Defending Neural Networks Against Adversarial Examples

by

ARMON BARTON

Presented to the Faculty of the Graduate School of
The University of Texas at Arlington in Partial Fulfillment
of the Requirements
for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT ARLINGTON
December 2018

Defending Neural Networks Against Adversarial Examples

The members of the Committee approve the doctoral
dissertation of Armon Barton

Jiang Ming
Supervising Professor _____

Matthew Wright

Gergely Zaruba

David Levine

Dean of the Graduate School

To my parents, I would not be who I am today without them.

To my wife, Ameriza Barton, for her endless love and support all the way through my PhD.

ACKNOWLEDGEMENTS

I would like to thank my original supervising professor Dr. Matthew Wright for his invaluable advice and support during the course of my doctoral studies, and my final supervising professor Dr. Jiang Ming for his advice and encouragement through the last half of my doctoral studies. I wish to thank my academic advisor Dr. Bahram Khalili for encouraging and inspiring me to pursue doctoral studies, and Dr. Gergely Zaruba and Mr. David Levine for their interest in my research and for taking the time to serve on my dissertation committee.

Finally, I would like to express my deep gratitude to my mother and father who encouraged and inspired me to pursue my doctoral studies. I am extremely fortunate to be so blessed. I am also extremely grateful to my wife for her encouragement and patience. I also thank several of my friends who have helped me throughout my career.

11-26-2018

ABSTRACT

Defending Neural Networks Against Adversarial Examples

Armon Barton, Ph.D.

The University of Texas at Arlington, 2018

Supervising Professor: Jiang Ming

Deep learning is becoming a technology central to the safety of cars, the security of networks, and the correct functioning of many other types of systems. Unfortunately, attackers can create *adversarial examples*, small perturbations to inputs that trick deep neural networks into making a misclassification. Researchers have explored various defenses against this attack, but many of them have been broken. The most robust approaches are *Adversarial Training* and its extension, *Adversarial Logit Pairing*, but Adversarial Training requires generating and training on adversarial examples from any possible attack. This is not only expensive, but it is inherently vulnerable to novel attack strategies.

We propose *PadNet*, a stacked defense against adversarial examples that does not require knowledge of the attack techniques used by the attacker. PadNet combines two novel techniques: *Defensive Padding* and *Targeted Gradient Minimizing* (TGM). Prior research suggests that adversarial examples exist near the decision boundary of the classifier. Defensive Padding is designed to reinforce the decision boundary of the model by introducing a new class of augmented data within the training set that exists near the decision boundary, called the *padding class*. Targeted Gradient

Minimizing is designed to produce low gradients from the input data point toward the decision boundary, thus making adversarial examples more difficult to find.

In this study, we show that: 1) PadNet significantly increases robustness against adversarial examples compared to adversarial logit pairing, and 2) PadNet is adaptable to various attacks without knowing the attacker's techniques, and therefore allows the training cost to be fixed unlike adversarial logit pairing.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

CHAPTER 1

Introduction

1.1 Statement of the Problem

In recent times, deep neural networks have demonstrated remarkable performance in many difficult machine-learning tasks such as image recognition [1] and speech recognition [2]. However, recent research has shown that an attacker can trick a neural network into misclassifying the input [3, 4, 5]. To do this, the attacker's algorithm perturbs the input by some small amount in a targeted way to ensure misclassification by the neural network. By reverse engineering back propagation, the attacker can use gradient decent to find small perturbations in the input that successfully cause the model to misclassify [3]. In the literature, the perturbed inputs are known as *adversarial examples*.

There are four current approaches for defending against this attack: 1) increasing robustness by introducing adversarial examples within the training set, called *adversarial training* [3, 4], 2) forcing the model to have flat gradients so that it becomes more difficult for the attacker to find adversarial examples using gradient decent based attacks [6], 3) detecting adversarial examples prior to classification [7, 8], and 4) externally processing the input to remove perturbation before classification [9, 10]. Approaches (1) and (2) aim to increase robustness of the defended network, while approaches (3) and (4) aim to filter or reject incoming adversarial examples while leaving the defended network unchanged. Among approaches that aim to increase robustness of the defended network, Adversarial Training has been the most successful

defense. Recently, Kannan et al. introduced Adversarial Logit Pairing (ALP) [11], the latest state-of-the-art defense, which relies heavily on Adversarial Training.

All of these current approaches have limitations. In (1) and (3), the trainer of the neural network must generate and then incorporate adversarial examples from all known attacks into the training process. Thus, the training cost is likely to dramatically increase. Moreover, the adversary may not be limited to the attack techniques used by the trainer and could bypass the defense with a different technique. In (2), Defensive Distillation [12] was proposed to hide the model's gradient information. However, Carlini et al. [5] showed that the gradient information could be recovered. In (4), MagNet [9] was proposed to reform or reject adversarial examples prior to classification. However, their results showed that a window of opportunity exists for the attacker to craft adversarial examples that pass both the reformer and rejector.

Existing research suggests that adversarial examples exist near the *decision boundary* of the model [13]. Using this as motivation for defending against adversarial examples, we propose *PadNet*, a defense that combines two novel approaches: 1) *Defensive Padding*, and 2) *Targeted Gradient Minimizing* (TGM). In our proposal, these two approaches are combined to form a hybrid defense. Defensive Padding is designed to reinforce the decision boundary of the model by introducing a new class of augmented data within the training set that exists near the decision boundary between all classes. This new class is called the *padding class*.

TGM is designed to produce low gradients from the input data points toward the decision boundary. This is achieved by introducing a regularization parameter in the cost function that acts as a penalty if the gradient with respect to the padding class is high. Compared to other gradient-based methods, which affect all gradients, TGM focuses only on the decision boundary.

1.2 Contributions

We show that PadNet significantly increases robustness against adversarial examples compared to ALP. Additionally, we show that PadNet is adaptable to various attacks, and therefore the training cost is fixed unlike in Adversarial Training. Moreover, PadNet is compatible with MagNet [9] and hence could work with MagNet to address weaknesses in that defense. In summary, the contributions of this paper are as follows.

- We propose *PadNet*, a defense that combines two novel approaches for defending against adversarial examples: 1) *Defensive Padding*, and 2) *Targeted Gradient Minimizing* (TGM).
- We evaluate *PadNet* against state-of-the art attacks using a variety of attack models and show: 1) *PadNet* significantly improves robustness against adversarial examples compared to ALP, 2) *PadNet* is adaptive to various attacks unlike ALP, and 3) *PadNet* has a fixed training cost unlike ALP.
- We evaluate *PadNet* on the MNIST and CIFAR-10 datasets.

CHAPTER 2

Background

Deep neural networks have become progressively more prominent in modern society and have demonstrated excellent performance on several difficult machine-learning tasks such as image recognition [14, 1], speech recognition [2], and natural language processing [15]. In security-critical domains, deep neural networks play an important role in autonomous control for robots and vehicles [16, 17], as well as medical imaging analysis [18]. However, recent work suggests that neural networks are vulnerable to attack by an adversary who specifically crafts the inputs in order to fool the neural net and cause misclassification. Szegedy et al. [3] was the first to identify this vulnerability in the image classification domain. They found that it was possible to perturb an image by such a small amount that the perturbation was undetectable by humans, yet would still causes misclassification by the neural net. These specifically crafted images are called *adversarial examples*. More recently, researchers have developed more efficient algorithms for constructing adversarial examples with minimal amount of perturbation [4, 6, 19, 5] and with the intent to cause the neural net to classify the input as *any* class the adversary chooses – called *targeted attacks*. In Figure 2.1, we show original images from the MNIST dataset of handwritten digits compared to crafted adversarial examples. While the original images classify as the correct digit, the adversarial examples all classify as the number '9'.

2.1 Neural Networks

A neural network is a function $F(x) = y$ where $x \in R^n$ is the input, and $y \in R^m$ is the output. The model F also depends on parameters θ. Each layer F_i within the neural network F has output

$$F_i(x) = \sigma(\theta_{wi} \cdot x + \theta_{bi}),$$

where σ is some non-linear activation function, θ_{wi} is a weight matrix, and θ_{bi} is a bias vector. The final output is given by

$$F(x) = softmax(F_n(F_n - 1(...(F_1(x))))).$$

The *softmax* function converts the output of the last layer, called the *logits*, into a probability distribution such that $y_1 + ... + y_m = 1$. In the image classification domain, input x is an $h * w$ pixel image such that $x \in R^{hw}$, and $argmax(y)$ is equal to the classification of x.

2.2 Crafting Adversarial Examples

Adversarial examples can be crafted to cause F to classify x as any target class y_t by adding some perturbation δx to x such that

$$x + \delta x = x', F(x') = y_t.$$

The adversarial example x' can be found efficiently through gradient descent and backpropagation. First, let $C(x, y_t)$ be any cost function such that the output is maximum if $F(x) \neq y_t$ and minimal if $F(x) = y_t$. Then, gradient decent can be used to iteratively adjust the pixels in x until the output of C is minimized and $F(x') = y_t$. For each iteration, x is adjusted such that

$$x' = x - \alpha \nabla_x C(x, y_t),$$

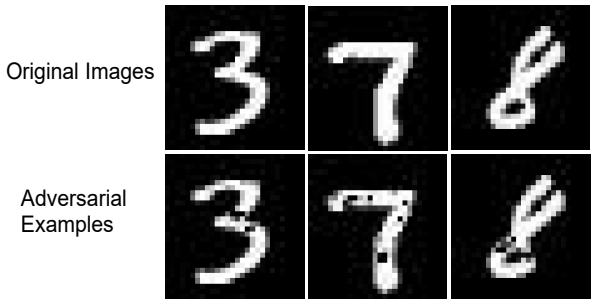


Figure 2.1: Original images compared to adversarial examples. The original images classify as the correct digit, while the adversarial examples all classify as the number '9'.

where the gradient vector $\nabla_x C(x, y_t)$ is the rate of change of cost function C with respect to changing x to the target class y_t, and α is a learning rate constant. This method was used to craft the adversarial examples shown in Figure 2.1, with one difference: instead of adjusting all the pixels for each iteration, we adjusted only one pixel – the pixel that corresponded to the steepest component in the gradient vector.

Ironically, gradient decent is also used for learning parameters θ during training of neural networks. For each iteration of gradient decent during training, θ is adjusted such that

$$\theta' = \theta - \alpha \nabla_\theta C(x, y, \theta).$$

Hence, the optimization method that makes deep neural networks possible is the same method used to attack them.

2.3 Distance Metrics

The perturbation δx should be sufficiently small as to be indistinguishable by humans. In the literature, researchers have proposed three metrics for quantifying δx. These three metrics are referred to as L_0, L_2, and L_∞, and are all L_p norms where

$$\| x - x' \|_p = (\sum_{i=1}^{n} (x - x')^p)^{1/p}.$$

- L_0 indicates the total number of pixels that were changed in the image.
- L_2 measures the Euclidean distance between the two images.
- L_∞ measures the greatest change to any pixel in the image.

For example, the adversarial examples shown in Figure 2.1 were optimized for the L_0 distance metric. The corresponding L_0 distances were 25, 60, and 19 for digits '3', '7', and '8' respectively. For digit '3', we only had to change 25 out of 784 pixels in order for this image to classify as a '9'.

2.4 Known Attacks

Methods for crafting adversarial examples can be categorized into two groups: 1) iterative methods, and 2) optimization methods. Both can be used for targeted or non-targeted attacks. The adversarial examples from Fig. 2.1 are one example of an iterative method. We now describe two other iterative methods from the literature.

2.4.1 Fast Gradient Sign Method (FGSM)

The fast gradient sign method [4] is optimized for the L_∞ distance metric. FGSM finds x' such that

$$x' = x - \epsilon \cdot sign(\nabla_x C(x, y, y_t)),$$

where ϵ is the amount that each pixel is changed. $sign(\nabla_x C(x, y, y_t))$ ignores all the magnitudes for each element in the gradient vector, and just keeps the sign of each element. Thus, each pixel is changed by amount ϵ in the direction of the gradient. FGSM is fast because it runs for one iteration only.

2.4.2 Iterative Gradient Sign Method (IGSM)

IGSM [20] is a refinement of FGSM where instead of changing the pixels by amount ϵ for one iteration, multiple iterations are taken of which pixels are changed by a minimum amount α. However, the change is clipped with respect to the original image, ensuring that no pixel is changed more than amount ϵ. IGSM finds x' such that

$$x' = clip_{\epsilon}(x - \alpha \cdot sign(\nabla_x C(x, y, y_t))).$$

2.4.3 Carlini Wagner Attack

The Carlini Wagner (CW) attack [5] is a suite of attacks that use optimization methods to find minimum perturbation according to the L_0, L_2, and L_∞ distance metrics. The attack is modeled in terms of an optimization problem as follows:

$$\begin{aligned} minimize \quad & D(x, x') + c \cdot f(x') \\ s.t. \quad & x' \in [0, 1]^n \end{aligned}$$

where the goal is to find x' such that the distance $D(x - x')$ is minimized. They add the constraint that, for x' to be a valid image, all pixels must be in the range $[0, 1]$. In this problem, $f(x')$ should also be minimized where f is an objective function that outputs a minimal value when $f(x') = y_t$ – i.e., when the attack succeeds. The constant c is used to increase or decrease the weight of f.

2.5 Threat Model

We assume an attacker may use an adversarial example x' to perform *targeted* and *non-targeted* attacks against a deep neural net F. For a targeted attack, the attack is successful if and only if the defended network F classifies adversarial examples as a particular target class of the adversary's choosing such that $F(x') = y_t$. A

non-targeted attack is successful if and only if the defended network F classifies the adversarial example as any class other than the original label such that $F(x') \neq y$.

We assume an attacker may perform *black-box* and *white-box* attacks against a deep neural net. A black-box attack is one in which the attacker does not know the parameters θ of F, or the training method. As such, the attacker generates adversarial examples on some state-of-the art deep neural net F', and subsequently transfers those example to be classified by the defended network F. A white-box attack is one in which the attacker has full knowledge of the defended network F and thus knows the parameters θ and the training method. As such, the attacker may generate adversarial examples directly on the defended network F.

CHAPTER 3

Related Work

Due to the limited understanding of the nature and extent of vulnerabilities in deep neural nets, defending against adversarial examples has turned out to be a major challenge [21]. Current defensive strategies fall into the following categories: 1) *Adversarial Training*, 2) *gradient masking*, 3) *detecting adversarial examples*, and 4) *external pre-processing*. Approaches (1) and (2) aim to increase robustness of the defended network, while approaches (3) and (4) aim to filter or reject incoming adversarial examples while leaving the defended network unchanged. Currently, the best defenses are a hybrid of two or more of the listed techniques. However, all current defenses fall short in fully securing deep neural nets against adversarial examples.

3.1 Adversarial Training

Currently, the best known method for increasing robustness for a defended neural net is to introduce Adversarial Training [4, 3] – a technique in which crafted adversarial examples along with their ground truth labels are included within the training set. Unfortunately, Goodfellow et al. showed that the FGSM attack still had a 17.6% success rate against a network defended by Adversarial Training. Recently, Kannan et al. proposed Adversarial Logit Pairing (ALP) [11] which is a refinement of Adversarial Training. In ALP, a term is added to the cost function that encourages the *logits*, the final inputs to the softmax function, to be similar for an adversarial example compared to its benign counterpart. For example, consider a model with cost function $C(M, \theta)$ used for Adversarial Training where M is a minibatch of benign

examples $\{x_1, ..., x_m\}$ and their corresponding adversarial examples $\{x'_1, ..., x'_m\}$. Let $f(x, \theta)$ be a function mapping the inputs to their *logits* within the model. Then, adversarial logit pairing is given by minimizing the cost

$$C(M, \theta) + \lambda \frac{1}{M} \sum_{i=1}^{m} (f(x_i, \theta), f(x'_i, \theta)).$$

The authors showed that ALP significantly increased robustness compared to vanilla Adversarial Training. However, ALP inherits several disadvantages from Adversarial Training. Papernot et al. [21] pointed out that, since Adversarial Training is not adaptive, it is essential to include adversarial examples produced by all known attacks. Generating adversarial examples, however, is expensive for most techniques. Therefore, the training cost is likely to increase dramatically. Additionally, the adversary may not be limited to the attack techniques used by the trainer.

3.2 Gradient masking

Gradient masking techniques aim to increase robustness by forcing the model to produce near-zero gradients. Gu et al. [22] proposed adding a layer-wise smoothness penalty within the feed-forward neural net that enables the model to achieve "flatness" around the input data points. Their results suggest that the defense increased distortion, however, they did not suggest that the defense decreased the attack success rate. Papernot et al. [12] introduced defensive distillation which hides the model's gradient information by replacing the model's last layer with a "harder" softmax function. However, Carlini et al. [5] showed that the defense can be bypassed by calculating the gradient directly from the pre-softmax layer.

3.3 Detecting adversarial examples

Grosse et al. [7] proposed using statistical distance metrics for detecting adversarial examples by measuring maximum mean discrepancy and energy distance. Metzen et al. [8] proposed using separate classification networks for detecting adversarial examples. They construct a deep neural net classifier to detect whether an input is adversarial. Similar to Adversarial Training, it is essential for these methods to include adversarial examples produced by all known attacks in order to generalize well. Moreover, the attacker may not be limited to the attack techniques used for training the detection method.

3.4 External pre-processing

In external pre-processing, the defended neural network remains unchanged and therefore lacks robustness. Instead, the input data passes through some pre-processing step before entering the defended neural net.

Xu et al. [10] proposed feature squeezing. In their design, they stack N identical deep neural net models in parallel. The input data passes directly into the first model, and simultaneously passes through an external pre-processing filter before passing through the other models. Each external filter performs a unique feature squeezing process on the input data that reduces the search space available to an adversary. One example of this would be to reduce the color bit depth of each pixel before passing the image to the model. For the final output, they use prediction inconsistency which detects adversary examples when the model reports disagreements among the predictions of the sub-models. According to their results, the method rejects adversarial examples with low perturbations, but is increasingly vulnerable as perturbations increase.

Meng et al. [9] proposed MagNet, a framework for defending against adversarial examples by using an external detector network and a reformer network. The detector detects adversarial examples when they are far from the manifold of normal examples. The reformer network reforms adversarial examples by moving them towards the manifold. According to their results, there exists windows of opportunity where adversarial examples within a particular distortion range pass through both the detector and reformer, and these examples thus attack the network successfully since the defended network itself is unchanged.

Both of these methods are hybrids of detection and external pre-processing and are compatible with methods that increase robustness of the defended network such as Adversarial Training and gradient masking.

CHAPTER 4

Motivation and design

In this section, we explain how knowledge about adversarial examples informs our design for a new defense against adversarial examples and describe those designs in detail. There are two strategies that we focus on: 1) a training strategy, and 2) a gradient minimization strategy. Unlike Adversarial Training, our training strategy does not rely on knowing the methods the attacker uses to create adversarial examples. Therefore, our training set size remains fixed with respect to the number of possible attacks, and our training method does not change regardless of the attacker's method. Likewise, unlike known gradient masking techniques, our technique targets and reduces the gradients where adversarial examples are known to exist rather than obfuscating the existing gradients.

4.1 Training strategy.

In the supervised learning domain of machine learning, the task is to infer a function from labeled training data. Each training example consists of an input object and a desired output value. The supervised learning algorithm analyzes the training data and produces an inferred *decision boundary* that can be used to classify new examples. The placement of this decision boundary is thus very important to the effectiveness of the classifier in generalizing to classify new inputs correctly.

The decision boundary is also important to adversarial examples. Cao et al. [13] performed an experiment in which they formed a small hyper cube around the center

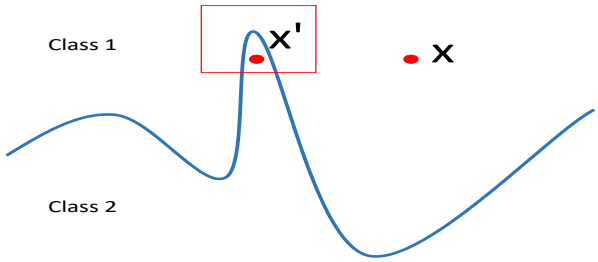


Figure 4.1: Data points from within a hyper cube centered around adversarial example x' classify partly as the ground truth label, and partly as other adjacent classes.

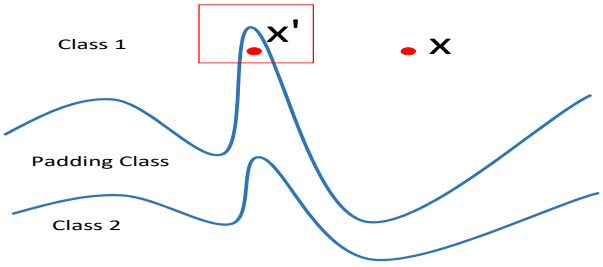


Figure 4.2: In defensive padding, a padding class is introduced within the training set in order to reinforce the boundary layer between classes.

of adversarial inputs x'. They then passed 10,000 data points from within this hyper cube through a deep neural net. This experiment is visualized in Figure 4.1. They found that a majority of the data points classified as the correct class, while the rest of the data points classified as one or more other classes. This suggests that adversarial examples occur near the *decision boundary*, and that decision boundaries of multiple classes can exist adjacently to one another.

Using this as motivation for a defense against adversarial examples, we propose *Defensive Padding* – a training strategy that is meant to reinforce the boundary between classes. This is done by introducing what we term as a *padding class* between classes of deep neural nets. This augmented model can be visualized in Figure 4.2.

We introduce two forms of padding: 1) *mean padding*, and 2) *uniform padding*. In mean padding, we take the mean of one random data point from one class and another random data point from another class. Then, we add Gaussian noise to this

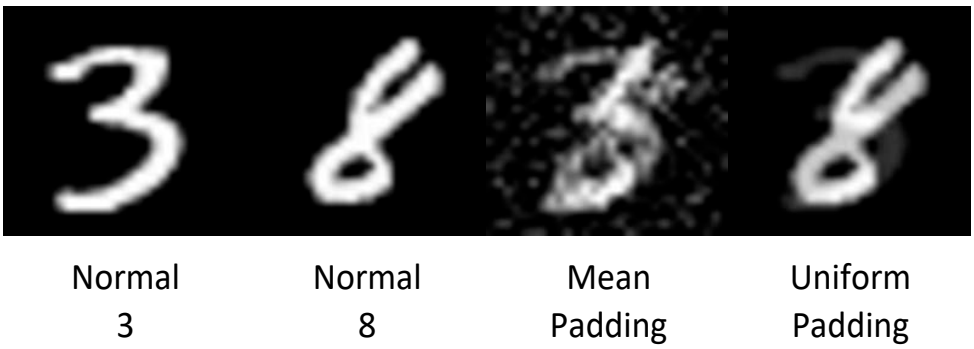


Figure 4.3: Mean padding and uniform padding.

sample. Figure 4.3 shows an example of mean padding taken from two MNIST data points. In uniform padding, we draw a random variable α from the range $[.2, .8]$. Then we take the weighted average of the two data points using the weights α and $1 - \alpha$. Figure 4.3 shows an example of uniform padding from MNIST with $\alpha = 0.2$.

While different amounts and combinations of mean padding and uniform padding are possible, we apply a simple approach. For each training data point, we added one mean padding data point and one uniform padding data point based on it. For each class (call it class A), we generate data points pairing data points from class A with each other class in equal measure.

The padding samples shown in Figure 4.3 are an amalgamation of both of the original samples and should reside in a space that is in between the two original classes while far from the respective centroids of those classes. We show in Section 5 that this defense is adaptive to a variety of attacks. Introducing these additional training samples effectively tripled our training set, though constructing each sample is very easy to do. On the other hand, Adversarial Training requires the trainer to both construct and incorporate adversarial examples from all known attacks, causing the training cost to dramatically increase.

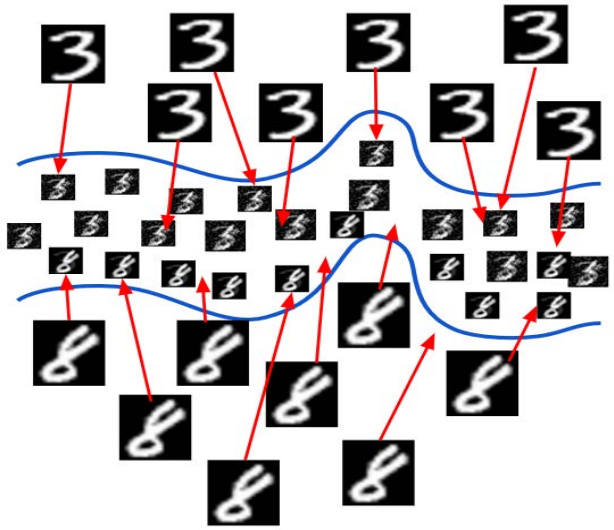


Figure 4.4: Visualizing the synergistic effect of combining TGM and defensive padding. The goal is to minimize the gradient vector (red arrows) starting at the original sample directed toward the padding class.

For the MNIST dataset, Defensive Padding works well with perturbation to every pixel. We found, however, that accuracy of a model trained on CIFAR-10 significantly decreased when all pixels were perturbed during defensive padding. We believe that, since there are three color channels, perturbing more than one third of the pixels results in excessive loss of information. As such, we used a similar padding approach for CIFAR-10 in which the only difference was that we randomly selected one-third of the pixels to perturb rather than all the pixels. The remaining two-thirds of the pixels remain the same as one of the two base images. In uniform padding, the base image is the one with the highest weight, while either image can be used arbitrarily in mean padding.

4.2 Gradient minimization strategy.

Existing gradient masking techniques aim to increase robustness by *arbitrarily* forcing the model to produce near-zero gradients, or by obfuscating the existing gra-

dient. On the other hand, evidence suggests that adversarial examples occur near the decision boundary [13]. As such, we propose *Targeted Gradient Minimizing* (TGM). Our goal is to produce near-zero gradients from input x toward the decision boundary. To achieve this, we add a term to the cost function $cost = C(x, y, \theta)$ such that

$$cost' = cost + \frac{1}{n}\sum_{i=1}^{n}(\nabla_x C(y_{pad}))^2,$$

where $\nabla_x C(y_{pad})$ is the gradient with respect to x when targeting any sample y_{pad} within the padding class. We take the mean of the squared gradient vector and penalize if that value is high. This increases likelihood that most components in the gradient vector from starting point x toward the *padding class* – and thus toward the *decision boundary* – are near zero. As such, TGM relies heavily on a well-formed padding class that truly reinforces the decision boundary. Figure 4.4 is a visualization of the synergistic effect of the combination of TGM and defensive padding. The red arrows represent the gradient vector starting at the original sample directed toward the padding class. Since we know adversarial examples exist near the decision boundary, we presume that an adversary's goal is to perturb benign samples in a direction *toward the decision boundary* using gradient based methods. By minimizing this gradient vector, we increase difficulty in an adversary's ability to find adversarial examples.

Athalye et al. [23] proposed Backward Pass Differentiable Approximation – an attack used to circumvent defenses that rely solely on gradient obfuscation. They identify three forms of gradient obfuscation in which the gradient information can be recovered: 1) shattered gradients, or networks that are non-differentiable due to some external preprocessing step such as MagNet, 2) stochastic gradients, caused by randomizing the output of the network during test time, and 3) exploding & vanishing gradients as found in Defensive Distillation where the output of one network is fed

as input to another during training. In these methods, the gradient exists within the model but is obfuscated on some level depending on the technique. However, TGM does not obfuscate the existing gradient of the model. Additionally, TGM does not satisfy the characteristics and behaviors laid out by Athalye et al. [23] that occur as a result of obfuscated gradients.

CHAPTER 5

Defense Evaluation

To evaluate *Defensive Padding* and *Targeted Gradient Minimizing*, we set up an experiment in which we trained four deep neural nets on MNIST and CIFAR-10. The four respective test networks are described as: 1) *No-Defense*, 2) *Padding-only*, 3) *TGM+Padding*, and 4) *Adversarial Logit Pairing (ALP)*. Apart from the Softmax layer, each network was trained with similar architecture and hyperparameters as used by Carlini and Wagner [5] and by Papernot et al. [12] and shown in Table 5.1 and Table 5.2. One key difference is that we used 11 neurons in the Softmax layer for any network that was trained with defensive padding.

The baseline classification accuracy for each network is shown in Table 5.3. For the MNIST models, the *TGM+Padding* and *Padding-only* networks had near state-of-the art accuracy of 97.43% and 98.98% respectively, while all other networks had state-of-the art accuracy of over 99%. For the CIFAR-10 models, *No-Defense* and *Padding-only* had similar accuracy with respect to this CNN architecture compared to prior work [5, 12]. There was a reduction in accuracy for *TGM+Padding* and *Adversarial Logit Pairing* by approximately 6% and 8% respectively compared to *No-Defense*.

5.1 Training

The *No-Defense* network was trained with standard training data for the MNIST and CIFAR-10 models respectively. The *Padding-only* network was trained with augmented training data that incorporated mean padding and uniform padding examples

Layer Type	MNIST	CIFAR-10
Convolution + ReLU	3 x 3 x 32	3 x 3 x 64
Convolution + ReLU	3 x 3 x 32	3 x 3 x 64
Max Pooling	2 x 2	2 x 2
Convolution + ReLU	3 x 3 x 64	3 x 3 x 128
Convolution + ReLU	3 x 3 x 64	3 x 3 x 128
Max Pooling	2 x 2	2 x 2
Fully Connected + ReLU	200	256
Fully Connected + ReLU	200	256
Softmax (No Padding) / (Padding)	10 / 11	

Table 5.1: Model architectures

Parameter	MNIST	CIFAR-10
Learning Rate	0.1	.01
Optimization Method	SGD	SGD
Dropout	0.5	0.5
Batch Size	10	10
Epochs	50	250

Table 5.2: Model parameters

as described in Chapter 4. The *TGM+Padding* network was trained with the same augmented training data as the Padding-only network, with the addition of a targeted gradient minimization term within the cost function. In Adversarial Logit Pairing, the FGSM adversarial examples were crafted with $\epsilon = 0.25$ and $\epsilon = 0.1$ for MNIST and CIFAR-10 respectively [3, 11]. We attacked these four test networks with FGSM, IGSM, and CW L_2, L_∞, and L_0 attacks under Black Box, White Box, Targeted, and Non-Targeted attack models.

Model	MNIST	CIFAR-10
No-Defense	99.52%	82.25%
Padding-only	98.98%	82.08%
TGM+Padding	97.43%	76.23%
Adversarial Logit Pairing	99.38%	73.9%

Table 5.3: Baseline accuracy comparisons

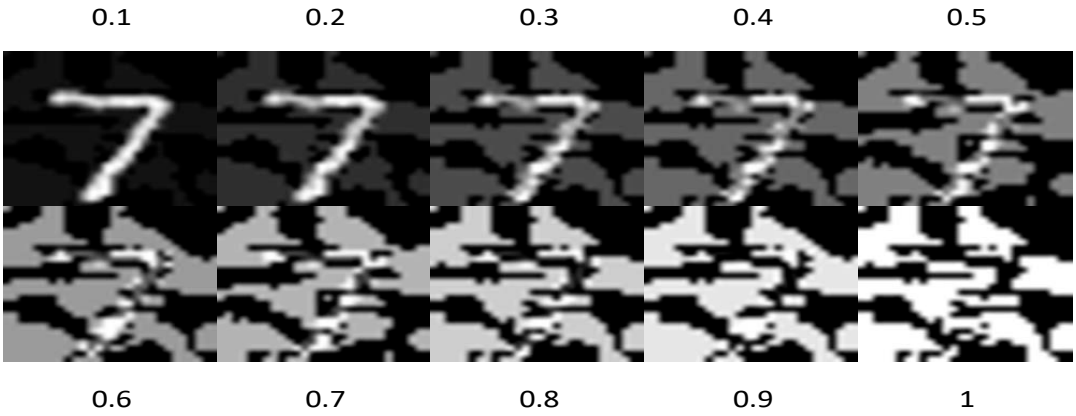
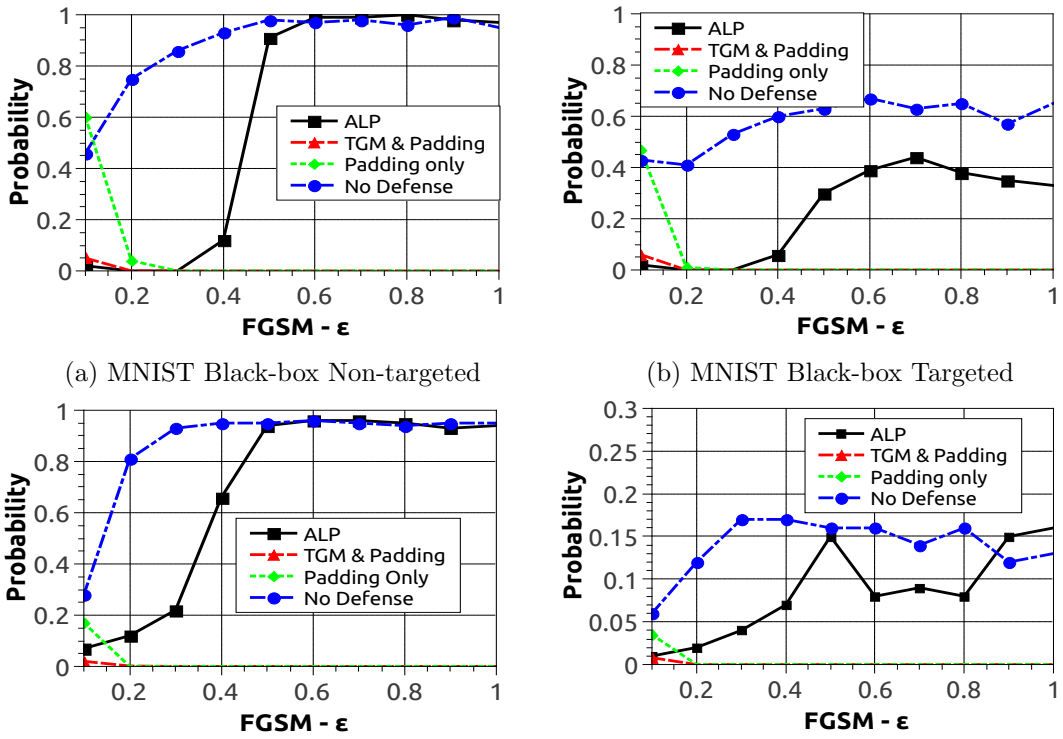


Figure 5.1: FGSM Examples for α ranging from 0.1 to 1.

5.2 Attacking with FGSM

We generated 1000 FGSM adversarial examples with ϵ ranging from 0.1 to 1 and from 0.05 to 0.5 for MNIST and CIFAR-10, respectively. Figure 5.1 shows an example of the added distortion on MNIST as it relates to ϵ. With $\epsilon = 0.1$, the distortion is minimal and hard to detect by humans. When $\epsilon > 0.3$, the distortion is already substantial, and the digit is nearly unrecognizable for $\epsilon > 0.8$. We show the attack success rate against MNIST models in Figure 5.2 and against CIFAR-10 models in Figure 5.3.



(a) MNIST Black-box Non-targeted

(b) MNIST Black-box Targeted

(c) MNIST White-box Non-targeted

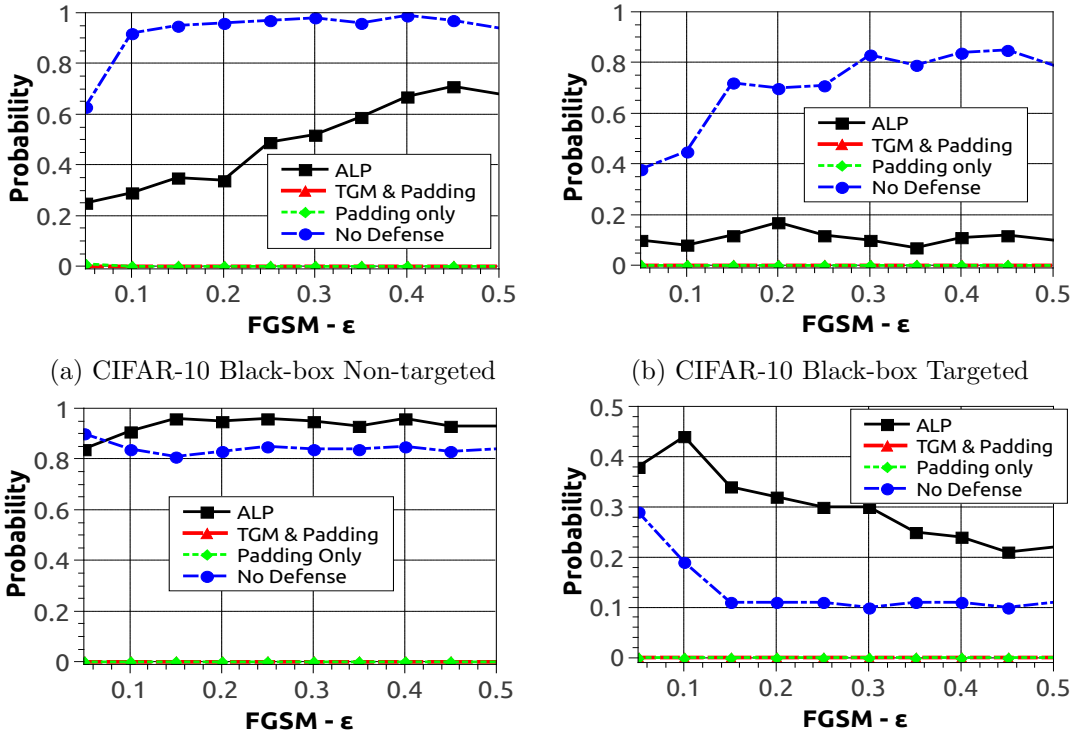
(d) MNIST White-box Targeted

Figure 5.2: FGSM black-box and white-box attack success probability for MNIST.

5.2.1 Defending MNIST against FGSM

5.2.1.1 No Defense

The FGSM attack was most successful against the undefended network compared to TGM+Padding and ALP. With $\epsilon = 0.1$, the attack success rate was 46%, 43%, 30%, and 6% for black-box non-targeted (5.2a), black-box targeted (5.2b), white-box non-targeted (5.2c), and white-box targeted (5.2d) respectively. As ϵ increased, the attack success rate increased to nearly 100% for black-box and white-box non-targeted, and to at most 65% and 17% for black-box targeted and white-box targeted respectively.



(a) CIFAR-10 Black-box Non-targeted

(b) CIFAR-10 Black-box Targeted

(c) CIFAR-10 White-box Non-targeted

(d) CIFAR-10 White-box Targeted

Figure 5.3: FGSM black-box and white-box attack success probability for CIFAR-10

5.2.1.2 TGM+Padding

The FGSM attack was least successful against the TGM+Padding network. With $\epsilon = 0.1$, the attack success rate was 5%, 6%, 2%, and 1% for black-box non-targeted (5.2a), black-box targeted (5.2b), white-box non-targeted (5.2c), and white-box targeted (5.2d) respectively.

For adversarial examples with low distortion ($\epsilon = 0.1$), the TGM+Padding defense improved robustness by 83% to 92% compared to the undefended network. For adversarial examples with major distortion $\epsilon = 1$, the TGM+Padding defense improved robustness by 100% compared to both the undefended network and ALP.

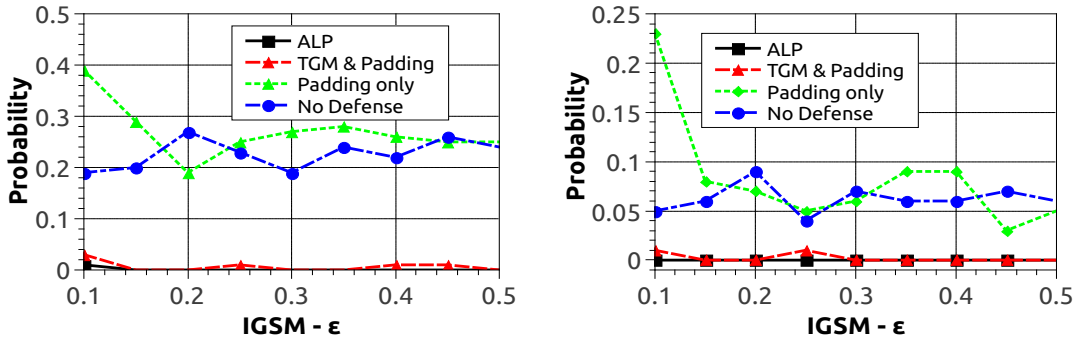
As ϵ increased from 0.2 to 1.0, the attack success rate remained zero under all attack models for TGM+Padding, and Padding-only, and thus no distinction was observed among these defenses comparatively.

5.2.1.3 Padding-only

Under the black-box non-targeted (5.2a) and targeted (5.2b) attack models with $\epsilon = 0.1$, the FGSM attack was most successful against Padding-only compared to all other test networks. The attack success rate was 60% and 47% for black-box non-targeted and targeted respectively.

5.2.1.4 Adversarial Logit Pairing

Under the black-box non-targeted (5.2a) and targeted (5.2b) attack models, with $\epsilon <= 0.25$, ALP provided similar robustness compared to TGM+Padding. However, Madry et al. [24] suggested that Adversarial Training loses robustness against adversarial examples that were crafted with greater ϵ values compared to the adversarial examples used for training. In our results, we observed that ALP inherits this disadvantage from Adversarial Training. As ϵ increased from 0.25 to 1.0, the attack success rate increased to nearly 100% under black-box non-targeted (5.2a) and at most 42% under black-box targeted (5.2b). In the white box cases, ALP had similar robustness with $\epsilon = 0.1$ compared to TGM+Padding. However, as ϵ increased to 1.0, the attack success rate increased to nearly 100% under white-box non-targeted (5.2c), and at most 16% under white-box targeted (5.2d)



(a) MNIST Black-box Non-targeted

(b) MNIST Black-box Targeted

(c) MNIST White-box Non-targeted

(d) MNIST White-box Targeted

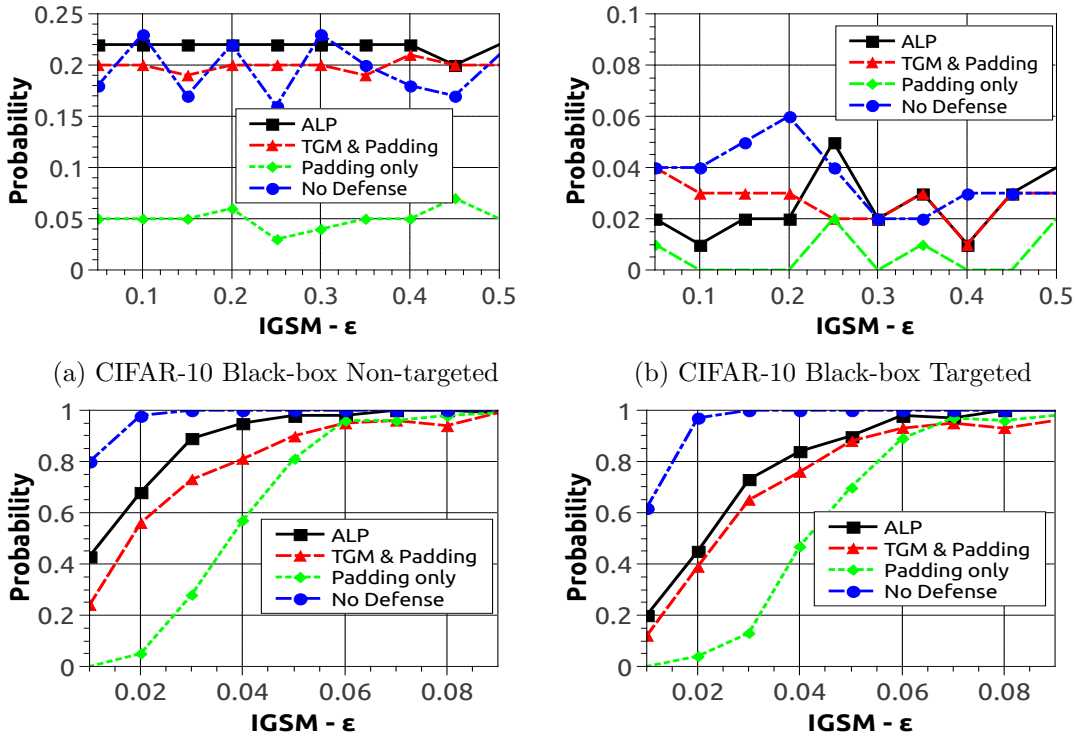
Figure 5.4: IGSM black-box and white-box attack success probability for MNIST.

5.2.2 Defending CIFAR-10 against FGSM

5.2.2.1 No Defense

Under the black-box model, the FGSM attack was most successful against the undefended network compared to TGM+Padding and ALP. With $\epsilon = 0.05$, the attack success rate was 63% and 38% for black-box non-targeted (5.3a) and black-box targeted (5.3b), respectively. As ϵ increased, the attack success rate increased to nearly 100% for black-box non-targeted and to at most 84% for black-box targeted.

The attack success rate against adversarial examples with low distortion ($\epsilon = 0.05$) was 95% and 29% for white-box non-targeted (5.3c) and white-box targeted (5.3d), respectively. Due to the lack of iterations in FGSM, as ϵ increased, the attack



(a) CIFAR-10 Black-box Non-targeted

(b) CIFAR-10 Black-box Targeted

(c) CIFAR-10 White-box Non-targeted

(d) CIFAR-10 White-box Targeted

Figure 5.5: IGSM black-box and white-box attack success probability for CIFAR-10. success rate decreased to approximately 82% and 10% for white-box non-targeted and white-box targeted respectively.

5.2.2.2 PadNet

PadNet provided the most robustness compared to the other test networks. The attack success rate remained approximately 0% against TGM+Padding and Padding only for all ϵ values under all tested attack models 5.3a, 5.3b, 5.3c, 5.3d.

5.2.2.3 Adversarial Logit Pairing

ALP provided limited robustness compared to PadNet. Under the black-box attack models with low distortion ($\epsilon = 0.05$), the attack success rate was reduced from

62% to 25% (5.3a), and from 38% to 10% (5.3b) compared to no defense. However, under the white-box non-targeted model (5.3c), the attack success rate for ALP was approximately 10% to 15% higher compared to no defense for ϵ values ranging from 0.1 to 0.5. Likewise, under the white-box targeted model (5.3d), the attack success rate for ALP was approximately 10% to 22% higher compared to no defense.

The contrasting results with respect to ALP's robustness on MNIST compared to CIFAR-10 for the white-box attacks indicate that its robustness is highly dependent on the characteristics of the particular data set that the defense is designed to protect.

5.3 Attacking with IGSM

We generated 1000 MNIST and CIFAR-10 IGSM adversarial examples. We show attack success rate against MNIST models in Figure 5.4 and against CIFAR-10 models in Figure 5.5.

5.3.1 Defending MNIST Against IGSM

5.3.1.1 No Defense

The IGSM attack was most successful against the undefended network compared to TGM+Padding and ALP. With $\epsilon = 0.1$, the attack success rate was 19%, 5%, 80%, and 58% for black-box non-targeted (5.4a), black-box targeted (5.4b), white-box non-targeted (5.4c), and white-box targeted (5.4d) respectively. As ϵ increased, the attack success rate increased to 100% for the white-box attacks, and to at most 27% and 9% for black-box non-targeted and targeted respectively.

5.3.1.2 Adversarial Logit Pairing

ALP provided similar robustness compared to PadNet under the black-box attack models. The attack success rate was approximately 0% to 1.5% for the targeted (5.4a) and non-targeted (5.4b) attacks for all tested ϵ values. Under the white-box attacks, ALP provided significantly less robustness compared to PadNet. With $\epsilon = 0.1$, the attack success rate was approximately 17% and 6% for white-box non-targeted (5.4c) and targeted (5.4d) respectively, and it increased to 84% and 59% as ϵ increased respectively.

5.3.1.3 TGM+Padding

Compared to ALP, TGM+Padding provided similar robustness against the black-box attacks and significantly more robustness against the white-box attacks. With $\epsilon = 0.1$, the attack success rate was 3%, 1%, 5%, and 2% for black-box non-targeted (5.4a), black-box targeted (5.4b), white-box non-targeted (5.4c), and white-box targeted (5.4d), respectively. For the black-box attacks, the attack success rate remained at approximately 0% as ϵ increased. On the other hand, the attack success rate increased to 50% and 42% for the white box attacks as ϵ increased.

For adversarial examples with low distortion ($\epsilon = 0.1$), the TGM+Padding defense improved robustness by 80% to 96% compared to the undefended network.

5.3.1.4 Padding-only

With $\epsilon = 0.1$, the IGSM attack was most successful against Padding-only compared to all other test networks. The attack success rate was 39%, 23%, 96%, and 95% for black-box non-targeted (5.4a), black-box targeted (5.4b), white-box non-targeted (5.4c), and white-box targeted (5.4d) respectively.

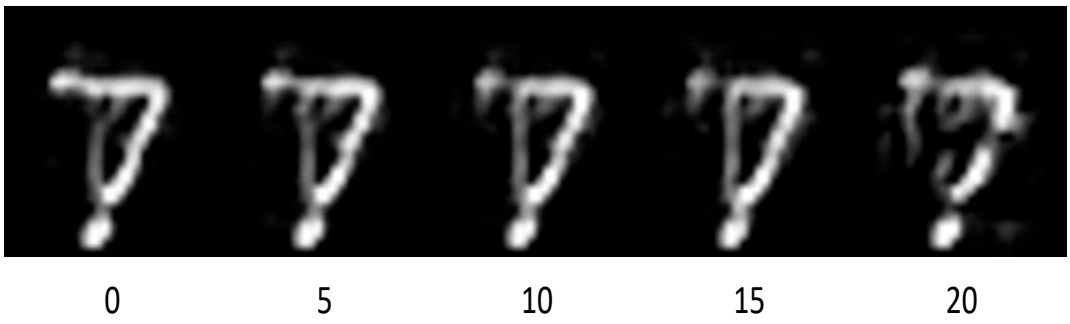
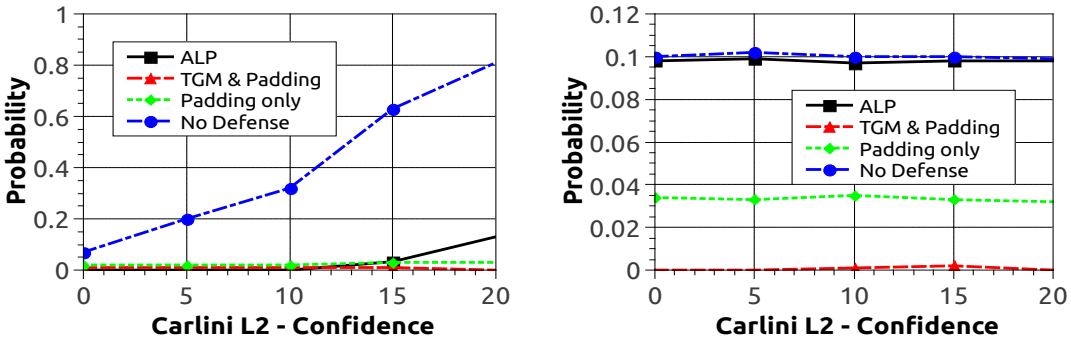


Figure 5.6: CW L_2 examples for increasing confidence values. As confidence is increased, the attack success rate and distortion both increase.



(a) MNIST (b) CIFAR-10

Figure 5.7: CW L_2 black-box attack success probability.

5.3.2 Defending CIFAR-10 Against IGSM

5.3.2.1 No Defense

Under the black-box non-targeted model (5.5a), the attack success rate was approximately 16% to 23% against the undefended network. Under the white-box attacks, the attack success rate was 80% and 62% for non-targeted (5.5c) and targeted (5.5d) respectively, and increased to approximately 100% as ϵ was increased to 0.02.

5.3.2.2 Padding Only

Under all attack models, Padding only provided the most robustness compared to both TGM+Padding and ALP. The attack success rate was approximately 5% against the black-box non-targeted (5.5a) attack. Against the white-box attacks

(5.5c), (5.5d), the attack success rate was approximately 0% for $\epsilon = 0.01$. However, as ϵ was increased for the white-box attacks, the attack success rate reached nearly 100%.

5.3.2.3 TGM+Padding and ALP

Under the White-box non-targeted attack model (5.5c), TGM+Padding provided slightly more robustness compared to ALP. With $\epsilon = 0.01$, the attack success rate was 24% and 42% for TGM+Padding and ALP respectively. For all other attack models, TGM+Padding and ALP provided similar robustness.

5.4 Carlini Wagner L_2 Attack

The CW L_2 attack allows the adversary to adjust attack strength by adjusting the confidence. As confidence is increased, distortion increases. We generated 500 CW L_2 examples with confidence values of 0, 5, 10, 15, and 20. Figure 5.6 shows one example of the added distortion on one MNIST sample as confidence is increased. Using these examples, we performed a black-box targeted attack on our test networks. The attack success rates for MNIST and CIFAR-10 are shown in Figures 5.7a and 5.7b respectively.

5.4.1 No Defense

The attack success rate for the undefended network on MNIST (5.7a) was 7% for $confidence = 0$ and increased to 81% as $confidence$ was increased to 20. The attack success rate for the undefended network on CIFAR-10 remained at approximately 10% for all $confidence$ values.

5.4.2 Adversarial Logit Pairing

For defending MNIST, ALP maintained similar robustness compared to PadNet for confidence values ranging from 0 to 15. However, when confidence was increased to 20, the attack success rate against ALP was approximately 14%. For defending CIFAR-10, ALP did not provide any additional robustness compared to no defense. The attack success rate for ALP on CIFAR-10 remained approximately 10% for all $confidence$ values.

5.4.3 TGM+Padding

For defending both MNIST and CIFAR-10, TGM+Padding provided the most robustness compared to the other test networks. The attack success rate for defending MNIST and CIFAR-10 did not exceed 1% for all tested confidence values.

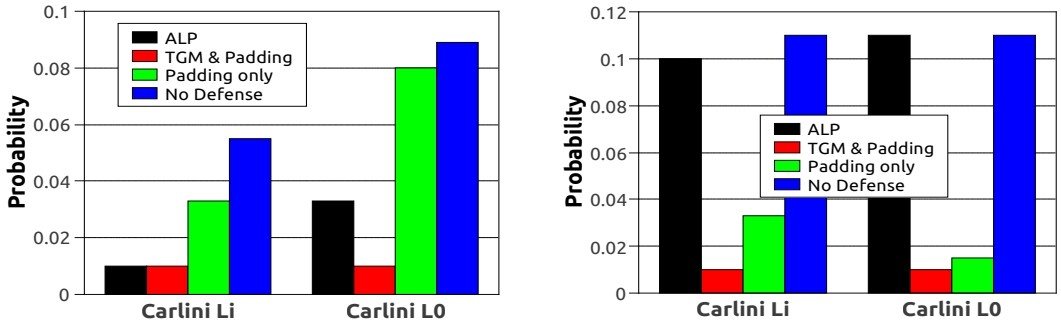
5.4.4 Padding-only

For defending MNIST, Padding-only provided similar robustness compared to TGM+Padding. The attack success rate for defending MNIST did not exceed 3% for any of the tested confidence values. For defending CIFAR-10, Padding-only significantly improved robustness compared to no defense. The attack success rate for defending CIFAR-10 was reduced from 10% to approximately 3.5% for Padding-only compared to No-Defense.

5.5 Carlini Wagner L_∞ & L_0 Attack

The CW L_∞ and L_0 attacks do not allow the adversary to adjust a confidence parameter. As such, we generated one set of 1,000 adversarial examples for each attack. In Figure 5.8, we show one CW L_0 and one CW L_∞ adversarial example from the MNIST dataset. Note that the distortion in the CW L_∞ example has a

Figure 5.8: CW L_0 & L_∞ adversarial examples.



(a) MNIST (b) CIFAR-10

Figure 5.9: CW L_∞ & L_0 black-box attack success probability.

similar pattern to the distortion in the FGSM adversarial examples shown in Figure 5.1. Though these are two distinct attacks, FGSM being iterative and CW being optimizational, they both optimize on the same L_∞ distance metric and thus produce similar distortion patterns. Using these examples, we performed a black-box targeted attack on our test networks. The attack success rates for MNIST and CIFAR-10 are shown in Figures 5.9a and 5.9b, respectively.

5.5.1 No Defense

The attack success rate was 6% and 9% for the L_∞ and L_0 attacks against MNIST respectively, and 11% for the L_∞ and L_0 attacks against CIFAR-10.

5.5.2 Adversarial Logit Pairing

For defending MNIST, ALP was as effective as TGM+Padding against the L_∞ attack, and slightly less effective against the L_0 attack compared to TGM+Padding. Against the L_∞ attack on MNIST (5.9a), the attack success rate was approximately 1% for ALP and TGM+Padding. Against the L_0 attack on MNIST (5.9a), the attack success rate was 3% for ALP.

For defending CIFAR-10, ALP did not provide any additional robustness compared to no defense. The attack success rate was 10% and 11% against the L_∞ and L_0 attack respectively.

5.5.3 TGM+Padding

For defending both MNIST and CIFAR-10 against the L_∞ and L_0 attacks, TGM+Padding provided the most robustness compared to the other test networks. The attack success rate did not exceed 1% against the L_∞ and L_0 attacks on MNIST and on CIFAR-10. For defending MNIST, TGM+Padding improved robustness by 89% against the L_0 attack compared to No-Defense, and by 81% against the L_∞ attack compared to No-Defense. For defending CIFAR-10, TGM+Padding improved robustness by 91% against both the L_0 and L_∞ attacks compared to No-Defense.

5.5.4 Padding Only

For defending against the L_0 attack on MNIST, the attack success rate for Padding only was 8%; Padding only did not provide any significant robustness compared to No-Defense. Conversely, for defending against the L_0 attack on CIFAR-10, the attack success rate for Padding only was 1.5%; Padding only provided similar robustness as compared to TGM+Padding. The contrasting results imply that robustness may very across diverse data sets.

CHAPTER 6

Discussion

6.1 Adversarial Logit Pairing

Madry et al. [24] pointed out that *Adversarial Training* becomes increasingly vulnerable as distortion increases beyond the ϵ value used for training. Our results suggest that ALP inherits this disadvantage from Adversarial Training. For example, ALP was trained using FGSM examples with $\epsilon = 0.25$. In the FGSM attacks on MNIST (Figures 5.2a, 5.2b, 5.2c), the attack success rate dramatically increased when ϵ was increased beyond 0.25.

Moreover, Papernot et al. [21] pointed out that adversarial training is not adaptive to crafted adversarial examples that differ from the adversarial examples used for training the model. Our results suggest that ALP also inherits this disadvantage. Notably, ALP did not adapt to the CW attacks. On MNIST, the attack success rate reached 14% for $confidence = 20$. On CIFAR-10, the attack success rate was not significantly different than No-Defense.

We conclude that, for ALP to be effective, the training set should incorporate adversarial examples from all known attacks. Since this not only involves more training but also the high computational cost of generating the adversarial examples, this is becomes a major disadvantage because the training cost is likely to increase dramatically. Moreover, the adversary is not limited to the maximum value of ϵ used in training nor the attack methods used by the trainer.

6.2 Diverse Data Sets

We observed some contrasting results with respect to MNIST and CIFAR-10 that may imply that robustness results may vary across diverse data sets. In the IGSM attacks, for example, we observed that Padding only significantly increased robustness for defending *CIFAR-10* compared to the other test networks. Conversely, for defending *MNIST* against IGSM, TGM+Padding significantly increased robustness compared to the other test networks. Secondly, for the L0 attack on *MNIST*, we found that Padding only did not provide any additional robustness compared to No-Defense. On the other hand, for the L0 attack on *CIFAR-10*, Padding only provided similar robustness compared to TGM+Padding. These results indicate that the best defense method is highly dependent on the particular data set.

6.3 Robustness / Accuracy Trade-off

In general, for defending CIFAR-10, Padding only provided the most robustness against the IGSM attacks and high robustness against the CW attacks. Given that the Padding only model for CIFAR-10 had an accuracy of 82.08%, we conclude that Padding only had the best robustness / accuracy trade-off for defending CIFAR-10. On the other hand, TGM+Padding provided the most overall robustness for defending MNIST. With an accuracy of 97.43% for the MNIST model, we conclude that TGM+Padding had the best robustness / accuracy trade-off for defending MNIST.

6.4 PadNet Training Cost

In our experiments, PadNet adapted well with respect to the different attack methods compared to ALP. This is advantageous for PadNet because the training set incorporated one set of padding examples, yet robustness was consistent across

attacks. Therefore, the training cost should be fixed with respect to the tested attack methods.

6.5 Combined Defense

One hybrid defense method known as *feature squeezing* [10] was found to reject adversarial examples with low distortion. However, as distortion increased, this method became increasingly vulnerable. In the MagNet defense [9], adversarial examples within a particular distortion range pass through both the detector and reformer. One advantage for PadNet is that it is compatible with defenses such as MagNet. Additionally, our results show that PadNet may have higher robustness for the distortion ranges that caused vulnerability within MagNet. As such, combining PadNet with MagNet would synergistically produce a more robust hybrid defense. We intend to explore this in future work.

6.6 Future Work

For other future work, we plan to improve upon our method for populating the padding class. In this work, we presented our first attempted methods for populating the padding class. However, there may be more effective solutions. Past research has focused on using *generative adversarial networks (GANs)* [25] to derive adversarial examples [26]. One idea would be to use GANs to derive a padding class.

As we noted above, we believe that robustness of the defense is highly dependent on the data set. As such, we plan on testing PadNet and the combined effect of PadNet with other defenses on more diverse datasets such as ImageNet.

CHAPTER 7

Conclusion

We have proposed *PadNet*, a stacked defense against adversarial examples. PadNet does not require knowledge of the attack techniques used by the attacker. PadNet combines two novel techniques: 1) *Defensive Padding*, and 2) *Targeted Gradient Minimizing* (TGM). Prior research suggests that adversarial examples exist near the decision boundary of the classifier. Defensive padding is designed to reinforce the decision boundary of the model by introducing a new class of augmented data within the training set that exists near the decision boundary, called the *padding class*. Targeted Gradient Minimizing is designed to produce low gradients from the input data point toward the decision boundary, thus making adversarial examples more difficult to find.

Our results showed that: 1) PadNet significantly increases robustness against adversarial examples compared to Adversarial Logit Pairing (current state-of-the-art), and 2) that PadNet is adaptable to various attacks without knowing the attacker's techniques, and therefore allowing the training cost to be fixed unlike Adversarial Logit Pairing.

REFERENCES

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[2] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, *et al.*, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," in *IEEE Signal Processing Magazine*, vol. 29, no. 6. IEEE, 2012, pp. 82–97.

[3] 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'13)*, 2013.

[4] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *International Conference on Learning Representations (ICLR'15)*, 2015.

[5] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *Security and Privacy (SP), 2017 IEEE Symposium on*. IEEE, 2017, pp. 39–57.

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

[7] K. Grosse, P. Manoharan, N. Papernot, M. Backes, and P. McDaniel, "On the (statistical) detection of adversarial examples," in *arXiv preprint arXiv:1702.06280*, 2017.

[8] J. H. Metzen, T. Genewein, V. Fischer, and B. Bischoff, "On detecting adversarial perturbations," 2017.

[9] D. Meng and H. Chen, "Magnet: a two-pronged defense against adversarial examples," in *Proceedings of the 24th ACM Conference on Computer and Communications Security (CCS'17)*, 2017.

[10] W. Xu, D. Evans, and Y. Qi, "Feature squeezing: Detecting adversarial examples in deep neural networks," in *arXiv preprint arXiv:1704.01155*, 2017.

[11] H. Kannan, A. Kurakin, and I. Goodfellow, "Adversarial logit pairing," *arXiv preprint arXiv:1803.06373*, 2018.

[12] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami, "Distillation as a defense to adversarial perturbations against deep neural networks," in *Security and Privacy (SP), 2016 IEEE Symposium on*. IEEE, 2016, pp. 582–597.

[13] X. Cao and N. Z. Gong, "Mitigating evasion attacks to deep neural networks via region-based classification," 2017.

[14] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," in *Proceedings of the IEEE*, vol. 86, no. 11. IEEE, 1998, pp. 2278–2324.

[15] D. Andor, C. Alberti, D. Weiss, A. Severyn, A. Presta, K. Ganchev, S. Petrov, and M. Collins, "Globally normalized transition-based neural networks," 2016.

[16] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, *et al.*, "End to end learning for self-driving cars," in *arXiv preprint arXiv:1604.07316*, 2016.

[17] C. Finn and S. Levine, "Deep visual foresight for planning robot motion," in *Robotics and Automation (ICRA), 2017 IEEE International Conference on*. IEEE, 2017, pp. 2786–2793.

[18] D. Shen, G. Wu, and H.-I. Suk, "Deep learning in medical image analysis," in *Annual Review of Biomedical Engineering*, no. 0. Annual Reviews 4139 El Camino Way, PO Box 10139, Palo Alto, California 94303-0139, USA, 2017.

[19] Y. Liu, X. Chen, C. Liu, and D. Song, "Delving into transferable adversarial examples and black-box attacks," in *International Conference on Learning Representations (ICLR'17)*, 2017.

[20] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," 2016.

[21] N. Papernot, P. McDaniel, A. Sinha, and M. Wellman, "Towards the science of security and privacy in machine learning," in *arXiv preprint arXiv:1611.03814*, 2016.

[22] S. Gu and L. Rigazio, "Towards deep neural network architectures robust to adversarial examples," in *arXiv preprint arXiv:1412.5068*, 2014.

[23] A. Athalye, N. Carlini, and D. Wagner, "Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples," *arXiv preprint arXiv:1802.00420*, 2018.

[24] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," *arXiv preprint arXiv:1706.06083*, 2017.

[25] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems*, 2014.

[26] J. Hayes and G. Danezis, "Machine learning as an adversarial service: Learning black-box adversarial examples," *arXiv preprint arXiv:1708.05207*, 2017.