tag{3.1}$$

$$\gamma = \frac{\tilde{\gamma}}{\|\mathbf{w}\|_2} \tag{3.2}$$

We also introduce *margin sensitivity* ϵ, which we define as a scaling factor in the relationship between functional and geometric margin. *Margin sensitivity* measures how much functional margin changes as we move towards the decision boundary. It is constant for affine classifier and equal to $\|\mathbf{w}\|_2$. However, this scaling factor for geometric margin is, in general, a function of training input $\mathbf{x}$ and training label y, which motivates our introduction of a new definition.

Simple arithmetic manipulations can be used to show that maximizing minimum geometric margin is equivalent to the following optimization problem:

$$\begin{aligned} &\min \|\mathbf{w}\|_2^2 \\ &\text{s.t. } y_i(\mathbf{w}^T\boldsymbol{x}_i + b) \geq 1 \quad \forall\, i = 1, \ldots, m \end{aligned} \tag{3.3}$$

where m is the number of training examples. The formulation above is known as hard-margin SVM. Hard-margin SVM works only for separable data and is sensitive to the outliers. Slack variables are usually introduced to maximize soft margin. The objective for soft-margin SVM is defined as follows:

$$\min C\|\mathbf{w}\|_2^2 + \sum_{i=1}^{m} \big(1 - y_i(\mathbf{w}^T\boldsymbol{x}_i + b)\big)_+ \tag{3.4}$$

where $z_+ = \max(z, 0)$ is a hinge loss.

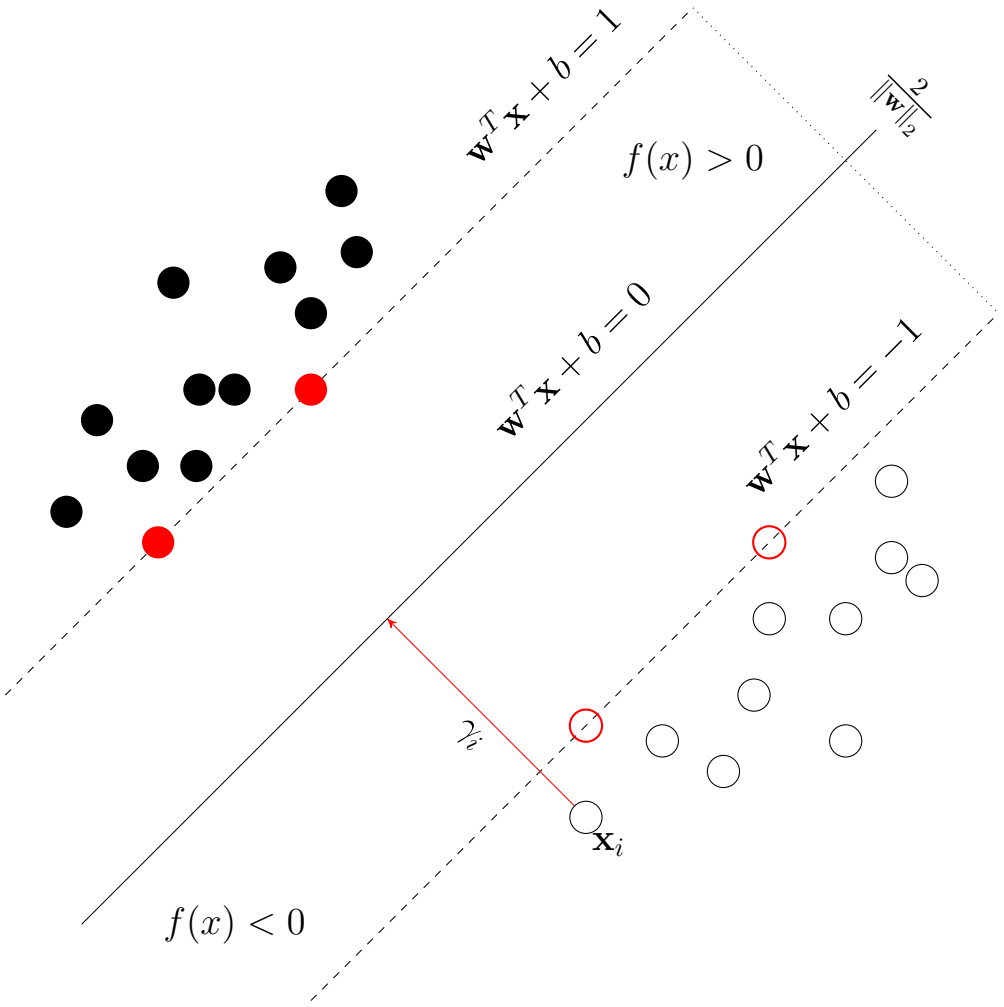


FIGURE 3.2: A decision surface of a linear SVM classifier. For a given datapoint $\mathbf{x}_i$ any input perturbation with norm less than γ_i does not change classifier prediction $\hat{y}_i$. In particular, hard-margin SVM, which maximizes minimum separating margin, is robust to any perturbation with the norm less than $1/\|w\|_2$. The diagram above is best viewed on screen.

The theory of SVM is well developed. A large classifier margin is usually associated with reduced classifier complexity and improved generalization ability of classifier [110]. By contrast, in this work we explore another facet of the large margin, namely an improved robustness of classifier. Classifier margin and its robustness are closely related. Indeed, classifier prediction $\hat{y} = f(\mathbf{x})$ for a given data point $(\mathbf{x}, y)$ is stable to any perturbation $\mathbf{r}$ with norm less than geometric margin γ for that point:

$$\operatorname{sign} f(\mathbf{x}+\mathbf{r}) = \operatorname{sign} f(\mathbf{x}) \quad \forall \, \|\mathbf{r}\|_2 \leq \gamma(\mathbf{x}) \tag{3.5}$$

In particular, for any given datapoint $(\mathbf{x}, y)$ hard-margin SVM is robust to any perturbation $\mathbf{r}$ with norm less than the optimal margin $1/\|\mathbf{w}\|_2$ (see fig. 3.2).

We measure classifier robustness as average geometric margin for each datapoint:

$$\bar{\gamma} = \frac{1}{m}\sum_{i=1}^{m}\gamma_i = \frac{1}{m}\sum_{i=1}^{m}\frac{y_i(\mathbf{w}^T\mathbf{x}_i + b)}{\|\mathbf{w}\|_2} \tag{3.6}$$

A robust classifier should have a large average margin. We can observe similarities between definition of average margin in Equation (3.6) and soft-margin SVM objective in Equation (3.4). Soft-margin SVM formulation maximizes average functional margin and minimizes margin sensitivity, which is the numerator and denominator in Equation (3.6). This observation is informal but provides useful intuition. We will use this intuition later to generalize soft-margin maximization objective to deep neural networks.

A formal connection between soft-margin SVM and robust optimization was established in [88]. Xu et al. [88, Theorem 3] showed that for non-separable dataset soft-margin SVM is equivalent to the following optimization problem:

$$\min : \max_{(\mathbf{r}_1,\ldots,\mathbf{r}_m)\in\mathcal{T}} \sum_{i=1}^{m}\left(1 - y_i\left(\mathbf{w}^T(\mathbf{x}_i - \mathbf{r}_i) + b\right)\right)_+ \tag{3.7}$$

where $\mathcal{T} = \{(\mathbf{r}_i, \ldots, \mathbf{r}_m) \,|\, \sum_{i=1}^{m}\|\mathbf{r}_i\|^* \leq C\}$ is an uncertainty set. The above theorem holds for any l_p weight norm in soft-margin formulation and its dual norm in uncertainty set $\mathcal{T}$. To simplify our discussion, we consider l_2-norm geometric margin which is used in standard soft-margin SVM. Generalization to l_p-norm should be straightforward.

Next, we generalize binary soft-margin SVM to multiclass classification. Let $f(\mathbf{x}) = \mathbf{W}^T\boldsymbol{x} + \mathbf{b}$ be a k-class classifier where $\mathbf{W} \in \mathbb{R}^{N\times k}$ and $\mathbf{b} \in \mathbb{R}^k$ are its parameters. Given the input $\mathbf{x}$, the prediction $\hat{y}$ is $\arg\max f(\mathbf{x})$. A multiclass classifier can be viewed as a collection of $\binom{k}{2}$ binary classifiers. A binary classifier for a pair of classes i and j is defined as $f_{i,j}(\mathbf{x}) = \mathbf{w}_{i,j}^T\mathbf{x} + b_{i,j}$ where $\mathbf{w}_{i,j}$ is the difference between columns i and j of weight matrix $\mathbf{W}$ and $b_{i,j}$ is the difference between class biases.

For a given input pair $(\mathbf{x}, y)$, its geometric margin is a minimum margin selected among all binary classifiers pairs:

$$\gamma = \min_{i \neq y} \frac{\mathbf{w}_{y,i}^T \mathbf{x} + b_{y,i}}{\|\mathbf{w}_{y,i}\|_2} \tag{3.8}$$

Geometrical margin γ for a multiclass classifier is the minimum distance to the side of the inscribed polyhedron.

We now formulate robust multiclass SVM using the definition of geometric margin in Equation (3.8):

$$\begin{aligned} \min \; & \sum_{i=1}^{k} C \max_{j \neq i, j > i} \|\mathbf{w}_{i,j}\|_2^2 + \sum_{i=1}^{m} \epsilon_i \\ \text{s. t. } & \mathbf{w}_{y_i,j}^T \mathbf{x}_i + b_{y_i,j} + \delta_{y_i,j} \geq 1 - \epsilon_i \quad \forall\, i, j \end{aligned} \tag{3.9}$$

where $\delta_{i,j}$ is the Kronecker delta; $\mathbf{w}_{i,j}$ is the difference between columns i and j in $\mathbf{W}$; $b_{i,j}$ is the difference between class biases. The equivalence to the robust optimization problem can be shown using a similar argument to [88, Theorem 3] where the equivalence was proved for binary soft-margin SVM. Due to a lack of space, we skip the proof here.

The closest to the proposed optimization problem is Crammer and Singer multiclass SVM [116]. Crammer and Singer [116] considered minimizing the average column difference of weight matrix $\mathbf{W}$, which is equivalent to minimizing matrix norm. By contrast, we propose to minimize the maximum column difference. The distinction is important as our formulation is equivalent to the robust optimization problem.

3.4 Deep Margin Maximization

Next, we develop a margin maximization objective for deep neural networks. Similar to the discussion in the previous section, we first consider binary classification.

Let $f(\mathbf{x}; \mathbf{W})$, shorthand $f(\mathbf{x})$, be a neural network where $\mathbf{W}$ represents the network parameters. The class prediction $\hat{y}$ is sign $f(\mathbf{x})$. Importantly, we consider that the output of the neural network is unnormalized. For a given datapoint $(\mathbf{x}, y)$, its l_2-margin, which is a minimum distance to the decision hyperplane, can be defined

as follows:

$$\gamma = \min\{\|\mathbf{r}\|_2 \,|\, f(\mathbf{x}+\mathbf{r}) = 0\} \tag{3.10}$$

where $\mathbf{r}$ is the input perturbation.

For a linear classifier, such as SVM, a geometric margin can be found using a closed-form solution. Non-linear decision boundary of a neural network requires approximation. We consider a first-order approximation of the function level set:

$$\begin{aligned} &\min \ \|\mathbf{r}\|_2 \\ &\text{s.t. } f(\mathbf{x}) + \nabla_{\mathbf{x}} f(\mathbf{x})^T \mathbf{r} = 0 \end{aligned} \tag{3.11}$$

where $\nabla_{\mathbf{x}} f(\mathbf{x}) = \frac{\partial f(\mathbf{x})}{\partial \mathbf{x}}$ is the gradient of the output w.r.t. input. Similar functional approximation was used for DeepFool attack [53] during computation of a minimal adversarial distortion.

The solution of Equation (3.11) can be found using a method of Lagrange multipliers. Then, l_2-distance to the decision boundary is:

$$\gamma = \frac{|f(\mathbf{x})|}{\|\nabla_{\mathbf{x}} f(\mathbf{x})\|_2} \tag{3.12}$$

And l_p geometric margin is:

$$\gamma = \frac{|f(\mathbf{x})|}{\|\nabla_{\mathbf{x}} f(\mathbf{x})\|_q} \tag{3.13}$$

where p and q such that $\frac{1}{p} + \frac{1}{q} = 1$. We can see from the above definition of the margin that *margin sensitivity* ϵ for non-linear classifier depends on the input $\mathbf{x}$ and is equal to the norm of the input gradient $\|\nabla_{\mathbf{x}} f(\mathbf{x})\|$.

Similar to the binary linear SVM, we propose soft-margin maximization objective for the binary neural network:

$$\min \sum_{i=1}^{m} C\|\nabla_{\mathbf{x}} f(\mathbf{x}_i)\|_2 + (1 - y_i f(\mathbf{x}_i))_+ \tag{3.14}$$

We can observe that the formulation in Equation (3.14) for deep linear networks reduces to the standard soft-margin SVM. Additionally, our formulation provides a novel margin interpretation of the methods [101, 114] that minimize the sum of squared input gradients.

The next theorem establishes a connection between the above objective and robust optimization for binary deep neural networks.

Theorem 3.4.1. Let $\mathcal{T}_i = \{\mathbf{r}_i \,|\, \|\mathbf{r}_i\|^* \leq C\}$ be an uncertainty set where $\mathbf{r}_i$ is the perturbation for $\mathbf{x}_i$. Then, the optimization problem in Equation (3.14) approximately minimizes the following robust optimization problem:

$$\min : \sum_{i=1}^{m} \max_{\mathbf{r}_i \in \mathcal{T}_i} (1 - y_i f(\mathbf{x}_i - \mathbf{r}_i))_+$$

Proof. After applying first-order approximation, we get:

$$\begin{aligned}
&\sum_{i=1}^{m} \sup_{\mathbf{r}_i \in \mathcal{T}_i} (1 - y_i f(\mathbf{x}_i - \mathbf{r}_i))_+ \\
\approx &\sum_{i=1}^{m} \sup_{\mathbf{r}_i \in \mathcal{T}_i} \left(1 - y_i (f(\boldsymbol{x}_i) - \nabla_{\mathbf{x}} f(\boldsymbol{x}_i)^T \mathbf{r}_i)\right)_+ \\
\leq &\sum_{i=1}^{m} (1 - y_i f(\boldsymbol{x}_i))_+ + \sup_{\mathbf{r}_i \in \mathcal{T}_i} (y_i \nabla_{\mathbf{x}} f(\boldsymbol{x}_i)^T \mathbf{r}_i) \\
= &\sum_{i=1}^{m} (1 - y_i f(\boldsymbol{x}_i))_+ + C \|\nabla_{\mathbf{x}} f(\boldsymbol{x}_i)\|
\end{aligned}$$

which completes the proof. □

Our proof mirrors the proof of [88, Theorem 3], where similar equivalence was established for binary soft-margin SVM. Unlike SVM, our result holds only in approximation. We can also observe that the above formulation generalizes soft-margin SVM. Indeed, soft-margin SVM for deep linear networks is equivalent to the proposed objective. Next, we generalize the formulation of the robust multiclass SVM Equation (3.9) to deep neural networks.

Neural network $f(\mathbf{x})$ for k-class classification can be viewed as a collection of $\binom{k}{2}$ binary neural networks with shared parameters for each pair of classes. For a pair of classes i and j, the binary classifier can be defined as follows:

$$f_{i,j}(\mathbf{x}) = f_i(\mathbf{x}) - f_j(\mathbf{x}) \tag{3.15}$$

where $f_i(\mathbf{x})$ is the output of i-logits unit in the neural network. The prediction is class i if $f_{i,j}(\mathbf{x}) \geq 0$ and class j otherwise.

Combining with Equation (3.12), geometric margin between class i and j is given by:

$$\gamma_{i,j} = \frac{|f_i(\mathbf{x}) - f_j(\mathbf{x})|}{\|\nabla_{\mathbf{x}} f_i(\mathbf{x}) - \nabla_{\mathbf{x}} f_j(\mathbf{x})\|} \tag{3.16}$$

Then, a geometric margin for an input $\mathbf{x}$ is a minimum margin among all class pairs given the target label y:

$$\gamma = \min_{y \neq j} \gamma_{y,j} = \min_{j \neq y} \frac{|f_y(\mathbf{x}) - f_j(\mathbf{x})|}{\|\nabla_{\mathbf{x}} f_y(\mathbf{x}) - \nabla_{\mathbf{x}} f_j(\mathbf{x})\|} \tag{3.17}$$

The geometric margin for a multiclass deep neural network has an interpretation of the shortest distance to the side of the decision region, which is approximated using polyhedron (see fig. 3.3). *Margin sensitivity* for the multiclass deep neural network is the maximum norm of the difference between the input gradient of the correct prediction and the incorrect prediction. If datapoint $(\mathbf{x}, y)$ is classified correctly with the functional margin at least some constant C, we can maximize geometric margin by minimizing maximum margin sensitivity.

Using the above definition of geometric margin, we propose a novel margin maximization objective for robust multiclass classification as follows:

$$\min \sum_{i=1}^{m} \max_{j \neq y_i} (1 + f_{y_i}(\mathbf{x}_i) - f_j(\mathbf{x}_i))_+ + C\epsilon(\mathbf{x}_i, y_i) \tag{3.18}$$

where $\epsilon(\mathbf{x}, y) = \max_{j \neq i} \|\nabla_{\mathbf{x}} f_i(\mathbf{x}) - \nabla_{\mathbf{x}} f_j(\mathbf{x})\|$ is the *margin sensitivity* for a given datapoint $\boldsymbol{x}$ with label y. In experiments, we considered minimization of l_2 and l_1 margin sensitivity. We note that minimizing l_1-norm corresponds to the maximization of l_∞-distance to the decision boundary.

The next theorem establishes an equivalence of the above objective and robust multiclass optimization.

Theorem 3.4.2. Let $\mathcal{T}_i = \{\mathbf{r}_i \,|\, \|\mathbf{r}_i\|_* \leq C\}$ be an uncertainty set where $\mathbf{r}_i$ is the perturbation for $\mathbf{x}_i$. Then, the optimization problem in Equation (3.18) approximately minimizes the following robust optimization problem:

$$\min : \sum_{i=1}^{m} \max_{\mathbf{r}_i \in \mathcal{T}_i; j \neq y_i} (1 + f_j(\hat{\mathbf{x}}_i) - f_{y_i}(\hat{\mathbf{x}}_i))_+ \tag{3.19}$$

where $\hat{\mathbf{x}}_i = \mathbf{x}_i - \mathbf{r}_i$ is the corrupted input.

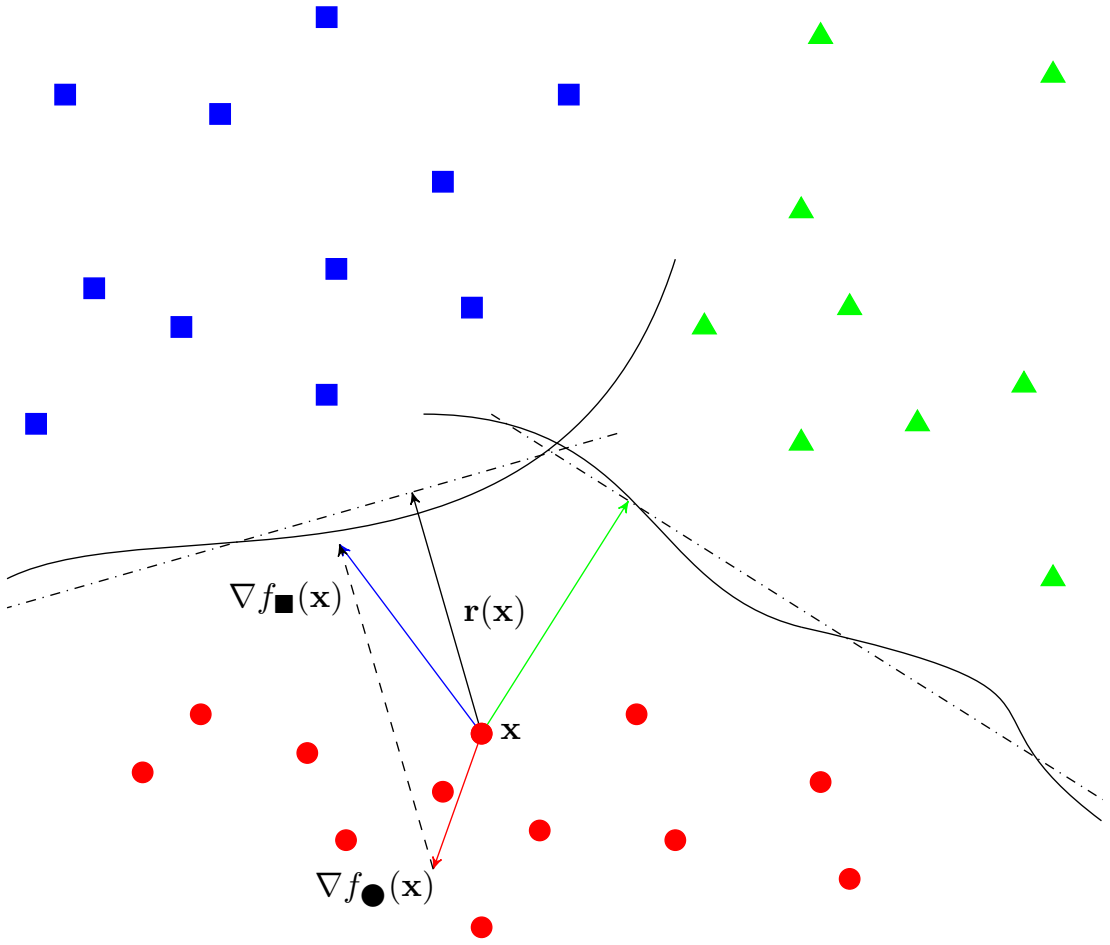


FIGURE 3.3: A visualization of margin maximization principle for a multiclass neural network. Prediction region $f(\mathbf{x}+\mathbf{r}) = f(\mathbf{x})$ is approximated using polytope. Adversarial perturbation $\mathbf{r}(\mathbf{x})$ is the minimum distance to its facet. Our method minimizes *margin sensitivity* which is the norm of the difference between input gradient for the correct prediction $\nabla_{\mathbf{x}} f_{\bullet}(\mathbf{x})$ and the incorrect one $\nabla_{\mathbf{x}} f_{\blacksquare}(\mathbf{x})$ (showed in dashed line). The diagram is best viewed on screen.

Proof. Let $f_{i,j}(\mathbf{x}) = f_i(\mathbf{x}) - f_j(\mathbf{x})$ be the difference between the prediction for class i and j. Now, for any i, the following holds:

$$
\begin{aligned}
&\sup_{\mathbf{r}_i \in \mathcal{T}_i} \max_{j \neq y_i} \left(1 - f_{j,y_i}(\mathbf{x}_i - \mathbf{r}_i)\right)_+ \\
\approx &\sup_{\mathbf{r}_i \in \mathcal{T}_i} \max_{j \neq y_i} \left(1 - f_{j,y_i}(\mathbf{x}_i) + \nabla_{\mathbf{x}} f_{j,y_i}(\mathbf{x}_i)^T \mathbf{r}_i\right) \\
\leq &\max_{j \neq y_i} \left(1 - f_{j,y_i}(\mathbf{x}_i)\right) + \sup_{\mathbf{r}_i \in \mathcal{T}_i} (\nabla_{\mathbf{x}} f_{j,y_i}(\mathbf{x}_i)^T \mathbf{r}_i) \\
= &\max_{j \neq y_i} \left(1 - f_{j,y_i}(\mathbf{x}_i)\right) + C \|\nabla_{\mathbf{x}} f_{j,y_i}(\mathbf{x}_i)\|
\end{aligned}
$$

Since each perturbation r_i is independent, summing up for all i completes the proof. □

The proposed margin maximization objective has an interpretation of minimizing input sensitivity of the classifier prediction in the direction of the closest decision boundary. Besides, the connection between contractive penalty [101] and our regularization method can be shown. Recall the contractive penalty is defined as a Frobenius norm of the network Jacobian $\mathcal{J}(\mathbf{x})$. Now, consider l-layer neural network in which the last layer is fully-connected. Then, our margin maximization can be viewed as minimizing the projection of the $(l-1)$-layer Jacobian $(\mathbf{w}_j^{(l)} - \mathbf{w}_y^{(l)})\mathbf{J}_{l-1}(\mathbf{x})$ where $\mathbf{w}_j^{(l)}$ is the j-column in the last layer weight matrix and y is the correct label.

3.5 Experiments

All computations were conducted using Theano [117] that supports the symbolic differentiation of complex expressions and computation on GPU. The source code to reproduce the experiments in this section is available online at `https://github.com/aam-at/margin_maximization`.

We performed several experiments to study regularization effects and robustness of the proposed margin objective on MNIST and CIFAR-10 datasets. Minimal adversarial perturbation $\mathbf{r}(\mathbf{x})$ was estimated using Deepfool algorithm [53]. We measured the average robustness of the neural network as follows:

$$\rho_{\text{adv}}(f) = \frac{1}{|\mathcal{D}|} \sum_{\mathbf{x} \in \mathcal{D}} \frac{\|\mathbf{r}(\mathbf{x})\|}{\|\mathbf{x}\|} \tag{3.20}$$

where $\mathcal{D}$ is the subset of test examples that are classified correctly by the model f. We note that Deepfool tends to overestimate a minimal adversarial perturbation for the regularized networks. To avoid this behavior, we clipped the norm of the perturbation at 0.5 for each iteration. The number of iterations was limited to 50.

In addition, we calculated test error on images corrupted using fast gradient sign method [5] for which adversarial direction is given as follows:

$$\mathbf{r}(\mathbf{x}) = \epsilon \operatorname{sign}(\nabla_{\mathbf{x}} \mathcal{L}(f, \mathbf{x}, y)) \tag{3.21}$$

where $\mathcal{L}$ is the categorical cross-entropy function.

3.5.1 MNIST experiments

MNIST is a dataset of handwritten digits images that consists of 60000 training and 10000 testing examples. Each image has 28×28 size. Its corresponding label is one of the numerals from 0 to 9. We randomly split the original 60000 training examples into 50000 and 10000 samples used for training and validation. We used the validation dataset for tuning hyperparameters. The images were preprocessed to be in $[0, 1]$ range. No data augmentation was used in all experiments. We studied the effects of the proposed regularization for two models: fully-connected network with three hidden layers of size $(1000, 1000, 1000)$, and convolutional network Lenet-5 [24]. We used Adam optimizer [29] with the default learning rate and exponential learning decay with 0.95 decay rate. All networks were trained for 100 epochs with minibatch of 100 examples.

First, we studied the dependency between test error and network robustness for different values of the regularization coefficient C. We trained convolutional neural network Lenet-5 and fully-connected neural network with l_2 margin regularization for $C \in [1.0, 10^{-6}]$. Average robustness and test error were computed for three separate runs with different random seeds. In this experiment, we considered only l_2 margin maximization as we observed that training with l_1 margin sensitivity is unstable for $C \geq 0.1$, which suggests that it has a stronger regularization effect. The results for the convolutional neural network are shown in Figure 3.4. We can observe in fig. 3.4 that the robustness of the convolutional neural network Lenet-5 steadily increases as we change the strength of our regularization objective. The results for fully-connected network are shown in Figure 3.5. We visually examined adversarial examples for $C \in [10^{-4}, 10^{-3}]$. We observed that, while the robustness is relatively high, adversarial examples are not visually meaningful. We believe that the extreme values of the robustness are the result of a failure of DeepFool attack [53] to find an adversarial perturbation due to gradient masking [64]. Using validation dataset, we selected optimal value of $C = 0.1$ for l_2-margin regularization and used $C = 0.01$ for l_1-margin regularization.

Next, we performed a numerical comparison of our l_1- and l_2 margin maximization method against baseline (no regularization), dropout [118], adversarial training (AT) [5], virtual adversarial training (VAT) [91]. For AT [5], we used fixed $\epsilon = 0.2$ which we selected using validation dataset. For VAT [91], we used implementation provided by the authors, which at the moment of this publication

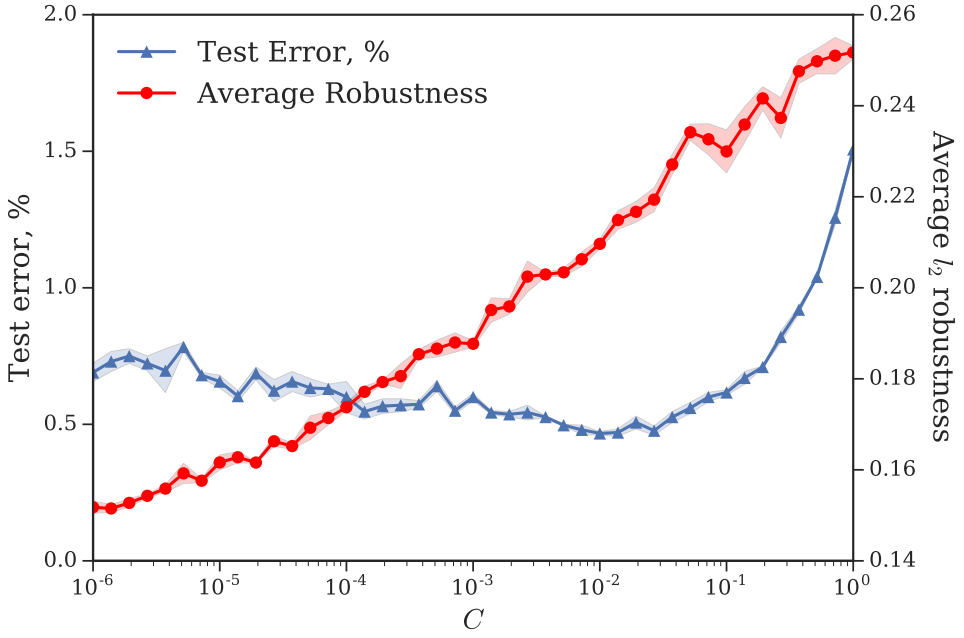


FIGURE 3.4: Test error and network robustness of convolutional neural network Lenet-5 for different values of C in the interval $[1, 10^{-6}]$. We report the results for 10 independent runs. The grade regions along the curves are the 3σ spread of the results between random runs.

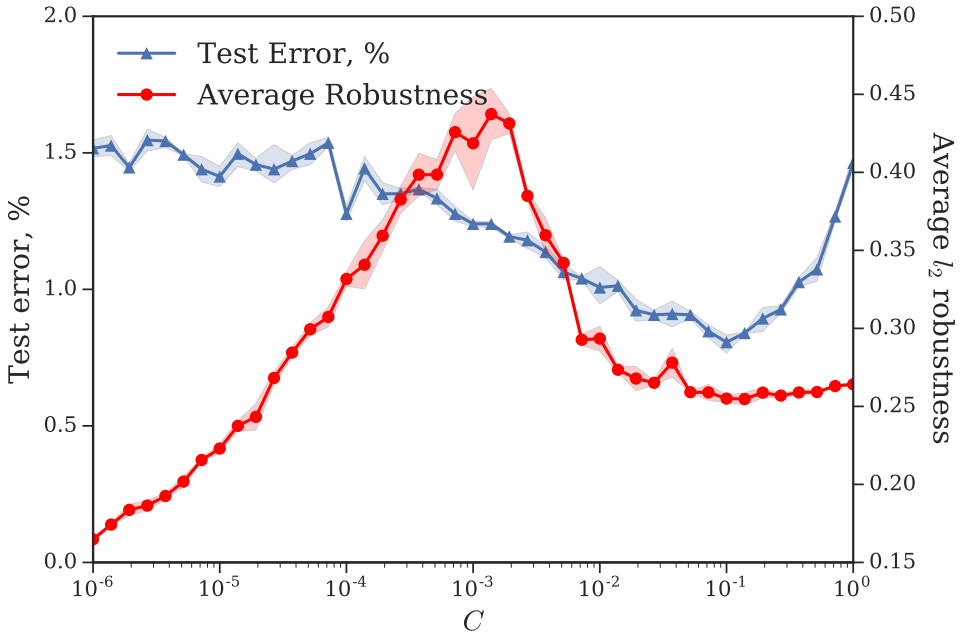


FIGURE 3.5: Test error and network robustness of fully-connected network for different values of C in the interval $[1, 10^{-6}]$. We report the results for 10 independent runs. The grade regions along the curves are the 3σ spread of the results between random runs.

Network	Error %	Adv. Error %	$\rho_{\text{adv}} \times 10^{-1}$
MLP baseline	1.42 ± 0.08	95.8 ± 1.89	1.14 ± 0.01
Dropout [118]	1.34 ± 0.05	93.4 ± 2.08	1.20 ± 0.01
AT [5]	1.19 ± 0.06	$\mathbf{10.17 \pm 0.69}$	1.60 ± 0.05
VAT [91]	$\mathbf{0.87 \pm 0.04}$	24.33 ± 1.38	$\mathbf{2.69 \pm 0.02}$
Our l_1	$\mathbf{0.84 \pm 0.03}$	32.43 ± 1.25	$\mathbf{2.73 \pm 0.08}$
Our l_2	$\mathbf{0.86 \pm 0.04}$	42.56 ± 1.37	2.59 ± 0.05

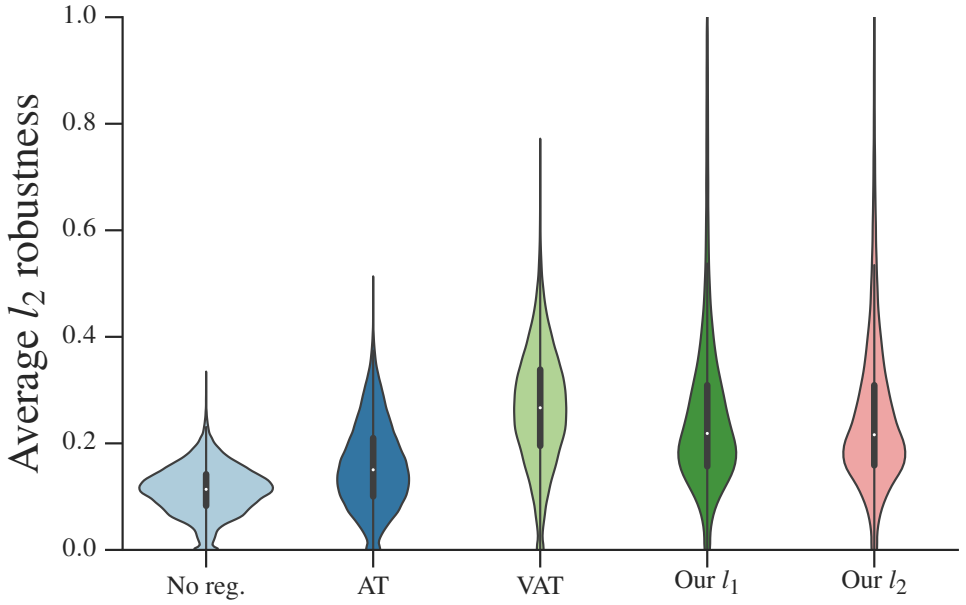
(A)

Network	Error %	Adv. Error %	$\rho_{\text{adv}} \times 10^{-1}$
Lenet-5 baseline	0.72 ± 0.06	72.14 ± 2.20	1.54 ± 0.04
Dropout [118]	$\mathbf{0.58 \pm 0.03}$	61.66 ± 3.45	1.70 ± 0.05
AT [5]	0.73 ± 0.05	$\mathbf{4.95 \pm 1.26}$	2.00 ± 0.03
Our l_1	0.64 ± 0.02	20.47 ± 2.52	$\mathbf{2.22 \pm 0.05}$
Our l_2	0.62 ± 0.04	28.26 ± 3.49	$\mathbf{2.17 \pm 0.06}$

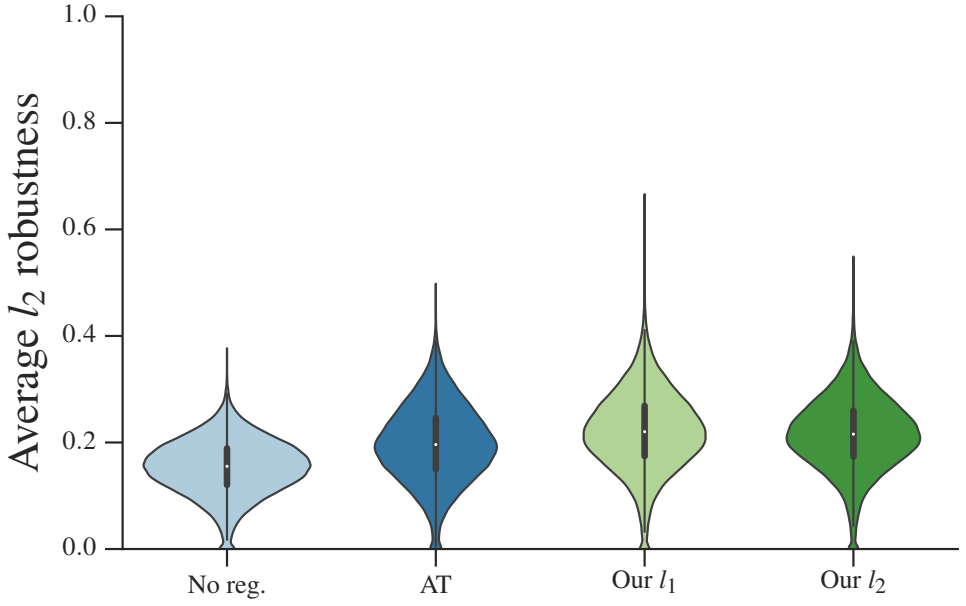
(B)

TABLE 3.1: A comparison between various defenses for fully-connected network (MLP) in Table 3.1a and convolutional network Lenet-5 in Table 3.1b on MNIST dataset. We report the summarized results for 10 random runs.

supports training only of fully-connected neural networks. For our method, we considered l_2 and l_1 margin regularization with $C = 0.1$ and $C = 0.01$ respectively. We trained neural networks for ten separated runs with different random seeds. We reported results for test error, test error on images distorted using FastGrad [5] with $\epsilon = 0.2$, and average robustness. The results are shown in Table 3.1. We can observe that three hidden layer MLP noticeably overfits without any regularization. The proposed margin objective and VAT significantly reduces overfitting and achieves high robustness scores. AT [5] performs the best on images corrupted using fast gradient sign method but is noticeably less robust. Arguably, this is the result of overfitting to the particular noise type when using data regularization. Baseline convolutional network Lenet-5 is more robust than the fully-connected network. We think this is due to using prior information about image structure in convolutional layers, e.g. translation invariance. We also note that minimization of l_1 margin sensitivity performs consistently better than l_2 margin loss on images corrupted using fast gradient sign. This result is in line with our theoretical analysis because l_1 margin objective maximizes the robustness to l_∞ perturbation.



(A)



(B)

FIGURE 3.6: Robustness ρ_{adv} of the models for various defenses. The distributions were estimated using kernel density estimator. The values for the robustness were computed for 10 different runs. The plots in figs. 3.6a and 3.6b show robustness for multilayer perceptron and Lenet-5 networks respectively.

In addition, we compared histograms of adversarial noise for the networks trained with different regularization strategies. Distribution histograms are shown in Figure 3.6. Our margin regularization and VAT method significantly improve the

FIGURE 3.7: Adversarial examples for a fully-connected neural network. The first row shows the original images. The rows from the second to the bottom show adversarial examples for the network trained with no regularization, AT [5], VAT [91], l_2-margin and l_1-margin.

robustness of the models. However, even after the regularization was applied, the networks remain relatively vulnerable to small perturbations $\rho_{\text{adv}} \leq 0.1$. Nonzero density for extreme values of the robustness suggests that Deepfool algorithm [53] often fails to estimate small adversarial distortion for the strongly regularized networks.

Lastly, we visually examined the quality of the generated adversarial images. Ideally, adversarial examples for the robust neural network should match the human perception of the visually confusing images. Adversarial examples are shown in Figure 3.7. We can observe that adversarial noise is visually meaningful. For example, digit "4" was changed to digit "9" by adding an upper stroke. Similarly, digit "1" was changed to digit "7" or digit "4". Thus, our defense quantitatively and qualitatively improves the robustness of DNNs. We suggest that future work should consider the human perception of visually confusing images as a way to compare different regularization methods.

3.5.2 CIFAR-10 experiments

CIFAR-10 is a 10-class dataset of 32×32 which consists of 50000 training and 10000 testing images. We used the last 5000 examples from the training dataset for validation. Data were preprocessed using global contrast normalization and ZCA whitening, which is commonly applied for this task. We trained Network in Network (NIN) [119] for 200 epochs with batch size 128 using stochastic gradient descent with momentum 0.9. The learning rate was divided by 5 at epochs $[80, 120, 160]$. Image flipping and random translation were used for data augmentation. Our reference implementation achieved 9.10% with data augmentation and 11.30% without augmentation. We found that NIN with multiclass hinge loss and our margin objective is unstable and often diverges. Due to its instability, we considered training NIN with categorical cross-entropy loss. We disabled weight decay when training NN with our margin maximization objective as it reduced the validation accuracy. We computed test error on clean and corrupted images using fast gradient sign [5] with $\epsilon = 0.05$ and estimate average robustness using Equation (3.20). We report the results for CIFAR-10 dataset in Table 3.2. We observe a small improvement in test and adversarial accuracy and robustness with the proposed regularization method. In future work, we plan to address the scalability of the proposed method to more complex datasets, such as ImageNet dataset.

Model	Error %	Adv. Error %	$\rho_{\text{adv}} \times 10^{-2}$
NIN w/o data augm.	11.30	68.85	5.12
NIN l_2, $C = 0.01$	11.13	**55.88**	**6.83**
NIN l_2, $C = 0.001$	**10.42**	61.53	6.45
NIN with data augm.	**9.10**	65.98	5.41
NIN l_2, $C = 0.005$	10.02	**52.80**	**7.77**

TABLE 3.2: A comparison between various defenses for Network in Network [119] on CIFAR-10 dataset.

3.6 Conclusion

We proposed a novel regularization objective for DNNs inspired by the margin maximization principle. Our formulation generalizes SVM margin objective. We

theoretically proved that the proposed margin objective approximately minimizes robust optimization problem. In experiments, we showed that the introduced regularization method: 1) improves quantitatively and qualitatively the robustness of neural networks; 2) reduces overfitting. Adversarial examples for a robust neural network should be confusing for a human observer. Ideally, future work should consider how humans perceive generated adversarial images as a reference to compare regularization methods. Another limitation of the proposed defense is that we compute the second-order gradient during training, which significantly increases the training time. We plan to address the scalability of the proposed defense to more complex datasets and larger network models in future work.

Chapter 4

Improved Network Robustness with Adversary Critic

Ideally, what confuses a neural network should be confusing to humans. However, recent experiments have shown that small, imperceptible perturbations, or adversarial noise, can change the network prediction. To address this gap in perception, we propose a novel approach for learning a robust classifier. Our main idea is: adversarial examples for the robust classifier should be indistinguishable from the regular data of the adversarial target. We formulate a problem of learning a robust classifier in the framework of Generative Adversarial Networks (GAN), where the adversarial attack on the classifier acts as a generator, and the critic network learns to distinguish between regular and adversarial images. The classifier cost is augmented with the objective that its adversarial examples should confuse the adversary critic. To improve the stability of the adversarial mapping, we introduce adversarial cycle-consistency constraint, which ensures that the adversarial mapping of the adversarial examples is close to the original. In the experiments, we show the effectiveness of our defense. Our method surpasses in terms of robustness networks trained with adversarial training. Additionally, we verify in the experiments with human annotators on MTurk that adversarial examples are indeed visually confusing.

4.1 Introduction

Adversarial training [5, 90, 120] is the most popular approach to improve network robustness, which we review in Section 2.5. During training, adversarial examples are generated online using the latest snapshot of the network parameters. The generated adversarial examples are used to augment the training dataset. Then, the classifier is trained on the mixture of the original and the adversarial images. In this way, adversarial training smoothens a decision boundary in the vicinity of the training examples. Adversarial training (AT) is an intuitive and effective defense, but it has some limitations. AT is based on the assumption that adversarial noise is label non-changing. If the perturbation is too large, the adversarial noise may change the true underlying label of the input. Secondly, adversarial training discards the dependency between the model parameters and the adversarial noise. As a result, the neural network may fail to anticipate changes in the adversary and overfit the adversary used during training.

Ideally, what confuses a neural network should be confusing to humans. So the changes introduced by the adversarial noise should be associated with removing identifying characteristics of the original label and adding identifying characteristics of the adversarial label [121]. For example, images that are adversarial to the classifier should be visually confusing to a human observer. Current techniques [5, 90, 120] improve robustness to input perturbations from a selected uncertainty set. Yet, the model's adversarial examples remain semantically meaningless. To address this gap in perception, we propose a novel approach for learning the robust classifier. Our core idea is that adversarial examples for the robust classifier should be indistinguishable from the regular data of the attack's target class. We depict a diagram of our idea in Figure 4.1.

We formulate the problem of learning a robust classifier in the framework of Generative Adversarial Networks (GAN) [122]. The adversarial attack on the classifier acts as a generator, and the critic network is trained to distinguish between natural and adversarial images. We also introduce a novel targeted adversarial attack, which we use as the generator. The classifier cost is augmented with the objective that its adversarial images generated by the attack should confuse the adversary critic. The attack is fully-differentiable and implicitly depends on the classifier

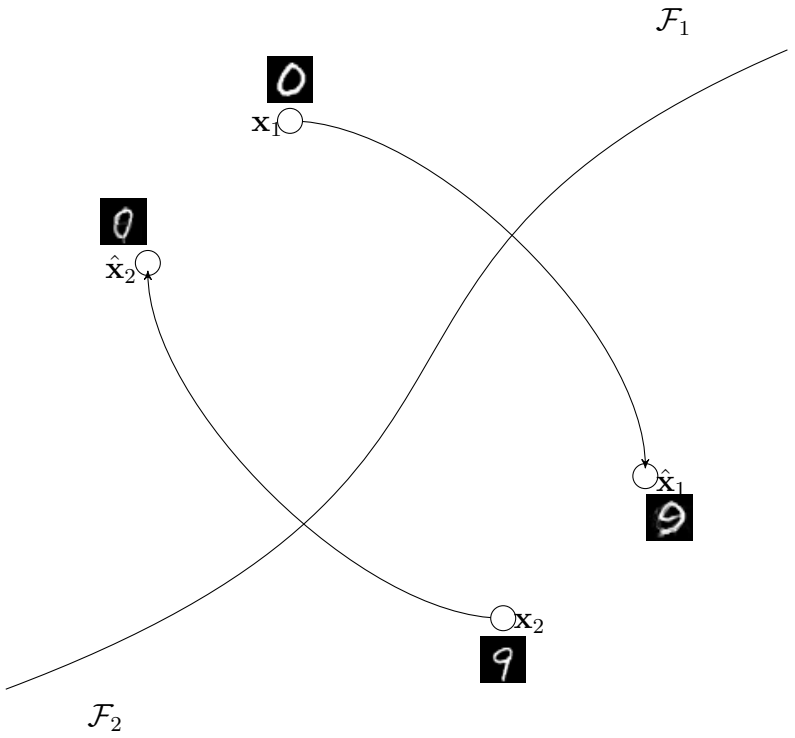


FIGURE 4.1: An intuitive visualization of the idea that adversarial examples should be indistinguishable from the regular data of the adversarial target. The images in the figure above were generated using Carlini and Wagner [18] l_2-attack on the network trained with our defense, such that the confidence of the prediction on the adversarial images is 95%. The prediction confidence on the original images $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ was 99%.

parameters. We train the classifier and the adversary critic jointly with back-propagation. To improve the stability of the adversarial mapping, we introduce adversarial cycle-consistency constraint that ensures that the adversarial mapping of the adversarial examples is close to the original. Unlike adversarial training, our method does not require adversarial noise to be label non-changing. To the contrary, we require that the changes introduced by adversarial noise should change the "true" label of the input to confuse the critic. In the experiments, we demonstrate the effectiveness of the proposed approach. Our method surpasses in terms of robustness networks trained with adversarial training. Additionally, we verify in the experiments with human annotators that adversarial examples are indeed visually confusing.

4.2 Related work

Adversarial attacks Szegedy et al. [6] originally introduced a targeted adversarial attack. Their attack generates adversarial noise by optimizing the likelihood of input for some adversarial target using a box-constrained L-BFGS method. Fast Gradient Sign method (FGSM) [5] is a one-step attack that uses a first-order approximation of the likelihood loss. Basic Iterative Method (BIM) [8], which is also known as Projected Gradient Descent (PGD) [17], iteratively applies the first-order approximation and projects the perturbation after each step. [52] proposed an iterative method which at each iteration selects a single most salient pixel and perturbs it. DeepFool [53] iteratively generates adversarial perturbation by taking a step in the direction of the closest decision boundary. The decision boundary is approximated with first-order Taylor series to avoid complex non-convex optimization. Then, the geometric margin can be computed in the closed-form. Carlini and Wagner [18] proposed an optimization-based attack on a modified loss function with implicit box-constraints. [64] introduced a black-box adversarial attack based on the transferability of adversarial examples. Adversarial Transformation Networks (ATN) [123] trains a neural network to attack.

Defenses against adversarial attacks Adversarial training (AT) [5] augments training batch with adversarial examples which are generated online using Fast Gradient Sign method. Virtual Adversarial training (VAT) [91] minimizes Kullback-Leibler divergence between the predictive distribution of clean inputs and adversarial inputs. Notably, adversarial examples can be generated without using label information; VAT was successfully applied in semi-supervised settings. [17] applies iterative Projected Gradient Descent (PGD) attack to adversarial training. Stability training [109] minimizes a task-specific distance between the output on clean and the output on corrupted inputs. However, only a random noise was used to distort the input. [55, 124] proposed to maximize a geometric margin to improve classifier robustness. Parseval networks [125] are trained with the regularization constraint, so the weight matrices have a small spectral radius. Most of the existing defenses are based on robust optimization and improve the robustness to perturbations from a selected uncertainty set.

In our work, the adversary critic is somewhat similar to the adversary detector.

However, unlike adversary-detection methods, we use information from the adversary critic to improve the robustness of the guarded model during training. We do not use the adversary critic during testing.

Generative Adversarial Networks [122] introduced a generative model where the learning problem is formulated as an adversarial game between discriminator and generator. The discriminator is trained to distinguish between real images and generated images. The generator is trained to produce naturally looking images which confuse the discriminator. A two-player minimax game is solved by alternatively optimizing two models. Recently several defenses have been proposed which use GAN framework to improve the robustness of neural networks. Defense-GAN [126] used the generator at test time to project the corrupted input on the manifold of the natural examples. Lee et al. [127] introduced Generative Adversarial Trainer (GAT) in which the generator is trained to attack the classifier. Like Adversarial Training [5], GAT requires that adversarial noise does not change the label. Compared to defenses based on robust optimization, we do not put any prior constraint on the adversarial attack. To the contrary, we require that adversarial noise for a robust classifier should change the "true" label of the input to confuse the critic. Our formulation has three components (the classifier, the critic, and the attack) and is also related to Triple-GAN [128]. But, in our work: 1) the generator also fools the classifier; 2) we use the implicit dependency between the model and the attack to improve the robustness of the classifier. Also, we use a fixed algorithm to attack the classifier.

4.3 Limitations of Robust Optimization

We first recall a mathematical formulation for the robust multiclass classification. We review in detail approaches based on robust optimization (RO) in Section 2.5. RO seeks a solution robust to the worst-case input perturbations:

$$\min_{\mathbf{W}} \sum_{i=1}^{N} \max_{\boldsymbol{r}_i \in \mathcal{U}_i} \mathcal{L}(f(\boldsymbol{x}_i + \boldsymbol{r}_i; \mathbf{W}); y_i) \tag{4.1}$$

where $\mathcal{L}$ is the training loss; $\boldsymbol{r}_i$ is an arbitrary (even adversarial) perturbation for the input $\boldsymbol{x}_i$; $\mathcal{U}_i$ is the uncertainty set, e.g. l_p-norm ϵ-ball $\mathcal{U}_i = \{\boldsymbol{r}_i : \|\boldsymbol{r}_i\|_p \leq \epsilon\}$. We select the uncertainty set $\mathcal{U}$ using prior information about the problem.

FIGURE 4.2: Adversarial examples for the model regularized with our adversary critic objective. Images off-diagonal are corrupted with the adversarial noise generated by CW [18] l_2-norm attack, so the prediction confidence on the adversarial images is at least 95%. The prediction confidence on the original images is 99%.

Selecting a good uncertainty set $\mathcal{U}$ for robust optimization is crucial. A poorly chosen uncertainty set may result in an overly conservative robust model. Most importantly, each perturbation $\boldsymbol{r} \in \mathcal{U}$ should leave the "true" class of the original input $\boldsymbol{x}$ unchanged. To ensure that the changes of the network prediction are indeed fooling examples, Goodfellow et al. [5] argued in favor of a max-norm perturbation constraint for image classification problems. However, simple disturbance models (e.g. l_2- and l_∞-norm ϵ-ball used in adversarial training) are inadequate in practice because the distance to the decision boundary for different examples may significantly vary. Several methods have been developed to adapt uncertainty set to the problem at hand. [129] constructed a data-dependent uncertainty set using statistical hypothesis tests. In this work, we propose a novel approach to learning a robust classifier that is orthogonal to prior robust optimization methods.

Ideally, inputs that are adversarial to the classifier should be confusing to a human observer. So the changes introduced by the adversarial noise should be associated with the removing of identifying characteristics of the original label and adding

the identifying characteristics of the adversarial target. For example, adversarial images in Figure 4.2 are visually confusing. The digit '1' (second row, eighth column) after adding the top stroke was classified by the neural network as digit '7'. Likewise, the digit '7' (eighth row, second column) after removing the top stroke was classified by the network as digit '1'. For other images in Figure 4.2, the model's "mistakes" can be predicted visually. Such behavior of the classifier is expected and desired for the problems in computer vision. Additionally, it improves the interpretability of the model. We study image classification problems in this work, but our formulation can be extended to the classification tasks in other domains, e.g. audio or text.

Based on the above intuition, we develop a novel formulation for learning a robust classifier. A classifier is robust if its adversarial examples are indistinguishable from the regular data of the adversarial target (see fig. 4.1). We formulate the following mathematical problem to learn a robust classifier:

$$\min \sum_{i=1}^{N} \mathcal{L}(f(\boldsymbol{x}_i), y_i) + \lambda \mathcal{D}\left[p_{\text{data}}\left(\boldsymbol{x}, y\right), p_{\text{adv}}\left(\boldsymbol{x}, y\right)\right] \tag{4.2}$$

where $\mathcal{D}$ is the distance between two distributions, e.g. KL-divergence; $p_{\text{data}}\left(\boldsymbol{x}, y\right)$ and $p_{\text{adv}}\left(\boldsymbol{x}, y\right)$ is the distribution of the natural and the adversarial for f examples and the parameter λ controls the trade-off between accuracy and robustness. Note that the distribution $p_{\text{adv}}\left(\boldsymbol{x}, y\right)$ is constructed by transforming natural samples $(\boldsymbol{x}, y) \sim p_{\text{data}}\left(\boldsymbol{x}, y\right)$ with $y \neq y_{\text{adv}}$, so that adversarial example $\boldsymbol{x}_{\text{adv}} = \mathcal{A}_f\left(\boldsymbol{x}; y_{\text{adv}}\right)$ is classified by f as the attack's target y_{adv}.

The first loss in eq. (4.2), e.g. NLL, fits the model's predictive distribution to the data distribution. The second term measures the probabilistic distance between the regular and adversarial images' distribution and constrains the classifier, so its adversarial examples are indistinguishable from the regular inputs. It is important to note that we minimize a probabilistic distance between joint distributions because the distance between marginal distributions $p_{\text{data}}(\boldsymbol{x})$ and $p_{\text{adv}}(\boldsymbol{x})$ is trivially minimized when $\boldsymbol{r} \sim 0$. Compared with adversarial training, the proposed formulation does not impose the assumption that adversarial noise is label non-changing. To the contrary, we require that adversarial noise for the robust classifier should be visually confusing and, thus, it should change the underlying label of the input. In the section, we describe the implementation details of the proposed defense.

4.4 Robust Learning with Adversary Critic

As we have argued in the previous section, adversarial examples for the robust classifier should be indistinguishable from the regular data of the adversarial target. Minimizing the statistical distance between $p_{\text{data}}(\boldsymbol{x}, y)$ and $p_{\text{adv}}(\boldsymbol{x}, y)$ in eq. (4.2) requires probability density estimation which in itself is a difficult problem. Instead, we adopt the framework of Generative Adversarial Networks [122]. We rely on a discriminator, or *an adversary critic*, to estimate a measure of difference between two distributions. The discriminator given an input-label pair $(\boldsymbol{x}, y)$ classifies it as either natural or adversarial. For the k-class classifier f, we implement the adversary critic as a k-output neural network (see fig. 4.3). The objective for the k-th output of the discriminator D is to correctly distinguish between natural and adversarial examples of the class y_k:

$$\mathcal{L}(f^*, D_k) = \min_{D_k} \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x}|y_k)} \left[\log D_k(\boldsymbol{x})\right] + \mathbb{E}_{y:y\neq y_k} \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x}|y)} \left[\log\left(1 - D_k\left(\mathcal{A}_{f^*}(\boldsymbol{x}; y_k)\right)\right)\right] \tag{4.3}$$

where $\mathcal{A}_f(\boldsymbol{x}, y_k)$ is the targeted adversarial attack on the classifier f which transforms the input $\boldsymbol{x}$ to the adversarial target y_k. An example of such attack is Projected Gradient Descent [8] which iteratively takes a step in the direction of the target y_k. Note that the second term in eq. (4.3) is computed by transforming the regular inputs $(\boldsymbol{x}, y) \sim p_{\text{data}}(\boldsymbol{x}, y)$ with the original label y different from the adversarial target y_k.

Our architecture for the discriminator in Figure 4.3 is slightly different from the previous work on joint distribution matching [128] where the label information was added as the input to each layer of the discriminator. We use class label only in the final classification layer of the discriminator. In the experiments, we observe that with the proposed architecture: 1) the discriminator is more stable during training; 2) the classifier f converges faster and is more robust. We also regularize the adversary critic with a gradient norm penalty [130]. For the gradient norm penalty, we do not interpolate between clean and adversarial images. Instead, we compute the penalty at the real and adversarial data separately. Interestingly, regularizing the gradient of the binary classifier has the interpretation of maximizing the geometric margin [55].

The objective for the classifier f is to minimize the number of mistakes subject to that its adversarial examples generated by the attack $\mathcal{A}_f$ fool the adversary critic D:

$$\mathcal{L}(f, D^*) = \min_f \mathbb{E}_{\boldsymbol{x},y\sim p_{\text{data}}(x,y)}\mathcal{L}(f(\boldsymbol{x}), y) + \lambda \sum_{y_k} \mathbb{E}_{y:y\neq y_k}\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x}|y)}\left[\log D_k^*\left(\mathcal{A}_f(\boldsymbol{x}; y_k)\right)\right] \tag{4.4}$$

where $\mathcal{L}$ is a standard supervised loss such as negative log-likelihood (NLL) and the parameter λ controls the trade-off between test accuracy and classifier robustness. To improve stability of the adversarial mapping during training, we introduce adversarial cycle-consistency constraint which ensures that adversarial mapping $\mathcal{A}_f$ of the adversarial examples should be close to the original:

$$\mathcal{L}_{\text{cycle}}(y_s, y_t) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x}|y_s)}\left[\|\mathcal{A}_f(\mathcal{A}_f(\boldsymbol{x}, y_t), y_s) - \boldsymbol{x}\|_2\right] \;\; \forall y_s \neq y_t \tag{4.5}$$

where y_s is the original label of the input and y_t is the adversarial target. Adversarial cycle-consistency constraint is similar to cycle-consistency constraint which was introduced for image-to-image translation [131]. But, we introduce it to constraint the adversarial mapping $\mathcal{A}_f$ and it improves the robustness of the classifier f. Next, we discuss implementation of our targeted adversarial attack $\mathcal{A}_f$.

Our defense requires that the adversarial attack $\mathcal{A}_f$ is differentiable. Additionally,

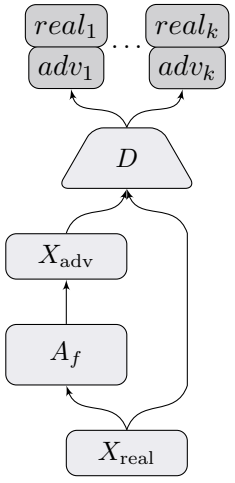


FIGURE 4.3: A diagram of a multiclass adversary critic, which is trained to distinguish regular and adversarial examples.

Algorithm 1 High-Confidence Attack $\mathcal{A}_f$

Input: Image $\boldsymbol{x}$, target y, network f, confidence C.
Output: Adversarial image $\hat{\boldsymbol{x}}$.
$\hat{\boldsymbol{x}} \leftarrow \boldsymbol{x}$
while $p_y(\hat{x}) < C$ **do**
  $f \leftarrow \log C - \log p_y(\hat{\boldsymbol{x}})$
  $\boldsymbol{w} \leftarrow \nabla \log p_y(\hat{\boldsymbol{x}})$
  $\boldsymbol{r} \leftarrow \frac{f}{\|\boldsymbol{w}\|_2^2}\boldsymbol{w}$
  $\hat{\boldsymbol{x}} \leftarrow \hat{\boldsymbol{x}} + \boldsymbol{r}$
end while

adversarial examples generated by the attack $\mathcal{A}_f$ should be misclassified by the network f with high confidence. Adversarial examples which are close to the decision boundary are likely to retain some identifying characteristics of the original class. An attack which optimizes for the mistakes, e.g. DeepFool [53], guarantees the confidence of $\frac{1}{k}$ for k-way classifier. To generate high-confidence adversarial examples, we propose a novel adversarial attack which iteratively maximizes the confidence of the adversarial target. The confidence of the target k after adding perturbation $\boldsymbol{r}$ is $p_k(\boldsymbol{x}+\boldsymbol{r})$. The goal of the attack is to find the perturbation, so the adversarial input is misclassified as k with the confidence at least C:

$$\begin{aligned} &\min \|\boldsymbol{r}\| \\ &\text{s. t. } p_k(\boldsymbol{x}+\boldsymbol{r}) \geq C \end{aligned} \tag{4.6}$$

We apply a first-order approximation to the constraint inequality:

$$\begin{aligned} &\min \|\boldsymbol{r}\| \\ &\text{s. t. } p_k(\boldsymbol{x}) + \boldsymbol{r}^T \nabla_{\boldsymbol{x}} p_k(\boldsymbol{x}) \geq C \end{aligned} \tag{4.7}$$

Softmax in the final classification layer saturates quickly and shatters the gradient. To avoid small gradients, we use log-likelihood instead. Finally, the l_2-norm minimal perturbation can be computed using a method of Lagrange multipliers as follows:

$$\boldsymbol{r}_k = \frac{\log C - \log p_k(\boldsymbol{x})}{\|\nabla_{\boldsymbol{x}} \log p_k(\boldsymbol{x})\|_2} \tag{4.8}$$

Because we use the approximation of the non-convex decision boundary, we iteratively update perturbation $\boldsymbol{r}$ for $N_{\max}$ steps using eq. (4.8) until the adversarial input $\boldsymbol{x}_{\text{adv}}$ is misclassified as the target k with the confidence C. Our attack can be equivalently written as $\boldsymbol{x}_{\text{adv}} = \boldsymbol{x} + \prod_{i=1}^{N_{\max}} I(p(\boldsymbol{x} + \sum_{j=1}^{i} \boldsymbol{r}_j) \leq C)\boldsymbol{r}_i$ where I is an indicator function. The discrete stopping condition introduces a non-differentiable path in the computational graph. We replace the gradient of the indicator function I with sigmoid-adjusted straight-through estimator during backpropagation [132]. This is a biased estimator but it has low variance and performs well in the experiments.

The proposed attack is similar to Basic Iterative Method (BIM) [8]. BIM takes a fixed ϵ-norm step in the direction of the attack target while our method uses an adaptive step $\gamma = \frac{|\log C - \log p_y(\hat{\boldsymbol{x}})|}{\|\nabla_x \log p_y(\hat{\boldsymbol{x}})\|}$. The difference is important for our defense:

1. BIM introduces an additional parameter ϵ. If ϵ is too large, then the attack will not be accurate. If ϵ is too small, then the attack will require many iterations to converge.

2. Both attacks are differentiable. However, for BIM attack during backpropagation, all the gradients $\frac{\partial \boldsymbol{r}_i}{\partial \boldsymbol{w}}$ have an equal weight ϵ. For our attack, the gradients will be weighted adaptively depending on the distance γ to the attack's target. The step γ for our attack is also fully-differentiable.

A full listing of our attack algorithm is shown in algorithm 1. Next, we discuss how to select the adversarial target y_t and the attack's target confidence C during training.

The classifier f approximately characterizes a conditional distribution $p\,(y|\boldsymbol{x})$. If the classifier f^* is optimal and robust, its adversarial examples generated by the attack $\mathcal{A}_f$ should fool the adversary critic D. Therefore, the attack $\mathcal{A}_f$ to fool the critic D should generate adversarial examples with the confidence C equal to the confidence of the classifier f on the regular examples. During training, we maintain a running mean of the confidence score for each class on the regular data. The attack target y_t for the input $\boldsymbol{x}$ with the label y_s can be sampled from the masked uniform distribution. Alternatively, the class with the closest decision boundary [53] can be selected. The latter formulation resulted in a more robust classifier f. We attacked the class with the closest decision boundary in all our experiments. This is similar to support vector machine formulation, which maximizes the minimum margin.

Finally, we train the classifier f and the adversary critic D jointly using stochastic gradient descent by alternating minimization of Equations (4.3) and (4.4). Our formulation has three components (the classifier f, the critic D, and the attack $\mathcal{A}_f$). In this sense, our formulation is similar to Triple-GAN [128], but the generator in our formulation also fools the classifier.

4.5 Experiments

Adversarial training [5] discards the dependency between the model parameters and the adversarial noise. In this work, it is necessary to retain the implicit dependency between the classifier f and the adversarial noise so that we can backpropagate through the adversarial attack $\mathcal{A}_f$. For these reasons, all experiments were conducted using Tensorflow [133], which supports symbolic differentiation and computation on GPU. The source code to reproduce the experiments in this section is available online at `https://github.com/aam-at/adversary_critic`. Backpropagation through our attack requires second-order gradients $\frac{\partial^2 f(\boldsymbol{x};\boldsymbol{w})}{\partial x \partial w}$, which increases the computational complexity of our defense. At the same time, this allows the model to anticipate the changes in the adversary and, as we show, significantly improves the model robustness both numerically and perceptually.

We performed experiments on MNIST dataset. While MNIST is a simple classification task, it remains unsolved in the context of robust learning. We evaluated the robustness of the models against l_2-norm attacks. Minimal adversarial perturbation $\boldsymbol{r}$ was estimated using DeepFool [53], Carlini and Wagner [18], and the proposed attack. To improve the accuracy of DeepFool and our attack during testing, we clipped the l_2-norm of perturbation at each iteration to 0.1. Note that our attack with the fixed step is equivalent to Basic Iterative Method [8]. We set the maximum number of iterations for DeepFool and our attack to 500. The target confidence C for our attack was set to the prediction confidence on the original input $\boldsymbol{x}$. DeepFool and our attack do not handle domain constraints explicitly, so we projected the perturbation after each update. For Carlini and Wagner [18], we used the implementation provided by the authors with default settings for the attack, but we reduced the number of optimization iterations from 10000 to 1000 to reduce the computational complexity of the attack. As suggested in [53], we measured the robustness of the model as follows:

$$\rho_{\text{adv}}(\mathcal{A}_f) = \frac{1}{|\mathcal{D}|} \sum_{\mathbf{x} \in \mathcal{D}} \frac{\|\mathbf{r}(\mathbf{x})\|_2}{\|\mathbf{x}\|_2} \tag{4.9}$$

where $\mathcal{A}_f$ is the attack on the classifier f and $\mathcal{D}$ is the test set.

We compared our defense with reference (no defense), Adversarial Training [5, 120] ($\epsilon = 0.1$), Virtual Adversarial Training (VAT) [91] ($\epsilon = 2.0$), and l_2-norm

Margin Maximization [55] ($\lambda = 0.1$) defense. We studied the robustness of two networks with rectified activation: 1) a fully-connected neural network with three hidden layers of size 1200 units each; 2) Lenet-5 convolutional neural network. We train both networks using Adam optimizer [29] with batch size 100 for 100 epochs. Next, we describe the training details for our defense.

Our critic network had two layers with 1200 units each and leaky rectified activation. We also added Gaussian noise to the input of each layer. We trained both the classifier and the critic using Adam [29] with the momentum $\beta_1 = 0.5$. The starting learning rate was set to $5 \cdot 10^{-4}$ and 10^{-3} for the classifier and the discriminator respectively. We trained our defense for 100 epochs. The learning rate was halved every 40 epochs. We set $\lambda = 0.5$ for fully-connected network and $\lambda = 0.1$ for Lenet-5 network, which we selected using validation dataset. Both networks were trained with $\lambda_{\text{rec}} = 10^{-2}$ for the adversarial cycle-consistency loss and $\lambda_{\text{grad}} = 10.0$ for the gradient norm penalty. The number of iterations for our attack $\mathcal{A}_f$ was set to 5. The attack confidence C was set to the running mean classifier's confidence on natural images. We pre-trained the classifier f for 1 epoch without any regularization to get the initial estimate of the class confidence scores C.

Defense	%	[53]	[18]	Our
No defense	1.46	0.131	0.124	0.173
[5]	0.90	0.228	0.210	0.299
[91]	0.84	0.244	0.215	0.355
[55]	0.84	0.262	0.230	0.453
Our	1.18	**0.290**	**0.272**	**0.575**

(A)

Defense	%	[53]	[18]	Our
No defense	0.64	0.157	0.148	0.207
[5]	0.55	0.215	0.191	0.286
[91]	0.60	0.225	0.195	0.330
[55]	0.54	0.248	0.225	0.470
Our	0.93	**0.288**	**0.278**	**0.590**

(B)

TABLE 4.1: A comparison between various defenses for fully-connected and Lenet-5 convolutional networks on MNIST dataset. We report the results on MNIST dataset for fully-connected network in table 4.1a and for Lenet-5 convolutional network in table 4.1b. Column 1 shows the test error on original images. Column 3-5 reports the robustness ρ under DeepFool [53], Carlini and Wagner [18], and the proposed attack.

Our results for 10 independent runs are summarized in Table 4.1, where the second column shows the test error on the clean images. The subsequent columns report the robustness ρ to DeepFool [53], Carlini and Wagner [18], and our attacks. Our defense significantly increased the robustness of the model to adversarial examples. Some adversarial images for the neural network trained with our defense are shown in Figure 4.4. Adversarial examples were generated using Carlini and Wagner [18] attack with default parameters. As we can observe, adversarial examples at the decision boundary in Figure 4.4b are visually confusing. At the same time, high-confidence adversarial examples in Figure 4.4c closely resemble natural images of the adversarial target. To further investigate this, we compared various defenses based on how many of its adversarial "mistakes" are actual mistakes that confuse humans.

(A) (B) (C)

FIGURE 4.4: Adversarial examples at different levels of the attack's target confidence C. Figure 4.4a shows a random subset of test images (average confidence 97%). Figure 4.4b shows adversarial examples at the class decision boundary (average confidence 34%). Figure 4.4c shows high-confidence adversarial images (average confidence 98%).

We conducted the experiment with human annotators on Amazon Mechanical Turk (MTurk). We asked the workers to label adversarial examples. The interface for the task is shown in Figure 4.5. Adversarial examples were generated from the test set using the proposed attack. The attack's target was set to class closest to the decision boundary. The target confidence was set to the model's confidence on the original examples. We split 10000 test images into 400 assignments. Each assignment was completed by one unique annotator. On average, 50-100 unique annotators performed the task depending on the availability of the workers. We report the results for four defense methods in Table 4.2. For the model trained without any defense, adversarial noise does not change the label of the input at all.

When the model is trained with our defense, the high-confidence adversarial noise changes the input's label.

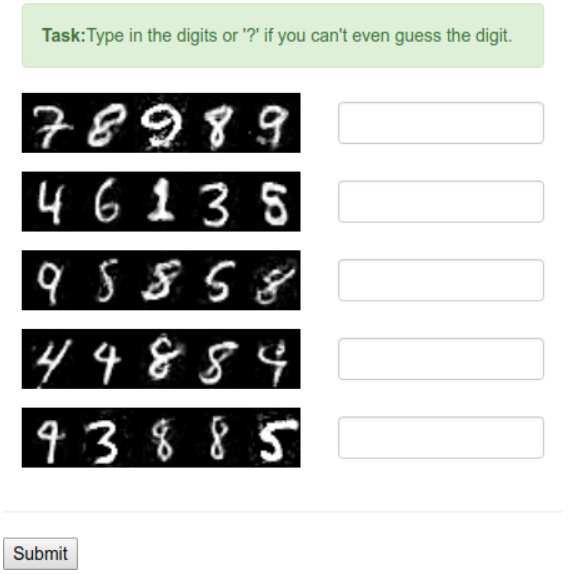


FIGURE 4.5: An interface for the experiment with human annotators on Amazon Mechanical Turk. Each human annotator was asked to label 25 images, overall 400 tasks.

4.6 Conclusion

In this chapter, we introduce a novel approach for learning a robust classifier. Our defense is based on the intuition that adversarial examples for the robust classifier should be indistinguishable from the regular data of the adversarial target. We formulate the problem of learning a robust classifier in the framework of Generative Adversarial Networks. Unlike prior work based on robust optimization, our method does not put any prior constraints on adversarial noise. Our method surpasses in terms of robustness networks trained with adversarial training. In experiments with human annotators, we also show that adversarial examples for our defense are indeed visually confusing. In the future work, we plan to scale our defense to more complex datasets and apply it to the classification tasks in other domains, such as audio or text.

Defense	% Change	% No change
No defense	0.57	98.74
[5]	19.02	77.21
[91]	35.08	59.68
[55]	60.47	34.52
Our	87.99	9.86

(A)

Defense	% Change	% No change
No defense	2.54	96.53
[5]	19.1	75.94
[91]	26.8	67.73
[55]	81.77	13.15
Our	92.29	6.51

(B)

TABLE 4.2: A comparison between various defenses for fully-connected network and Lenet-5 convolutional network on the experiment with human annotators. We report the results of the experiment for fully-connected network in table 4.2a and for Lenet-5 convolutional network in fig. 4.4c. Column 2 shows the percent of adversarial images which human annotator label with its adversarial target. This means that adversarial noise changed the "true" label of the input. Column 3 shows the percent of the adversarial images that human annotator label with its original label, so adversarial noise did not change the underlying label of the input.

Chapter 5

Primal-Dual Proximal Gradient Descent Adversarial Attack

State-of-the-art deep neural networks are sensitive to small input perturbations. Since the discovery of this intriguing vulnerability, a myriad of defense methods has been proposed that attempt to improve robustness to adversarial noise. To assess the progress in the area of robust training of DNNs, it is important to understand when existing defenses fail. However, evaluating adversarial robustness has proven to be an extremely challenging task. Existing adversarial attacks require thousands of iterations and multiple restarts to find the minimal adversarial perturbation. On the other hand, fast attacks ignore the norm minimization penalty and solve a simpler perturbation-bounded problem. In this work, we introduce a principled adversarial attack that directly solves the original non-convex constrained minimization problem. We interpret optimizing the Lagrangian of the adversarial attack as a two-player game: the first player minimizes the Lagrangian wrt the adversarial noise; the second player maximizes the Lagrangian wrt the regularisation penalty. Our attack algorithm simultaneously optimizes primal and dual variables to find the minimal adversarial perturbation. For non-smooth l_p-norm minimization, such as l_∞-, l_1-, and l_0-norms, we introduce primal-dual proximal gradient descent attack. Experimental results show that our attack method is significantly stronger than current state-of-the-art attacks on MNIST, CIFAR-10, and Restricted ImageNet datasets against unregularised and adversarially trained models. We hope that our attack will be considered as a benchmark for a future comparison between different defense methods.

5.1 Introduction

The goal of the adversarial attack is to find a minimal l_p-norm perturbation that changes the model prediction. Solving this non-convex constrained minimization problem has proven to be extremely challenging [68]. Existing methods [6, 18, 75] in place of the original constrained problem solve a sequence of unconstrained minimization problems for multiple values of the regularisation weight λ, which is selected using linear or bisection search. The attack optimization search needs to be restarted for each value of λ. This greatly increases the computational cost of the attack and makes it infeasible for attacking large models. Fast methods, such as FGM [5] and PGD [8, 17] attacks, reformulate the original norm minimization problem with non-convex misclassification constraint as a surrogate loss minimization with convex l_p-norm perturbation constraint. For this "simpler" problem, projected gradient descent attack (PGD) is an optimal first-order adversary [17]. However, PGD attack does not explicitly minimize the l_p-norm of the perturbation. Instead, PGD attack computes model accuracy at the threshold ϵ. PGD attack needs to be restarted several times to evaluate the robust accuracy at multiple thresholds.

This work introduces a principled adversarial attack that directly solves the original non-convex constrained minimization problem. We interpret optimizing the Lagrangian of the adversarial attack as playing a two-player game. The first player minimizes the Lagrangian wrt the adversarial noise, while the second player maximizes the Lagrangian wrt the regularisation penalty, which penalizes the first player for the violation of the misclassification constraint. Then, we apply a primal-dual gradient descent algorithm to simultaneously update primal and dual variables to find the minimal adversarial perturbation. In addition, for non-smooth l_p-norm minimization, such as l_∞-, l_1-, and l_0-norms, we introduce primal-dual proximal gradient descent attack. Existing formulation of l_0-norm proximal operator minimizes the number of non-zero elements in the vector. We derive group $l_{0,G}$-norm proximal operator, which we can be used to minimize the number of non-zero pixels. Experimental results show that our attack method is significantly stronger than current state-of-the-art attacks on MNIST, CIFAR-10, and Restricted ImageNet datasets against unregularised and adversarially trained models. We hope that our attack will be considered as a benchmark for a future comparison between various defense methods.

5.2 Primal-Dual Gradient Descent Attack

As we extensively discussed in Section 2.4, the problem of finding minimal adversarial perturbation is challenging to solve due to the non-convexity of DNNs and non-differentiability of the misclassification constraint. Existing methods in place of the original problem solve an unconstrained problem in Equation (2.14) for multiple values of λ selected using linear or bisection search as described in [6, 18]. The attack optimization search needs to be restarted for every value of λ. This greatly increases the computational cost of the attack and makes it infeasible for evaluating the robustness of large models. Other methods, such as FGM [5], PGD [8, 17] and DAA [74] attacks, reformulate the original norm minimization problem with non-convex misclassification constraint as non-convex surrogate loss minimization with convex l_p-norm perturbation constraint as in Equation (2.15). For this "simpler" problem, projected gradient descent attack (PGD) is an optimal first-order adversary [17]. In this work, we propose a novel algorithm that solves the original non-convex constrained l_p-norm minimization problem directly and efficiently.

Let us revisit the Lagrangian of the original non-convex constrained l_p-norm minimization problem in Equation (2.13):

$$\mathcal{L}(\mathbf{r}, \lambda) = \|\mathbf{r}\| + \lambda \mathrm{I}\left[\hat{k}\left(\mathbf{x} + \mathbf{r}\right) \neq y\right] \tag{5.1}$$

where λ is a dual variable that controls the penalty weight for the misclassification constraint; I is an indicator function. For brevity, we omit the domain constraints $\mathbf{x} + \mathbf{r} \in \mathrm{C}$ as they are easy to enforce for typical inputs, e.g. $[0, 1]$ box-projection Π_C for natural images. As $\lambda \to \infty$, the solution of the problem in Equation (5.1) converges to the feasible solution of the original non-convex constrained norm minimization problem.

Optimizing the Lagrangian in Equation (5.1) can be interpreted as playing a two-player game: the $\mathbf{r}$-player seeks the Lagrangian wrt primal variable $\mathbf{r}$; the λ-player wishes to maximize the Lagrangian wrt dual variable λ. The dual variable λ penalizes the $\mathbf{r}$-player for the violation of the misclassification constraint. C&W attack [18] uses a bisection search to find an optimal value of the dual variable λ^*. For a fixed λ, each iteration of the bisection search requires full optimization of the Lagrangian to find an optimal value of the primal variable $\mathbf{r}^*$ which significantly increases the running time of the attack.

We propose a primal-dual gradient algorithm to optimize primal and dual variables simultaneously. Unfortunately, we cannot calculate the constraint gradients in order to optimize the Lagrangian $\mathcal{L}(\mathbf{r}, \lambda)$ using the first-order algorithm because the misclassification constraint is non-differentiable. In line with the previous research, we express the error constraint in terms of the prediction margin. We define the prediction margin $m_y(\mathbf{x})$ for the input $\mathbf{x}$ with label y as follows:

$$m_y(\mathbf{x}) = f(\mathbf{x})_y - \max_{i \neq y} f(\mathbf{x})_i \tag{5.2}$$

Then, the input $\mathbf{x}$ with the label y is misclassified if and only if $m_y(\mathbf{x}) \leq 0$. As our surrogate loss of 0/1-indicator function, we adopt shifted hinge loss $\max(0, \delta - m_y(\mathbf{x})$ where δ is the target margin. To generate minimal adversarial perturbation, we set the margin δ to 0 because minimal adversarial example should be close to the decision boundary.

Using our surrogate loss for the misclassification constraint, we introduce two proxy-Lagrangians for the $\mathbf{r}$-player and for the λ-player as follows:

$$\begin{aligned} \mathcal{L}_{\mathbf{r}}(\mathbf{r}, \lambda) &= \lambda_1 \|\mathbf{r}\| + \lambda_2 \max\left(0, -m_y\left(\mathbf{x} + \mathbf{r}\right)\right) \\ \mathcal{L}_{\lambda}(\mathbf{r}, \lambda) &= \lambda_2 \mathrm{I}\left[-m_y\left(\mathbf{x} + \mathbf{r}\right)\right] \end{aligned} \tag{5.3}$$

where $(\lambda_1, \lambda_2) \in \Lambda$ and $\Lambda \subseteq R_+^2$ is 2-dimensional simplex. In the formulation above, we represent $\lambda \in \mathrm{R}_+$ as point in 2-dimensional simplex $\lambda = {}^{\lambda_2}/_{\lambda_1}$. Notice that only the $\mathbf{r}$-player uses surrogate hinge loss in place of the misclassification constraint, while the λ-player uses original non-differentiable misclassification constrain. The λ-player chooses how much the $\mathbf{r}$-player should penalize surrogate misclassification constraint, but does so in a way as to satisfy original non-differentiable constraint.

The proxy-Lagrangians formulation in Equation (5.3) avoids the issue of the non-differentiable error constraint. The $\mathbf{r}$-player wishes to find parameters $\mathbf{r}$ to minimise $\mathcal{L}_{\mathbf{r}}(\mathbf{r}, \lambda)$, while the λ-player wants to maximise $\mathcal{L}(\mathbf{r}, \lambda)$. Unfortunately, the proxy-Lagrangian formulation corresponds to a non-zero-sum game because two players optimise now different functions. Fortunately, the proxy-Lagrangian formulation admits a weaker type of equilibrium, Φ-correlated equilibrium (see [134] for the details).

Next, we describe our primal-dual gradient descent attack (PDGD) algorithm in Algorithm 2. The $\mathbf{r}$-player minimizes the external regret, while the λ-player minimizes

the swap regret. We perform gradient descent on primal variables using Adam optimiser [29], which we found produces the smallest perturbation. For the λ-player, we perform gradient ascent in the log domain, which is equivalent to a multiplicative weight update algorithm. Intuitively, if at an iteration k, the misclassification constraint is not satisfied, we need to increase the penalty λ_2 for the $\mathbf{r}$-player. On the other hand, if the constraint is satisfied, we can reduce the penalty weight λ_2. Finally, we record and store the best perturbation found during the optimization process in Lines 16 to 19.

Algorithm 2 Primal-Dual Gradient Descent

Require: Image $\mathbf{x}$, label y, initial perturbation $\mathbf{r}^{(0)}$, the total number of iterations T.

Ensure: Adversarial perturbation $\mathbf{r}$.

1: $\mathbf{r} \leftarrow \mathbf{0}$
2: **for** $k \leftarrow 1$ to T **do**
3: Let $\nabla_{\mathbf{r}}^{(k)}$ be a gradient of $\mathcal{L}_{\mathbf{r}}(\mathbf{r}^{(k)}, \lambda^{(k)})$
4: Let $\nabla_{\lambda}^{(k)}$ be a gradient of $\mathcal{L}_{\lambda}(\mathbf{r}^{(k)}, \lambda^{(k)})$
5: Update $\mathbf{r}^{(k+1)} = \Pi_C \left(\mathbf{r}^{(k)} - \theta_{\mathbf{r}} \nabla_{\mathbf{r}}^{(k)} \right)$
6: Update $\lambda^{(k+1)} = \Pi_{\Lambda} \left(\lambda^{(k)} + \theta_{\lambda} \nabla_{\lambda}^{(k)} \right)$
7: **if** $\hat{k}(\mathbf{x} + \mathbf{r}) \neq y$ **and** $\left\| \mathbf{r}^{(k+1)} \right\| \leq \|\mathbf{r}\|$ **then**
8: $\mathbf{r} \leftarrow \mathbf{r}^{(k+1)}$
9: **end if**
10: **end for**

Our primal-dual gradient descent algorithm has several shortcomings. First, we use gradient descent to minimize the external regret of the $\mathbf{r}$-player. Gradient descent for smooth convex functions has convergence rate of $\mathcal{O}\left(1/k\right)$ where k is the number of gradient iterations. For the non-smooth functions, e.g. l_1-norm, the convergence rate of subgradient descent is $\mathcal{O}\left(1/\sqrt{k}\right)$ which is considerably slower than sub-linear convergence of gradient descent for smooth functions. In addition, our attack cannot be used to minimize l_0-quasinorm because we cannot compute the gradients. In the next section, we introduce a proximal formulation of the Algorithm 2 that has a faster convergence rate for l_∞- and l_1-norm minimization. This proximal attack can also be used to minimize l_0-norm of the perturbation.

5.3 Primal-Dual Proximal Gradient Descent

In the previous section, we introduced a primal-dual gradient descent attack (PDGD) attack. PDGD can be used to minimize any smooth, differentiable function, such as l_2-norm. However, PDGD has several limitations: 1) a slow convergence rate of $\mathcal{O}\left(1/\sqrt{k}\right)$ for non-smooth functions, e.g. l_∞- and l_1-norms; 2) requires gradient, so it cannot be used for the minimization of non-differentiable l_0-quasinorm. In this section, we derive a proximal formulation of PDGD attack that addresses the above shortcomings. Our primal-dual proximal gradient attack (PDPGD) can be used for the direct minimization of any norm or function for which the proximal operator can be computed easily, including but not limited l_∞-, l_1-, and l_2-norms, and l_0-quasinorm. In this section, we also derive a novel l_0-proximal operator for multichannel images, which measures the total number of the modified pixels.

First, we review some basics of proximal algorithms before introducing our attack. A detailed overview of proximal algorithms can be found in [135]. We define the proximal operator of the scaled function λf where $\lambda > 0$ as follows:

$$\text{prox}_{\lambda f}(x) = \arg\min_{\mathbf{u}} \left(f(u) + \frac{1}{2\lambda}\|u - x\|_2^2 \right) \tag{5.4}$$

The following useful relation holds true for any proximal operator of the proper closed function f:

$$\text{prox}_{\lambda f}(x) + \lambda \text{prox}_{\lambda^{-1} f^*}(x/\lambda) = x \tag{5.5}$$

where f^* is the convex conjugate of f. The equation above is known as Moreau decomposition. Moreau decomposition is useful for deriving proximal operators of l_p-norm functions. In particular, it implies that for any general norm $\|\cdot\|$:

$$\text{prox}_{\lambda\|\cdot\|}(x) + \lambda \Pi_{\mathcal{B}}(x/\lambda) = x \tag{5.6}$$

where Π is the projection operator.

Let us revisit the problem of minimizing the $\mathbf{r}$-player proxy-Lagrangian:

$$\mathcal{L}_{\mathbf{r}}(\mathbf{r}, \lambda) = \lambda_1 \|\mathbf{r}\| + \lambda_2 \max\left(0, -m_y\left(\mathbf{x} + \mathbf{r}\right)\right) \tag{5.7}$$

The goal of the r-player is to maximize the proxy-Lagrangian function. However, if the norm $\|\cdot\|$ is non-smooth, first-order subgradient descent needs $\mathcal{O}(1/\epsilon^2)$ iterations to find ϵ-error local minimum. Moreover, for l_0-norm minimization, we cannot use first-order algorithms because the gradient of l_0-norm is $\emptyset$ almost everywhere. The above limitations can be addressed in the framework of the proximal algorithms.

We can write the proxy-Lagrangian of the **r**-player as follows:

$$\mathcal{L}_{\mathbf{r}}(\mathbf{r}, \lambda) = \lambda\|r\| + g(\mathbf{x} + \mathbf{r}) \tag{5.8}$$

where $\lambda = \lambda_1/\lambda_2$ and g is a differentiable surrogate loss of the original non-differentiable misclassification constraint, e.g. hinge loss function. Consider the quadratic approximation of the **r**-player proxy-Lagrangian at step k where we ignore not necessarily differentiable l_p-norm penalty:

$$\begin{aligned}\mathbf{r}^{(k+1)} =& \arg\min_{\mathbf{u}} \|\mathbf{u}\| + g(\mathbf{r}^{(k)}) + \nabla_{\mathbf{r}} g(\mathbf{r}^{(k)})^T(\mathbf{u} - \mathbf{r}^{(k)}) \\ &+ \frac{1}{2t}\left\|\mathbf{u} - \mathbf{r}^{(k)}\right\|_2^2 \\ =& \arg\min_{\mathbf{u}} \|\mathbf{u}\| + \frac{1}{2t}\left\|\mathbf{u} - \left(\mathbf{r}^{(k)} - t\nabla_{\mathbf{r}} g(\mathbf{r}^{(k)})\right)\right\|_2^2\end{aligned} \tag{5.9}$$

which is by the definition of the proximal operator is equivalent to:

$$\mathbf{r}^{(k+1)} = \text{prox}_{\lambda\|\|,t}\left(\mathbf{r}^{(k)} - t\nabla_{\mathbf{r}} g(\mathbf{r}^{(k)})\right) \tag{5.10}$$

where t is the step size. The above method is known as proximal gradient descent, and it consists of a gradient update followed by a proximal operator.

Our complete primal-dual proximal gradient (PDPGD) attack is listed in Algorithm 3. PDPGD attack can be used to minimize any function with a closed-form proximal operator. In this work, we limit our discussion to the minimization of l_p-norm perturbations. Next, we derive and list proximal operators for l_∞-, l_2-, l_1-, and l_0-norms.

Algorithm 3 Primal-Dual Proximal Gradient Descent

Require: Image $\mathbf{x}$, label y, initial perturbation $\mathbf{r}^{(0)}$, the total number of iterations T.

Ensure: Adversarial perturbation $\mathbf{r}$.

$\mathbf{r} \leftarrow \mathbf{0}$
for $k \leftarrow 1$ to T **do**
  Let $\nabla_{\mathbf{r}}^{(k)}$ be a gradient of $g(\mathbf{x} + \mathbf{r}^{(k)})$ where g is the surrogate loss of the misclassification constraint
  Let $\nabla_{\lambda}^{(k)}$ be a gradient of $\mathcal{L}_{\lambda}(\mathbf{r}^{(k)}, \lambda^{(k)})$
  Update $\mathbf{r}^{(k+1)} = \Pi_C \left(\text{prox}_{\lambda \| \cdot \|, \theta_{\mathbf{r}}} \left(\mathbf{r}^{(k)} - \theta_{\mathbf{r}} \nabla_{\mathbf{r}}^{(k)} \right) \right)$
  Update $\lambda^{(k+1)} = \text{proj}_{\Lambda} \left(\lambda^{(k)} + \theta_{\lambda} \nabla_{\lambda}^{(k)} \right)$
  if $\hat{k}(\mathbf{x} + \mathbf{r}) \neq y$ **and** $\left\| \mathbf{r}^{(k+1)} \right\| \leq \|\mathbf{r}\|$ **then**
    $\mathbf{r} \leftarrow \mathbf{r}^{(k+1)}$
  end if
end for

5.3.1 l_∞-proximal operator

l_∞-norm of the vector $\mathbf{x}$ returns the largest absolute element of the vector $\mathbf{x}$: $l_\infty(\mathbf{x}) = \max |\mathbf{x}|$. Using Moreau decomposition in Equation (5.6), we can easily show that:

$$\boxed{\text{prox}_{\lambda \| \cdot \|_\infty}(x) = x - \lambda \text{proj}_{\{\| \cdot \|_1 \leq 1\}}(x/\lambda)} \tag{5.11}$$

where proj is a projection operator. l_1-norm projection can be computed efficiently in $\mathcal{O}(n \log n)$ time [78].

5.3.2 l_2-proximal operator

Using Moreau decomposition in Equation (5.6), we can show that:

$$\boxed{\text{prox}_{\lambda \| \cdot \|_2}(x) = (1 - \lambda / \|x\|_2)_+ x} \tag{5.12}$$

This operator is known as a block soft thresholding operator.

5.3.3 l_1-proximal operator

l_1-norm proximal operator is well-known in signal processing [76]. It is defined as follows:

$$\boxed{\text{prox}_{\lambda\|\cdot\|_1}(x) = \mathcal{T}_\lambda(x)} \tag{5.13}$$

where $\mathcal{T}_\lambda(x) = \text{sign}(x)(x-\lambda)_+$ is a soft-thresholding operator.

5.3.4 l_0-proximal operator

l_0-norm is non-convex quasinorm. It measures the cardinality of the vector x. The proximal operator of l_0-norm is defined as follows:

$$\boxed{\text{prox}_{\lambda\|\cdot\|_0}(x) = \mathcal{H}_{\sqrt{2\lambda}}(x)} \tag{5.14}$$

where $\mathcal{H}_\lambda(x) = \mathrm{I}\left[x-\lambda\right]x$ is a hard-thresholding operator.

The proximal operator in eq. (5.14) minimizes the total number of non-zero elements in the vector x. However, for the multichannel images, the attack's goal is to minimize the number of non-zero pixels. We define group $l_{0,G}$-norm of the vector x for the groups $\mathcal{G} = (g_1, g_2, \ldots, g_G)$ as the number of groups for which at least one element of the group is non-zero:

$$\|x\|_{0,G} = \sum_{i=1}^{G} \mathrm{I}\left[\max|x|_{g_i} > 0\right] \tag{5.15}$$

For RGB images, the group partition $\mathcal{G}$ naturally corresponds to the pixels in the image. Then, we can derive the proximal operator of $l_{0,G}$-quasinorm as follows:

$$\boxed{\text{prox}_{\lambda\|\cdot\|_0}(x) = \mathcal{H}^G_{\sqrt{2\lambda}}(x)} \tag{5.16}$$

where $\mathcal{H}^G_\lambda(x) = \mathrm{I}\left[\max x_g - \lambda\right]x$ is a hard-thresholding group operator which sets all elements in the group to 0 if the maximal element in the group is less than the threshold λ.

5.3.5 Other proximal operators

Above, we presented proximity operators for standard l_p-norms. Our algorithm can minimize any function for which the proximal operator is available in closed-form. For example, our algorithm can minimize total variation regularizer, since TV proximal operator can be efficiently computed [136]. Another advantage of the proximal framework is that it is possible to minimize a convex combination of multiple functions with closed-form proximal operators using the properties of the composition of proximal operators [137].

5.4 Experiments

Models: To study the effectiveness of our attack method, we compared our attack to state-of-the-art attacks on MNIST, CIFAR-10 and Restricted ImageNet datasets [138]. For each dataset, we considered a naturally trained model (*plain*) and l_∞ (l_∞-AT) and l_2 (l_2-AT) adversarially trained models as in [17]. *Plain* and l_∞-AT models on MNIST can be found at `https://github.com/MadryLab/mnist_challenge`, while the l_2-AT was trained to be robust to l_2-norm perturbations. For CIFAR-10 dataset, we used models available at `https://github.com/fra31/fab-attack` (architecture with 8 convolutional layers and 2 dense layers), while on Restricted ImageNet dataset we used models from [138] which can be downloaded from `https://github.com/MadryLab/robust-features-code`.

The models on MNIST achieved the following clean accuracy on the test dataset (first 1000 images of the test set): *plain* 99.17% (98.7%), l_∞-AT 98.53% (98.5%), and l_2-AT 98.95% (98.7%). The models on CIFAR-10 achieve the following clean test accuracy (first 1000 images of the test set): *plain* 88.38% (89.4%), l_∞-AT 79.9% (80.4%), and l_2-AT 80.44% (80.7%).

Attacks: We tested the robustness of each model wrt to l_∞-, l_2-, l_1-, and l_0-norm adversaries. We compared the performance of our attacks to attacks representing state-of-the-art for each norm: Brendel & Bethge attack (B&B, l_∞-, l_2-, l_1-, l_0-norms) [139]; Carlini-Wagner l_2-attack (C&W, l_2-norm) [18]; Decoupled Direction and Norm l_2-attack (DDN, l_2-norm) [72]; DeepFool (DF, l_∞-, and l_2-norm) [53]; Distributionally Adversarial Attack (DAA, l_∞-norm) [74]; Elastic-net

attack (l_1-norm) [75]; Fast Adaptive Boundary Attack (FAB, l_∞-, l_2-, l_1-norms) [54]; Jacobian-based Saliency Map attack (JSMA, l_0-norm) [52]; One Pixel attack (Pixel, l_0-norm) [51]; Projected Gradient on the cross-entropy loss (PGD, l_∞-, l_2-, and l_1-norms) [8, 17, 77]; Sparsefool (SF, l_1-norm) [69]. We use DDN and B&B attacks from [140], Pixel attack from [141], C&W, EAD, JSMA, and PGD attacks as in [142], SF, DF, and FAB with the code from the original papers, while we reimplemented DAA attack. We implemented our attack and conducted all experiments in Tensorflow [133] [1]. Next, we describe the choice of attack specific parameters.

Attack parameters:

- Brendel & Bethge attack [139] as in [140] with the following parameters for all l_p-norms: 1000 iterations, initial learning rate 1.0, learning rate decay every 20 steps on MNIST and every 100 steps on CIFAR-10 datasets.
- Carlini-Wagner l_2-attack (C&W) [18] as in [142] with the following parameters: 9 binary search steps, 10000 iterations, learning rate 0.01, initial const 0.01, and no early stopping.
- Decoupled Direction and Norm l_2-attack (DDN) [72] as in [140] with the following parameters: 1000 iterations, initial epsilon 1.0, and gamma 0.05.
- l_2- and l_∞ DeepFool (DF) [53] as in `https://github.com/LTS4/DeepFool` with the following parameters for all norms: 50 iterations and overshoot 0.02.
- We reimplemented Distributionally Adversarial Attack (DAA) [74]. We used Lagrangian Blob Attack algorithm. We set the number of iterations to 200. For each ϵ, dataset and model, we selected the optimal epsilon step from $[\epsilon, \epsilon/2, \epsilon/5, \epsilon/10, \epsilon/25, \epsilon/50, \epsilon/100]$. For each threshold ϵ, we reported the lowest robust accuracy.
- Elastic-net attack (EAD) [75] as in [142] with the following parameters: 9 binary search steps, 1000 iterations, learning rate 0.01, initial const 0.01, beta 0.05, l_1 decision rule, and no early stopping.
- Fast Adaptive Boundary (FAB) [54] as in `https://github.com/fra31/fab-attack` with 100 iterations. The remaining parameters were set to the values recommended by the authors.

[1] Code to reproduce the experiments is available at `https://github.com/aam-at/pdpgd`.

- Jacobian-based Saliency Map attack (JSMA) [52] as in [142] with the following parameters: theta in [-1.0, 1.0] (allow to add and remove features), gamma 1.0 (allow to perturb all features). JSMA requires selecting the target. We attacked all targets and reported the minimal adversarial perturbation (in total $2 \times (k-1)$ JSMA attacks for each example).

- One Pixel attack (Pixel) [51] as in [141] with the following parameters: differential evolution strategy, 400 population size, and 100 iterations.

- Projected Gradient on the cross-entropy loss (PGD) [8, 17] as in [142]. We set the number of iterations to 100. For each ϵ, dataset and model, we selected the optimal epsilon step from $[\epsilon, \epsilon/2, \epsilon/5, \epsilon/10, \epsilon/25, \epsilon/50, \epsilon/100]$. For each threshold ϵ, we reported the lowest robust accuracy. For l_1-norm, we use Sparse Projected Gradient [77] with sparsity levels of 1% on MNIST and CIFAR-10, and 10% on Restricted ImageNet.

- We used SparseFool attack (SF) [69] as in `https://github.com/LTS4/SparseFool` with the following parameters: 20 iterations, epsilon 0.02, and lambda set to 1.0.

For our attack, we set the number of iterations to 500, so the computational cost of robust evaluation with our attack is similar to the cost of evaluation using Projected Gradient Descent (PGD) [17] attack with 100 iterations at 5 thresholds. We used PDGD with Adam optimizer [29] to minimise the l_2-norm perturbation. We used PDPGD attack to minimise l_∞-, l_1-, and l_0-norm perturbation. The learning rate for the primal variables was selected from [0.01, 0.05, 0.1, 0.5, 1.0] to minimize the adversarial perturbation. During 500 iterations of our attack, we linearly decreased the learning rate to 10^{-2} of its initial value. We performed gradient ascent on dual variables in the log domain. The learning rate for dual variables was set to 0.1 for all experiments. We applied exponential moving average to dual variables to smooth the value of the regularisation weight. We set the initial value of the regularisation weight λ (dual variable) to 0.01.

Evaluation metrics: We define the robust accuracy of the model at the threshold ϵ as the classification accuracy of the model if the adversary is allowed to perturb the input with the perturbation of l_p-norm smaller than the ϵ in order to change the model prediction. A strong attack is expected to reduce the model accuracy more significantly compared to weaker attacks. We fixed five thresholds per model

and per dataset to calculate the robust accuracy for each attack (we selected the thresholds as in [54]). In addition, we compare the strength of the attacks by measuring the average norm of the perturbation if the adversary is allowed to perturb the input (perturb only correctly classified inputs). The strong attack should produce smaller adversarial perturbations on average compared to weaker attacks. Please note that the comparison using the average norm of the adversarial perturbation is only available for the attacks that minimize the perturbation norm, e.g. C&W and EAD, and excludes attacks that solve the perturbation-constrained attack problem in Equation (2.15), e.g. PGD and DAA attacks.

In this section, we present the comparison of the attack methods on MNIST and CIFAR-10 datasets in Section 5.4.1 and Section 5.4.2 respectively. The results of the experiments on Restricted ImageNet dataset can be found in the full version of the paper.

5.4.1 MNIST

We evaluated the attacks on the first 1000 images of the test set. We reported the robust accuracy for l_∞-, l_2-, l_1-, and l_0-norm attacks at five different thresholds for *plain*, l_∞-AT, and l_2-AT models. For our attack with random restarts, we sampled the initial perturbation from a uniform distribution $\mathcal{U} = [-0.5, 0.5]$. The results for a naturally trained model and an l_∞- and l_2 adversarially trained models are shown in Table 5.2, Table 5.3, and Table 5.4 respectively. We reported the average l_p-norm of the adversarial perturbation found by the attacks (when successful, excluding already misclassified points) for every model in Table 5.1. All attacks, except DF and SF, succeeded in generating adversarial perturbation. l_2-DF and SparseFool attacks fail for l_∞-AT model, likely due to gradient masking, (success rate of 76% and 82% respectively).

Our primal-dual gradient descent attack with 100 restarts (PDGD-100) is the best l_2-norm attack. The improvement over the second-best attack, FAB [54], is particularly significant for l_∞-AT model [17]. Our primal-dual proximal gradient descent attack (PDPGD) is the best l_∞- and l_1-norm attack. Our l_1-norm attack can generate substantially smaller perturbations. For example, we reduce the average l_1-norm perturbation for l_∞-AT model from 3.45 down to 2.31. Our proximal attack is also the best attack for l_0-norm perturbation minimization except for MNIST

naturally trained for which our attack is a close second after B&B attack [139]. We show examples of adversarial perturbations generated using our attack for 10 randomly selected test images in Figure 5.1.

TABLE 5.1: A comparison of various attacks based on the estimated robustness of the models on MNIST dataset. We reported the mean l_p-norm of the adversarial perturbations found by the attacks (when successful, excluding the already misclassified points) for every model.

$l_\infty \times 10^{-1}$			DF	B&B	FAB	PDPGD
plain			0.82	0.68	0.66	**0.63**
l_∞-at			5.28	3.31	3.26	**3.15**
l_2-at			2.59	1.80	1.70	**1.66**
l_2	DF	C&W	DDN	B&B	FAB	PDGD
plain	1.13	1.01	1.00	1.03	0.99	**0.98**
l_∞-at	4.84	2.08	1.84	1.47	1.12	**1.07**
l_2-at	3.08	2.35	2.31	2.42	2.25	**2.19**
l_1	Sparsefool		EAD	B&B	FAB	PDPGD
plain	8.70		6.36	7.19	6.04	**5.91**
l_∞-at	200.56		7.85	13.96	3.45	**2.31**
l_2-at	16.46		12.19	15.22	12.16	**10.99**
l_0				JSMA	B&B	PDPGD
plain				13.75	9.09	**8.36**
l_∞-at				59.26	28.75	**4.07**
l_2-at				26.57	16.82	**13.59**

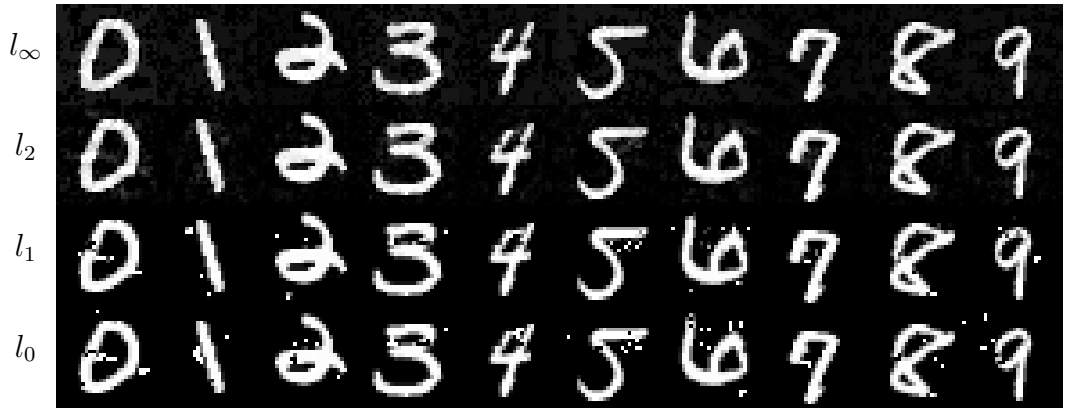
5.4.2 CIFAR10

We evaluated the attacks on the first 1000 images of the CIFAR-10 test set. We reported the robust accuracy for l_∞-, l_2-, l_1-, and l_0-norm attacks at five different thresholds for *plain*, l_∞-AT, and l_2-AT models on CIFAR-10 dataset in Table 3.2. For our attack with random restarts, we sampled the initial perturbation from a uniform distribution $\mathcal{U} = [-0.25, 0.25]$. The results for a naturally trained model and an l_∞- and l_2 adversarially trained models are shown in Table 5.6, Table 5.7, and Table 5.8 respectively. We reported the average l_p-norm of the adversarial perturbation found by the attacks (when successful, excluding already misclassified points) for every model in Table 5.5.

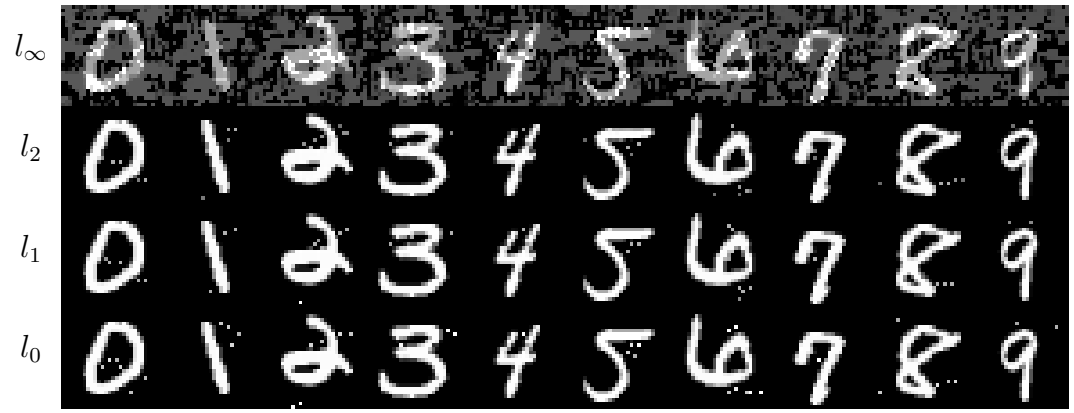
Our primal-dual gradient descent (PDGD) and proximal gradient descent (PDPGD) attacks are the strongest attacks across all models and all perturbation norms. The

FIGURE 5.1: l_∞-, l_2-, l_1-, and l_0-norm adversarial examples for a naturally trained model and an l_∞- and l_2- adversarially trained models on MNIST.

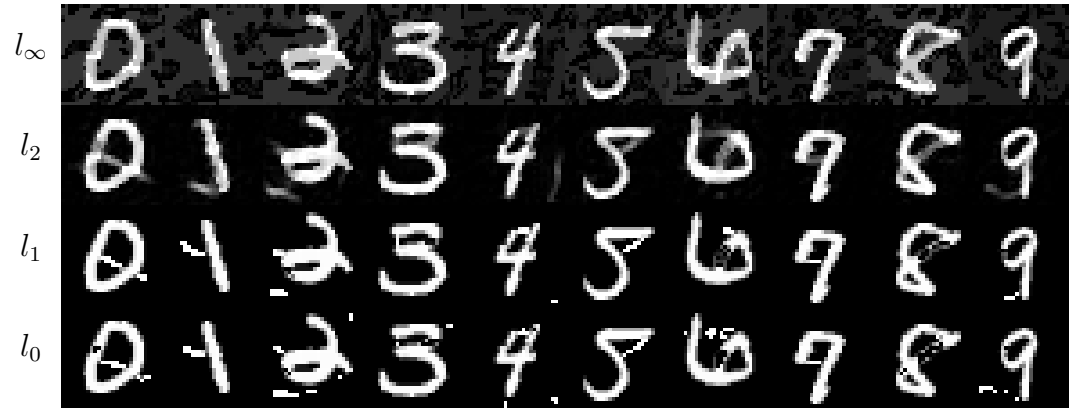
(A) Original images



(B) Adversarial examples for a naturally trained model



(C) Adversarial examples for an l_∞-AT model



(D) Adversarial examples for an l_2-AT model

improvements for l_0-norm minimization is particularly significant because our proximal attack minimizes the number of modified pixels while JSMA and B&B attacks minimize the number of modified values (see discussion in section 5.3.4).

5.5 Conclusion

Fast and accurate estimation of robust accuracy and average robustness of the models is critical for comparing various defense methods. Evaluating the robustness of DNNs has proved to be challenging. The original non-convex constrained norm minimization attacker's problem is difficult to solve. Existing attack methods either do not explicitly minimize the norm of the perturbation (PGD, DAA) or require thousands of iterations to converge (C&W, EAD). In this work, we introduce a principled adversarial attack that directly solves the original non-convex constrained minimization problem. We interpret optimizing the Lagrangian of the adversarial attack as playing a two-player game. The first player minimizes the Lagrangian wrt the adversarial noise, while the second player maximizes the Lagrangian wrt the regularisation penalty, which penalizes the first player for the violation of the misclassification constraint. Then, we apply a primal-dual gradient descent algorithm to simultaneously update primal and dual variables in order to find the minimal adversarial perturbation. In addition, for non-smooth l_p-norm minimization, such as l_∞-, l_1-, and l_0-norms, we introduce primal-dual proximal gradient descent attack. We also derive group $l_{0,G}$-norm proximal operator, which we use to minimize the number of non-zero adversarial pixels. Our method is fast and accurate. In the experiments, we show that our method is substantially stronger than existing attacks for measuring l_p-norm robustness of deep neural networks.

TABLE 5.2: A comparison of l_∞-, l_2-, l_1, and l_0-norm attacks for a naturally trained model on MNIST.

metric	ϵ	DF	B&B	DAA-1	DAA-50	PGD-1	PGD-10	PGD-100	FAB-1	FAB-10	FAB-100	PDPGD-1	PDPGD-10	PDGD-100
	0.03	93.6	92.0	**91.9**	**91.9**	92.0	**91.9**	**91.9**	92.0	92.0	92.0	**91.9**	**91.9**	**91.9**
	0.05	84.4	77.4	79.2	76.8	76.3	75.3	74.8	77.2	76.8	76.2	75.9	73.4	**73.2**
l_∞	0.07	65.0	45.6	62.2	53.8	43.4	41.2	40.5	44.6	43.4	42.8	43.5	**37.7**	**37.7**
	0.09	38.9	18.3	46.4	35.7	16.0	14.0	12.8	16.2	15.0	14.3	16.5	10.7	**10.4**
	0.11	16.8	3.7	30.4	19.7	3.3	2.5	2.2	3.4	2.8	2.6	3.1	1.4	**1.3**

	ϵ	DF	C&W	DDN	B&B	PGD-1	PGD-10	PGD-100	FAB-1	FAB-10	FAB-100	PDGD-1	PDGD-10	PDGD-100
	0.5	92.1	91.1	91.1	91.1	91.1	91.1	91.1	91.1	91.1	91.1	90.9	**90.8**	**90.8**
	1	59.9	48.4	47.5	52.1	50.4	48.8	47.7	48.7	47.7	47.1	48.5	45.3	**45.2**
l_2	1.5	18.6	9.4	8.4	10.3	10.2	8.8	8.5	8.5	8.0	7.6	10.0	6.5	**6.1**
	2	2.0	0.9	0.8	0.8	1.2	1.0	0.8	0.7	0.6	0.6	1.0	0.4	**0.3**
	2.5	0.1	**0.0**	0.1	**0.0**	0.4	0.1	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**

	ϵ	SparseFool	EAD	B&B	PGD-1	PGD-10	PGD-100	FAB-1	FAB-10	FAB-100	PDPGD-1	PDPGD-10	PDPGD-100
	2		94.9	93.4	94.8	94.4	94.2	94.0	94.2	93.7	93.5	94.3	93.4	**93.3**
	4		86.7	76.8	83.4	81.0	78.0	76.7	80.3	76.2	75.2	80.7	75.9	**74.3**
l_1	6		69.9	47.7	60.1	58.7	50.4	48.0	53.2	47.5	43.3	56.2	44.8	**41.7**
	8		50.5	26.5	36.5	39.8	30.2	26.6	31.3	25.2	22.6	34.6	23.3	**21.1**
	10		33.1	11.1	19.3	25.2	16.2	12.7	15.1	10.2	8.4	18.5	9.3	**7.0**

	ϵ	JSMA	Pixel	B&B	PDPGD-1	PDPGD-10	PDPGD-100
	1	88.6	71.9	83.5	80.0	71.7	**67.3**
	3	82.6	42.7	71.5	58.6	41.6	**33.0**
l_0	5	75.7	36.7	58.0	38.4	23.3	**14.9**
	8	65.2	35.3	42.7	18.8	7.3	**3.9**
	12	53.3	33.7	25.3	6.8	1.7	**0.9**

TABLE 5.3: A comparison of l_∞-, l_2-, l_1, and l_0-norm attacks for an l_∞ adversarially trained model on MNIST.

metric	ϵ	DF	B&B	DAA-1	DAA-50	PGD-1	PGD-10	PGD-100	FAB-1	FAB-10	FAB-100	PDPGD-1	PDPGD-10	PDGD-100
l_∞	0.2	95.2	95.0	94.1	**93.7**	94.8	94.2	93.8	94.7	94.2	94.1	94.5	93.9	**93.7**
	0.25	94.7	93.6	92.4	91.2	93.2	91.8	91.4	93.3	91.9	91.6	92.8	91.4	**91.1**
	0.3	93.9	91.2	88.8	87.3	91.6	88.9	87.9	91.4	89.6	88.6	89.1	87.8	**87.2**
	0.325	91.9	73.3	67.3	60.5	79.1	71.9	66.9	86.3	83.4	80.9	65.6	58.1	**54.7**
	0.35	89.6	29.8	15.4	9.4	34.6	20.8	14.9	49.7	31.5	23.5	16.2	9.5	**7.2**

	ϵ	DF	C&W	DDN	B&B	PGD-1	PGD-10	PGD-100	FAB-1	FAB-10	FAB-100	PDGD-1	PDGD-10	PDGD-100
l_2	1	94.3	87.9	87.7	79.7	91.6	89.9	89.0	82.9	71.7	64.5	76.2	64.6	**60.3**
	1.5	93.3	73.7	69.2	44.3	86.9	81.3	76.9	45.1	20.7	11.8	31.7	12.8	**9.0**
	2	92.1	53.9	41.9	16.1	78.9	69.0	58.7	14.4	1.8	0.8	4.9	1.0	**0.3**
	2.5	90.0	32.1	15.4	4.1	67.7	47.2	31.1	2.6	0.1	**0.0**	0.2	**0.0**	**0.0**
	3	87.5	13.7	4.1	0.8	49.4	23.4	10.5	1.0	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**

	ϵ	SparseFool	EAD	B&B	PGD-1	PGD-10	PGD-100	FAB-1	FAB-10	FAB-100	PDPGD-1	PDPGD-10	PDPGD-100
l_1	2.5	94.4	91.8	89.1	93.7	93.4	93.0	84.1	74.2	58.1	65.7	45.6	**38.1**
	5	92.7	72.3	76.0	88.4	86.3	83.6	60.4	46.5	18.9	17.0	5.6	**3.4**
	7.5	92.3	45.7	61.1	82.5	76.6	72.2	42.9	32.5	5.6	4.4	0.4	**0.1**
	10	91.8	23.9	48.1	78.3	68.1	61.1	31.8	25.4	2.0	1.0	**0.0**	**0.0**
	12.5	91.4	12.4	39.2	71.9	57.9	45.3	24.0	20.1	0.7	**0.0**	**0.0**	**0.0**

	ϵ	JSMA	Pixel	B&B	PDPGD-1	PDPGD-10	PDPGD-100
l_0	1	98.5	92.3	96.3	96.0	91.8	**90.9**
	3	97.7	61.3	88.6	84.6	66.8	**53.6**
	5	97.1	31.8	80.4	68.3	36.3	**20.1**
	8	96.9	22.1	67.5	36.8	10.1	**4.2**
	12	96.2	18.4	52.7	12.2	1.5	**0.1**

TABLE 5.4: A comparison of l_2-, l_2-, l_1, and l_0-norm attacks for an l_2 adversarially trained model on MNIST.

metric	ϵ	DF	B&B	DAA-1	DAA-50	PGD-1	PGD-10	PGD-100	FAB-1	FAB-10	FAB-100	PDPGD-1	PDPGD-10	PDGD-100
	0.05	96.7	96.4	96.4	**96.3**	**96.3**	**96.3**	**96.3**	96.4	**96.3**	**96.3**	96.4	96.4	96.4
	0.1	93.8	91.7	90.7	90.2	90.6	90.5	90.2	90.8	90.4	90.4	90.0	89.7	**89.6**
l_∞	0.15	86.9	76.9	73.3	71.3	74.2	72.5	71.6	74.0	72.2	71.8	72.8	70.5	**69.7**
	0.2	76.3	37.2	31.2	24.6	31.9	26.7	23.8	33.7	27.6	24.4	25.1	20.7	**19.3**
	0.25	59.8	3.1	0.9	0.7	1.7	1.0	0.7	2.0	0.9	0.7	1.8	0.8	**0.5**

	ϵ	DF	C&W	DDN	B&B	PGD-1	PGD-10	PGD-100	FAB-1	FAB-10	FAB-100	PDGD-1	PDGD-10	PDGD-100
	1	93.6	92.3	92.4	93.2	92.5	92.5	92.4	92.3	92.3	**92.2**	92.3	**92.2**	**92.2**
	1.5	87.3	84.5	83.7	86.0	84.1	83.6	83.6	84.2	83.7	83.7	84.5	83.6	**83.0**
l_2	2	79.3	70.6	68.5	72.7	69.0	68.1	67.7	70.2	69.1	68.1	70.1	67.7	**66.6**
	2.5	67.6	46.2	44.4	50.7	44.9	40.8	38.5	46.4	41.7	39.5	42.8	37.6	**35.4**
	3	54.0	17.6	16.6	20.3	16.0	11.6	10.3	19.0	13.2	11.7	14.2	10.2	**8.8**

	ϵ	SparseFool	EAD	B&B	PGD-1	PGD-10	PGD-100	FAB-1	FAB-10	FAB-100	PDPGD-1	PDPGD-10	PDPGD-100
	5		91.6	89.9	93.0	90.7	90.3	90.2	90.5	90.2	90.1	90.3	89.1	**88.7**
	8.75		80.2	71.6	81.4	75.6	73.5	72.5	75.2	73.7	72.5	74.1	69.2	**66.9**
l_1	12.5		65.4	45.3	64.3	60.6	56.4	54.1	55.2	49.2	45.5	49.7	40.2	**33.9**
	16.25		48.8	20.5	39.1	46.5	39.0	32.6	33.0	24.3	20.4	24.1	15.3	**13.0**
	20		31.5	8.4	20.6	37.4	24.6	15.8	15.3	9.5	7.4	10.6	5.4	**3.7**

	ϵ	JSMA	Pixel	B&B	PDPGD-1	PDPGD-10	PDPGD-100
	5	96.3	97.3	94.4	94.7	92.2	**90.8**
	10	87.9	96.6	77.3	82.0	73.0	**67.6**
l_0	15	80.4	96.6	51.1	64.0	46.6	**33.8**
	20	64.1	96.2	23.5	39.7	20.0	**12.7**
	25	49.9	95.8	10.1	21.3	7.8	**4.3**

TABLE 5.5: A comparison of various attacks based on the estimated robustness of the models on CIFAR-10 dataset. We reported the mean l_p-norm of the adversarial perturbations found by the attacks (when successful, excluding the already misclassified points) for every model.

$l_\infty \times 10^{-2}$			DF	B&B	FAB	PDPGD
plain			0.75	0.58	0.57	**0.54**
l_∞-at			2.98	2.59	2.37	**2.32**
l_2-at			2.42	2.1	1.94	**1.89**
$l_2 \times 10^{-1}$	DF	C&W	DDN	B&B	FAB	PDGD
plain	2.66	2.13	2.12	2.18	2.06	**2.02**
l_∞-at	9.08	7.28	7.66	7.77	7.16	**6.96**
l_2-at	8.96	7.12	7.49	7.60	6.99	**6.82**
l_1	Sparsefool		EAD	B&B	FAB	PDPGD
plain	6.72		3.00	3.79	2.87	**2.63**
l_∞-at	10.54		5.75	7.38	6.03	**5.22**
l_2-at	13.76		7.96	10.05	8.05	**7.35**
l_0				JSMA	B&B	PDPGD
plain				18.37	9.80	**3.37**
l_∞-at				25.03	11.02	**4.73**
l_2-at				26.14	11.58	**5.64**

TABLE 5.6: A comparison of l_∞-, l_2-, l_1, and l_0-norm attacks for a naturally trained model on CIFAR-10.

metric	ϵ	DF	B&B	DAA-1	DAA-50	PGD-1	PGD-10	PGD-100	FAB-1	FAB-10	FAB-100	PDPGD-1	PDPGD-10	PDGD-100
l_∞	$1/255$	62.4	57.3	63.5	62.2	56.2	55.8	**55.6**	56.5	56.4	56.4	56.3	56.2	55.7
	$1.5/255$	49.6	39.8	61.3	58.5	39.0	38.3	37.7	38.7	38.6	38.4	37.3	36.4	**36.1**
	$2/255$	37.5	24.4	59.0	54.8	24.3	23.2	22.5	22.8	22.7	22.7	22.2	21.6	**20.7**
	$2.5/255$	27.7	14.3	57.3	52.0	15.6	14.4	14.0	12.9	12.8	12.7	13.1	11.3	**10.9**
	$3/255$	20.5	7.1	55.5	47.7	10.4	8.8	8.4	6.7	6.6	6.5	6.5	4.8	**4.4**

	ϵ	DF	C&W	DDN	B&B	PGD-1	PGD-10	PGD-100	FAB-1	FAB-10	FAB-100	PDGD-1	PDGD-10	PDGD-100
l_2	0.1	71.8	68.8	68.2	69.1	67.9	67.6	**67.5**	68.2	68.0	67.8	**67.5**	**67.5**	**67.5**
	0.15	62.1	56.3	55.3	57.4	55.6	55.3	55.0	55.9	55.1	54.9	54.3	53.9	**53.6**
	0.2	52.3	44.1	43.9	46.0	43.8	43.1	43.0	43.6	42.9	42.5	42.5	41.9	**41.4**
	0.3	35.2	22.1	21.6	24.8	25.4	23.6	23.4	21.5	20.3	19.9	20.6	19.8	**19.5**
	0.4	21.5	9.3	9.8	13.2	17.6	15.8	15.0	9.9	8.4	8.2	8.1	8.0	**7.3**

	ϵ	SparseFool	EAD	B&B	PGD-1	PGD-10	PGD-100	FAB-1	FAB-10	FAB-100	PDPGD-1	PDPGD-10	PDPGD-100
l_1	2	70.1	54.6	60.1	55.1	53.7	53.0	53.6	52.4	50.2	54.4	50.4	**48.8**
	4	55.1	24.4	35.1	28.2	26.2	25.3	29.5	24.8	22.6	25.0	21.8	**20.3**
	6	42.1	9.1	18.2	16.4	14.1	12.0	15.4	10.0	8.2	10.2	6.6	**5.6**
	8	31.3	2.8	7.7	11.0	7.8	6.7	7.5	3.6	2.5	3.9	1.8	**1.3**
	10	22.4	1.0	3.4	7.2	5.0	3.7	4.1	1.5	1.0	1.8	0.6	**0.4**

	ϵ	JSMA	Pixel	B&B	PDPGD-1	PDPGD-10	PDPGD-100
l_0	1	88.6	71.9	83.5	80.0	71.7	**67.3**
	3	82.6	42.7	71.5	58.6	41.6	**33.0**
	5	75.7	36.7	58.0	38.4	23.3	**14.9**
	8	65.2	35.3	42.7	18.8	7.3	**3.9**
	12	53.3	33.7	25.3	6.8	1.7	**0.9**

TABLE 5.7: A comparison of l_∞-, l_2-, l_1, and l_0-norm attacks for an l_∞ adversarially trained model on CIFAR-10.

metric	ϵ	DF	B&B	DAA-1	DAA-50	PGD-1	PGD-10	PGD-100	FAB-1	FAB-10	FAB-100	PDPGD-1	PDPGD-10	PDGD-100
l_∞	$2/255$	66.9	66.2	66.4	66.1	**65.5**	**65.5**	**65.5**	65.8	65.8	65.7	65.6	65.6	65.6
	$4/255$	53.2	50.2	60.9	58.5	49.7	49.1	49.0	49.2	49.2	48.9	49.2	49.1	**48.8**
	$6/255$	43.3	36.9	58.4	54.9	37.4	36.5	36.1	35.3	34.8	34.7	34.8	34.7	**34.3**
	$8/255$	33.4	27.0	56.5	52.3	29.1	28.2	27.7	23.9	23.5	23.3	23.4	23.0	**22.6**
	$10/255$	25.2	18.3	54.6	50.1	22.9	21.4	20.5	15.4	14.7	14.2	14.3	**13.6**	**13.6**

	ϵ	DF	C&W	DDN	B&B	PGD-1	PGD-10	PGD-100	FAB-1	FAB-10	FAB-100	PDGD-1	PDGD-10	PDGD-100
l_2	0.25	67.4	65.5	64.9	66.3	**64.7**	**64.7**	**64.7**	65.0	**64.7**	**64.7**	64.8	**64.7**	**64.7**
	0.5	53.7	49.1	49.4	52.1	49.8	49.2	49.0	49.3	49.1	49.0	49.0	48.7	**48.6**
	0.75	42.8	33.7	34.4	38.5	38.7	37.9	37.4	34.2	33.4	33.3	33.3	**32.7**	**32.7**
	1	32.8	22.1	23.9	27.9	34.5	33.4	32.8	21.6	21.1	20.9	20.5	20.1	**19.8**
	1.25	24.4	11.7	15.0	18.8	33.5	32.2	31.0	12.2	11.5	11.3	10.7	10.2	**10.0**

	ϵ	SparseFool	EAD	B&B	PGD-1	PGD-10	PGD-100	FAB-1	FAB-10	FAB-100	PDPGD-1	PDPGD-10	PDPGD-100
l_1	5	53.4	36.5	43.9	46.8	44.8	43.3	43.4	40.4	38.9	39.6	35.2	**33.1**
	8.75	38.2	18.6	27.1	31.0	28.5	27.3	25.9	22.5	20.5	21.4	16.8	**14.5**
	12.5	27.3	7.4	15.3	22.2	19.4	17.2	14.1	10.6	9.1	8.5	6.3	**5.7**
	16.25	19.2	2.8	7.6	16.3	13.2	12.6	7.2	4.2	3.7	4.0	2.3	**1.9**
	20	12.7	0.9	4.3	13.2	11.4	10.4	4.1	1.8	1.6	1.3	0.5	**0.4**

	ϵ	JSMA	Pixel	B&B	PDPGD-1	PDPGD-10	PDPGD-100
l_0	1	80.3	68.6	76.6	73.5	69.2	**66.0**
	3	75.3	49.5	65.8	56.2	45.9	**40.9**
	5	68.7	46.9	52.8	44.1	32.5	**24.6**
	8	60.1	46.0	40.5	29.3	16.9	**11.8**
	12	51.6	46.7	28.2	14.9	7.0	**4.4**

TABLE 5.8: A comparison of l_2-, l_2-, l_1, and l_0-norm attacks for an l_2 adversarially trained model on CIFAR-10.

metric	ϵ	DF	B&B	DAA-1	DAA-50	PGD-1	PGD-10	PGD-100	FAB-1	FAB-10	FAB-100	PDPGD-1	PDPGD-10	PDGD-100
l_∞	$^2/_{255}$	64.2	63.5	65.7	64.7	62.6	62.6	**62.4**	62.7	62.6	62.6	62.7	62.7	62.7
	$^4/_{255}$	49.2	46.3	62.6	60.6	45.1	44.8	44.7	44.4	44.2	44.2	44.2	**44.1**	**44.1**
	$^6/_{255}$	36.9	29.4	60.2	56.6	32.1	30.8	30.6	27.2	26.8	26.7	26.7	26.2	**26.0**
	$^8/_{255}$	26.5	17.7	57.6	53.5	23.9	22.5	21.8	15.1	14.1	13.7	13.8	13.0	**12.7**
	$^{10}/_{255}$	19.2	10.5	55.4	51.2	18.3	16.4	15.3	8.6	8.1	7.9	7.3	6.6	**6.5**

	ϵ	DF	C&W	DDN	B&B	PGD-1	PGD-10	PGD-100	FAB-1	FAB-10	FAB-100	PDGD-1	PDGD-10	PDGD-100
l_2	0.25	66.6	65.8	**65.3**	66.8	65.4	65.4	**65.3**	65.4	65.4	**65.3**	65.5	65.4	**65.3**
	0.5	54.5	49.1	49.2	51.6	50.1	49.7	49.6	49.5	49.2	49.0	48.7	**48.5**	**48.5**
	0.75	42.2	33.0	33.1	36.6	38.2	37.7	37.1	33.2	32.5	32.4	32.1	31.9	**31.8**
	1	30.3	20.1	22.6	25.4	35.2	33.8	32.4	20.6	20.0	19.8	19.4	19.1	**18.7**
	1.25	22.5	11.0	14.4	17.1	33.4	31.9	30.8	11.4	11.1	10.6	9.9	9.7	**9.6**

	ϵ	SparseFool	EAD	B&B	PGD-1	PGD-10	PGD-100	FAB-1	FAB-10	FAB-100	PDPGD-1	PDPGD-10	PDPGD-100
l_1	3	67.3	62.6	64.8	65.1	64.9	64.6	63.5	63.3	62.9	63.5	62.4	**61.7**
	6	58.3	45.5	51.1	50.5	49.4	48.4	49.1	46.8	45.4	47.8	43.4	**41.5**
	9	47.7	28.0	36.8	37.6	35.6	34.6	33.9	30.4	28.9	31.8	27.1	**24.8**
	12	37.7	18.1	27.0	30.7	28.5	27.4	24.1	19.6	17.6	20.7	16.3	**14.8**
	15	30.9	10.1	18.2	26.9	23.9	22.0	15.8	12.2	10.9	13.1	9.7	**8.3**

	ϵ	JSMA	Pixel	B&B	PDPGD-1	PDPGD-10	PDPGD-100
l_0	1	79.6	71.2	76.4	74.6	70.5	**68.9**
	3	75.4	58.1	67.5	64.8	55.8	**49.7**
	5	71.4	55.2	58.4	52.3	39.3	**32.7**
	10	59.7	55.5	35.5	29.2	16.2	**10.1**
	15	49.7	54.5	20.3	12.2	5.1	**2.5**

Chapter 6

Conclusions

Deep neural networks have made tremendous progress in the area of image, language, and speech understanding. Yet, experiments with adversarial examples have shown that state-of-the-art deep neural networks are broken in one intriguing way: they are sensitive to small imperceptible perturbations in the input data. On the other hand, a human visual system is remarkably robust to changes in the object's appearance, shape, and pose. Computer vision systems to be trusted should be as accurate and reliable as the human visual system. Understanding the nature of adversarial examples and designing classifier's robust to adversarial noise is the main goal of this thesis.

The lack of deep neural networks' robustness to small, invisible perturbations is counter-intuitive, undesirable, and risky. If the model exceeds human-level performance on the task, how is it possible that the same model performs worse than random on the inputs that are to a human eye indistinguishable from the original correctly classified inputs? The existence of adversarial examples also puts into question the generalization ability of DNNs: do they really learn how to perform the task, or do they learn how to mimic the answers? Additionally, adversarial examples limit the use of DNNs in safety and security-critical applications where the adversary can potentially exploit the vulnerability to adversarial noise to gain access to restricted resources. Overall, if AI system is not robust, it is difficult to trust and rely on its predictions.

In this dissertation, we outlined the problem of adversarial examples and developed several partial solutions to it. In Chapter 3, we established a connection

between the classifier's margin and its robustness. We generalized a support vector machine (SVM) margin maximization objective to deep neural networks. We also proved that our formulation is equivalent to robust optimization. In Chapter 4, we proposed an ideal solution to the problem of adversarial examples: adversarial examples for the robust classifier should be indistinguishable from the regular data of the adversarial target. We formulated a problem of learning a robust classifier in the framework of generative adversarial networks (GAN) where an auxiliary network, or an adversary critic, is trained to distinguish regular and adversarial data. The classifier is trained to correctly classify the inputs subject to the regularization constraint that its adversarial examples fool the adversary critic. Our approach can be viewed as robust optimization with data-dependent uncertainty sets. Lastly, in Chapter 5, we proposed a norm constrained project gradient descent attack, which allows to estimate model robustness precisely. We hope that our attack will be used as a benchmark for a future comparison between various defenses.

The area of adversarial machine learning has made significant progress in reducing the vulnerability of deep neural networks to adversarial examples. However, the success of existing defenses is largely limited to small problems, e.g. MNIST or CIFAR-10 datasets. Even for the easy problem of classifying digits on MNIST, the best defenses still remain vulnerable to adversarial noise. Future works need to improve the scalability and efficiency of robustifying defenses. Additionally, existing adversarial attacks find a minimal norm additive adversarial perturbations and do not encompass other types of perturbations, e.g. multiple perturbations (a sum of several l_p-norms), translation or rotation of the image. Robustness to such unrestricted adversarial examples should be addressed in future work. We hope that the development of techniques in adversarial machine learning will improve the accuracy, robustness, and interpretability of machine learning systems.

Journal Articles

[1] **Alexander Matyasko** and Lap-Pui Chau. PDPGD: Primal-Dual Proximal Gradient Descent Adversarial Attack. In: under review in IEEE Transactions on Neural Networks and Learning Systems.

Conference Proceedings

[2] **Alexander Matyasko** and Lap-Pui Chau Improved Network Robustness with Adversary Critic. In: *Conference on Neural Information Processing Systems (NeurIPS)* 2018, Montreal, Canada, December 3-8, 2018. (Preliminary work was presented at *ICML 2017 Workshop on Implicit Models*)

[3] **Alexander Matyasko** and Lap-Pui Chau Margin Maximization for Robust Classification using Deep Learning. In: *International Joint Conference on Neural Networks (IJCNN)* 2017, Anchorage, USA, May 14-18, 2017.

[4] Jie Chen and **Alexander Matyasko** and Lap-Pui Chau A light field sparse representation structure and its fast coding technique. In: *19th International Conference on Digital Signal Processing (DSP)* 2014, Hong Kong, China, August 20-23, 2014.

Bibliography

[1] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1026–1034, 2015. 1

[2] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97, Nov 2012. ISSN 1558-0792. doi: 10.1109/MSP.2012.2205597. 1, 19

[3] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 3104–3112. Curran Associates, Inc., 2014. URL `http://papers.nips.cc/paper/5346-sequence-to-sequence-learning-with-neural-networks.pdf`. 1, 19

[4] C. Szegedy, Wei Liu, Yangqing Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9. IEEE, June 2015. ISBN 9781467369640. doi: 10.1109/cvpr.2015.7298594. URL `https://doi.org/10.1109/cvpr.2015.7298594`. 2, 18, 19

[5] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations (ICLR)*, 2015. 2, 3, 5, 18, 20, 23, 24, 25, 29, 32, 33, 38, 40, 41, 50, 51, 53, 55, 56, 60, 62, 63, 64, 70, 71, 74, 76, 77

[6] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. In *International Conference on Learning Representations (ICLR)*, 2014. URL `https://openreview.net/forum?id=kklr_MTHMRQjG`. 2, 19, 20, 23, 25, 29, 30, 38, 40, 41, 62, 76, 77

[7] Anh Nguyen, Jason Yosinski, and Jeff Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *2015 IEEE*

Conference on Computer Vision and Pattern Recognition (CVPR), pages 427–436. IEEE, June 2015. ISBN 9781467369640. doi: 10.1109/cvpr.2015.7298640. URL https://doi.org/10.1109/cvpr.2015.7298640. 2, 40

[8] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. Adversarial examples in the physical world. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Workshop Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=HJGU3Rodl. 2, 23, 25, 29, 31, 62, 66, 69, 70, 76, 77, 85, 86

[9] K. Eykholt, I. Evtimov, E. Fernandes, B. Li, A. Rahmati, C. Xiao, A. Prakash, T. Kohno, and D. Song. Robust physical-world attacks on deep learning visual classification. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1625–1634, June 2018. doi: 10.1109/CVPR.2018.00175. 2, 3, 19, 31

[10] Suranjana Samanta and Sameep Mehta. Towards Crafting Text Adversarial Samples. *arXiv e-prints*, art. arXiv:1707.02812, Jul 2017. 2

[11] Nicholas Carlini, Pratyush Mishra, Tavish Vaidya, Yuankai Zhang, Micah Sherr, Clay Shields, David Wagner, and Wenchao Zhou. Hidden voice commands. In *25th USENIX Security Symposium (USENIX Security 16)*, pages 513–530, Austin, TX, 2016. USENIX Association. ISBN 978-1-931971-32-4. URL https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/carlini. 2

[12] N. Carlini and D. Wagner. Audio adversarial examples: Targeted attacks on speech-to-text. In *2018 IEEE Security and Privacy Workshops (SPW)*, pages 1–7, 2018. 19

[13] Yao Qin, Nicholas Carlini, Ian Goodfellow, Garrison Cottrell, and Colin Raffel. Imperceptible, Robust, and Targeted Adversarial Examples for Automatic Speech Recognition. *arXiv e-prints*, art. arXiv:1903.10346, March 2019. 2

[14] I. Evtimov, K. Eykholt, E. Fernandes, T. Kohno, B. Li, A. Prakash, A. Rahmati, and D. Song. Robust Physical-World Attacks on Deep Learning Models. *ArXiv e-prints*, July 2017. 3, 19, 31

[15] Mahmood Sharif, Sruti Bhagavatula, Lujo Bauer, and Michael K. Reiter. Accessorize to a crime. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security - CCS'16*, CCS '16, pages 1528–1540, New York, NY, USA, 2016. ACM Press. ISBN 9781450341394. doi: 10.1145/2976749.2978392. URL https://doi.org/10.1145/2976749.2978392. 3, 19, 31

[16] D. Amodei, C. Olah, J. Steinhardt, P. Christiano, J. Schulman, and D. Mané. Concrete problems in AI safety. *ArXiv e-prints*, June 2016. 3

[17] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=rJzIBfZAb`. 3, 5, 20, 23, 25, 29, 32, 33, 62, 76, 77, 84, 85, 86, 87

[18] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 39–57. IEEE, May 2017. ISBN 9781509055333. doi: 10.1109/sp.2017.49. URL `https://doi.org/10.1109/sp.2017.49`. 5, 20, 23, 24, 29, 61, 62, 64, 70, 71, 72, 76, 77, 84, 85

[19] Tom M Mitchell et al. Machine learning. WCB, 1997. 8

[20] Nikola K. Kasabov. *Time-Space, Spiking Neural Networks and Brain-Inspired Artificial Intelligence (Springer Series on Bio- and Neurosystems)*. Springer Publishing Company, Incorporated, 1st edition, 2018. ISBN 3662577135. 10

[21] Lukas Paulun, Anne Wendt, and Nikola Kasabov. A retinotopic spiking neural network system for accurate recognition of moving objects using neucube and dynamic vision sensors. *Frontiers in computational neuroscience*, 12: 42–42, Jun 2018. ISSN 1662-5188. doi: 10.3389/fncom.2018.00042. URL `https://doi.org/10.3389/fncom.2018.00042`. PMC6006267[pmcid]. 10

[22] D. H. Hubel and T. N. Wiesel. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of Physiology*, 160(1):106–154, January 1962. ISSN 0022-3751. doi: 10.1113/jphysiol.1962.sp006837. URL `https://doi.org/10.1113/jphysiol.1962.sp006837`. 12

[23] Kunihiko Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36(4):193–202, April 1980. ISSN 0340-1200, 1432-0770. doi: 10.1007/bf00344251. URL `https://doi.org/10.1007/bf00344251`. 12

[24] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, November 1998. ISSN 0018-9219. doi: 10.1109/5.726791. URL `https://doi.org/10.1109/5.726791`. 12, 14, 38, 51

[25] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems*, 2(4):303–314, December 1989. ISSN 0932-4194, 1435-568X. doi: 10.1007/bf02551274. URL `https://doi.org/10.1007/bf02551274`. 14

[26] P. Werbos. *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*. PhD thesis, Harvard University, Cambridge, MA, 1974. 15

[27] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Parallel distributed processing: Explorations in the microstructure of cognition, vol. 1. chapter Learning Internal Representations by Error Propagation, pages 318–362. MIT Press, Cambridge, MA, USA, 1986. ISBN 0-262-68053-X. URL http://dl.acm.org/citation.cfm?id=104279.104293. 15

[28] Geoffrey Hinton, Nitish Srivastava, and Kevin Swersky. Neural networks for machine learning, lecture 6a: overview of mini-batch gradient descent. 15

[29] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1412.6980. 15, 24, 51, 71, 79, 86

[30] Sepp Hochreiter. Untersuchungen zu dynamischen neuronalen netzen. *Master's thesis, Institut fur Informatik, Technische Universitat, Munchen*, 1991. 15

[31] Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. Greedy layer-wise training of deep networks. In *Advances in Neural Information Processing Systems 19, Proceedings of the Twentieth Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 4-7, 2006*, pages 153–160, 2006. URL http://papers.nips.cc/paper/3048-greedy-layer-wise-training-of-deep-networks. 16

[32] Geoffrey E. Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554, July 2006. ISSN 0899-7667, 1530-888X. doi: 10.1162/neco.2006.18.7.1527. URL https://doi.org/10.1162/neco.2006.18.7.1527. 16

[33] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In Geoffrey Gordon, David Dunson, and Miroslav Dudík, editors, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pages 315–323, Fort Lauderdale, FL, USA, 11–13 Apr 2011. PMLR. URL http://proceedings.mlr.press/v15/glorot11a.html. 16

[34] Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv e-prints*, art. arXiv:1207.0580, Jul 2012. 16

[35] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778. IEEE, June 2016. ISBN 9781467388511. doi: 10.1109/cvpr.2016.90. URL https://doi.org/10.1109/cvpr.2016.90. 16, 19, 21

[36] Yoshua Bengio and Olivier Delalleau. On the expressive power of deep architectures. In Jyrki Kivinen, Csaba Szepesvári, Esko Ukkonen, and Thomas Zeugmann, editors, *Lecture Notes in Computer Science*, volume 6925 of *Lecture Notes in Computer Science*, pages 18–36. Springer Berlin Heidelberg, 2011. ISBN 9783642244117, 9783642244124. doi: 10.1007/978-3-642-24412-4_3. URL `https://doi.org/10.1007/978-3-642-24412-4_3`. 16

[37] Guido F. Montufar, Razvan Pascanu, Kyunghyun Cho, and Yoshua Bengio. On the number of linear regions of deep neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N.d. Lawrence, and K.q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2924–2932. Curran Associates, Inc., 2014. URL `http://papers.nips.cc/paper/5422-on-the-number-of-linear-regions-of-deep-neural-networks.pdf`. 16

[38] Bruno A. Olshausen and David J. Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381(6583):607–609, June 1996. ISSN 0028-0836, 1476-4687. doi: 10.1038/381607a0. URL `https://doi.org/10.1038/381607a0`. 16

[39] Dumitru Erhan, Yoshua Bengio, Aaron Courville, and Pascal Vincent. Visualizing higher-layer features of a deep network. Technical Report 1341, University of Montreal, June 2009. Also presented at the ICML 2009 Workshop on Learning Feature Hierarchies, Montréal, Canada. 16, 17

[40] K. Simonyan, A. Vedaldi, and A. Zisserman. Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps. *ArXiv e-prints*, December 2013. 17, 25

[41] Aravindh Mahendran and Andrea Vedaldi. Understanding deep image representations by inverting them. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5188–5196. IEEE, June 2015. ISBN 9781467369640. doi: 10.1109/cvpr.2015.7299155. URL `https://doi.org/10.1109/cvpr.2015.7299155`. 17

[42] Anh Nguyen, Alexey Dosovitskiy, Jason Yosinski, Thomas Brox, and Jeff Clune. Synthesizing the preferred inputs for neurons in neural networks via deep generator networks. In *Advances in Neural Information Processing Systems 29*. 2016. URL `https://arxiv.org/abs/1605.09304`. 17

[43] Matthew D. Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision – ECCV 2014*, pages 818–833. Springer International Publishing, Cham, 2014. ISBN 9783319105895, 9783319105901. doi: 10.1007/978-3-319-10590-1_53. URL `https://doi.org/10.1007/978-3-319-10590-1_53`. 17

[44] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin A. Riedmiller. Striving for simplicity: The all convolutional net. *CoRR*, abs/1412.6806, 2014. URL http://arxiv.org/abs/1412.6806. 17

[45] Julius Adebayo, Justin Gilmer, Michael Muelly, Ian Goodfellow, Moritz Hardt, and Been Kim. Sanity checks for saliency maps. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 9505–9515. Curran Associates, Inc., 2018. URL http://papers.nips.cc/paper/8160-sanity-checks-for-saliency-maps.pdf. 18

[46] Sara Hooker, Dumitru Erhan, Pieter-Jan Kindermans, and Been Kim. A benchmark for interpretability methods in deep neural networks. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 9737–9748. Curran Associates, Inc., 2019. URL http://papers.nips.cc/paper/9167-a-benchmark-for-interpretability-methods-in-deep-neural-networks.pdf. 18

[47] Nicholas Frosst and Geoffrey E. Hinton. Distilling a neural network into a soft decision tree. In Tarek R. Besold and Oliver Kutz, editors, *Proceedings of the First International Workshop on Comprehensibility and Explanation in AI and ML 2017 co-located with 16th International Conference of the Italian Association for Artificial Intelligence (AI*IA 2017), Bari, Italy, November 16th and 17th, 2017*, volume 2071 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2017. URL http://ceur-ws.org/Vol-2071/CExAIIA_2017_paper_3.pdf. 18

[48] Mike Wu, Michael C. Hughes, Sonali Parbhoo, Maurizio Zazzi, Volker Roth, and Finale Doshi-Velez. Beyond sparsity: Tree regularization of deep models for interpretability. In Sheila A. McIlraith and Kilian Q. Weinberger, editors, *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 1670–1678. AAAI Press, 2018. URL https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16285. 18

[49] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-CNN: Towards real-time object detection with region proposal networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 91–99. Curran Associates, Inc., 2015. URL http://papers.nips.cc/paper/5638-faster-r-cnn-towards-real-time-object-detection-with-region-proposal-netwo
pdf. 19

[50] Bin Liang, Hongcheng Li, Miaoqiang Su, Pan Bian, Xirong Li, and Wenchang Shi. Deep text classification can be fooled. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*,

pages 4208–4215. International Joint Conferences on Artificial Intelligence Organization, 7 2018. doi: 10.24963/ijcai.2018/585. URL `https://doi.org/10.24963/ijcai.2018/585`. 19

[51] J. Su, D. V. Vargas, and K. Sakurai. One pixel attack for fooling deep neural networks. *IEEE Transactions on Evolutionary Computation*, 23(5):828–841, 2019. 19, 29, 85, 86

[52] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z. Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. In *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 372–387. IEEE, March 2016. ISBN 9781509017515, 9781509017522. doi: 10.1109/eurosp.2016.36. URL `https://doi.org/10.1109/eurosp.2016.36`. 19, 29, 38, 62, 85, 86

[53] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. DeepFool: A simple and accurate method to fool deep neural networks. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2574–2582. IEEE, June 2016. ISBN 9781467388511. doi: 10.1109/cvpr.2016.282. URL `https://doi.org/10.1109/cvpr.2016.282`. 20, 23, 26, 27, 38, 40, 46, 50, 51, 55, 62, 68, 69, 70, 71, 72, 84, 85

[54] Francesco Croce and Matthias Hein. Minimally distorted Adversarial Examples with a Fast Adaptive Boundary Attack. *arXiv e-prints*, art. arXiv:1907.02044, Jul 2019. 20, 27, 85, 87

[55] A. Matyasko and L. P. Chau. Margin maximization for robust classification using deep learning. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 300–307, May 2017. doi: 10.1109/IJCNN.2017.7965869. 20, 62, 66, 71, 74

[56] Gamaleldin Elsayed, Shreya Shankar, Brian Cheung, Nicolas Papernot, Alexey Kurakin, Ian Goodfellow, and Jascha Sohl-Dickstein. Adversarial examples that fool both computer vision and time-limited humans. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 3910–3920. Curran Associates, Inc., 2018. URL `http://papers.nips.cc/paper/7647-adversarial-examples-that-fool-both-computer-vision-and-time-limited-pdf`. 20

[57] David K. Duvenaud, Oren Rippel, Ryan P. Adams, and Zoubin Ghahramani. Avoiding pathologies in very deep networks. In *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics, AISTATS 2014, Reykjavik, Iceland, April 22-25, 2014*, pages 202–210, 2014. URL `http://jmlr.org/proceedings/papers/v33/duvenaud14.html`. 20, 21

[58] Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Logan Engstrom, Brandon Tran, and Aleksander Madry. Adversarial examples are not

bugs, they are features. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 125–136. Curran Associates, Inc., 2019. URL http://papers.nips.cc/paper/8307-adversarial-examples-are-not-bugs-they-are-features.pdf. 21

[59] Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. Analysis of classifiers' robustness to adversarial perturbations. *Machine Learning*, 107(3):481–508, August 2017. ISSN 0885-6125, 1573-0565. doi: 10.1007/s10994-017-5663-3. URL https://doi.org/10.1007/s10994-017-5663-3. 21

[60] Alhussein Fawzi, Seyed-Mohsen Moosavi-Dezfooli, and Pascal Frossard. Robustness of classifiers: From adversarial to random noise. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 1632–1640. Curran Associates, Inc., 2016. URL http://papers.nips.cc/paper/6331-robustness-of-classifiers-from-adversarial-to-random-noise.pdf. 21, 41

[61] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Jonathan Uesato, and Pascal Frossard. Robustness via curvature regularization, and vice versa. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 21

[62] X. Yuan, P. He, Q. Zhu, and X. Li. Adversarial examples: Attacks and defenses for deep learning. *IEEE Transactions on Neural Networks and Learning Systems*, 30(9):2805–2824, Sep. 2019. ISSN 2162-2388. doi: 10.1109/TNNLS.2018.2886017. 21, 22, 35

[63] Naveed Akhtar and Ajmal Mian. Threat of adversarial attacks on deep learning in computer vision: A survey. *IEEE Access*, 6:14410–14430, 2018. ISSN 2169-3536. doi: 10.1109/access.2018.2807385. URL https://doi.org/10.1109/access.2018.2807385. 21, 22, 35

[64] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z. Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security - ASIA CCS '17*, pages 506–519. ACM Press, 2017. ISBN 9781450349444. doi: 10.1145/3052973.3053009. URL https://doi.org/10.1145/3052973.3053009. 22, 30, 51, 62

[65] Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 274–283, Stockholmsmässan, Stockholm Sweden, 2018. PMLR. URL http://proceedings.mlr.press/v80/athalye18a.html. 22, 30

[66] Pedro Tabacof and Eduardo Valle. Exploring the space of adversarial images. In *2016 International Joint Conference on Neural Networks (IJCNN)*, pages 426–433. IEEE, July 2016. ISBN 9781509006205. doi: 10.1109/ijcnn.2016.7727230. URL `https://doi.org/10.1109/ijcnn.2016.7727230`. 23, 41

[67] N. Carlini and D. Wagner. Defensive Distillation is Not Robust to Adversarial Examples. *ArXiv e-prints*, July 2016. 24

[68] Nicholas Carlini, Anish Athalye, Nicolas Papernot, Wieland Brendel, Jonas Rauber, Dimitris Tsipras, Ian Goodfellow, Aleksander Madry, and Alexey Kurakin. On Evaluating Adversarial Robustness. *arXiv e-prints*, art. arXiv:1902.06705, Feb 2019. 24, 25, 29, 30, 76

[69] Apostolos Modas, Seyed-Mohsen Moosavi-Dezfooli, and Pascal Frossard. Sparsefool: A few pixels make a big difference. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 27, 85, 86

[70] S. Sabour, Y. Cao, F. Faghri, and D. J. Fleet. Adversarial Manipulation of Deep Representations. In *International Conference on Learning Representations (ICLR)*. 2015. 27, 28, 40

[71] L. A. Gatys, A. S. Ecker, and M. Bethge. A Neural Algorithm of Artistic Style. *ArXiv e-prints*, August 2015. 28

[72] J. Rony, L. G. Hafemann, L. S. Oliveira, I. Ben Ayed, R. Sabourin, and E. Granger. Decoupling direction and norm for efficient gradient-based l2 adversarial attacks and defenses. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4317–4325, 2019. 28, 84, 85

[73] Y. Dong, F. Liao, T. Pang, H. Su, J. Zhu, X. Hu, and J. Li. Boosting adversarial attacks with momentum. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9185–9193, 2018. 28

[74] Tianhang Zheng, Changyou Chen, and Kui Ren. Distributionally adversarial attack. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019*, pages 2253–2260. AAAI Press, 2019. doi: 10.1609/aaai.v33i01.33012253. URL `https://doi.org/10.1609/aaai.v33i01.33012253`. 28, 77, 84, 85

[75] Pin-Yu Chen, Yash Sharma, Huan Zhang, Jinfeng Yi, and Cho-Jui Hsieh. EAD: elastic-net attacks to deep neural networks via adversarial examples. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 10–17, 2018. 29, 76, 85

[76] Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2 (1):183–202, January 2009. ISSN 1936-4954. doi: 10.1137/080716542. URL https://doi.org/10.1137/080716542. 29, 83

[77] Florian Tramer and Dan Boneh. Adversarial training and robustness for multiple perturbations. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 5866–5876. Curran Associates, Inc., 2019. URL http://papers.nips.cc/paper/8821-adversarial-training-and-robustness-for-multiple-perturbations.pdf. 29, 85, 86

[78] John Duchi, Shai Shalev-Shwartz, Yoram Singer, and Tushar Chandra. Efficient projections onto the l1-ball for learning in high dimensions. In *ICML '08: Proceedings of the 25th international conference on Machine learning*, pages 272–279, New York, NY, USA, 2008. URL http://doi.acm.org/10.1145/1390156.1390191. 29, 82

[79] Osbert Bastani, Yani Ioannou, Leonidas Lampropoulos, Dimitrios Vytiniotis, Aditya Nori, and Antonio Criminisi. Measuring neural net robustness with constraints. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 2613–2621. Curran Associates, Inc., 2016. URL http://papers.nips.cc/paper/6339-measuring-neural-net-robustness-with-constraints.pdf. 29, 40

[80] Guy Katz, Clark Barrett, David L. Dill, Kyle Julian, and Mykel J. Kochenderfer. Reluplex: An efficient SMT solver for verifying deep neural networks. In Rupak Majumdar and Viktor Kunčak, editors, *Computer Aided Verification*, pages 97–117, Cham, 2017. Springer International Publishing. ISBN 978-3-319-63387-9. 29

[81] Shiqi Wang, Kexin Pei, Justin Whitehouse, Junfeng Yang, and Suman Jana. Formal security analysis of neural networks using symbolic intervals. In *27th USENIX Security Symposium (USENIX Security 18)*, pages 1599–1614, Baltimore, MD, 2018. USENIX Association. ISBN 978-1-931971-46-1. URL https://www.usenix.org/conference/usenixsecurity18/presentation/wang-shiqi. 29

[82] Chuan Guo, Jacob Gardner, Yurong You, Andrew Gordon Wilson, and Kilian Weinberger. Simple black-box adversarial attacks. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2484–2493, Long Beach, California, USA, 09–15 Jun 2019. PMLR. URL http://proceedings.mlr.press/v97/guo19a.html. 30

[83] Andrew Ilyas, Logan Engstrom, Anish Athalye, and Jessy Lin. Black-box adversarial attacks with limited queries and information. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 2137–2146, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR. URL `http://proceedings.mlr.press/v80/ilyas18a.html`. 30

[84] Wieland Brendel, Jonas Rauber, and Matthias Bethge. Decision-based adversarial attacks: Reliable attacks against black-box machine learning models. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=SyZI0GWCZ`. 30

[85] Jiajun Lu, Hussein Sibai, Evan Fabry, and David Forsyth. NO Need to Worry about Adversarial Examples in Object Detection in Autonomous Vehicles. *arXiv e-prints*, art. arXiv:1707.03501, Jul 2017. 31

[86] Anish Athalye, Logan Engstrom, Andrew Ilyas, and Kevin Kwok. Synthesizing robust adversarial examples. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 284–293, Stockholmsmässan, Stockholm Sweden, 2018. PMLR. URL `http://proceedings.mlr.press/v80/athalye18b.html`. 31

[87] Huan Xu, Constantine Caramanis, and Shie Mannor. Robust regression and lasso. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems 21*, pages 1801–1808. Curran Associates, Inc., 2009. URL `http://papers.nips.cc/paper/3596-robust-regression-and-lasso.pdf`. 32

[88] Huan Xu, Constantine Caramanis, and Shie Mannor. Robustness and regularization of support vector machines. *J. Mach. Learn. Res.*, 10:1485–1510, December 2009. ISSN 1532-4435. URL `http://dl.acm.org/citation.cfm?id=1577069.1755834`. 32, 39, 41, 42, 44, 45, 47

[89] Uri Shaham, Yutaro Yamada, and Sahand Negahban. Understanding adversarial training: Increasing local stability of supervised models through robust optimization. *Neurocomputing*, 307:195–204, September 2018. ISSN 0925-2312. doi: 10.1016/j.neucom.2018.04.027. URL `https://doi.org/10.1016/j.neucom.2018.04.027`. 32

[90] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. Ensemble adversarial training: Attacks and defenses. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=rkZvSe-RZ`. 32, 60

[91] T. Miyato, S.-i. Maeda, M. Koyama, K. Nakae, and S. Ishii. Distributional Smoothing with Virtual Adversarial Training. In *International Conference on Learning Representation*. 2015. 33, 38, 40, 51, 53, 55, 62, 70, 71, 74

[92] Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric Xing, Laurent El Ghaoui, and Michael Jordan. Theoretically principled trade-off between robustness and accuracy. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 7472–7482, Long Beach, California, USA, 09–15 Jun 2019. PMLR. URL `http://proceedings.mlr.press/v97/zhang19p.html`. 33

[93] J. Hendrik Metzen, T. Genewein, V. Fischer, and B. Bischoff. On Detecting Adversarial Perturbations. In *International Conference on Learning Representations*, 2017. URL `https://openreview.net/forum?id=SJzCSf9xg`. 34

[94] R. Feinman, R. R. Curtin, S. Shintre, and A. B. Gardner. Detecting Adversarial Samples from Artifacts. *ArXiv e-prints*, March 2017. 34

[95] B. Liang, H. Li, M. Su, X. Li, W. Shi, and X. Wang. Detecting adversarial image examples in deep neural networks with adaptive noise reduction. *IEEE Transactions on Dependable and Secure Computing*, pages 1–1, 2018. ISSN 2160-9209. doi: 10.1109/TDSC.2018.2874243. 34

[96] Tianyu Pang, Chao Du, Yinpeng Dong, and Jun Zhu. Towards robust detection of adversarial examples. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 4579–4589. Curran Associates, Inc., 2018. URL `http://papers.nips.cc/paper/7709-towards-robust-detection-of-adversarial-examples.pdf`. 34

[97] Nicholas Carlini and David Wagner. Adversarial examples are not easily detected. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security - AISec '17*, pages 3–14. ACM Press, 2017. ISBN 9781450352024. doi: 10.1145/3128572.3140444. URL `https://doi.org/10.1145/3128572.3140444`. 34

[98] X. Li and F. Li. Adversarial examples detection in deep networks with convolutional filter statistics. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 5775–5783, Oct 2017. doi: 10.1109/ICCV.2017.615. 34

[99] Weilin Xu, David Evans, and Yanjun Qi. Feature squeezing: Detecting adversarial examples in deep neural networks. In *25th Annual Network and Distributed System Security Symposium, NDSS 2018, San Diego, California, USA, February 18-21, 2018*, 2018. 34

[100] Shixiang Gu and Luca Rigazio. Towards deep neural network architectures robust to adversarial examples. In *Workshop contribution in International Conference on Learning Representations (ICLR)*. 2014. URL `http://arxiv.org/abs/1412.5068`. 34, 40

[101] Salah Rifai, Pascal Vincent, Xavier Muller, Xavier Glorot, and Yoshua Bengio. Contractive auto-encoders: Explicit invariance during feature extraction. In *Proceedings of the 28th International Conference on Machine Learning, ICML 2011, Bellevue, Washington, USA, June 28 - July 2, 2011*, pages 833–840, 2011. 34, 39, 40, 41, 46, 50

[102] Yang Song, Taesup Kim, Sebastian Nowozin, Stefano Ermon, and Nate Kushman. Pixeldefend: Leveraging generative models to understand and defend against adversarial examples. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=rJUYGxbCW`. 34

[103] A. van den Oord, N. Kalchbrenner, O. Vinyals, L. Espeholt, A. Graves, and K. Kavukcuoglu. Conditional Image Generation with PixelCNN Decoders. *ArXiv e-prints*, June 2016. 34

[104] Chuan Guo, Mayank Rana, Moustapha Cisse, and Laurens van der Maaten. Countering adversarial images using input transformations. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=SyJ7ClWCb`. 34

[105] Cihang Xie, Jianyu Wang, Zhishuai Zhang, Zhou Ren, and Alan Yuille. Mitigating adversarial effects through randomization. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=Sk9yuql0Z`. 34

[106] E. Raff, J. Sylvester, S. Forsyth, and M. McLean. Barrage of random transforms for adversarially robust defense. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6521–6530, 2019. 34

[107] Cihang Xie, Yuxin Wu, Laurens van der Maaten, Alan L. Yuille, and Kaiming He. Feature denoising for improving adversarial robustness. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 34

[108] Huan Xu and Shie Mannor. Robustness and generalization. *Machine Learning*, 86(3):391–423, November 2011. ISSN 0885-6125, 1573-0565. doi: 10.1007/s10994-011-5268-1. URL `https://doi.org/10.1007/s10994-011-5268-1`. 38, 41

[109] Stephan Zheng, Yang Song, Thomas Leung, and Ian Goodfellow. Improving the robustness of deep neural networks via stability training. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4480–4488. IEEE, June 2016. ISBN 9781467388511. doi: 10.1109/cvpr.2016.485. URL `https://doi.org/10.1109/cvpr.2016.485`. 38, 41, 62

[110] Vladimir N. Vapnik. *The Nature of Statistical Learning Theory*. Springer New York, New York, NY, USA, 1995. ISBN 9781475724424,

9781475724400. doi: 10.1007/978-1-4757-2440-0. URL https://doi.org/10.1007/978-1-4757-2440-0. 39, 41, 42, 43

[111] Yichuan Tang. Deep learning using linear support vector machines. In *Workshop on Representational Learning, ICML*, 2013. 39, 41

[112] G. Hinton, O. Vinyals, and J. Dean. Distilling the Knowledge in a Neural Network. *ArXiv e-prints*, March 2015. 41

[113] Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a defense to adversarial perturbations against deep neural networks. In *2016 IEEE Symposium on Security and Privacy (SP)*, pages 582–597. IEEE, May 2016. ISBN 9781509008247. doi: 10.1109/sp.2016.41. URL https://doi.org/10.1109/sp.2016.41. 41

[114] H. Drucker and Y. Le Cun. Double backpropagation increasing generalization performance. In *IJCNN-91-Seattle International Joint Conference on Neural Networks*, volume ii, pages 145–150 vol.2. IEEE, July 1991. ISBN 0780301641. doi: 10.1109/ijcnn.1991.155328. URL https://doi.org/10.1109/ijcnn.1991.155328. 41, 46

[115] Constantine Caramanis, Shie Mannor, and Huan Xu. Robust optimization in machine learning. In Suvrit Sra, Sebastian Nowozin, and Stephen J Wright, editors, *Optimization for machine learning*, pages 369–402. Mit Press, Cambridge, MA, USA, 2012. ISBN 0-262-01646-X. 41

[116] Koby Crammer and Yoram Singer. On the learnability and design of output codes for multiclass problems. *Machine Learning*, 47(2):201–233, 2002. ISSN 1573-0565. doi: 10.1023/A:1013637720281. URL http://dx.doi.org/10.1023/A:1013637720281. 45

[117] The Theano Development Team, R. Al-Rfou, G. Alain, A. Almahairi, C. Angermueller, D. Bahdanau, N. Ballas, F. Bastien, J. Bayer, A. Belikov, A. Belopolsky, Y. Bengio, A. Bergeron, J. Bergstra, V. Bisson, J. Bleecher Snyder, N. Bouchard, N. Boulanger-Lewandowski, X. Bouthillier, A. de Brébisson, O. Breuleux, P.-L. Carrier, K. Cho, J. Chorowski, P. Christiano, T. Cooijmans, M.-A. Côté, M. Côté, A. Courville, Y. N. Dauphin, O. Delalleau, J. Demouth, G. Desjardins, S. Dieleman, L. Dinh, M. Ducoffe, V. Dumoulin, S. Ebrahimi Kahou, D. Erhan, Z. Fan, O. Firat, M. Germain, X. Glorot, I. Goodfellow, M. Graham, C. Gulcehre, P. Hamel, I. Harlouchet, J.-P. Heng, B. Hidasi, S. Honari, A. Jain, S. Jean, K. Jia, M. Korobov, V. Kulkarni, A. Lamb, P. Lamblin, E. Larsen, C. Laurent, S. Lee, S. Lefrancois, S. Lemieux, N. Léonard, Z. Lin, J. A. Livezey, C. Lorenz, J. Lowin, Q. Ma, P.-A. Manzagol, O. Mastropietro, R. T. McGibbon, R. Memisevic, B. van Merriënboer, V. Michalski, M. Mirza, A. Orlandi, C. Pal, R. Pascanu, M. Pezeshki, C. Raffel, D. Renshaw, M. Rocklin, A. Romero, M. Roth, P. Sadowski, J. Salvatier, F. Savard, J. Schlüter, J. Schulman, G. Schwartz, I. Vlad Serban, D. Serdyuk, S. Shabanian, É. Simon, S. Spieckermann, S. Ramana Subramanyam, J. Sygnowski, J. Tanguay, G. van Tulder, J. Turian,

S. Urban, P. Vincent, F. Visin, H. de Vries, D. Warde-Farley, D. J. Webb, M. Willson, K. Xu, L. Xue, L. Yao, S. Zhang, and Y. Zhang. Theano: A Python framework for fast computation of mathematical expressions. *ArXiv e-prints*, May 2016. 50

[118] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014. URL `http://jmlr.org/papers/v15/srivastava14a.html`. 51, 53

[119] M. Lin, Q. Chen, and S. Yan. Network In Network. In *International Conference on Learning Representations (ICLR)*. December 2014. 56

[120] A. Kurakin, I. Goodfellow, and S. Bengio. Adversarial Machine Learning at Scale. In *International Conference on Learning Representations*, 2017. URL `https://openreview.net/forum?id=BJm4T4Kgx`. 60, 70

[121] David Warde-Farley and Ian Goodfellow. Adversarial perturbations of deep neural networks. In George Papandreou Tamir Hazan and Daniel Tarlow, editors, *Perturbations, Optimization, and Statistics*, chapter 11. The MIT Press, 2016. ISBN 9780262337939. doi: 10.7551/mitpress/10761.003.0012. URL `https://doi.org/10.7551/mitpress/10761.003.0012`. 60

[122] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 2672–2680, 2014. URL `http://papers.nips.cc/paper/5423-generative-adversarial-nets`. 60, 63, 66

[123] Shumeet Baluja and Ian Fischer. Learning to attack: Adversarial transformation networks. In *Proceedings of AAAI-2018*, 2018. URL `http://www.esprockets.com/papers/aaai2018.pdf`. 62

[124] Gamaleldin Elsayed, Dilip Krishnan, Hossein Mobahi, Kevin Regan, and Samy Bengio. Large margin deep networks for classification. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 842–852. Curran Associates, Inc., 2018. URL `http://papers.nips.cc/paper/7364-large-margin-deep-networks-for-classification.pdf`. 62

[125] Moustapha Cissé, Piotr Bojanowski, Edouard Grave, Yann Dauphin, and Nicolas Usunier. Parseval networks: Improving robustness to adversarial examples. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, pages 854–863, 2017. URL `http://proceedings.mlr.press/v70/cisse17a.html`. 62

[126] Pouya Samangouei, Maya Kabkab, and Rama Chellappa. Defense-GAN: Protecting classifiers against adversarial attacks using generative models. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=BkJ3ibb0-. 63

[127] H. Lee, S. Han, and J. Lee. Generative Adversarial Trainer: Defense to Adversarial Perturbations with GAN. *ArXiv e-prints*, May 2017. 63

[128] Chongxuan LI, Taufik Xu, Jun Zhu, and Bo Zhang. Triple generative adversarial nets. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 4088–4098. Curran Associates, Inc., 2017. URL http://papers.nips.cc/paper/6997-triple-generative-adversarial-nets.pdf. 63, 66, 69

[129] Dimitris Bertsimas, Vishal Gupta, and Nathan Kallus. Data-driven robust optimization. *Mathematical Programming*, 167(2):235–292, February 2017. ISSN 0025-5610, 1436-4646. doi: 10.1007/s10107-017-1125-8. URL https://doi.org/10.1007/s10107-017-1125-8. 64

[130] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein GANs. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5767–5777. Curran Associates, Inc., 2017. URL http://papers.nips.cc/paper/7159-improved-training-of-wasserstein-gans.pdf. 66

[131] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2242–2251. IEEE, October 2017. ISBN 9781538610329. doi: 10.1109/iccv.2017.244. URL https://doi.org/10.1109/iccv.2017.244. 67

[132] Yoshua Bengio, Nicholas Léonard, and Aaron C. Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *CoRR*, abs/1308.3432, 2013. URL http://arxiv.org/abs/1308.3432. 68

[133] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mane, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viegas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale Machine Learning on Heterogeneous Distributed Systems. *ArXiv e-prints*, March 2016. 70, 85

[134] Andrew Cotter, Heinrich Jiang, and Karthik Sridharan. Two-player games for efficient non-convex constrained optimization. In Aurélien Garivier

and Satyen Kale, editors, *Proceedings of the 30th International Conference on Algorithmic Learning Theory*, volume 98 of *Proceedings of Machine Learning Research*, pages 300–332, Chicago, Illinois, 2019. PMLR. URL http://proceedings.mlr.press/v98/cotter19a.html. 78

[135] Neal Parikh and Stephen Boyd. Proximal algorithms. *Found. Trends Optim.*, 1(3):127–239, January 2014. ISSN 2167-3888. doi: 10.1561/2400000003. URL https://doi.org/10.1561/2400000003. 80

[136] Álvaro Barbero Jiménez and Suvrit Sra. Modular proximal optimization for multidimensional total-variation regularization. *J. Mach. Learn. Res.*, 19: 56:1–56:82, 2018. URL http://jmlr.org/papers/v19/13-538.html. 84

[137] Yao-Liang Yu. On decomposing the proximal map. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 91–99. Curran Associates, Inc., 2013. URL http://papers.nips.cc/paper/4863-on-decomposing-the-proximal-map.pdf. 84

[138] Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. Robustness may be at odds with accuracy. In *International Conference on Learning Representations (ICLR)*, 2019. URL https://openreview.net/forum?id=SyxAb30cY7. 84

[139] Wieland Brendel, Jonas Rauber, Matthias Kümmerer, Ivan Ustyuzhaninov, and Matthias Bethge. Accurate, reliable and fast robustness evaluation. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 12861–12871. Curran Associates, Inc., 2019. URL http://papers.nips.cc/paper/9446-accurate-reliable-and-fast-robustness-evaluation.pdf. 84, 85, 88

[140] Jonas Rauber, Wieland Brendel, and Matthias Bethge. Foolbox: A python toolbox to benchmark the robustness of machine learning models. In *Reliable Machine Learning in the Wild Workshop, 34th International Conference on Machine Learning*, 2017. URL http://arxiv.org/abs/1707.04131. 85

[141] Maria-Irina Nicolae, Mathieu Sinn, Minh Ngoc Tran, Beat Buesser, Ambrish Rawat, Martin Wistuba, Valentina Zantedeschi, Nathalie Baracaldo, Bryant Chen, Heiko Ludwig, Ian Molloy, and Ben Edwards. Adversarial robustness toolbox v1.2.0. *CoRR*, 1807.01069, 2018. URL https://arxiv.org/pdf/1807.01069. 85, 86

[142] Nicolas Papernot, Fartash Faghri, Nicholas Carlini, Ian Goodfellow, Reuben Feinman, Alexey Kurakin, Cihang Xie, Yash Sharma, Tom Brown, Aurko Roy, Alexander Matyasko, Vahid Behzadan, Karen Hambardzumyan, Zhishuai Zhang, Yi-Lin Juang, Zhi Li, Ryan Sheatsley, Abhibhav Garg,

Jonathan Uesato, Willi Gierke, Yinpeng Dong, David Berthelot, Paul Hendricks, Jonas Rauber, and Rujun Long. Technical report on the cleverhans v2.1.0 adversarial examples library. *arXiv preprint arXiv:1610.00768*, 2018. 85, 86