Enhancing Adversarial Robustness of Deep Neural Networks

by

Jeffrey Zhang

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Computer Science and Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2019

Author .....
Department of Electrical Engineering and Computer Science
June 7, 2019

Certified by .....
Aleksander Madry
Associate Professor of Computer Science
Thesis Supervisor

Accepted by .....
Katrina LaCurts
Chair, Master of Engineering Thesis Committee

Enhancing Adversarial Robustness of Deep Neural Networks

by

Jeffrey Zhang

Submitted to the Department of Electrical Engineering and Computer Science
on June 7, 2019, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Computer Science and Engineering

Abstract

Logit-based regularization and pretrain-then-tune are two approaches that have recently been shown to enhance adversarial robustness of machine learning models. In the realm of regularization, Zhang et al. (2019) proposed TRADES, a logit-based regularization optimization function that has been shown to improve upon the robust optimization framework developed by Madry et al. (2018) [14, 9]. They were able to achieve state-of-the-art adversarial accuracy on CIFAR10. In the realm of pretrain-then-tune models, Hendrycks el al. (2019) demonstrated that adversarially pretraining a model on ImageNet then adversarially tuning on CIFAR10 greatly improves the adversarial robustness of machine learning models. In this work, we propose Adversarial Regularization, another logit-based regularization optimization framework that surpasses TRADES in adversarial generalization. Furthermore, we explore the impact of trying different types of adversarial training on the pretrain-then-tune paradigm.

Thesis Supervisor: Aleksander Madry
Title: Associate Professor of Computer Science

Acknowledgments

I would like to thank Professor Aleksander Madry for allowing me to work in an intellectually stimulating environment with extremely talented individuals and giving me good directions to pursue in research. I would like to thank Andrew Ilyas and Aleksandar Makelov for providing great research insights and directly mentoring me with respect to the project. I would like to thank the rest of the Madry Lab for their help and receptiveness to discussing random topics about Deep Learning. I would like to thank MIT EECS for an amazing four years. Finally, I would like to thank my family for their everlasting love and support.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

With cutting edge advances in machine learning, machine learning models are being used in an increasing number of real-world applications. Some of these applications include safety-critical tasks such as perception for autonomous vehicle and facial recognition for identification purposes [10, 13]. Thus, understanding the security and limitations of these models is becoming increasingly important.

Academic literature previously pointed out that naturally trained state-of-the-art Deep Neural Network models are vulnerable to *adversarial examples*, which are slightly perturbed, usually imperceivably different, versions of inputs that may cause erroneous results. With sufficient knowledge of a model's architecture, one can generate an adversarial example by starting with the natural example and iteratively applying small, carefully-crafted, constrained perturbations in the direction that maximizes some loss metric. In practice, this idea is captured in the projected gradient descent (PGD) attack [8].

Adversarial robustness is a quality that provides practical benefits. Adversarial robustness provides security against adversaries that may cause real-life harm using adversarial examples that cause machine learning decision systems to behave incorrectly. By viewing the challenge of training adversarially-robust models through the lens of robust optimization, Madry et al. (2017) showed that adversarial robustness can be achieved through a natural minimax objective function [9]. Their results motivate the idea that training machine learning models on adversarially perturbed ver-

sions of training sets is a fruitful approach to training deep machine learning models robust to adversarial examples. They also motivate the PGD adversary as a complete first-order adversary for various convex constraint sets that the threat model is defined for. See Figure 3-1 for an example of an adversarial example [3].

This work explores methods of further enhancing the current state-of-the-art adversarially-robust machine learning models though regularized minimax optimization as well as dataset pretraining. In doing so, a deeper dive into the factors that affect robust generalization will also inevitably be explored.

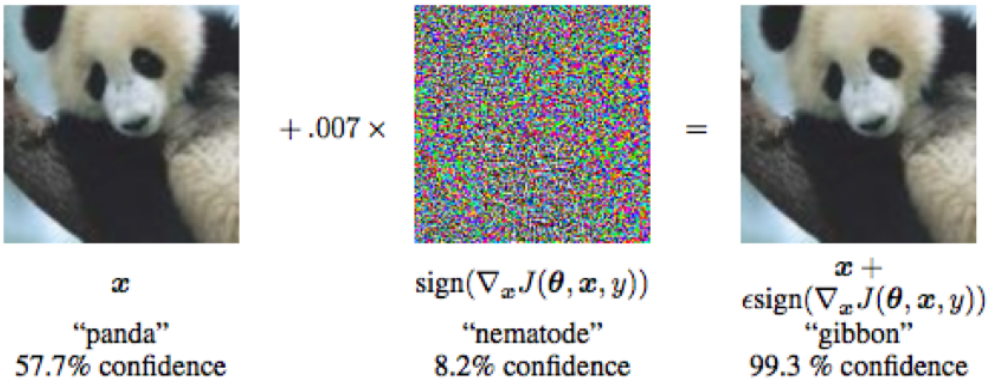

Figure 1-1: Example of Adversarial Example [3]

Chapter 2

Background

To motivate specific enhancements to the adversarial robust learning process, we must first formalize the methodology used to evaluate adversarial robustness of machine learning models. First, we define the natural classification task and evaluation metrics. Then, we formally introduce the adversarial classification task and associated evaluation metrics. Next, we introduce two recently proposed ideas that motivate deeper understanding of adversarially-robust models, potentially motivating enhancements to the current state-of-the-art. Finally, we make a few notes about how these ideas are implemented in practice.

2.1 Tasks at Hand

2.1.1 Natural Classification Task

Suppose we are given n data points, denoted by $X_n = \{x_1, ..., x_n\}$ where $x_i \in \mathbb{R}^d$ is a d dimensional input vector, with corresponding labels, denoted by $Y_n = \{y_1, ..., y_n\}$ where $y_i \in \{c_1, ..., c_k\}$ is a categorical variable and c_i represents a class. Typically, we assume that these (x_i, y_i) pairs are sampled from some underlying distribution, denoted by F, and that there is some latent relationship between x_i and y_i that can be learned to correctly identify y_i given x_i. A *classification algorithm*, denoted by $\hat{f}$, is a function that takes an arbitrary x and maps it a class label $y \in \{c_1, ..., c_k\}$). Al-

ternatively, instead of returning a class label prediction, a classification algorithm can also return a probability distribution over the possible class categories, representing relative fit of a data point to each class.

In empirical risk minimization, classification algorithms are learned by minimizing the expectation over the training data set (X_n, Y_n), which is essentially n samples from F, of some loss objective function, denoted by l. The loss objective function is some measurement of how incorrect a classification algorithm's prediction, $\hat{f}(x)$, is in predicting the true class label, y, for some (x, y) pair. It is important to note that different loss functions will typically result in different "optimal" classification algorithms. The optimal classification algorithm for a particular loss function is:

$$\hat{f} = \arg\min_{f} \mathbb{E}_{(x,y)\in(X_n,Y_n)}[l(f(x), y)]$$

Currently, state-of-the-art classification algorithms use deep neural networks as model architectures. By selecting a model architecture a priori, we fix the class of functions that our classification algorithm can assume. Search for the function that minimizes our loss metric is then conducted as iterative optimization over the model parameters which are the network weights.

We evaluate how well our learned classification algorithm, $\hat{f}$, generalizes to the underlying distribution F by calculating generalization loss, denoted by L:

$$L(\hat{f}) = \mathbb{E}_{(x,y)\in F}[l(\hat{f}(x), y)]$$

However, we typically do not know what the exact distribution over F is like. In practice, we approximate F with the "test set", which is essentially another set of data points that are independently sampled from the same F as the original training set.

2.1.2 Adversarially-Robust Classification

We begin by formulating the adversarial threat model that we are interested in. An adversarial example of a natural example for a network is any point belonging to a certain set around the natural example which has a different label, according to the network, from the natural example. Mathematically, we look for an adversarial perturbation, denoted by δ, to our input x by finding a point $x + \delta$ that yields the greatest increase in some loss metric. We typically search for δ over a bounded set of allowed perturbations that formalizes the manipulative strength of the adversary. Intuitively, we expect large δ's to increase the loss metric more as the adversarial example is farther away from the natural example.

Specifically, Goodfellow et al. (2015) and Madry et al. (2017) both recommend the ℓ_∞-ball around the input x as a commonly used adversary to consider [3, 9]. Thus, we define our adversary for model $\hat{f}$ as:

$$\delta^*(x, y) = \arg\max_{\delta \in \mathbb{B}_\epsilon^\infty} [l(\hat{f}(x + \delta), y)]$$

where we define the ℓ_∞-ball of radius ϵ around the origin as:

$$\mathbb{B}_\epsilon^\infty = \{z \in \mathbb{R}^d : \|z\|_p \leq \epsilon\}$$

It is important to note that these adversarial examples are model dependent; in other words, it depends on the $\hat{f}$ being considered. This optimization problem is solved using a multi-step variant of the Fast Gradient Sign Method (FGSM) [3, 8], which has the update step:

$$x_{adv} = x + \epsilon * sgn(\nabla_x [l(\hat{f}(x), y)])$$

where we $sgn(\cdot)$ takes the sign of the components of the input element-wise. Note that the multi-step variant of FGSM is just projected gradient descent on the negative loss function constrained by the ℓ_∞-ball.

Madry et al. (2017) [9] discovered that an intuitive procedure to train a model that is robust to this strong adversary is to rewrite the desired optimization problem

as:

$$\min_f \Big[\mathbb{E}_{(x,y)\in(X_n,Y_n)} \max_{\delta\in\mathbb{B}_\epsilon^\infty}[l(f(x+\delta),y)]\Big]$$

This provides a meaningful interpretation of adversarial robustness. Specifically, we seek to find a classifier where the loss on a worst case adversarial example is still expected to be low.

We evaluate the adversarial robustness of a learned classification algorithm, $\hat{f}$, by calculating the adversarial generalization loss, denoted by L_{adv}, over the underlying distribution F as follows:

$$L(\hat{f}) = \mathbb{E}_{(x,y)\in F}\Big[\max_{\delta\in\mathbb{B}_\epsilon^\infty}[l(\hat{f}(x+\delta),y)]\Big]$$

Again, we calculate this metric by approximating F with the "test set", which is created by sampling a new set of points from F.

2.2 Regularization

Kannan et al. (2018) proposed a method called *Adversarial Logit Pairing* (ALP), which is a regularized form of the adversarial classification objective. Let us denote $logit(\cdot)$ as a function that maps inputs to the logits of the model, which represents a distribution over the potential class labels [7]. The proposed regularized objective function is as follows:

$$\min_f \Big\{\mathbb{E}_{(x,y)\in(X_n,Y_n)}\Big[l(f(x),y) + \lambda L_{alp}(logit(x+\delta^*), logit(x))\Big]\Big\}$$

where L_{alp} is the alp-regularization distance metric (Kannan et al. experimented with L_2 norm regularization) and δ^* is the bounded adversarial perturbation that satisfies:

$$\delta^*(x,y) = \arg\max_{\delta\in\mathbb{B}_\epsilon^\infty}\Big[l(\hat{f}(x+\delta),y)\Big]$$

which is the adversarial perturbation found by considering the ℓ_∞-bounded adversary.

This regularization technique is motivated by the idea that an input and its ad-

versarially perturbed counterpart should have similar latent embeddings. Intuitively, this form of regularization is thought to guide the machine learning model towards a more structured internal representation of the data.

Engstrom et al. (2018) [2] evaluates this implemented method and notes that the classification loss component of the objective function ($l(f(x), y)$) is evaluated on the natural input rather than the adversarially perturbed input, fundamentally changing the minimax robust optimization perspective previously described by Madry et al. Furthermore, they experimentally show that this ALP implementation may not be as robust to the ℓ_∞-bounded adversary as claimed. Despite this shortcoming, the idea of adding some form of regularization to the adversarial classification problem may still have merit.

In a recent publication, Zhang et al. (2019) [14] improved upon ALP by finding adversarial examples that maximize some logit-regularization distance metric. Furthermore, instead of using L_2 distance as the distance metric, Zhang et al. was able to achieve a better state-of-the-art adversarial accuracy by using the Kullback Leibler divergence as the distance metric. They proposed the following optimization function which they coined as TRADES:

$$\min_f \left\{ \mathbb{E}_{(x,y)\in(X_n,Y_n)} \Big[l(f(x), y) + \lambda * \text{KLDiv}(logit(x + \delta^*), logit(x)) \Big] \right\}$$

where we have:

$$\delta^*(x, y) = \arg\max_{\delta \in \mathbb{B}_\epsilon^\infty} \Big[\text{KLDiv}(logit(x + \delta^*), logit(x)) \Big]$$

2.3 Pretraining on Larger Dataset

Schmidt et al. (2018) demonstrated that adversarially-robust generalization requires more data than regular generalization [11]. Specifically, they show a trend of increasing test set accuracy as training set size increases for various standard computer vision data sets. Their findings pose the question, "how exactly does size of data set affect model robustness?" Additionally, it makes us wonder if the current state-of-the-art

is limited by data set size as opposed to carefully crafted optimization functions and training methodology.

Furthermore, He et al. (2018) summarized a common practice to deal with training machine learning models when one has insufficient data [4]. He describes a paradigm known as "pretrain-then-tune" where we pretrain the model on a much larger data set, such as ImageNet, and tune the pretrained model by training it on the data set of interest, which is usually CIFAR10 in computer vision research. The intuition for why such a method would work is that the feature representations learned during pretraining should transfer well to other tasks after some task-specific tuning. However, while other researchers have claimed that pretraining enhances the current state-of-the-art in the natural classification task, He et al. argued that it merely speeds up convergence in the learning process and that learned models that lack ImageNet pretraining may do just as well.

The conclusion of He et al. focuses almost solely on the natural classification task. In another recently published paper, Hendrycks et al. empirically evaluate the effect of pretraining on the adversarial classification task and show that, while vanilla pretraining does not greatly improve a model's natural accuracy, adversarial pretraining can greatly improve a model's adversarial robustness [6]. Their experiments were motivated by a speculative point made by Schmidt et al. [11] where a significant increase in CIFAR10 training data was hypothesized to improve the adversarial generalization accuracy of state-of-the-art adversarially-robust machine learning models. In a sense, the results of the experiments conducted by Hendrycks et al. give the intuition that adversarial pretraining methods are able to learn transferable feature representations that are also adversarially-robust.

2.4 In Practice

With the exception of pretraining, the methods described in this chapter are not necessarily trivial to implement in practice. pretraining is the most straightforward as we simply train a model on a large data set for some finite number of epochs (for the pretrain step) then continue training the same model on the task-specific data set (for the tune step).

The robust optimization objective that Madry et al. described [9] is:

$$\min_{f} \Big[\mathbb{E}_{(x,y)\in(X_n,Y_n)} \max_{\delta\in\mathbb{B}_\epsilon^\infty} [l(f(x+\delta),y)] \Big]$$

In practice, we can implement this optimization function by first creating adversarial examples for the considered training batch. Next, we treat this batch of adversarial examples as the new training batch and run back-propagation to update network weights.

TRADES is an optimization function described by Zhang et al. that has the following form [14]:

$$\min_{f} \Big\{ \mathbb{E}_{(x,y)\in(X_n,Y_n)} \Big[l(f(x),y) + \lambda * \text{KLDiv}(\mathit{logit}(x+\delta^*), \mathit{logit}(x)) \Big] \Big\}$$

where we have:

$$\delta^*(x,y) = \arg\max_{\delta\in\mathbb{B}_\epsilon^\infty} \Big[\text{KLDiv}(\mathit{logit}(x+\delta^*), \mathit{logit}(x)) \Big]$$

In practice, we again start by first creating adversarial examples for the considered training batch. However, since the optimization function relies on both natural and adversarial examples, we must use both types of examples to calculate the loss which we seek to run back-propagation to minimize.

The discussion of how to implement these theoretical optimization functions in practice is included primarily to give the reader a sense of the training procedures and experimental techniques used in this thesis.

Chapter 3

Overview of Work

In this chapter, we give an overview of the details related to our results. We present the problem setting that our models consider, the evaluation criteria of interest, and a high-level outline of what to expect in the rest of this work.

3.1 Problem Setting

We will evaluate natural and adversarial robustness of machine learning models in the context of the computer vision classification task. We use CIFAR10 as the main computer vision data set that we train and evaluate our methods on as it is widely used and studied by computer vision researchers. With regards to pretraining, we use CINIC10 as the pretrain data set as it contains 3.6× as many examples as CIFAR10 and can be adversarially trained much quicker than ImageNet.

With respect to model architecture, we choose to use ResNet18 because ResNets, in general, are able to achieve state-of-the-art computer vision performance and ResNet18s, in particular, allow for quicker training and evaluation iterations, which is essential for researching different training methods [5]. See Figure 3-1 for a visualization of the ResNet18 architecture.

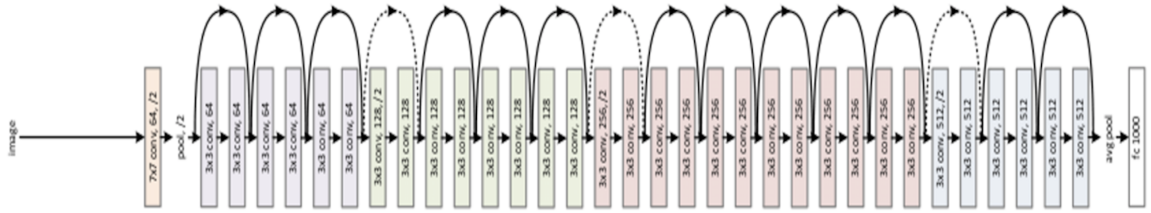

Figure 3-1: Architecture of ResNet18 [5]

3.2 Evaluation Criteria

It is important to specify the evaluation metrics in which we compare our learned models. Before doing so, we describe the adversarial threat model that we use to create adversarial examples. We consider a ℓ_∞-bounded PGD white-box adversary, a commonly used threat model in adversarial robustness research, with maximum perturbation magnitude of 0.03, which is standard for CIFAR10. For the training process, we run 10 iterations of PGD with step size 0.007. For the evaluation process, we run 20 iterations of PGD with step size 0.003. We evaluate our models on the following metrics:

- Natural Trainset Accuracy : Proportion of correctly classified data points in natural training set
- Natural Testset Accuracy : Proportion of correctly classified data points in natural testing set
- Adversarial Trainset Accuracy : Proportion of correctly classified data points in adversarially perturbed training set.
- Adversarial Testset Accuracy : Proportion of correctly classified data points in adversarially perturbed testing set.

3.3 Outline

At this point, we have discussed most of the background methodology necessary to understand the rest of the research conducted for this work. In this section, we give

an outline of the rest of this work. Chapter 4 will focus on work conducted on regularization techniques. While the work of Zhang et al. (2019) [14] will greatly overlap with some of the experiments described in this section, it is still worthwhile to present these results as many of the experiments were conducted before the publishing of their paper. Chapter 5 will focus on experiments that deal with pretraining techniques and their impact on robustness. Chapter 6 will summarize our results and present ideas for future work.

Chapter 4

Regularization

In this chapter, we present the results obtained from experiments directly related to regularization. To research the impact of different forms of regularization, we will first present the different regularized optimization functions that will be considered. We then present results of experiments that implement these proposed methods.

4.1 Theory of Regularization

Utilizing regularization in enhancing adversarial robustness is still relatively new and unexplored. The largest breakthrough in using this sort of method came at the beginning of 2019 in the form of Zhang et al.'s TRADES optimization function.

4.1.1 Robust Optimization

In the robust optimization framework, Madry et al. recommended minimizing the following optimization function over the training set [9]:

$$\mathbb{E}_{(x,y)\in(X_n,Y_n)}\Big[\max_{\delta\in\mathbb{B}_\epsilon^\infty}[l(f(x+\delta),y)]\Big]$$

While the robust optimization approach does not use regularization at all, it is included in this section because it acts as a baseline to evaluate how regularization affects adversarial robustness. We summarize this method in Algorithm 1:

Algorithm 1 Robust Optimization

Input: Mini-batch $B = \{x_1, ..., x_m\}$ of m data points. Number of PGD iterations K. Learning rate for back-propagation η_1. Step size for PGD η_2. Radius of PGD adversary ϵ.
Output: Network weights θ
for $x_i \in B$ **do**
    $x_i^{'} = x_i$
    for $j = 1, ..., K$ **do**
        $x_i^{'} = \Pi_{\mathbb{B}^{\infty}_{(x_i,\epsilon)}}\Big(\eta_2 * \text{sign}\Big(\nabla_{x_i^{'}} l(f(x_i^{'}), y_i)\Big) + x_i^{'}\Big)$ ▷ Π is the projection operator
    end for
end for
$\theta = \theta - \frac{\eta_1}{m}\sum_{i=1}^{m} \nabla_\theta (l(f(x_i^{'}), y_i))$
Return: θ

4.1.2 Regularization - TRADES Approach

The first successful use of regularization to improve the adversarial robustness of machine learning models occurred in a training procedure proposed by Zhang et al. [14] known as TRADES. The optimization problem that they seek to solve is the following:

$$\mathbb{E}_{(x,y)\in(X_n,Y_n)}\Big[l(f(x), y) + \lambda * \max_{\delta\in\mathbb{B}^{\infty}_{\epsilon}}\Big[\text{KLDiv}(logit(x+\delta), logit(x))\Big]\Big]$$

Their hypothesis for why this optimization function may outperform the standard robust optimization is that it can learn more adversarially-robust feature representations because it accurately captures the trade-off between natural and robust errors [14]. The first term encourages natural accuracy to be optimized by penalizing based on how poorly the natural example fits to the correct classification label. The second component encourages the output to be smooth by minimizing how different the model predicts natural examples with its adversarial counterpart. The *regularization parameter*, λ, determines how much each of these loss components should relatively contribute towards finding the optimal set of model weights. We outline how TRADES works in Algorithm 2:

Algorithm 2 TRADES

Input: Mini-batch $B = \{x_1, ..., x_m\}$ of m data points. Number of PGD iterations K. Learning rate for back-propagation η_1. Learning rate for PGD η_2. PGD bound ϵ. Regularization parameter λ.

Output: Network weights θ

for $x_i \in B$ **do**

  $x_i^{'} = x_i$

  for $j = 1, ..., K$ **do**

    $x_i^{'} = \Pi_{\mathbb{B}^{\infty}_{(x_i,\epsilon)}}\Big(\eta_2 * \text{sign}\Big(\nabla_{x_i^{'}} \text{KLDiv}(logit(x_i^{'}), logit(x_i))\Big) + x_i^{'}\Big)$

  end for

end for

$\theta = \theta - \frac{\eta_1}{m}\sum_{i=1}^{m} \nabla_\theta \Big(l(f(x_i), y_i) + \lambda * \text{KLDiv}(logit(x_i^{'}), logit(x_i))\Big)$

Return: θ

4.1.3 Adversarial Regularization

In analyzing the optimization function of TRADES, we note that the first component of loss function utilizes natural classification loss as opposed to some robust variant. We note that the adversarial robustness of this function may be highly sensitive to λ as we expect model trained using the corresponding objective function with small values of λ to not be robust to adversarial perturbations. This is because the first component dominates and no adversarial robustness is really introduced. As part of the research into regularization's effect on adversarial robustness, we propose an innovative optimization similar to TRADES except that the classification loss utilizes the robust variant. We coin this training method as *Adversarial Regularization*, and have the following objective function:

$$\mathbb{E}_{(x,y)\in(X_n,Y_n)}\Big\{\max_{\delta\in\mathbb{B}^{\infty}_{\epsilon}}\Big[l(f(x+\delta), y) + \lambda * \text{KLDiv}(logit(x+\delta), logit(x))\Big]\Big\}$$

We expect this optimization function to behave similarly to TRADES in the sense that, through this objective function, we are able to encode some classification power through the first term and smoothness in output through the second regularization term. However, by using adversarial classification loss, we additionally emphasize *robust* classification as the main task of interest as opposed to simple vanilla classifi-

cation. We include pseudo-code for our idea in Algorithm 3:

Algorithm 3 Adversarial Regularization

Input: Mini-batch $B = \{x_1, ..., x_m\}$ of m data points. Number of PGD iterations K. Learning rate for back-propagation η_1. Learning rate for PGD η_2. PGD bound ϵ. Regularization parameter λ.
Output: Network weights θ
for $x_i \in B$ **do**
  $x_i^{'} = x_i$
  for $j = 1, ..., K$ **do**
    $g_j = \nabla_{x_i^{'}} \Big[l(f(x_i^{'}), y_i) + \lambda * \text{KLDiv}(logit(x_i^{'}), logit(x_i)) \Big]$
    $x_i^{'} = \Pi_{\mathbb{B}^{\infty}_{(x_i, \epsilon)}} \Big(\eta_2 * \text{sign}(g_j) + x_i^{'} \Big)$
  end for
end for
$\theta = \theta - \frac{\eta_1}{m} \sum_{i=1}^{m} \nabla_{\theta} \Big(l(f(x_i^{'}), y_i) + \lambda * \text{KLDiv}(logit(x_i^{'}), logit(x_i)) \Big)$
Return: θ

4.2 Experiments

In this section, we will detail the regularization-based experiments conducted.

4.2.1 Description

Using the optimization functions described in the previous section, we trained numerous ResNet18 models with varying degrees of regularization on the CIFAR10 training set. We train each model for a total of 100 epochs with batch sizes of 100 examples per batch. We evaluate the final model (after the 100^{th} epoch of training) on the CIFAR10 train and test sets both naturally and adversarially. The adversary used to create the adversarial examples is a 20-step ℓ_∞-bounded adversary with maximum perturbation of 0.03.

Table 4.1 documents the effect of increasing the regularization parameter in the TRADES optimization function while Table 4.2 documents the effect of increasing the regularization parameter in the Adversarial Regularization optimization function. Note that for both Tables, we include the Robust Optimization model as a baseline. It is represented as the 0 regularization model in both tables.

Additionally, we plot each of the evaluation metrics with respect to size of regularization parameter for both algorithms to better visualize the strength and weaknesses of each. Figure 4-1 records the natural trainset accuracy. Figure 4-2 records the natural testset accuracy. Figure 4-3 records the adversarial trainset accuracy. Figure 4-4 records the adversarial testset accuracy. In each of these graphs, the blue plot represents models learned by using the TRADES optimization function while the red plot represents models learned by using the Adversarial Regularization optimization functions.

4.2.2 TRADES Results

The results in Table 4.1 demonstrate the ability of TRADES regularization to improve the adversarial robustness of machine learning models.

Compared to the Robust Optimization algorithm, we see that utilizing a small regularization parameter in the TRADES optimization function increases natural trainset and testing accuracies while decreasing the adversarial counterparts. This is not surprising because the TRADES optimization function utilizes natural classification loss and all of its adversarial robustness is derived from logit layer regularization or, in other words, minimizing the distance between the logit representations of natural and adversarial examples. When the regularization parameter is small, the natural classification loss dominates the optimization function and the gradient components that promote adversarial robustness are, as a result, negligible.

As we increase the regularization parameter, we see that the natural trainset and testset accuracies gradually decrease, the adversarial trainset accuracy initially increases then decreases, and the adversarial testset accuracy generally increases. The negatively-correlated trend between the natural metrics and the adversarial metrics reflect an inherent trade-off between these metrics. This phenomenon has been previously explored and discussed on separate occasions by Tsipras et al. and Zhang et al. [12, 14].

The increasing then decreasing behavior of the adversarial trainset accuracy is also not unexpected. Intuitively, as we increase the regularization parameter, the

Regularization	Nat. Train	Nat. Test	Adv. Train	Adv. Test
0.0	0.9577	0.8293	0.7628	0.4803
0.1	0.9995	0.9062	0.3724	0.3203
0.2	0.9978	0.8875	0.5317	0.4101
0.4	0.9977	0.8799	0.6152	0.4248
0.8	0.9924	0.8634	0.7177	0.4740
1.0	0.9891	0.8583	0.7135	0.4790
2.0	0.9770	0.8429	0.7143	0.4864
3.0	0.9664	0.8405	0.7158	0.5041
5.0	0.9367	0.8161	0.7032	0.5267

Table 4.1: Evaluation of TRADES

component of the loss-minimizing gradient update that conveys information about correct classification is relatively smaller. In fact, we note that the model with the best adversarial trainset accuracy is the model that was trained using vanilla robust optimization. This observation is unsurprising as the robust optimization objective function is perfectly aligned with the adversarial trainset accuracy metric. Despite this observation, we see that the TRADES trained models with high regularization parameters tend to have better adversarial testset accuracy, hinting at the idea that logit layer regularization promotes learning adversarially-robust feature representations that generalize better to out-of-sample adversarial examples than those learned by the robust optimization framework. Most remarkably, we see that the best performing TRADES-trained model outperforms robust optimization on our specified model architecture by about **4.6%** in the adversarial testset accuracy metric.

4.2.3 Adversarial Regularization Results

The results in Table 4.2 demonstrate that training machine learning models based on the Adversarial Regularization optimization function can improve the adversarial robustness of the resulting models.

Compared to the Robust Optimization training algorithm, we see that using small regularization parameters in the Adversarial Regularization framework immediately improves adversarial testset accuracy without significantly changing the other met-

Regularization	Nat. Train	Nat. Test	Adv. Train	Adv. Test
0.0	0.9577	0.8293	0.7628	0.4803
0.1	0.9580	0.8314	0.7560	0.4986
0.2	0.9713	0.8443	0.7535	0.4822
0.4	0.9634	0.8396	0.7520	0.5092
0.8	0.9543	0.8240	0.7475	0.5126
1.0	0.9474	0.8230	0.7510	0.5200
2.0	0.9224	0.8054	0.7313	0.5372
3.0	0.9237	0.8128	0.7111	0.5257
5.0	0.8646	0.7652	0.6822	0.5390

Table 4.2: Evaluation of Adversarial Regularization

rics. In contrast to the TRADES optimization function, using a small regularization parameter in the Adversarial Regularization optimization function does not harm the adversarial evaluation metrics because the Adversarial Regularization objective function utilizes the adversarial classification loss as opposed to natural classification loss. In an extreme case where the regularization parameter is taken to be very small, Adversarial Regularization behaves much like vanilla robust optimization, approaching equality as the parameter goes to 0.

Similar to the TRADES results, we see that the natural trainset and testset accuracies generally decrease as we increase the logit regularization parameter. Again, this is another instance of the inherent trade-off between natural accuracy and adversarial robustness [12, 14]. The decrease in the natural metrics is much more pronounced in this case because both components of the Adversarial Regularization objective function are biaed towards learning robust feature representations and, additionally, in direct contrast with TRADES, Adversarial Regularization has no component that directly tries to optimize for natural metrics.

With respect to the adversarial metrics, Adversarial Regularization exhibits a greater degree of robustness when compared to TRADES. In considering adversarial trainset accuracy, we observe that adversarial trainset accuracy almost monotonically decreases with respect to regularization parameter. This is expected to happen because increasing the regularization parameter in the Adversarial Regularization op-

timization function can be interpreted simply as shifting weight from the adversarial classification task, which is perfectly aligned with the adversarial trainset accuracy metric, to the regularization component. In considering adversarial testset accuracy, we observe that it generally increases with respect to increasing regularization parameter and seems to generalize better than both Robust Optimization and TRADES. In fact, on our specified model architecture, the best Adversarial Regularization based model outperforms the Robust Optimization model by **5.9%** and the best TRADES-trained model by **1.3%** in the adversarial testset accuracy metric.

4.3 TRADES vs Adversarial Regularization

In this section, we summarize the information in Table 4.1 and Table 4.2 to create plots to better visualize and compare TRADES (blue) and Adversarial Regularization (red). Furthermore, trends apparent to both algorithms may be trends inherent to logit-based regularization methods.

4.3.1 Natural Classification Results

Figure 4-1 and Figure 4-2 captures how increasing the regularization parameter affects the natural trainset and testset accuracies respectively. When the regularization parameter is small, logit-based regularization initially increases the natural metrics for both algorithms, implying that some degree of logit-based regularization provides some inherent benefit to the natural classification and generalization tasks when compared to Robust Optimization. As the regularization parameter increases, both natural metrics decline for both algorithms, reflecting the idea that weight is being moved from the classification task component of the loss to the logit distance minimization task. Finally, for both metrics, we see that TRADES outperforms Adversarial Regularization at every regularization parameter value. This result reflects the key idea that the TRADES optimization function contains a natural classification loss term and, thus, directly optimizes for the natural classification metrics.

4.3.2 Adversarial Classification Results

Figure 4-3 and Figure 4-4 captures how increasing the regularization parameter affects the adversarial trainset and testset accuracies respectively. For the adversarial trainset accuracy metric, we notice that both algorithms perform worse than models trained using Robust Optimization. This is not unexpected as the Robust Optimization objective function is perfectly aligned with the task of optimizing for the adversarial trainset accuracy metric. In analyzing the adversarial testset accuracy graphs, it is clear that logit-based regularization promotes adversarial generalization. This is evident from the apparent positively-correlated relationship between regularization parameter and adversarial test accuracy. Furthermore, at every value of regularization parameter, Adversarial Regularization consistently outperforms TRADES for the task of adversarial generalization, which is measured represented by adversarial testset accuracy. It is also able to achieve a higher adversarial testset accuracy at a lower degree of regularization.

4.4 Final Remarks

In this chapter, we presented two alternative optimization functions that can be used to train models that are robust to adversarial examples: TRADES [14] and Adversarial Regularization, both of which use logit-based regularization. From the training experiments conducted, including logit-based regularization in the training objective function promotes learning adversarially-robust feature representations that generalize well to out-of-sample adversarial examples. TRADES was developed by Zhang et al. [14] and is motivated by maintaining a balance in the trade-off between natural and adversarial accuracy. Adversarial Regularization performs the best on the adversarial generalization task.

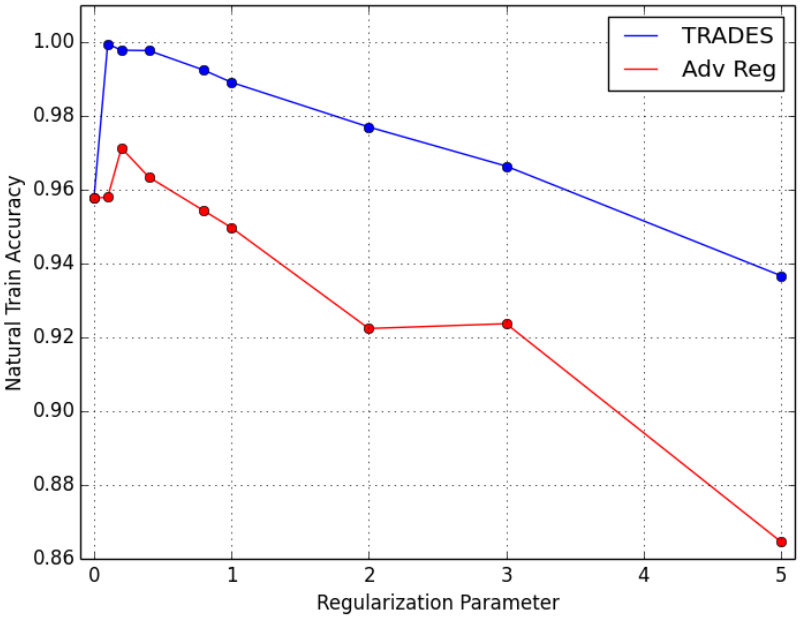

Figure 4-1: TRADES vs Adversarial Regularization: Natural Trainset Accuracy.

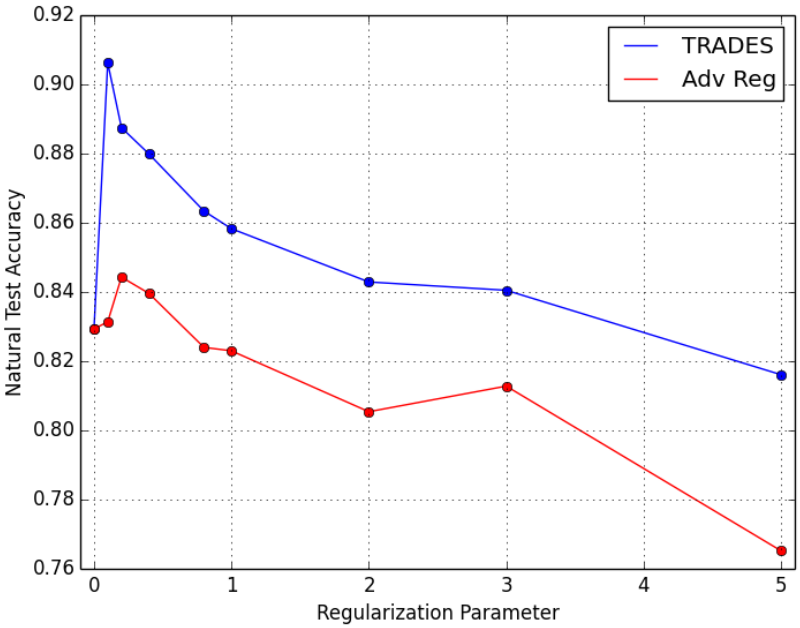

Figure 4-2: TRADES vs Adversarial Regularization: Natural Testset Accuracy.

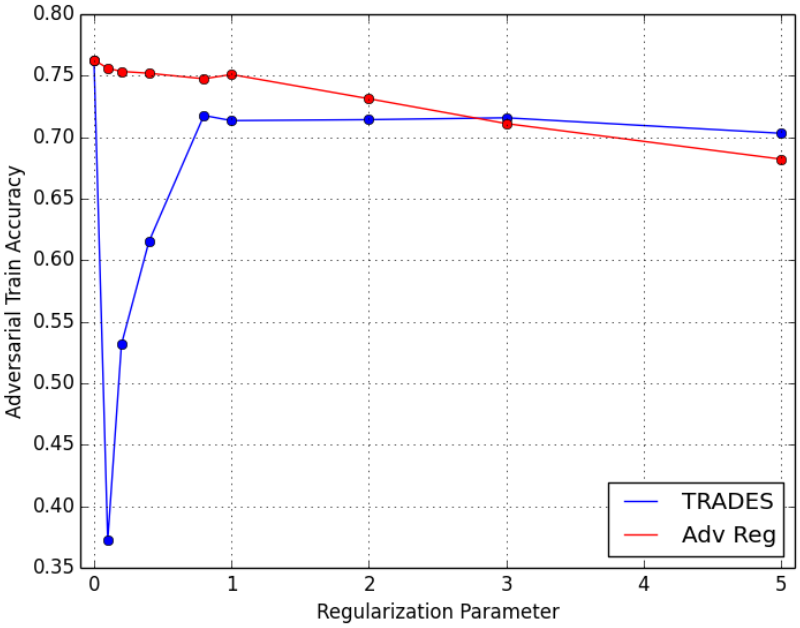

Figure 4-3: TRADES vs Adversarial Regularization: Adversarial Trainset Accuracy.

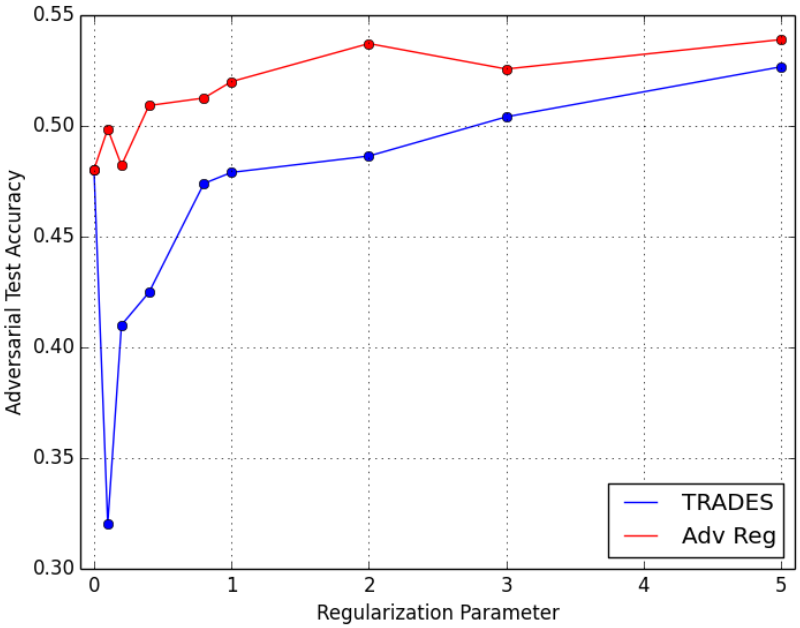

Figure 4-4: TRADES vs Adversarial Regularization: Adversarial Testset Accuracy.

Chapter 5

Pretraining

In this chapter, we present the results obtained from experiments related to pretraining. We start by giving an overview of the related work and intuition behind pretraining. Next, we discuss the experiment settings used to train and evaluate our models. Finally, we present the results achieved through our experiments. Another interesting idea that will be investigated is how we can combine pretraining with the ideas from regularization to further enhance adversarial robustness.

5.1 Related Work

Prior to 2019, the vast majority of research exploring the effects of pretraining on model performance was primarily focused on the natural classification task. Initially, researchers believed that feature representations learned after pretraining a model on a much larger dataset could transfer effectively to a specific task with the addition of domain tuning, and would enable better feature representation learning in general. The pretrain step typically has a larger learning rate than the tune step. This was experimentally observed to be misguided when He et al. (2018) showed that, while ImageNet pretraining is nevertheless a useful practice, it is not necessary to achieve state-of-the-art performance in the natural classification metrics on the CIFAR10 test set [4]. They discovered that ImageNet pretraining merely speeds up the learning process and that models trained for a sufficiently long duration from random initialization

would be able to achieve the same performance. Thus, He et al. arrived at the conclusion that the feature representations learned by the pretrain-then-tune paradigm and the feature representations learned by simply training on domain-specific data should be similar given enough training compute time and resources.

The effect of pretraining on the adversarial classification task was first explored by Hendrycks et al. (2019) [6]. In contrast to the pretraining methods used by He et al., Hendrycks et al. conducted pretraining using the robust optimization objective function (Algorithm 1). From their experiments, they showed that this form of ImageNet pretraining is able to drastically improve the adversarial robustness of CIFAR10 models.

5.2 Experiments Overview

Due to the large size of ImageNet and the computational overhead of adversarial training, we use CINIC10 as the pretraining dataset proxy for ImageNet [1]. CINIC10 contains examples from CIFAR10 as well as downsampled ImageNet examples of the categorical classes present in CIFAR10. In the train set, CINIC10 contains 180,000 examples, exactly 3.6× the size of CIFAR10.

We continue to use ResNet18 as the model architecture. We will pretrain the models on CINIC10 trainset for 100 epochs with batch sizes of 100 examples per batch. We will tune the pretrained model on CIFAR10 trainset for an additional 100 epochs. For the pretrain step, we will use learning rate 0.1. For the tune step, we will use learning rate 0.001. The adversary used to create adversarial examples and evaluate adversarial robustness is a 20-step ℓ_∞-bounded adversary with maximum perturbation of 0.03.

Even though natural pretraining does not make much of a difference on the learned model, we seek to extend the work of Hendrycks et al. and dive deeper into the effects of using different adversarially-robust pretraining and tuning methods on the learned model [6]. We have the following experiments:

1. First, we will consider how using different degrees of regularization in the TRADES optimization function for the pretrain and tune steps can enhance or harm the resulting models.

2. Secondly, we will examine how the tuning process affects the model's performance on the pretraining and tuning datasets.

3. Finally, we will investigate the effect of using a smaller pretrain dataset size on model robustness.

5.3 TRADES Pretrain/Tune

5.3.1 Description

Tables 5.1, 5.2, 5.3, 5.4 document the natural trainset accuracy, natural testset accuracy, adversarial trainset accuracy, and adversarial testset accuracy, respectively, of the tuned models after 100 epochs of pretraining and 100 epochs of tuning. In each of the tables, the first row documents the pretraining method while the first column describes the tuning method. In the tuning method column, the "None" entry refers to no tuning conducted and represents the metrics achieved by the model right after the pretraining step.

5.3.2 Results

We notice that the issues that were present in using TRADES with a low regularization parameter from Chapter 4 are still apparent in the pretraining and tuning steps. Specifically, when the regularization parameter is low, we see that the adversarial accuracy metrics are much lower. We also notice a similar trend where increasing the regularization parameter in either pretraining or tuning step generally decreases the natural accuracy metrics while increasing the adversarial accuracy metrics. Unsurprisingly, the model with the best adversarial trainset accuracy is the one where we use robust optimization for both the pretrain and tune step.

Tune/Pretrain	Rob Opt	TRADES 0.1	TRADES 0.5	TRADES 1.0	TRADES 3.0
None	0.846080	0.968800	0.934240	0.911840	0.889560
Rob Opt	0.968540	0.861000	0.965680	0.971160	0.978420
TRADES 0.1	0.999240	0.980340	0.981000	0.966020	0.994920
TRADES 0.5	0.995180	0.958880	0.975980	0.969760	0.987080
TRADES 1.0	0.984300	0.920980	0.965180	0.972480	0.979480
TRADES 3.0	0.954580	0.851400	0.936560	0.941480	0.957580
TRADES 5.0	0.934060	0.808040	0.910260	0.931080	0.937980

Table 5.1: Pretraining: Natural Trainset Accuracy

Tune/Pretrain	Rob Opt	TRADES 0.1	TRADES 0.5	TRADES 1.0	TRADES 3.0
None	0.819700	0.935800	0.898500	0.878100	0.853100
Rob Opt	0.862500	0.802800	0.873900	0.870900	0.877200
TRADES 0.1	0.908000	0.932100	0.924000	0.908500	0.914500
TRADES 0.5	0.888600	0.893800	0.905500	0.902400	0.898700
TRADES 1.0	0.874300	0.860700	0.888800	0.893400	0.881100
TRADES 3.0	0.854700	0.803600	0.862400	0.861700	0.864500
TRADES 5.0	0.840000	0.770700	0.844000	0.856200	0.854900

Table 5.2: Pretraining: Natural Testset Accuracy

Tune/Pretrain	Rob Opt	TRADES 0.1	TRADES 0.5	TRADES 1.0	TRADES 3.0
None	0.588920	0.010580	0.376040	0.483960	0.540520
Rob Opt	0.794400	0.536420	0.725320	0.760840	0.808760
TRADES 0.1	0.527100	0.054940	0.409960	0.487680	0.543220
TRADES 0.5	0.675160	0.356520	0.577300	0.607440	0.671280
TRADES 1.0	0.702340	0.447840	0.623900	0.664360	0.707180
TRADES 3.0	0.699860	0.478060	0.642000	0.671980	0.709560
TRADES 5.0	0.698380	0.470520	0.634000	0.664280	0.700140

Table 5.3: Pretraining: Adversarial Trainset Accuracy

Tune/Pretrain	Rob Opt	TRADES 0.1	TRADES 0.5	TRADES 1.0	TRADES 3.0
None	0.537900	0.009100	0.358900	0.445200	0.506900
Rob Opt	0.522000	0.450400	0.500500	0.504000	0.514300
TRADES 0.1	0.437100	0.057500	0.369700	0.437300	0.459400
TRADES 0.5	0.487900	0.322200	0.476600	0.494800	0.508500
TRADES 1.0	0.510500	0.401800	0.506700	0.519500	0.524000
TRADES 3.0	0.533300	0.438700	0.536300	0.538700	0.550300
TRADES 5.0	0.548800	0.444200	0.542500	0.544700	0.557500

Table 5.4: Pretraining: Adversarial Testset Accuracy

In comparison to the models trained in Chapter 4, the models trained using the pretrain-then-tune paradigm tend to have better performance across all of the metrics than their non-pretrained counterparts. We notice that incorporating regularization in the tune step yields a greater improvement in adversarial testset accuracy than incorporating regularization in the pretrain step. This implies that the tune step is more sensitive to logit-based regularization and that logit-based regularization in the tune step enables the model to learn feature representations that are more adversarially-robust. The model with the best adversarial generalization accuracy that we were able to train using pretrain-then-tune involved pretraining using TRADES with a regularization parameter of 3.0 and then tuning using TRADES with a regularization parameter of 5.0. It is able to achieve state-of-the-art adversarial testset accuracy that surpasses the best non-pretrained model by $\approx 3.0\%$.

5.4 Effect of Tuning on Pretraining Dataset Accuracy

5.4.1 Description

For the next set of experiments, we train 3 different models using the pretrain-then-tune paradigm. We seek to investigate how different forms of regularization in the tune step affect the pretrained models. To do this, we find the best performing models using TRADES and Adversarial Regularization in both pretrain and tune steps. We have the following models:

- Pretrained using robust optimization. Tuned using robust optimization.
- Pretrained using TRADES 3.0. Tuned using TRADES 5.0.
- Pretrained using Adversarial Regularization 1.0. Tuned using Adversarial Regularization 5.0

For each model, we save an image of the model at each epoch of training. We then evaluate each epoch of the model on the pretrain dataset (CINIC10). Figures

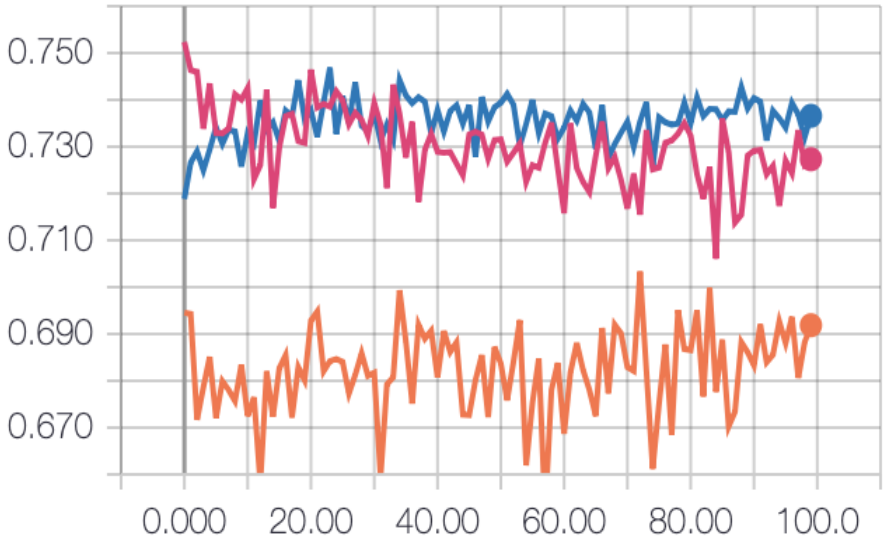

Figure 5-1: CINIC10: Natural Trainset Accuracy vs Tuning Epoch. Blue: Pretrained using robust optimization, Tuned using robust optimization. Pink: Pretrained using TRADES 3.0, Tuned using TRADES 5.0. Orange: Pretrained using Adversarial Regularization 1.0, Tuned using Adversarial Regularization 5.0

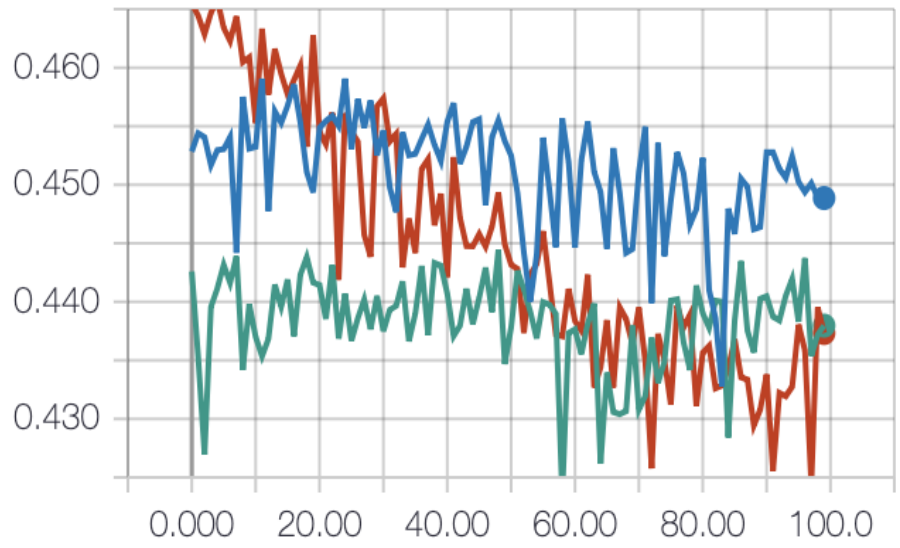

Figure 5-2: CINIC10: Adversarial Trainset Accuracy vs Tuning Epoch. Red: Pretrained using robust optimization, Tuned using robust optimization. Green: Pretrained using TRADES 3.0, Tuned using TRADES 5.0. Blue: Pretrained using Adversarial Regularization 1.0, Tuned using Adversarial Regularization 5.0

5-1 and 5-2 show the outcome of the evaluation experiment on the CINCI10 natural trainset accuracy and CINCI10 adversarial trainset accuracy metrics respectively.

5.4.2 Results

In Figure 5-1, we see that the robust optimization tuning method maintains the highest CINIC10 natural trainset accuracy with a low amount of variation. Tuning with logit-based regularization methods tend to cause greater fluctuations in the natural trainset accuracy metric. Unsurprisingly, we see TRADES perform consistently better than Adversarial Regularization in the natural trainset accuracy metrics for reasons detailed in Chapter 4. In each of the models, while there is variance, there is no directional drift in the natural trainset accuracy over tuning epoch. This implies that the feature representations learned in the tuning step are fairly consistent in the natural classification task.

In Figure 5-2, we see a more interesting phenomenon. Tuning with the robust optimization function results in a negative drift that implies decreasing CINIC10 adversarial trainset accuracy with respect to tuning epoch. The models that are tuned using some form of logit-based regularization have variance in the CINIC10 adversarial trainset metric, but no directional drift. Adversarial Regularization consistently outperforms TRADES in the CINIC10 adversarial trainset metric, which is consistent with the reasons outlined in Chapter 4. These observations imply that tuning using a logit-based regularization method learns feature representations that are consistently adversarially-robust. Conversely, tuning using the robust optimization method on CIFAR10 trainset causes some of the adversarial robustness learned in pretraining to be forgotten. An interesting point of curiosity to note is that the robust optimization model starts off with a greater degree of adversarial robustness, and that tuning to CIFAR10 causes this adversarial robustness to degrade.

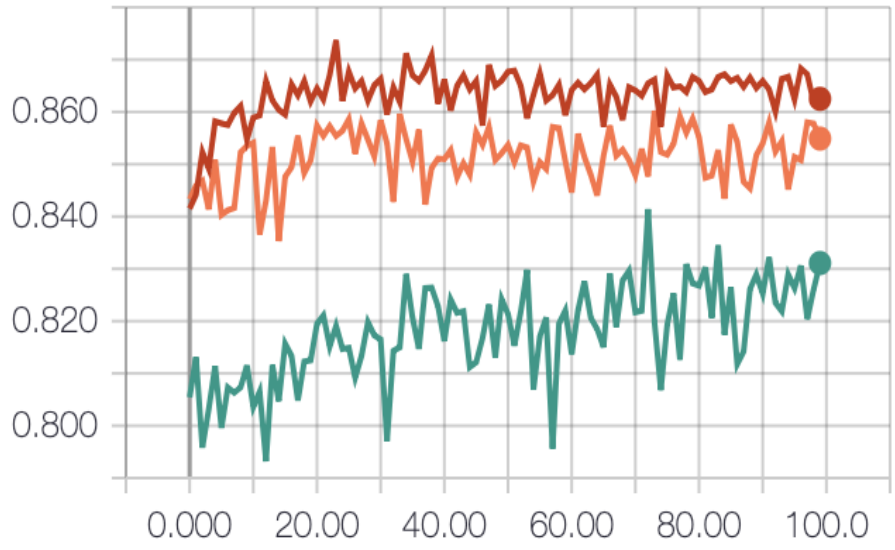

Figure 5-3: CIFAR10: Natural Testset Accuracy vs Tuning Epoch. Red: Pretrained using robust optimization, Tuned using robust optimization. Orange: Pretrained using TRADES 3.0, Tuned using TRADES 5.0. Green: Pretrained using Adversarial Regularization 1.0, Tuned using Adversarial Regularization 5.0

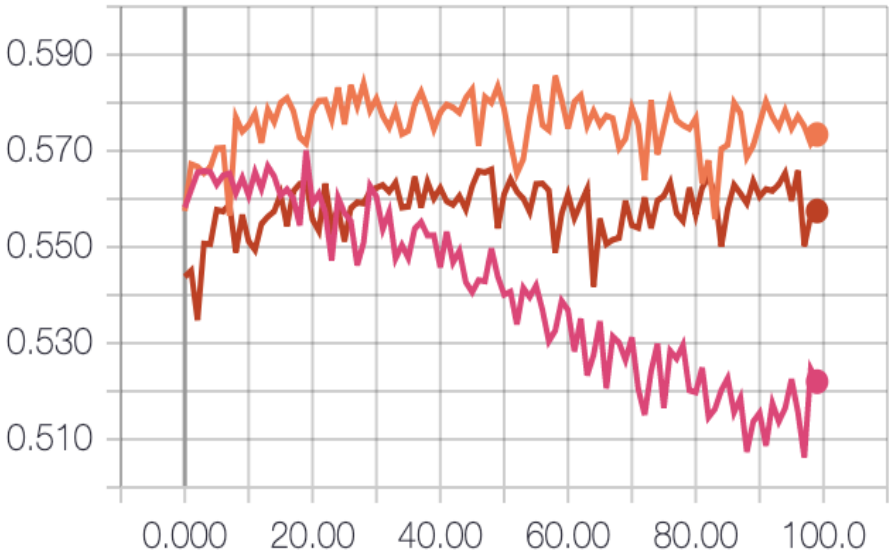

Figure 5-4: CIFAR10: Adversarial Testset Accuracy vs Tuning Epoch. Pink: Pretrained using robust optimization, Tuned using robust optimization. Red: Pretrained using TRADES 3.0, Tuned using TRADES 5.0. Orange: Pretrained using Adversarial Regularization 1.0, Tuned using Adversarial Regularization 5.0

5.5 Effect of Tuning on Generalization

5.5.1 Description

For the next set of experiments, we will use the same 3 models trained epoch-by-epoch from the previous section. We seek to investigate how different forms of regularization in the pretrain and tune steps affect the generalization capacity of the pretrained models. Using the same saved images of the training processes of each model, we evaluate each epoch of each model on the out-of-sample CIFAR10 testset. Figures 5-3 and 5-4 show the outcome of the evaluation experiment on the natural testset accuracy and adversarial testset accuracy metrics respectively.

5.5.2 Results

In Figure 5-3, we see a clear order on which models perform the best on the natural testset accuracy metric. This order is consistent with the ideas present in Chaper 4. Specifically, for models trained using higher logit-based regularization parameters, we expect the natural testset accuracy metric to decrease as the higher regularization parameter takes signal away from the classification component of the loss function. Greater amounts of robust optimization or TRADES tuning does not seem to affect the natural testset accuracy of the models in a directional manner. On the other hand, it is interesting to note that there seems to be a positive trend in increasing natural testset acccuracy with respect to tuning epoch for the Adversarial Regularization tuning method. This may indiciate that this model may have been under trained.

In Figure 5-4, we see that both logit-based regularization tuning methods initially increase the adversarial testset accuracy in the first few epochs. Afterwards, they exhibit little to no directional drift in the adversarial testset accuracy metric with respect to tuning epoch. This hints at the idea that logit-based regularization in the tune step helps pretrained models transfer to domain specific tasks while maintaining and enhancing adversarial robustness. In other words, tuning with logit-based regularization methods successfully adapts the adversarially-robust feature represen-

tations learned during pretraining to the domain specific task at hand. On the other hand, we see that using robust optimization in the tune step significantly harms adversarial testset accuracy. This may indicate that robust optimization in the tune step causes the adversarially-robust feature representations learned during pretraining to overfit to the adversarial train set and generalize poorly.

5.6 Pretraining on Smaller Dataset

5.6.1 Description

In the next set of experiments, we train 3 models, using the same algorithms as outlined in the previous two experiments, on a sample of $50,000$ CINIC10 train-set examples. The purpose of this experiment is to examine how pretrain dataset size affects the performance of the final model trained using the pretrain-then-tune paradigm. Previously, the observations made by Schmidt et al. (2018) indicate that, in the non-pretrain-then-tune paradigm, smaller dataset size is correlated with worse performance [11]. Through this experiment, we explore if a similar phenomenon occurs for the pretraining step. The results of this experiment are recorded in Table 5.5.

5.6.2 Results

Inspecting Table 5.5, unsurprisingly, we see that using a sample of $50,000$ CINIC10 examples as the pretrain dataset results in a drop in all of the performance metrics across the board. We also notice that the adversarial metrics experience considerably larger drops in performance, reaffirming Schmidt et al.'s idea that achieving adversarial robustness is a more data heavy task than achieving good natural accuracy.

Another interesting observation is that the models trained using Adversarial Regularization drops 5% when we decrease from the full dataset to the $50,000$ samples dataset. This hints at the idea that Adversarial Regularization, as an objective function, is more efficient at using additional data to learn feature representations that

generalize well to out-of-sample adversarial examples.

The results from these experiments show that achieving adversarial generalization is a data intensive task. While Schmidt et al. showed this in the non-pretrained context, the results from our experiments show that it also holds for the pretrain-then-tune paradigm.

Model Algorithm	**Metric**	**Full Dataset**	**Sample**
Rob. Opt.	Natural Training	0.968540	0.942140
Rob. Opt.	Natural Testing	0.862500	0.833700
Rob. Opt.	Adversarial Training	0.794400	0.698420
Rob. Opt.	Adversarial Testing	0.522000	0.489400
TRADES	Natural Training	0.937980	0.901660
TRADES	Natural Testing	0.854900	0.818700
TRADES	Adversarial Training	0.700140	0.626860
TRADES	Adversarial Testing	0.557500	0.522900
Adv. Reg.	Natural Training	0.902280	0.853740
Adv. Reg.	Natural Testing	0.831100	0.785700
Adv. Reg.	Adversarial Training	0.700760	0.618300
Adv. Reg.	Adversarial Testing	0.573400	0.522000

Table 5.5: Pretraining: Full vs Sample Dataset

5.7 Final Remarks

In this chapter, we presented the paradigm of pretrain-then-tune as a potential method to improve the adversarial robustness of our machine learning models. In particular, we explored the impact of using regularization in conjunction with pretrain-then-tune. We did an in-depth search over regularization parameters for pretraining and tuning with TRADES. We observed the tradeoff between natural accuracy and adversarial accuracy and found that increasing the regularization parameter for the tune step was more effective in enhancing adversarial robustness. In general, the pretrain-then-tune models outperformed their non-pretrained counterparts across all the metrics, hinting at the idea that pretrain-then-tune encourages the learning of a more adversarially-robust feature representation.

We also took a deep dive into how the tuning step differed when we use 3 different training techniques: Robust Optimization, TRADES, and Adversarial Regularization. The results from evaluating each model on the pretrain dataset demonstrates that tuning with TRADES or Adversarial Regularization maintains adversarial robustness whereas tuning with Robust Optimization causes the pretrained adversarial robustness to be slowly forgotten. Furthermore, we examine how each of these models behave over the tuning process on the adversarial generalization task. We see that tuning with TRADES or Adversarial Regularization enhances the learning of adversarially-robust feature representations. On the other hand, tuning with Robust Optimization degrades the adversarial generalizability of the pretrained feature representation.

Our final set of experiments explored how the size of pretrain dataset affects the adversarial robustness of the final model. We observed that using a smaller sample of $CINIC10$ examples as the pretrain dataset causes all of the metrics, both natural and adversarial variants, to drop for all of the models. For each model, we also saw that the adversarial metrics dropped more than the natural metrics, hinting that achieving adversarial robustness is a more data-intensive task than achieving good natural accuracy.

Chapter 6

Conclusion

In this work, we explored mechanisms to train machine learning models that are more robust to adversarial examples. Specifically, we implement and extend upon two very recently proposed approaches to enhancing adversarial robustness.

Kannan et al. (2018) first proposed a method that incorporates some form of regularization in the optimization function to potentially enhance adversarial robustness [7]. They called their method "Adversarial Logit Pairing" and was shown to not be as adversarially-robust as claimed in a follow-up paper by Engstrom et al. (2018) [2]. The first successful instance of a regularized, adversarially-robust optimization function was proposed and implemented by Zhang et al. (2019) [14]. They coined their method TRADES and showed that it successfully enhances performance on the adversarial generalization on CIFAR10 when the regularization parameter is sufficiently large.

Pretrain-then-tune is a common training paradigm where we first "pretrain" a model on a larger, general dataset (usually ImageNet) then we tune the pretrained model on a smaller, task-specific dataset. While this method was previously thought to be necessary to achieve state-of-the-art performance in the natural classification task, He et al. (2018) showed that this is not true and that pretraining simply speeds up convergence of models [5]. Hendrycks et al. (2019) was the first to apply this pretrain-then-tune paradigm to the adversarial classification task [6]. They were able to show that using an adversarially-robust pretraining algorithm to pretrain a model

does indeed enhance the adversarial robustness of the final, tuned model.

6.1 Summary of Contributions

In this section, we will summarize our specific contributions made to researching methods that enhance adversarial robustness of machine learning models:

- Proposed novel regularization method that enhances adversarial robustness more than TRADES. We call this method "Adversarial Regularization" and has the following optimization function:

$$\mathbb{E}_{(x,y)\in(X_n,Y_n)}\Big\{\max_{\delta\in\mathbb{B}_\epsilon^\infty}\Big[l(f(x+\delta),y)+\lambda*\text{KLDiv}(logit(x+\delta),logit(x))\Big]\Big\}$$

- Implemented and trained models using Robust Optimization [9], TRADES [14], and Adversarial Regularization.

- Studied how each of the methods affects adversarial robustness, concluding that Adversarial Regularization performs the best with respect to adversarial generalization.

- Implemented CINIC10 pretraining using various adversarially-robust optimization functions (Robust Optimization, TRADES, and Adversarial Regularization).

- Evaluated robustness of pretrain-then-tune paradigm coupled with logit-based regularization.

- Studied the effect of applying different tuning algorithms on the adversarial robustness of models with respect to tune epoch.

- Designed experiments to investigate how pretrain dataset size affects the adversarial robustness of the final model.

6.2 Key Takeaways

In this section, we will give an overview of the conclusions and implications that we can drawn from the research conducted:

- Logit-based regularization successfully enhances adversarial generalization. The feature representations learned through these methods generalize well to out-of-sample adversarial examples.
- Adversarial Regularization is a newly proposed regularized training algorithm that trains machine learning models that generalize better to adversarial examples than TRADES.
- Experimental results show a consistent trade-off between natural accuracy and adversarial accuracy, a result consistent with the results achieved by Tsipras et al. (2018) [12].
- CINIC10 pretraining followed by CIFAR10 tuning using adverarially robust training methods have been shown to improve adversarial robustness. Logit-based regularization methods further enhance robustness.
- Logit-based regularization methods in the tune step maintain and enhance the adversarial generalization learned by the pretrained model. Robust Optimization in the tune step causes adversarial generalization learned by the pretrained model to drop.
- Pretrain dataset size is crucial to the ability of the pretrain-then-tune paradigm to significantly enhance adversarial robustness.

6.3 Future Work

Naturally, there are two main avenues for conducting future research. With respect to regularization techniques, there are a pleathora of different ideas to try out. One could experiment with various types of distance metrics for the regularization term.

Some good follow-up candidates include total variation distance and symmetric KL-Divergence. Furthermore, a deeper dive into *how* logit-based regularization techniques affects the optimization landscape and decision boundaries learned by the models would give a greater insight into achieving a greater degree of adversarial robustness. With respect to pretrain-then-tune reserach, one cruicial follow-up experiment to conduct would be to try out various sample sizes for the pretrain dataset. In our experiments, we tried out one sample size. Having more sample sizes would better allow us to detect trends between final adversarial robustness and pretrain dataset size. Additionally, another follow-up experiment would be to use logit-based regularization methods with ImageNet pretraining with CIFAR10 classes omitted from the dataset. This type of experiment should give greater insight into how the adversarial robustness achieved by logit-based regularization generalizes.

Bibliography

[1] L. N. Darlow, E. J. Crowley, A. Antoniou, and A. J. Storkey. Cinic-10 is not imagenet or cifar-10. arXiv:1810.03505 [cs.CV].

[2] L. Engstrom, A. Ilyas, and A. Athalye. Evaluating and understanding the robustness of adversarial logit pairing. arXiv:1807.10272 [stat.ML].

[3] I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. arXiv:1412.6572 [stat.ML].

[4] K. He, R. Girshick, and P. Dollar. Rethinking imagenet pre-training. arXiv:1811.08883 [cs.CV].

[5] K. He, X. Zhang, S. Ren, and S. Sun. Deep residual learning for image recognition. arXiv:1512.03385 [cs.CV].

[6] D. Hendrycks, K. Lee, and M. Mazeika. Using pre-training can improve model robustness and uncertainty. arXiv:1901.09960 [cs.LG].

[7] H. Kannan, A. Kurakin, and I. J. Goodfellow. Adversarial logit pairing. arXiv:1803.06373 [cs.LG].

[8] A. Kurakin, I. J. Goodfellow, and S. Bengio. Adversarial machine learning at scale. arXiv:1611.01236 [cs.CV].

[9] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu. Towards deep learning models resistant to adversarial attacks. arXiv:1706.06083 [stat.ML].

[10] M. S. Ramanagopal1, C. Anderson, R. Vasudevan, and M. Johnson-Roberson. Failing to learn: Autonomously identifying perception failures for self-driving cars. arXiv:1707.00051 [cs.CV].

[11] L. Schmidt, S. Santurkar, D. Tsipras, K. Talwar, and A. Madry. Adversarially robust generalization requires more data. arXiv:1804.11285 [cs.LG].

[12] D. Tsipras, S. Santurkar, L. Engstrom, A. Turner, and A. Madry. Robustness may be at odds with accuracy. arXiv:1805.12152 [stat.ML].

[13] M. Wang and W. Deng. Deep face recognition: A survey. arXiv:1804.06655 [cs.CV].

[14] H. Zhang, Y. Yu, J. Jiao, E. P. Xing, G. EL Ghaoui, and M. I. Jordan. Theoretically principled trade-off between robustness and accuracy. arXiv:1901.08573 [cs.LG].