Enhancing Adversarial Robustness Through Model Optimization on Clean Data in Deep Neural Networks

by

Bokai Yang

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science

in

Software Engineering and Intelligent System

Department of Electrical and Computer Engineering

University of Alberta

Abstract

Adversarial robustness has emerged as a critical area in deep learning due to the increasing application of deep neural networks (DNNs) and the consequent demand for their security. Adversarial examples, which are inputs modified with imperceptible perturbations to deceive DNNs, have garnered significant attention since their introduction in 2014. Despite the importance of this issue, current defense methods against adversarial attacks either demand extensive computational resources or offer limited effectiveness, often relying on specific attack assumptions. In this thesis, we propose two novel methods to enhance the adversarial robustness of neural networks without the need for adversarial examples involving in the training process.

The first approach utilizes temperature scaling of the cross-entropy function to enhance adversarial defense against untargeted attacks. Through both theoretical analysis and empirical observations, we demonstrate that increasing the temperature during model training promotes a more balanced learning process. This adjustment helps prevent bias in optimization towards challenging samples, leading to smoother surfaces and reduced gradient updates across non-target classes. As a result, this implicit debiased optimization strategy significantly improves robustness. Our experiments confirm that training at elevated temperatures effectively defends against untargeted adversarial attacks without requiring additional computational resources.

The second approach utilizes implicit dimension reduction of input data/features and online knowledge distillation to enhance model robustness. Recent re-

search in adversarial defense has demonstrated that training networks with low-dimensional input vectors can improve robustness. However, this method typically sacrifices the model's ability to generalize well on clean data. Based on this theory, we introduce a teacher-student framework, where a teacher model trained with low-dimensional inputs obtains strong adversarial robustness and is used to guide the optimization of a student model trained with higher-dimensional inputs. This framework encourages the student model to inherit the teacher's adversarial robustness while maintaining strong generalization capabilities. Extensive experiments validate that this approach significantly improves adversarial robustness with only a small impact on generalization performance.

Overall, our research presents two innovative strategies for improving the adversarial robustness of neural networks. The temperature scaling method provides a straightforward yet effective way to bolster defenses against untargeted attacks, while the teacher-student framework offers a balanced solution to maintain generalization ability alongside enhanced robustness. These contributions advance the field of adversarial machine learning, offering practical solutions for developing more secure and reliable deep learning models.

Preface

This thesis was conducted under the supervision of Professor Xingyu Li. The content presented in Chapter 3 of this dissertation was a part of a big project conducted jointly with Huan Xuan. Our collaborative effort resulted in a paper titled "Exploring the Impact of Temperature Scaling in Softmax for Classification and Adversarial Robustness," authored by H. Xuan, B. Yang, and X. Li, which has been submitted to the Asian Conference on Machine Learning (ACML) 2024. Chapter 4 of the thesis is my independent, original work and is being prepared for submission to the Winter Conference on Applications of Computer Vision (WACV) 2024.

Acknowledgements

I would like to thank Professor Xingyu Li. Without her decisive support back then, I would not have had the opportunity to complete a thesis like this. Her professional and patient guidance has been the most important pillar in my scientific research journey. I would also like to thank all the teachers I have met over the years. The knowledge they have imparted has become the cornerstone of my scientific research.

Contents

1 Introduction **1**
- 1.1 Motivation . . . 1
- 1.2 Thesis Scope . . . 3
- 1.3 Thesis Outline . . . 4

2 Background **5**
- 2.1 Deep Learning . . . 5
- 2.2 Theories of Adversarial Robustness . . . 6
- 2.3 Adversarial Attacks . . . 11
  - 2.3.1 Taxonomy of Adversarial Perturbations . . . 12
  - 2.3.2 Adversarial Examples . . . 13
- 2.4 Defense Method . . . 15
  - 2.4.1 Adversarial Training . . . 15
  - 2.4.2 Adversarial purification . . . 17
- 2.5 Discussion & Conclusion . . . 18

3 Improve Adversarial robustness by Temperature Scaling **20**
- 3.1 Introduction . . . 20
- 3.2 Related Works . . . 21
- 3.3 Preliminary . . . 23
  - 3.3.1 Softmax Function . . . 23
  - 3.3.2 Problem Definition and Notation . . . 23
  - 3.3.3 Debias Effects of Elevated Temperature . . . 24
- 3.4 Empirical Analysis and Discussion . . . 25
  - 3.4.1 Experiment Setting . . . 25
  - 3.4.2 Experiment Results & Observations . . . 26
  - 3.4.3 Gradient Analysis for Adversarial Generation . . . 30
  - 3.4.4 Class Prototypes Analysis . . . 31
  - 3.4.5 Extending to Adversarial Training . . . 31
  - 3.4.6 Further Discussion on Adversarial Robustness . . . 33
- 3.5 Conclusion & Limitation . . . 34
- 3.6 Appendix: Complete Experimental Results . . . 34
  - 3.6.1 Experiment on Common Corruptions and Perturbations 35
  - 3.6.2 Experiment on Adversarial Attacks . . . 35

4 Low Dimension Distillation for Adversarial Robustness **43**
- 4.1 Introduction . . . 43
- 4.2 Methodology . . . 44
  - 4.2.1 LDD Architecture . . . 45
  - 4.2.2 Online Distillation . . . 47
- 4.3 Experimentation & Discussion . . . 49
  - 4.3.1 Experimental Setting . . . 49
  - 4.3.2 Experimental Results . . . 51

4.3.3 Extension: Finetune for Further Improvement 53
4.4 Conclusion 55

5 Conclusion and Future Work **56**
5.1 Conclusion 56
5.2 Future Work 57

References **59**

List of Tables

3.1 Model performance and Robustness against Common Corruptions and Adversarial attacks (%) under different temperatures with ResNet50 trained from scratch using SGD optimizer. -C in the table represents the corresponding Common Corruptions and Perturbations set. 26
3.2 Model performance and Robustness against Common Corruptions and Adversarial attacks (%) under different temperatures with Transformer. Transformer models are using Vit-small-patch16-224 as the backbone and are trained on Imagenet-21k and fine-tuned on the target dataset using Adam optimizer. -C in the table represents the corresponding Common Corruptions and Perturbations set. 27
3.3 Preliminary experiments of adversarial training on CIFAR-10 with temperature control. The training scheme uses Madry *et al*. [41] and the model is ResNet50. 33
3.4 Accuracy(%) under natural corruption CIFAR10-C dataset. Models use Resnet50 as the backbone. 36
3.5 Accuracy(%) under natural corruption CIFAR100-C dataset. Models use Resnet50 as the backbone. 37
3.6 Accuracy(%) under natural corruption Tiny-Imagenet-C dataset. Models use Resnet50 as the backbone. Attack Strength is set to 3. 38
3.7 Accuracy(%) under natural corruption CIFAR10-C dataset. Models use 'vit_small_patch16_224' as the backbone. 39
3.8 Accuracy(%) under natural corruption CIFAR100-C dataset. Models use 'vit_small_patch16_224' as the backbone. 40
3.9 Clean and Adversarial accuracy(%) under different temperatures. Models use Resnet50 as the backbone and are trained on CIFAR10 dataset using SGD optimizer. 41
3.10 Clean and Adversarial accuracy(%) under different temperatures. Models use Resnet50 as the backbone and are trained on CIFAR100 dataset using SGD optimizer. 41
3.11 Clean and Adversarial accuracy(%) under different temperatures. Models use 'vit_small_patch16_224' as the backbone and are trained on CIFAR10 dataset using Adam optimizer. 42
3.12 Clean and Adversarial accuracy(%) under different temperatures. Models use 'vit_small_patch16_224' as the backbone and are trained on CIFAR100 dataset using Adam optimizer. 42

4.1 Model performance under different dimension reduction implementations on CIFAR-100. The backbone is WideResNet. . . 51
4.2 Illustrate performance of different target dimension in Cifar100 on Resnet50. 32x32, 16x16, 8x8, 4x4 means the dimension of the input vector of teacher network. 52

4.3 Model performance under different knowledge distillation (KD) loss function 52
4.4 Model performance and Robustness against Adversarial attacks (%) 53
4.5 Model performance and Robustness against Common Corruptions with different Teacher feature space. -C in the table represents the corresponding Common Corruptions and Perturbations set. 32x32, 16x16, 8x8 means the dimension of the input vector of teacher network. 54
4.6 Illustrate performance of distillate pseudo label 54

List of Figures

2.1 Example of Adversarial Perturbation generated following *Goodfellow et al.* [23]. Even if the perturbation is hardly noticeable, it can still significantly affect the neural network. 7
2.2 Schematic of the hypothesis that create by [52]. There are many low-probability traps in feature space. Even if all samples have been clearly separated by the classifier, samples in the trap may still lead to classification errors. 9
2.3 Schematic of the Boundary tilting hypothesis [53]. The decision boundary (red line) and the submanifold of sampled data have a small angle tilting between each other. This gap makes adversarial examples possible. 11

3.1 T-SNE [55] visualization of the CIFAR10 sample distribution after the ResNet50 encoder with different temperatures. 28
3.2 T-SNE [55] visualization of the CIFAR10 sample distribution after the VIT encoder with different temperatures. 29
3.3 A demonstration of the Euclidean distance and cosine similarity between the encoded sample $f(x)$ and all class prototypes for one sample, with different temperature configurations. The red lines indicate the Euclidean distance while the blue lines stand for cosine similarity. 32
3.4 Euclidean Distance 32
3.5 Box plot of the variance of the Euclidean distance and cosine similarity calculated from each sample. The variances are calculated across all negative class prototypes, therefore, lower variance indicates a more uniform distribution of all negative class distances. Each box is a model trained with a different temperature, the green line shows the median value across all variances and the orange line is the mean value of all variances. 32

4.1 The diagram of our Low Dimension Distillation (LDD) Architecture. 44
4.2 This diagram shows the heat maps of different method trained WideResnet. The heat maps are create by the GradCAM method. The teacher network is trained with 8x8 input vectors 50

Chapter 1

Introduction

1.1 Motivation

In recent years, with the improvement of computational power and the application of large models, deep learning algorithms have gradually demonstrated their potential. Tools like ChatGPT and DALL-E 2 have already become integral parts of people's lives, bringing significant commercial success to AI companies. Even in traditional image classification, the performance of neural networks has seen substantial improvements due to the usage of large datasets. In some applications, their performance has reached and even surpassed human levels [34]. Deep learning models are increasingly being used in security-sensitive fields, such as autonomous driving. Consequently, more research is beginning to focus on the security and robustness of neural networks.

Researchers have found that adding imperceptible perturbations to input data can deceive state-of-the-art classifiers, leading them to make incorrect predictions [52]. Further research has shown that such perturbations can also affect recognition and natural language processing (NLP) tasks. These perturbations are known as **adversarial perturbations**, and samples containing such perturbations are called **adversarial examples**. Although different studies offer various explanations for the causes of adversarial examples, there is a consensus that improving neural networks' defenses against such perturbations is crucial for their future use in safety-critical applications. In adversarial defense research, two primary methods are frequently discussed: adversarial training and input purification.

Adversarial training is the most common defense method against such perturbations, first proposed by *Goodfellow et al.* [23] in 2015. It involves using adversarial examples to train neural networks. However, this approach is quite ad hoc, often designed to counter specific types of adversarial examples. When a new attack method is developed, a corresponding training method is created. Overfitting is another problem with adversarial training. It may occur due to the lack of comprehensive adversarial examples during training, resulting in a significant gap between training robustness accuracy and test robustness accuracy. The third issue with this defense method is the trade-off between standard accuracy and robustness accuracy; improving robustness accuracy usually leads to a decrease in standard accuracy. Lastly, adversarial training is time-consuming. According to this method's assumptions, all adversarial examples need to be generated on-the-fly, requiring multiple backpropagation processes for each adversarial example.

Input purification is another widely used approach to counter adversarial examples. However, most purification methods cannot achieve a "useful" level of robustness accuracy against adversarial attacks. They usually need to be combined with adversarial training to achieve reasonable performance. Time consumption can also be an issue for some adversarial purification techniques. For example, recent works using diffusion models for purification take considerable time, and this time consumption occurs during the inference phase, which is even more problematic than if it occurred during training.

The problems faced by the above two methods will greatly restrict their application in real life, especially the need for adversarial examples during the training process and the time consumption during the testing process. Thus, this thesis would answer the question:

"Can we defend adversarial attacks without involving adversarial samples in model training and heavy computation and time overhead in model deployment?"

To answer this question, we investigate temperature scaling within the generalized softmax function. We find that training vanilla models at elevated temperatures enhances their resilience against untargeted adversarial attacks.

Additionally, we develop a teacher-student online distillation approach for training robust models. This strategy encourages the student model to prioritize robust feature learning, thereby enhancing its ability to withstand attacks.

1.2 Thesis Scope

The objective of this thesis is to explore the possibility of increasing the robustness of deep neural networks without relying on adversarial samples generated based on specific hypotheses and heavy computational cost in inference. We approach this objective from two difference perspectives: temperature scaling and teaching-student online distillation. The main contributions of this thesis are as follows:

- We empirically identify the effect of temperature scaling in model optimization on adversarial robustness without sacrificing standard accuracy. We conducted an in-depth study on the "temperature," an important hyperparameter of the softmax function. Through extensive experiments, we explored how temperature affects the model training process. We found that temperature not only shapes the model's optimization process, but also influences adversarial sample generation process. Our experiments demonstrate that a high temperature helps the model resist common corruptions, natural perturbations, and non-targeted adversarial attacks. Remarkably, we found no significant trade-off between robustness and standard accuracy.

- We propose a Low Dimension Distillation Method (LDD) that significantly enhances the robust accuracy of neural networks without relying on adversarial examples or additional data during training. Previous research in adversarial defense has shown that training networks with low-dimensional input vectors can improve adversarial robustness. However, this improvement often comes at the cost of reduced generalizability of the model. Building on this insight, our approach introduces a teacher-student framework where a teacher model analyzes data in a low-dimensional

space to enhance robustness, while a student model directly processes the original high-dimensional data. In this setup, the teacher model, trained with low-dimensional inputs, guides the optimization of the student model trained on high-dimensional inputs. This distillation process ensures that the student model naturally inherits the robustness of the teacher model.

1.3 Thesis Outline

This thesis is divided into four chapters, outlined as follows:

- **Chapter 1 - Introduction** introduces the motivation behind this thesis, emphasizing the importance of adversarial robustness for neural networks. It also provides a summary of our work and contributions.
- **Chapter 2 - Background** reviews adversarial machine learning and robustness. It covers the causes of neural network vulnerabilities, theories explaining the existence of adversarial examples, and current adversarial attack and defense methods.
- **Chapter 3 - Improve Adversarial robustness by Scaling Temperature** is the first technical chapter of the thesis, specifying our effort on exploring the temperature scaling for robust classification models.
- **Chapter 4 - Low Dimension Distillation for Adversarial Robustness** is the second technical chapter of the thesis, detailing our teacher-student distillation design for robust representation learning.
- **Chapter 5 - Conclusion and Future Work** summarizes the findings of this thesis and discuss potential future research directions.

Chapter 2

Background

2.1 Deep Learning

The core objective of a machine learning model is to approximate the projections from data X to the target output Y. It uses neural networks or multilayer perceptrons (MLPs) to achieve this goal. A *deep neural network* (DNN) is one of the most renowned architectures for approximating these projection functions, $f : X \rightarrow Y$. This network can be divided into three parts: an input layer to map the input raw data to the numerical *input vector*, multiple hidden layers to project the *input vector* into the *latent feature space*, and an output layer to complete the task-specific inference based on the numerical representation in the *feature space*, for example, classifying a sample. This architecture is usually represented by the following formula:

$$f(\mathbf{x};\boldsymbol{\theta}) = \phi_l(\mathbf{W}^{(L)} \dots \phi_2(\mathbf{W}^{(2)}\phi_1(\mathbf{W}^{(1)}\mathbf{x} + \mathbf{b}^{(1)}) + \mathbf{b}^{(2)}) \cdots + \mathbf{b}^{(L)}). \quad (2.1)$$

In this formula, $\mathbf{x}$ represents the *input data*, $\boldsymbol{\theta}$ represents all the parameter variables, expressed as $\boldsymbol{\theta} = \{\mathbf{W}^{(l)}, \mathbf{b}^{(l)} : l = 1, 2 \dots L\}$. Here, $\mathbf{W}^{(l)}$ are weight matrices and $\mathbf{b}^{(l)}$ are bias vectors. ϕ_l are activation functions, such as the *rectified linear unit* (ReLU). $l \in \mathbf{L}$, where L denotes the set of hidden layers, and l specifies a particular layer within the set.

Given a training set $\{(x_n, y_n) : n = 1, 2 \dots N\}$, the process of predicting $\hat{y}_n$ from x_n is called **feedforward**. This term refers to the sequential layer-by-layer processing of information by the neural network during inference. The process of updating the parameters $\boldsymbol{\theta}$ is called **backpropagation**, where a loss function

is used to quantify the distance between the predicted value $\hat{y}_n$ and the true value y_n and then update the parameters $\boldsymbol{\theta}$ for loss function minimization through gradient descent. A widely used loss function in the classification task is the cross-entropy loss [8].

$$L_{CE} = -\sum_{n=1}^{N} y_n \log f(\mathbf{x}_n; \boldsymbol{\theta}) \tag{2.2}$$

The mean squared error (MSE) loss also performs well in classification tasks,

$$L_{MSE} = \frac{1}{\mathbf{N}} \sum_{n=1}^{N} [f(\mathbf{x}_n; \boldsymbol{\theta}) - y_n]^2. \tag{2.3}$$

However, due to the limitation of the samples $(\mathbf{x}_n, y_n)$ and the greedy property of the gradient descent based model optimization, this training process can lead to an overfitting problem, where the network learns specific features that are not necessary for classification. For example, consider a training set intended to train a neural network to classify whether there is a swan in an image. If the training set only includes images of white swans, the neural network will learn that being white is a critical feature of a swan, causing black swans to be misclassified as "not a swan." Such problems significantly reduce the generalization ability of the neural network. One consequence of this problem is the weak robustness of deep models against adversarial attacks. Figure 2.1 shows an example of adversarial attacks which is generated following *Goodfellow et al.* [23]. Even if the perturbation is hardly noticeable by human, it can still significantly affect the neural network.

2.2 Theories of Adversarial Robustness

In traditional machine learning methods, researchers manually extract specific features, which are carefully selected through empirical and theoretical methods, to project the *input vectors* to the *feature space*. In deep learning, the neural network automatically extracts features from the input images to the *latent representation*. Ideally, this process would automatically extract correct and important features from the input data, such as using the shape of a cat's ear

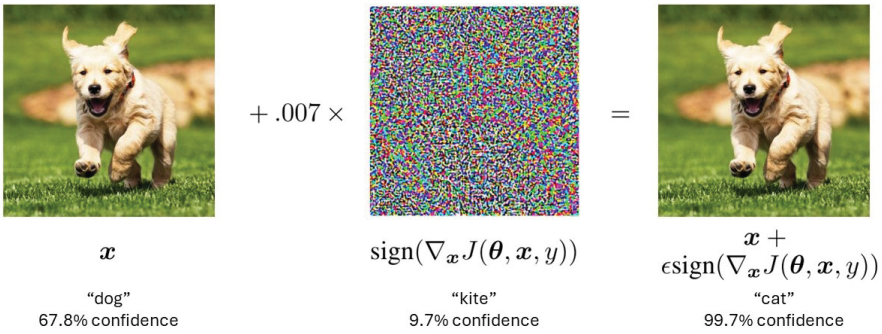


Figure 2.1: Example of Adversarial Perturbation generated following *Goodfellow et al.* [23]. Even if the perturbation is hardly noticeable, it can still significantly affect the neural network.

(feature) to predict that the picture is of a cat. However, the reality is not that simple and adding a maliciously designed, imperceptible, small perturbation to a test image can lead the neural network to make an incorrect prediction. In fundamental machine learning research, numerous studies and hypotheses have been proposed to explain the causes of deep model's vulnerability to adversarial attacks. In this section, we revisit some of the representative hypotheses.

When we look into deep neural networks, they can be seen as a collection of numerous kernel machines, as defined by **Schölkopf** [49] in 1999,

$$f(x) = b + \sum_i \alpha_i K(x_i, x_i). \tag{2.4}$$

$K(x_i, x_i)$ represents the *kernel function*, which matches the distribution between x and x_i. Most *kernel functions* can be considered as a dot product of two feature points in the feature space:

$$K(x, x_i) = \phi(x) \cdot \phi(x_i), \tag{2.5}$$

where $\phi(x)$ is a non-linear transformation that transfers x into a high-dimensional feature space. According to Equation 2.4 and Equation 2.5, we can see that the kernel function always focuses on the local distribution of x and x_i. To generalize the result from the local kernel, *Bengio* [8] proposed an assumption called the *smoothness prior*, which states: *"the target function is smooth or can be well approximated with a smooth function."* [8]. A good feature space can also be derived from this assumption. It needs to be able to smoothly express

the target space. This assumption is challenging to achieve because all the training data in the training set are finite, discrete points in the feature space. There may not be enough to construct a continuous surface of the feature space distribution [52].For example, in some areas of high-dimensional space, the surface may change drastically, but if the training data do not include samples from these areas, the plane may be smoothed incorrectly, making the neural network vulnerable. This problem is difficult to solve because the surface of the high-dimensional distribution is hard to visualize, making it challenging to determine which specific images should be added to the training set to interpolate the missing information.

Further studies also suggest that the **Shrinking of the Interior** can be considered an important reason for neural network vulnerability [30]. This property indicates that as the dimensionality of the space increases, the points within the distribution tend to get closer to the surface. This phenomenon can be expressed by the following inequality [11]:

$$\frac{volume((1-\epsilon)A)}{volume(A)} = (1-\epsilon)^d \leq e^{-\epsilon d} \tag{2.6}$$

In Equation 2.6, let A be an object in R^d. ϵ is a small factor that transforms A into another object. $volume((1-\epsilon)A)$ represents the volume of the interior region of object A, while $volume(A)$ represents the volume near the surface of object A. The factor $(1-\theta)^d$ affects the rate of shrinkage. If we fix θ and increase the dimension d, the fraction will gradually decrease. As a result, the points within object A will gradually move closer to the surface. If the dimension d approaches infinity, the fraction will approximately equal zero, meaning almost all points in object A will be near the surface, leaving the interior space close to zero. This phenomenon is common and fundamental for distributions in Euclidean space [18]. *Gregory* [30] provided numerical statistics on the proportion of points near the surface. The data shows that when the dimension d is equal to or greater than 128, nearly 100% of the points are near the surface. For most neural network classifiers, such as CNNs, the dimension of the *feature space* is far greater than 128. Therefore, the high-dimensional *feature space* itself contributes to and increases the vulnerability of neural

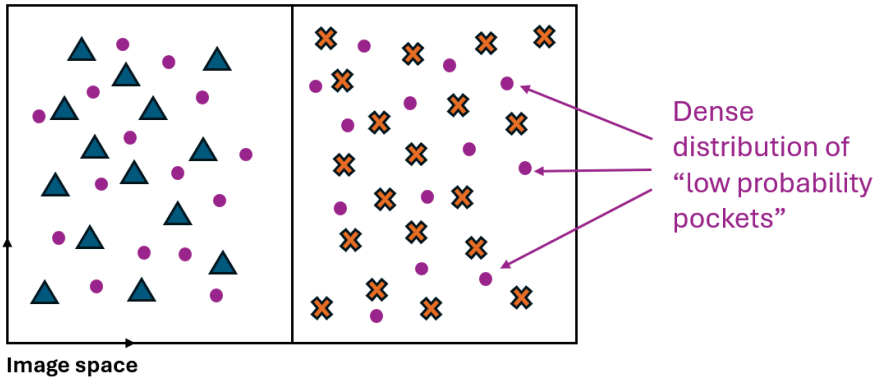


Figure 2.2: Schematic of the hypothesis that create by [52]. There are many low-probability traps in feature space. Even if all samples have been clearly separated by the classifier, samples in the trap may still lead to classification errors.

networks.

The term **adversarial example** was coined by *Szegedy et al.* [52] to describe input examples that have been modified with imperceptible perturbations, as mentioned in the previous section, that can deceive deep neural networks into making incorrect predictions. It is assumed that the number of adversarial examples is extremely low compared to the number of other examples; therefore, they are hard to find in the test set. However, even though the relative number of adversarial examples is small, the absolute number is still significant. Thus, they can be found using specially designed search methods. Figure 2.2 schematically represents this assumption. In this paper, Szegedy also provides an interesting example of this assumption: consider a classifier C designed to distinguish whether a real number x belongs to the category of "positive irrational or negative rational" numbers, or "negative irrational or positive rational" numbers. If the training set is randomly selected from real numbers, the classifier will easily learn to separate positive numbers from negative numbers without considering whether a number is rational or not. This is because the training set is likely to be predominantly composed of irrational numbers, given the relative abundance of irrational numbers compared to rational numbers. However, this classifier can be easily fooled by adversarial examples, as rational numbers close to the irrational input x are easy to find

among real numbers.

Goodfellow et al. [23] found that this kind of vulnerability not only occurs in deep neural networks but also in linear classifiers. Based on this observation, they assumed that this vulnerability might not be caused by "low-probability pockets." Instead, it may be due to small perturbations whose effects are magnified by the linear superposition resulting from the increase in dimensionality. This hypothesis is called the **linear hypothesis**, and they proved this hypothesis by building a shallow linear neural network. For the simplest linear neural network, it can be represented as a dot product between a weight vector w and an input x:

$$y = w^T x. \tag{2.7}$$

The adversarial example can be formulated as $\tilde{x} = x + \eta$, and its output can be represented as $\tilde{y} = w^T \tilde{x}$. In this context, η represents the adversarial perturbation, and it is constrained by $\|\eta\|_{\text{inf}} < \epsilon$. Here, ϵ is the regularization term that defines the maximum amount of perturbation to ensure that the perturbation remains imperceptible. Therefore, we can express the distance between the standard and adversarial output as follows:

$$Dis(y) = \tilde{y} - y = w^T \eta. \tag{2.8}$$

According to this equation, we can see that the distance is positively related to the perturbation η and the weight vector w. To maximize this distance, we can set η to its constraint ϵ by assigning $\eta = sign(w)$. The weight vector w can be formed by the dot product of a dimension vector n and a magnitude vector m. Therefore, this distance in Equation 2.8 can be formulated as:

$$Dis(y) = \epsilon mn \tag{2.9}$$

Since ϵ is a hyper-parameter, and the magnitude vector m and dimensions are independent of each other, the expression can be simplified to:

$$Dis(y) \propto n \tag{2.10}$$

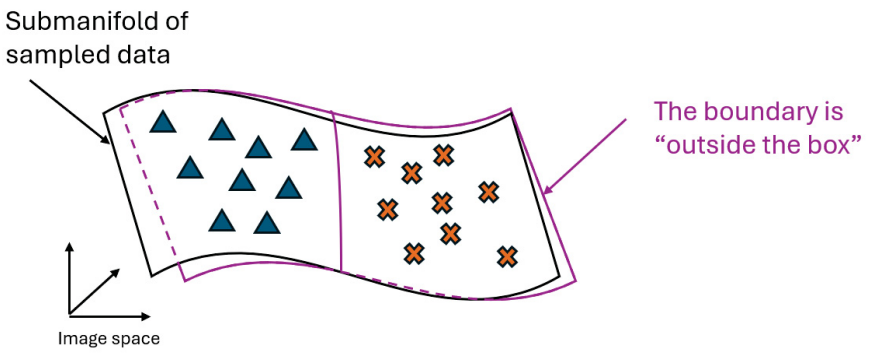


Figure 2.3: Schematic of the Boundary tilting hypothesis [53]. The decision boundary (red line) and the submanifold of sampled data have a small angle tilting between each other. This gap makes adversarial examples possible.

According to this Equation 2.10, we can see that the distance will grow linearly with the dimension n. Even though the perturbation is small, it can still cause a significant change in the output if the dimension of the weight vector is large enough.

The **Boundary Tilting Hypothesis** is another important hypothesis for analyzing the vulnerability of models, proposed by *Tanay et al.* [53]. They believe that the cause of adversarial examples is the classifier's decision boundary extending beyond the submanifold of input data. As shown in the Figure 2.3, there is a small gap between the submanifold of sampled data and the classifier's decision boundary (red line). Adversarial examples are the points not on the submanifold surface but close to it. *Khoury et al.* [31] also mathematically proved this idea using geometric analysis. *Tanay et al.* also claim that in low dimensions, an input image perturbed by a random perturbation is likely to cross to the different side of the class boundary. However, in high-dimensional space, a random perturbation is less likely to guide the point along the boundary, thus providing robustness to random perturbations, which is consistent with the results in [52].

2.3 Adversarial Attacks

In Section 2.2, we theoretically analyzed the possible causes of the existence of adversarial examples from different perspectives. In this section, we will review

specific methods to generate adversarial attacks.

2.3.1 Taxonomy of Adversarial Perturbations

After *Szegedy et al.* [52] proposed adversarial perturbations, researchers conducted extensive studies on methods to search such adversarial perturbations. As the research progressed, different angles of attacks on a model emerged.

The first and most direct distinction of attacks to deep models is the ability to access the training data. Attacks with access to training data are called **data poisoning attacks** or **training time attacks**. Depending on the level of access to the training data, this attack method is divided into the following three categories:

1. **Data Modification Attacks**: These have the maximum access permission, allowing modification of both the training sample's feature vectors and their labels.

2. **Insertion Attacks**: These attacks can only insert perturbed examples without changing the data in the original training set.

3. **Label Modification Attacks**: These can only modify some labels in the training set.

Attacks that can only access a trained model without touching the training data are called **decision-time** or **test-time attacks**. These attacks aim to find the weaknesses of a classifier and craft a feature vector to fool it at decision time. *Adversarial examples* are one such attack, which limits the magnitude of perturbations.

In the attack methods involving *adversarial examples*, based on the amount of information about the victim model obtained by the attackers, these can be divided into **white-box attacks** and **black-box attacks**. In a **white-box attack**, the attacker knows full details about the model's internals, such as model parameters, random seeds, and even the training strategy. In a **black-box attack**, under the original assumption, attackers can only access an API to query the target model [44]. In subsequent research, *Assion et al.* [5] further

divided it into four subcategories based on different levels of access to model output information: *output transparent black-box, query-limited black-box, label-only black-box,* and *full black-box.* In some work, black-box attackers are also allowed to know the model's architecture to enhance the attack.

Targeted and *untargeted* are another dimension (attack **goals**) to distinguish attack methods. In a targeted attack, the objective is to misclassify the output into a specific class. In contrast, an untargeted attack aims to craft a perturbation to maximize the classification loss, without concern for the specific outcome.

2.3.2 Adversarial Examples

In this section, we will overview some representative methods to generate white-box adversarial examples that are closed related to this dissertation. Black-box attacks can be generated using a surrogate models with these algorithms.

- **Limited Memory Broyden–Fletcher–Goldfarb–Shanno (L-BFGS) Algorithm** [52] is the most basic optimization-based method to find adversarial examples, introduced by *Szegedy et al.* in 2014. They proposed that perturbations with desired properties could be found by optimizing an objective function. To achieve this, the solution to the following box-constrained minimization problem needs to be found:

$$\min_{\epsilon} \|x - x'\|_2, \quad \text{subject to } f(x') = y', x' \in [0,1]^D \tag{2.11}$$

  where $f(\cdot)$ is the classifier, y' represents the incorrect label. x is the input example, x' is the adversarial example, and ϵ is the difference between x and x'

- **Carlini-Wagner ($C\&W$) Attack** [9] is an alternative method to the **L-BFGS attack**. *Carlini and Wagner* pointed out that the main weakness of the L-BFGS method is that the optimization problem is hard to solve. Therefore, they proposed using a penalty function to replace that constraint. They introduced their L^p attack by optimizing the following

problem:

$$\min_{w} \quad \|x - \frac{1}{2}(1 + tanh\xi)\|_p^2 + \alpha \cdot f(\frac{1}{2}(1 + tanh\xi)), \tag{2.12}$$

where α controls the impact of the objectives on the problem, and the function $f(\cdot)$ is defined as:

$$f_c(x') = y' \iff f(x') \leq 0. \tag{2.13}$$

Here, f_c is a classifier. They argued that using the unconstrained variable ξ is a better choice to achieve this objective as it is much easier to solve and can still produce effective results.

- **Fast Gradient Sign Method (FGSM)** [23] is one of the most significant methods in adversarial attacks. It was the first useful feature-based attack proposed by *Goodfellow et al.*, based on their linearity hypothesis. They used an equation to express the perturbation:

$$\delta = \epsilon sgn\nabla_x J(\theta; x, y), \tag{2.14}$$

where θ represents the parameters of the classifier, $J(\theta; x, y)$ is the loss function for the classifier, and ϵ is a hyper-parameter that defines the magnitude of the perturbation. This method is a simple but effective technique for finding adversarial examples, requiring only a single step of backpropagation to craft an adversarial example with a relatively high success rate.

- **Projected Gradient Descent (PGD)** [41] is an iterative version of the FGSM algorithm. It transforms FGSM's simple optimization function into an iterative optimization function. By applying repeated small-perturbation optimizations, PGD can find better adversarial examples with a higher success rate. The recurrence equation of this optimization process can be formulated as:

$$x'_{(t+1)} = x'_{(t)} + \epsilon sgn\nabla_x J(\theta; x, y). \tag{2.15}$$

For the first iteration, a random perturbation is applied to the input x.

- **Auto-PGD (APGD) Algorithm** [15] is an automatic scheme designed to address the problems faced by the PGD algorithm. Since the PGD algorithm can only use a fixed step size to update the iteration function, the performance of the iteration process depends heavily on hyperparameter settings, and, like other training processes, a fixed step size may not be the best option for finding the optimal convergence point. APGD starts the search with a larger step size, and when the trend of convergence begins to level off, it switches to a smaller step size to continue the search until the loss function can no longer converge. This method balances the effective optimization speed brought by large step sizes with the advantages of local optimization provided by small step sizes.

- **Square Attack** [4] is a notable example of a black-box attack. It employs a random search method to sample a random perturbation δ. The algorithm then determines whether this perturbation can improve the objective function. If it can, the perturbation is added to the current adversarial sample x'. This is a score-based attack method that does not rely on the gradient of the victim model. Instead, it uses a random search method to achieve a high success rate, which is comparable to that of white-box attacks.

2.4 Defense Method

Since "adversarial examples" were proposed, researchers have tried various methods to defend against this type of attacks. Adversarial training and input purification are methods that have achieved considerable results and are widely used. In this section, we will review these two methods.

2.4.1 Adversarial Training

The idea of adversarial training was proposed by *Szegedy et al.* [52] as well. They pointed out that training a network with a mixed dataset containing both adversarial and clean examples could enhance the network's adversarial

robustness. The following year, *Goodfellow et al.* proposed a regularization-based adversarial training method that adds a regularization term to the cross-entropy loss function to improve adversarial robustness. The loss function can be written as follows:

$$L(\theta) = \alpha J(\theta, x, y) + (1 - \alpha) J(\theta, x + \epsilon sign(\nabla_x J(\theta, x, y), y)), \tag{2.16}$$

where α is a parameter that controls the degree of attention the network pays to adversarial examples, and J is the cross-entropy loss. Goodfellow et al. found that this method can, to some extent, defend against the FGSM attack. *Kurakin et al.* [37] extended this method to ImageNet by modifying the ratio of adversarial examples in the mini-batch.

Madry et al. [41] further optimized the problem. They believe adversarial training can be described as a saddle point (min-max) optimization problem. They used the PGD method to iteratively find the maximum adversarial examples and used the cross-entropy loss to achieve the best prediction on them. It can be formulated as follows:

$$L(\theta) = min \mathbb{E}_{(x,y)\sim D} [\max_{\delta \in S} J(x_{pgd}, y; \theta)]. \tag{2.17}$$

Zhang et al. [63] pointed out that robust error can be decomposed into the sum of natural error and boundary error. They believe boundary error is caused by the small distance between data points and the decision boundary. Therefore, they proposed the TRADES method to minimize the error by solving the following problem:

$$\min_f \mathbb{E} L(f(x), y) + \max_{x' \in \mathbb{B}(x,\epsilon)} L(f(x), f(x'))/\lambda, \tag{2.18}$$

where λ is a coefficient to control the strength of regularization. Later studies on adversarial training have mostly focused on using data augmentation to improve performance [59], or making small optimizations to the loss function.

It should be noted that through adversarial training shows promising results, it suffers from several drawbacks. The most important issue is its 'ad hoc' nature. Adversarial training methods are always based on specific attack

methods and use adversarial examples generated by these attacks to reinforce weak points. The problem is that most current attack methods are directly or indirectly based on the hypotheses from *Goodfellow et al.* [23] and *Madry et al.* [41], who proposed the adversarial training method. Therefore, the robustness performance achieved by adversarial training might be due to the limitations of the attack methods themselves.

So far, there are no attack methods based on newer theories [17], [20], [21], [31], [53], [54]. However, this remains the sword of Damocles hanging over the adversarial method. Even under the limited attacks currently available, the performance of adversarial training is still not satisfactory. The trade-off between clean accuracy and adversarial accuracy is unavoidable in the adversarial training process. Most of these methods suffer from serious overfitting problems. To mitigate this issue, some state-of-the-art methods attempt to introduce additional data. However, there are challenges with this solution. For toy datasets, such as CIFAR10, it is easy to find useful additional data, but for large datasets, such as ImageNet, finding additional data becomes a significant challenge.

Time consumption is another problem for adversarial training methods. Adversarial training requires 'real-time' generated adversarial examples for the training process, and generating each adversarial example requires multiple iterations of backpropagation, which consumes a lot of time.

2.4.2 Adversarial purification

Adversarial purification can be considered as noise removal before feeding the query data to the pre-trained model. To filter out or mitigate the effects of adversarial perturbations on model predictions, it usually requires clean data and their adversarial counterparts to build a denoising model. Defense-GAN is the first input purification method proposed by *Samangouei et al.* [48], which builds a generator to produce clean images from attacked images. Subsequent work has used different methods to process input images, such as hand-crafted techniques [3], learned objectives [47], and auxiliary networks [62]. The state-of-the-art method uses diffusion models to purify input images [40]. This

method achieves almost the same performance as the latest adversarial training methods [66]. It should be noted that one biggest problem with this method is time consumption, and unlike adversarial training, this time consumption is added to the evaluation process.

2.5 Discussion & Conclusion

With the 'arms race' over the past few years in the field of adversarial robustness, increasingly powerful and efficient attacks are being proposed. However, in the area of defense, there is still a lack of effective means.

Adversarial training, proposed by Goodfellow et al. in 2015, involves training neural networks with adversarial examples to improve robustness. Despite its widespread adoption, several challenges persist. Firstly, the approach is often reactive, tailored to specific attack types, which necessitates continual adaptation as new attack methods emerge. Secondly, overfitting can occur due to the limited diversity of adversarial examples used during training, leading to a disparity between robustness accuracy during training and testing phases. Thirdly, there exists a well-known trade-off between standard accuracy and robustness accuracy; while improving robustness, the model's performance on clean data may deteriorate. Lastly, the computational overhead is considerable, as generating adversarial examples on-the-fly and performing multiple backpropagation steps per example significantly increases training time and complexity.

Input purification methods aim to preprocess inputs to remove adversarial perturbations or make them less effective. However, these techniques often fall short in achieving robustness without supplementary adversarial training. Many purification methods struggle to attain sufficiently high levels of robustness accuracy on their own and typically require reinforcement from adversarial training to yield satisfactory results. Moreover, certain purification approaches, such as those utilizing diffusion models, introduce significant computational overhead during the inference phase, posing challenges for real-time applications.

In conclusion, while adversarial training and input purification represent

important steps forward in defending against adversarial attacks, their limitations underscore the need for further research and development. Future efforts could focus on mitigating overfitting, reducing the trade-offs between accuracy metrics, and optimizing computational efficiency. By addressing these challenges, advancements can be made towards achieving more robust and reliable neural network defenses against adversarial threats in practical settings. Therefore, in this dissertation, we aim to explore feasible methods for enhancing the model's adversarial robustness without incorporating adversarial examples during model training or introducing additional computational overhead during inference.

Chapter 3

Improve Adversarial robustness by Temperature Scaling

3.1 Introduction

Deep learning has had dramatic breakthroughs in recent years with tasks like image classification [33], nature language processing [16], and semantic segmentation [65], achieving great success. One crucial component utilized by most deep learning methods is the standard softmax function which normalizes a set of real values into probabilities. A generalized softmax function incorporates a parameter, typically referred to as "temperature", which controls the degree of softness in the output distribution. While the temperature scaling factor has been extensively explored across various tasks such as knowledge distillation [27], contrastive learning [56], confidence calibration [45], and natural language processing, there has been limited investigation into its impact on adversarial robustness [1]. Particularly, in contrastive learning, low temperature has been shown to be beneficial by placing greater emphasis on the learning of hard samples [56]. Since the cross-entropy loss can be expressed in a manner similar to contrastive loss: measuring similarity between two terms, we became curious about whether the temperature in the softmax function plays a crucial role in model's adversarial robustness.

In this empirical study, we surprisingly find that models trained with high temperatures show tremendous robustness against non-targeted adversarial attacks such as Projected Gradient Descent (PGD). We verify this phenomenon on

CIFAR10, CIFAR100, and Tiny-Imagenet [39] using both CNN and Transformer architectures. Through our analysis, we determined that higher temperatures ultimately lead to models with a smoother loss surface where the gradient with respect to the input data is very small, leading to inefficient adversarial sample generation. However, this robustness improvement doesn't exist for targeted attacks. Lastly, we show preliminary to extend our findings regarding temperature into adversarial training [41] and find that applying temperature control to adversarial training can indeed regularize the model performance and elevated temperatures have the potential to boost the model's overall robustness.

3.2 Related Works

The softmax function has been a longstanding component of neural networks, usually used to normalize a vector of real values into probabilities. Modulating the temperature scaling factor within the softmax function allows for reshaping the probability distribution. This section provides a concise overview of the application of temperature scaling in various computational tasks.

Knowledge Distillation proposed by Hinton *et al.* [27] is one innovative way to transfer knowledge from a teacher model to a student model. Temperature is utilized during training to control both the student and teacher model's output. The author argues that lower temperatures make the distillation assign less weight to logits that are much smaller than the average. Conversely, employing larger temperatures softens the probability distribution and pays more attention to the unimportant part of the logit. Larger temperatures are proven to be beneficial in the distillation process since the hard-target term already ensures the dominant part of the logit (target class) is correct. By focusing on the remaining logit, the student model can capture more fine-grained information from the teacher model. Note that despite various temperatures used during training, it is set to 1 when the model is deployed.

Model Confidence Calibration [24], [38], [42] usually utilizes temperature scaling to address the over-confident issue in deep learning. It centers

on estimating predictive uncertainty to match its expected accuracy. Despite multiple generic calibration methods being proposed [35], [36], temperature scaling used by Guo *et al.* [24] remains a baseline method for being simple, effective and able to apply to various cases without major expense. The motivation behind temperature scaling is simple, since the goal is to control the network's confidence to match its accuracy, applying temperature to the softmax function that can directly modify the probability distribution seems a perfect fit for the problem. During training, a validation set is needed to find the ideal temperature parameter for the network, and the same temperature is used when deployed.

Contrastive Learning [43], [60] is one paradigm for unsupervised learning. To achieve a powerful feature encoder, it utilizes contrastive loss to pull similar samples close and push negative pairs away in the latent space:

$$L(x_i) = -\log\left[\frac{\exp(s_{i,i}/\tau)}{\sum_j \exp(s_{i,j}/\tau)}\right] \tag{3.1}$$

where $s_{i,j}$ is the similarity between x_i and x_j. Although the temperature control parameter has long existed as a hyper-parameter in contrastive loss, its actual mechanism has been relatively understudied. Wang and Liu [56] analyze the contrastive loss closely and find that as the temperature decreases, the distribution of the contrastive loss becomes sharper, which applies larger penalties to samples similar to x_i. Also, uniformity [58] of feature distribution increases, indicating the embedding feature distribution aligns with a uniform distribution better.

Temperature Scaling in Image Classification has occasionally been utilized in the experimental sections of prior studies [2], [19], [29], yet focused investigations on this subject remain limited. For example, certain studies aiming to improve adversarial robustness have utilized temperature scaling to adjust logits within their experimentation. However, these studies often integrate additional complex techniques such as Gaussian noise injection [2], adversarial training [19], [46], and innovative quadratic activation functions [29], making it challenging to isolate and understand the specific contribution of temperature scaling to the overall system performance. In contrast, our study

narrows its focus to investigating the direct impact of temperature scaling applied through the softmax function on model adversarial robustness. Among the few related works, "The Temperature Check" [1] is notably relevant to our discussion. It mainly explores the dynamics of model training by considering factors such as temperature, learning rate, and time, and presents an empirical finding that a model's generalization performance is significantly influenced by temperature settings. While our observations and analysis align with these findings, our study broadens the scope of inquiry by assessing the effect of temperature scaling on a model's resilience to common corruptions and adversarial attacks, thereby adding a new dimension to the existing research.

3.3 Preliminary

3.3.1 Softmax Function

Given a set of real numbers, $X = \{x_1, .., x_N\}$, the generalized softmax function can be used to normalize $\mathcal{X}$ into a probability distribution.

$$\mathbb{S}(X) = \frac{\exp(X/\tau)}{\sum_i \exp(x_i/\tau)}, \tag{3.2}$$

where $\mathbb{S}$ represents the softmax function and τ is the temperature scaling factor. The temperature τ controls the smoothness (softness) of the probability it produces. Specifically, when $\tau \to \infty$, the output tends toward a uniform distribution; while when $\tau = 0$, the softmax function assigns a probability of 1 to the element with the highest value and a probability of 0 to the rest. The standard (unit) softmax function, with $\tau = 1$, is widely used in conventional classification tasks.

3.3.2 Problem Definition and Notation

We consider multi-category classification in this study, where paired training data $\{\mathcal{X}, \mathcal{Y}\} = \{(x, y)|x \in \mathbb{R}^{H\times L\times N}, y \in \mathbb{R}^{1\times M}\}$ are drawn from a data distribution $\mathcal{D}$. Here, H, L, N are the dimension of a sample x, M is the number of categories, and y is a one-hot vector indicating the class of the input x. A classifier, $C : \mathcal{X} \to \mathcal{Y}$, is a function predicting the label y for a given data

x. That is $C(x) = y$. In the canonical classification setting, a neural network classifier, $C = (f, W)$, is usually composed of a feature extractor f parameterized by θ and a weight matrix W. f is a function mapping the input x to a real-valued vector $f(x)$ in the model's penultimate layer and $W = (w_1, ..., w_M)$ represents the coefficients of the last linear layer before the softmax layer. So the likelihood probability of data x corresponding to the M categories can be formulated as

$$\hat{y} = C(x) = \mathbb{S}(W^T f(x)). \tag{3.3}$$

Note that each vector w_i in matrix W can be considered as the prototype of class i and the production $W^T f(x)$ in Equation (3.3) quantifies the similarity between the feature $f(x)$ and different class-prototypes.

During training, the model $C = (f, W)$ is optimized to minimize a specific loss, usually a Cross-Entropy loss.

$$L_{ce}(x) = -y \log \hat{y} = -\log \left[\frac{\exp(w_i^T \cdot f(x)/\tau)}{\sum_{j=1}^{N} \exp(w_j^T \cdot f(x)/\tau)} \right] \tag{3.4}$$

When considering all N samples in one batch B_N, the compound loss of the N samples are

$$L_{ce}^{N}(x) = -\frac{1}{N} \sum_{x \in B_N} \log \left[\frac{\exp(w_i^T \cdot f(x)/\tau)}{\sum_{j=1}^{N} \exp(w_j^T \cdot f(x)/\tau)} \right]. \tag{3.5}$$

Though $\tau = 1$ is the default setting in classification tasks, we preserve τ in the equations to facilitate theoretical analysis in this section.

3.3.3 Debias Effects of Elevated Temperature

The fundamental optimization policy is to update the trainable parameters of the encoder based on the loss calculated from a batch of training data in Equation (3.5). When we take the temperature parameter into account, we find that temperature has an impact on the loss function. Specifically, at lower temperatures, the softmax function produces a sharper probability distribution, resulting in significant differences in loss values between misclassified (hard samples) and correctly classified (easy samples) data. Consequently, the encoder update is predominantly influenced by the misclassified data within the batch.

Conversely, higher temperatures lead to smaller differences in probability values across different classes, resulting in a smoother probability distribution. Consequently, all samples contribute more equally to the loss calculation and model update. Essentially, except for perfectly predicted samples, which are rare during model training, lower temperatures cause the model to prioritize learning from hard samples, whereas higher temperatures help mitigate bias in updates across all samples within the batch.

3.4 Empirical Analysis and Discussion

As discussed in the Preliminary Section, applying a small temperature encourages a model to learn more about hard (misclassified) samples. A low temperature, however, leads to more equitable learning across different data points. Theoretically, both approaches to optimize feature distribution sound reasonable. We argue that which optimization strategy is better for classification tasks remains an empirical problem.

3.4.1 Experiment Setting

We conduct image classification tasks on multiple benchmarks (i.e. CIFAR10, CIFAR100, and Tiny-ImageNet) and their extended Common Corruptions and Perturbations sets (i.e. CIFAR10-C, CIFAR100-C, and Tiny-ImageNet-C with corruption strentgh being 3) to investigate the impact of temperature scaling on model's adversarial robustness. To get a comprehensive evaluation, $\tau \in \{0.1, 0.5, 1, 10, 30, 50, 70, 100\}$. Unless stated otherwise, we takes ResNet50 [25] and VIT-small-patch16-224 as the CNN and transformer backbones, respectively. The ResNet50 is trained from scratch, with SGD optimizer and learning rate setting to 0.1. We also utilize the Cosine Annealing scheduler to better train the model. The transformer is pretrained on ImageNet-21K and finetuned on the target dataset using Adam optimizer. All experiments run on one RTX3090.

Using the standard classification performance as the performance baslines, we evaluate the model's robustness against PDG20 [41] and C&W [9]. Both

attacks are bounded by the l_∞ box with the same maximum perturbation $\epsilon = 8/255$ (please refer to the Section 3.6 for more adversarial attack performance).

To clarify, the temperature scaling only involves in model training in this study, but not model evaluation and attacks. All empirical evaluation and adversarial sample generation by PGD and C&W are based on the standard cross entropy, i.e. $\tau = 1$. Thus, attack gradients are not attenuated, reflecting model's true sensitivity to data perturbation.

3.4.2 Experiment Results & Observations

Table 3.1: Model performance and Robustness against Common Corruptions and Adversarial attacks (%) under different temperatures with ResNet50 trained from scratch using SGD optimizer. -C in the table represents the corresponding Common Corruptions and Perturbations set.

Temp.	CIFAR10				CIFAR100				Tiny-Imagenet			
	Clean	-C	PGD20	C&W	Clean	-C	PGD20	C&W	Clean	-C	PGD20	C&W
$\tau = 0.1$	90.05	73.31	0	27.79	70.39	44.52	0	14.32	54.53	12.63	0	23.17
$\tau = 0.5$	94.17	72.51	0	16.03	74.79	45.41	0	8.44	61.07	18.55	0	19.44
$\tau = 1$	94.26	72.53	0	19.19	74.58	46.47	0	11.26	62.93	18.66	0	19.09
$\tau = 10$	95.41	73.94	0.56	39.79	78.21	50.67	0.29	15.33	64.70	21.66	2.59	23.88
$\tau = 30$	95.26	74.93	91.09	43.35	78.27	50.17	68.47	18.81	63.60	21.30	49.45	26.50
$\tau = 50$	94.92	74.44	93.04	36.13	77.97	49.87	72.92	20.50	62.85	20.40	54.95	28.68
$\tau = 70$	95.05	74.26	93.85	35.43	77.20	49.61	73.49	21.66	62.14	20.57	55.54	30.14
$\tau = 100$	95.05	73.08	94.29	37.32	77.14	49.31	73.65	22.83	61.46	18.82	54.60	32.71

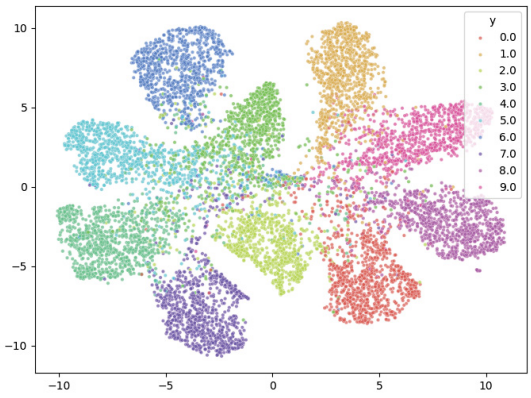
The quantitative results on CNN and Transformer are summarized in Table 3.1 and Table 3.2, respectively. For the CNN model, ResNet50, training from scratch, the standard accuracy increases with the temperature increase. Further more, CNN models trained at elevated temperatures shows more robust against adversarial perturbations and naturally corrected images. We believe that such improvements majorly attribute to the better model optimization with leveraged temperature. For the transformer finetuned on the target

Table 3.2: Model performance and Robustness against Common Corruptions and Adversarial attacks (%) under different temperatures with Transformer. Transformer models are using Vit-small-patch16-224 as the backbone and are trained on Imagenet-21k and fine-tuned on the target dataset using Adam optimizer. -C in the table represents the corresponding Common Corruptions and Perturbations set.

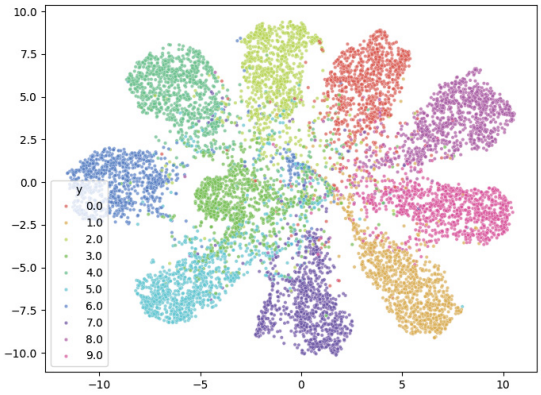
Temp.	CIFAR10				CIFAR100			
	Clean	-C	PGD20	C&W	Clean	-C	PGD20	C&W
$\tau = 0.1$	98.45	92.83	0	26.13	89.79	74.7	0	23.71
$\tau = 0.5$	98.33	91.60	0	26.26	90.53	74.9	0	29.25
$\tau = 1$	98.29	92.21	0	31.69	90.78	75.5	0	31.97
$\tau = 10$	98.06	92.19	89.07	31.89	89.94	75.5	58.71	34.96
$\tau = 30$	98.23	91.72	97.10	38.21	89.52	74.6	86.25	36.07
$\tau = 50$	98.22	91.43	97.75	39.52	89.28	73.8	87.29	33.64
$\tau = 70$	98.03	91.20	97.72	39.02	89.48	74.2	87.96	33.81
$\tau = 100$	98.07	91.56	97.87	38.26	89.13	73.47	86.99	31.84

set, the standard accuracy and robustness aginast natural corruptions and perturbations is quite stable. We hypothesize that such stable performance is due to the fact that ViT has already been pre-trained on ImageNet and has reached a relatively high-quality state.

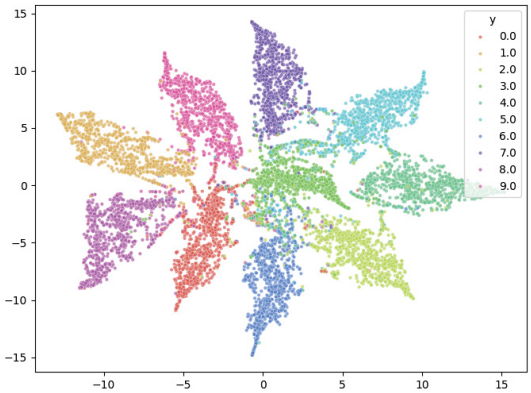
Clustering is a crucial metric when measuring how an encoder performs. In classification, a good encoder should be able to gather samples from the same class while separating clusters of different classes. Figure 3.1 and Figure 3.2 present 2D TSNE visualization of the CIFAR10 sample distribution by ResNet50 and transformer. We observe a similar trend: low temperatures lead to more mixed clusters, while models trained with elevated temperatures have better cluster effects. These empirical observations also explain the improved classification performance on clean and non-adversarial perturbations, as well as stronger adversarial robustness, with high temperature in Table 3.1 and Table 3.2.



(a) $\tau = 0.5$

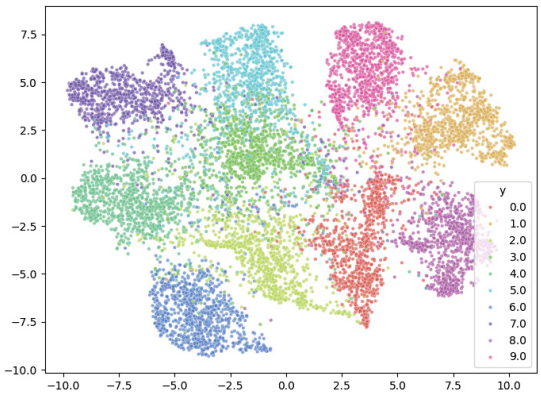


(b) $\tau = 1$

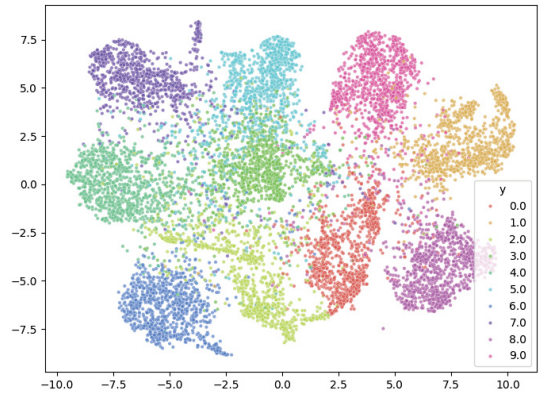


(c) $\tau = 50$

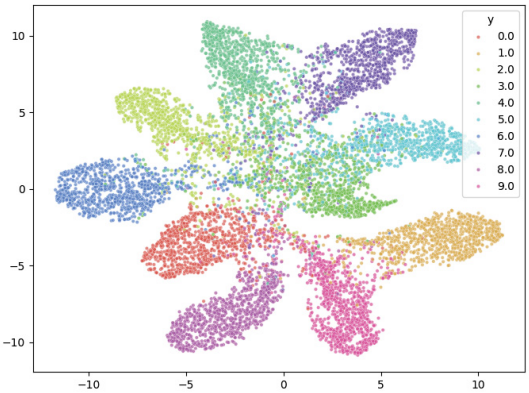
Figure 3.1: T-SNE [55] visualization of the CIFAR10 sample distribution after the ResNet50 encoder with different temperatures.



(a) $\tau = 0.5$



(b) $\tau = 1$



(c) $\tau = 50$

Figure 3.2: T-SNE [55] visualization of the CIFAR10 sample distribution after the VIT encoder with different temperatures.

3.4.3 Gradient Analysis for Adversarial Generation

Our empirical observation prompts questions regarding the mechanism behind the gained robustness. In this section, our focus is on investigating the model's behavior under adversarial attacks and understanding why the model demonstrates such robustness.

In order to discern the source of model robustness, we follow the work in [28] and study the gradient of the classification loss with respect to the input to analyze the direction of the PGD attack, which can be written as

$$\frac{\partial L_{ce}}{\partial x} = [(\mathbb{S}(w_i^T \cdot f(x)) - 1) \cdot w_i^T + \sum_{j \neq i} w_j^T \cdot \mathbb{S}(w_j^T \cdot f(x))] \cdot \frac{\partial f(x)}{\partial x} \tag{3.6}$$

As illustrated above, given a well-trained model, for most inputs where $\mathbb{S}(w_i^T \cdot f(x)) \approx 1$, the gradient does not have a noticeable portion in target class w_i on the early stage of the attack. This implies that rather than directly 'stepping away' from the target class, the attack will initially focus on approaching other class prototypes. Moreover, the second term, $\sum_{j \neq i} w_j^T \cdot \mathbb{S}(w_j^T \cdot f(x))$, indicates that all the other directions are weighted by their according probabilities. Therefore, untargeted attacks are actually targeted toward the *error-prone class*, which most commonly is the largest probability class other than the target class. However, if a model lacks an *error-prone class* given an input, all w_k will be weighted equally. Consequently, the gradient would point toward all negative class prototypes, making it exceptionally challenging to determine the optimal direction. We noticed that such a scenario occurs when a model is trained with large τ. Then let's focus on the gradient update strength. For a data sample x is classified correctly, $\mathbb{S}(w_j^T \cdot f(x))$ would be small when the model training temperature τ increases. That is, when a model is trained with high temperatures, not only the gradient direction to generate adversarial samples is not clear, but the gradient strength is also small. Both factors contribute to the robustness of the model when optimized with elevated temperatures.

3.4.4 Class Prototypes Analysis

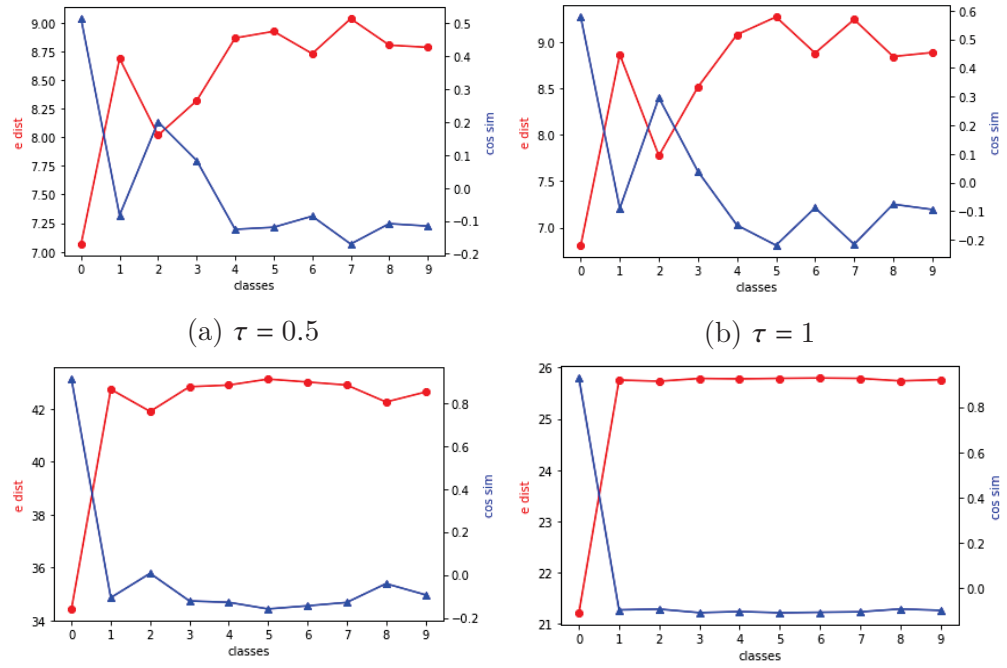
To further analyze the model behavior, we investigate the relation between the encoded feature, $f(x)$, and each class prototype, w_j. Here, we observe the Euclidean distance and cosine similarity. Figure 3.3 shows Euclidean distance and cosine similarity between one sample and all class prototypes. It is evident that as the training temperature goes up, the feature $f(x)$ tends to have an identical distance to all negative class prototypes. This indicates the model trained with high temperature is less likely to have an *error-prone class*, which is essential for untargeted attacks as we discuss above.

Furthermore, to illustrate that the phenomenon shown in Figure 3.3 is not limited to one or a few samples, we calculate the variance of Euclidean distance and cosine similarity of all negative class prototypes across all samples in CIFAR10 test set. Note that as illustrated in Figure 3.3, different models have very different ranges for Euclidean distance between encoded feature and class prototypes. Therefore, we map the value of different models into the same range to make a more direct comparison. Box plots are drawn in Figure 3.5 showing the overall variance results with each box being a model trained with a different temperature. We can observe a clear trend that when the temperature rises, the variance for both Euclidean distance and cosine similarity drops indicating the encoded sample, $f(x)$, has a more similar distance to all negative class prototypes. One might notice an increase in variance when the temperature reaches some threshold. We label them as extreme temperatures, which are so large that they can adversely affect the model's convergence.

3.4.5 Extending to Adversarial Training

Given that our temperature control method is used inside the Cross-Entropy Loss, it is possible to apply this method in adversarial training. Here, we do preliminary experiments on the adversarial training baseline proposed by Madry *et al.* [41] for the simplicity of its loss function. We add temperature control inside vanilla loss term forming

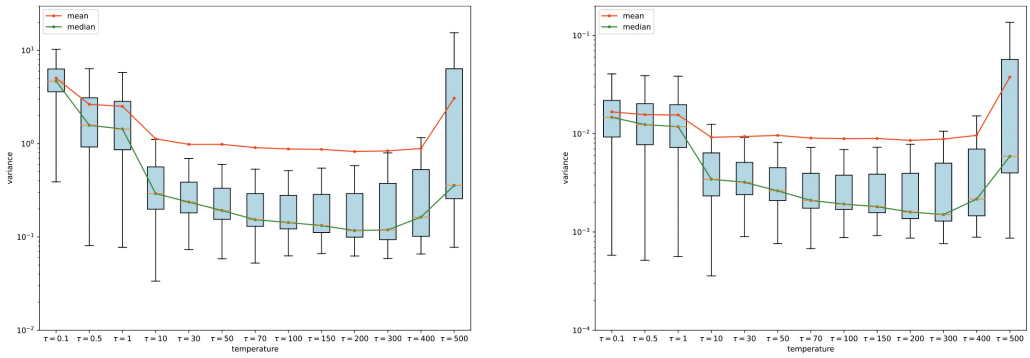
$$L_{AT}(x, x_{adv}, y, F) = L_{ce}(F(x)/\tau, y) + L_{ce}(F(x_{adv}), y), \tag{3.7}$$



(a) $\tau = 0.5$ (b) $\tau = 1$

(c) $\tau = 50$ (d) $\tau = 100$

Figure 3.3: A demonstration of the Euclidean distance and cosine similarity between the encoded sample $f(x)$ and all class prototypes for one sample, with different temperature configurations. The red lines indicate the Euclidean distance while the blue lines stand for cosine similarity.



(a) Euclidean Distance **(b)** Cosine Similarity

Figure 3.4: Euclidean Distance

Figure 3.5: Box plot of the variance of the Euclidean distance and cosine similarity calculated from each sample. The variances are calculated across all negative class prototypes, therefore, lower variance indicates a more uniform distribution of all negative class distances. Each box is a model trained with a different temperature, the green line shows the median value across all variances and the orange line is the mean value of all variances.

Table 3.3: Preliminary experiments of adversarial training on CIFAR-10 with temperature control. The training scheme uses Madry *et al*. [41] and the model is ResNet50.

Temperature	$\tau = 0.5$	$\tau = 1$	$\tau = 10$	$\tau = 30$	$\tau = 50$	$\tau = 70$	$\tau = 100$
Clean	88.98	85.67	81.71	82.62	83.75	84.28	84.27
PGD20	35.93	42.63	40.95	44.96	48.61	49.16	48.53

where $\boldsymbol{F}$ is a combination of encoder and class prototypes.

Our preliminary results are listed in Table 3.3. We can clearly observe that model robustness increases as the temperature increases with a slight trade-off with clean accuracy, which confirms the possibility of combining the temperature control method with adversarial training. While further extension to other adversarial training methods is possible, it remains a complex problem for most adversarial training involves complex loss functions that may introduce terms other than the Cross-Entropy function. Also, balancing the vanilla loss term and adversarial loss term largely relies on empirical experiments. Therefore, further exploration of fitting this into other adversarial training methods falls beyond the scope of this section.

3.4.6 Further Discussion on Adversarial Robustness

Despite the model trained with high temperatures showing superb robustness against untargeted PGD attack due to its nature attribute that discovers the weakness of PGD attack, it does not hold robustness against targeted attacks. The reason behind this is straightforward. In targeted attacks, (3.6) no longer holds, and the gradient is not obligated to move towards all the negative class prototypes with a weighted step size. Therefore, with the only source of the model robustness gained eliminated, it is naturally vulnerable when facing targeted attacks.

Remark: Even though many attacks claim themselves to be untargeted attacks, they actually optimize toward one self-selected target, which we do not consider untargeted attacks under this setting. One popular example is

the Difference of Logits Ratio(DLR) attack proposed by Croce and Hein [15]. Regardless of its ability to rescale the logit,

$$\mathrm{DLR}(x, y) = -\frac{z_y - \max_{i \neq y} z_i}{z_{\pi 1} - z_{\pi 3}} \tag{3.8}$$

shows that the DLR loss automatically selects the class holding the largest logit other than the target class as the attack target. Therefore, during optimization, it does not need to optimize toward all negative class prototypes. A similar example also includes FAB attack [14].

3.5 Conclusion & Limitation

In this section, we investigate the under-explored property of temperature scaling with the softmax function on image classification tasks. By performing gradient analysis with the Cross-Entropy classification loss and executing different empirical experiments, we show that temperature scaling can be a significant factor in model performance. Further experiments reveal applying high temperatures during training introduces enormous robustness against gradient-based untargeted adversarial attacks. We hope our work raises the interest of other researchers to utilize the simple temperature scaling in the common Cross-Entropy loss.

One limitation of this study was that we didn't report an explicit algorithm to set the best temperature values. We will work on this in our future work. One takehome note, as a hyperparameter, the tuning cost of the tempeerature is low as a wide range of temperatures (30 to 70) can provide improvements to the model.

3.6 Appendix: Complete Experimental Results

In this section, we report the complete experimental results of different models trained with various temperatures against common corruption and perturbations as well as various adversarial attacks. In all our experiments, the CNN model takes ResNet50 as the backbone and are trained using SGD optimizer with the learning rate set to 0.1 and a cosine annealing schedule

is also utilized during all training processes. The transformer model takes "vit_small_patch16_224" from the timm package, which version is 0.9.10. It has been fine-tuned for 30 epochs under the target datasets, with a learning rate of 1e-4.

3.6.1 Experiment on Common Corruptions and Perturbations

This section includes detailed experiment results on natural corruption including CIFAR10-C, CIFAR100-C, and Tiny-Imagenet-C.

Natural corruption results on CIFAR10-C, CIFAR100-C and TinyImageNet-C with ResNet50 are reported in Table 3.4, Table 3.5, and Table 3.6. As demonstrated in the tables, models trained with elevated temperatures have stronger noise resistant capability. We believe that the robust performance against the non-adversarial perturbation is due to the better model optimization under elevated temperature. Similarly, we report transformer's performance in Table 3.7 and Table 3.8. As we discussed in the section 3.4.2 , VIT's robustness against natural corruptions and perturbations is quite stable. We hypothesize that such consistent performance is due to the fact that ViT has already been pre-trained on ImageNet and has reached a relatively high-quality state.

3.6.2 Experiment on Adversarial Attacks

In this section, we report more results on adversarial perturbations. All the L_∞ attacks are using epsilon $8/255$, C&W-L2 attack is set with 0.01 learning rate, 1000 max iteration, and 0.01 cost, and the Square attack uses its default parameters from the autoattack package. Table 3.9 and Table 3.10 show the results on CIFAR10 and CIFAR100 using ResNet50 and Table 3.11 and Table 3.12 report the results using transformer.

Table 3.4: Accuracy(%) under natural corruption CIFAR10-C dataset. Models use Resnet50 as the backbone.

Temperature	Clean	**Noise**					**Blur**				
		Gauss.	Shot	Speckle	Impulse	Spatter	Defocus	Gauss.	Glass	Motion	Zoom
τ = 0.1	90.06	64.07	70.14	70.71	67.42	82.70	74.45	64.20	62.72	67.10	67.51
τ = 0.5	94.17	44.26	56.28	60.00	55.97	84.63	80.02	69.20	48.08	73.51	74.68
τ = 1	94.26	45.40	56.94	60.52	56.91	84.62	78.83	67.50	51.24	74.46	73.72
τ = 10	95.41	43.10	57.16	61.86	58.27	86.45	80.07	68.39	50.78	75.16	74.83
τ = 30	95.26	49.87	62.05	65.47	59.72	86.59	80.67	68.74	51.81	74.48	75.98
τ = 50	94.92	48.78	60.43	63.71	60.95	86.52	79.63	67.59	51.32	75.45	74.95
τ = 70	95.06	48.10	60.00	63.22	59.19	86.37	80.38	68.18	52.16	76.01	74.70
τ = 100	95.17	44.68	57.41	61.08	57.21	86.54	79.05	66.50	48.56	73.79	74.10

Temperature	**Average**	**Weather**				**Digital**				
		Snow	Fog	Brightness	Frost	Elastic	Pixelate	JPEG	Saturate	Contrast
τ = 0.1	73.31	75.45	78.03	87.04	74.08	78.82	75.72	83.29	84.48	64.94
τ = 0.5	72.51	79.41	85.78	92.48	74.88	82.13	71.46	80.65	90.67	73.76
τ = 1	72.53	78.29	85.35	92.17	74.19	82.03	72.42	79.98	90.62	72.91
τ = 10	73.94	81.85	87.90	93.77	77.28	83.05	74.26	80.04	91.99	78.62
τ = 30	74.93	82.52	87.94	93.87	77.68	83.38	73.30	78.82	91.95	78.76
τ = 50	74.44	81.91	87.31	93.55	77.61	82.94	72.35	79.36	91.74	78.26
τ = 70	74.26	81.90	86.64	93.69	76.68	82.91	72.63	79.40	91.88	76.73
τ = 100	73.08	81.45	86.76	93.41	76.43	82.52	72.50	78.69	91.63	76.25

Table 3.5: Accuracy(%) under natural corruption CIFAR100-C dataset. Models use Resnet50 as the backbone.

Temperature	Clean	**Noise**					**Blur**				
		Gauss.	Shot	Speckle	Impulse	Spatter	Defocus	Gauss.	Glass	Motion	Zoom
τ = 0.1	72.60	19.05	27.17	28.54	29.02	58.96	53.70	44.47	19.69	47.79	46.48
τ = 0.5	74.98	20.44	28.49	30.37	27.55	58.65	54.33	43.58	20.15	47.97	46.83
τ = 1	75.25	24.50	31.89	33.26	30.37	58.47	54.77	44.88	18.91	49.88	46.83
τ = 10	77.86	21.61	29.69	31.27	30.56	64.15	61.33	52.20	24.30	56.00	55.44
τ = 30	78.23	20.87	29.72	31.23	27.57	63.86	60.33	51.15	24.53	54.95	55.65
τ = 50	77.78	22.21	30.25	31.95	28.57	63.35	59.68	49.95	22.97	53.80	53.90
τ = 70	77.52	20.89	29.50	31.42	27.08	63.51	60.62	51.06	23.72	54.87	55.29
τ = 100	77.60	21.62	29.54	31.24	28.12	63.38	60.17	50.67	21.86	53.99	54.73

Temperature	**Average**	**Weather**				**Digital**				
		Snow	Fog	Brightness	Frost	Elastic	Pixelate	JPEG	Saturate	Contrast
τ = 0.1	44.52	48.21	56.71	66.89	42.18	55.15	47.09	51.84	57.50	45.50
τ = 0.5	45.41	49.51	58.71	69.60	42.64	56.78	48.38	52.16	59.84	46.85
τ = 1	46.47	49.90	59.07	70.13	42.31	56.82	45.67	53.74	62.33	49.15
τ = 10	50.67	56.21	65.29	73.66	51.38	62.08	53.76	54.43	64.07	55.39
τ = 30	50.17	56.39	65.09	73.43	51.16	61.37	53.87	52.24	64.42	55.44
τ = 50	49.87	55.70	64.46	73.33	50.53	61.11	53.20	53.28	64.40	54.84
τ = 70	49.61	55.15	64.55	72.65	48.93	60.62	52.45	51.91	63.80	54.57
τ = 100	49.31	55.63	64.68	72.69	49.05	59.80	50.86	50.61	63.83	54.45

Table 3.6: Accuracy(%) under natural corruption Tiny-Imagenet-C dataset. Models use Resnet50 as the backbone. Attack Strength is set to 3.

Temperature	Clean	**Noise**					**Blur**				
		Gauss.	Shot	Speckle	Impulse	Spatter	Defocus	Gauss.	Glass	Motion	Zoom
$\tau = 0.1$	54.53	6.06	10.93	7.62	8.29	16.74	8.39	7.04	8.72	11.87	9.44
$\tau = 0.5$	61.07	11.15	17.93	13.19	14.70	20.33	13.70	11.99	13.93	18.05	15.62
$\tau = 1$	62.93	9.08	16.73	11.23	12.32	19.66	13.94	11.70	13.58	18.91	16.13
$\tau = 10$	64.70	13.07	21.56	15.54	15.32	23.13	17.61	15.72	16.78	21.65	19.34
$\tau = 30$	63.60	14.99	23.96	17.99	17.19	22.71	15.92	13.84	15.75	20.74	18.63
$\tau = 50$	62.85	12.97	22.34	15.88	15.42	23.28	15.37	13.55	15.01	19.85	16.74
$\tau = 70$	62.14	12.94	22.01	16.14	15.87	22.77	15.68	13.20	14.59	19.92	17.19
$\tau = 100$	61.46	11.69	20.90	13.69	13.67	21.04	12.73	11.15	12.80	17.29	14.72

Temperature	**Average**	**Weather**				**Digital**				
		Snow	Fog	Brightness	Frost	Elastic	Pixelate	JPEG	Saturate	Contrast
$\tau = 0.1$	12.63	14.47	14.38	18.97	0.11	16.45	30.77	23.70	22.71	3.39
$\tau = 0.5$	18.55	19.96	22.02	27.60	0.11	23.96	38.96	32.18	31.91	5.11
$\tau = 1$	18.66	22.12	23.45	28.40	0.10	24.89	40.06	33.26	40.06	5.96
$\tau = 10$	21.66	24.51	26.72	31.10	0.11	27.49	42.26	35.29	42.26	9.85
$\tau = 30$	21.30	22.77	24.45	29.67	0.10	26.86	41.42	34.60	41.42	9.04
$\tau = 50$	20.40	23.10	24.23	29.28	0.10	25.51	40.65	33.43	40.65	8.05
$\tau = 70$	20.57	22.95	23.64	29.31	0.10	25.70	40.46	34.17	40.46	9.51
$\tau = 100$	18.82	22.34	22.63	27.70	0.11	23.35	39.33	32.05	39.33	7.31

Table 3.7: Accuracy(%) under natural corruption CIFAR10-C dataset. Models use 'vit_small_patch16_224' as the backbone.

Temperature	Clean	**Noise**					**Blur**			
		Gauss.	Shot	Speckle	Impulse	Spatter	Defocus	Gauss.	Motion	Zoom
$\tau = 0.1$	98.4	81.7	86.5	87.5	89.3	95.6	96.1	94.8	92.6	95.5
$\tau = 0.5$	98.3	74.9	81.8	83.4	86.3	95.1	95.8	94.7	92.4	95.0
$\tau = 1$	98.3	78.8	84.4	85.5	87.7	95.4	95.8	94.6	92.6	95.0
$\tau = 10$	98.1	80.1	85.3	85.5	88.9	95.4	94.9	93.4	91.5	94.2
$\tau = 30$	98.2	77.8	83.4	84.9	88.8	95.3	95.1	93.7	91.9	94.2
$\tau = 50$	98.2	78.1	83.1	83.7	86.4	95.2	94.7	93.0	90.8	93.5
$\tau = 70$	98.0	77.0	83.0	84.4	86.5	95.3	94.4	92.4	90.1	93.7
$\tau = 100$	98.1	77.7	83.7	84.8	85.7	94.7	94.5	92.4	91.2	93.2
Temperature	**Average**	**Weather**				**Digital**				
		Snow	Fog	Brightness	Frost	Elastic	Pixelate	JPEG	Saturate	Contrast
$\tau = 0.1$	92.83	94.6	93.2	98.0	95.0	93.9	93.1	90.5	96.6	90.7
$\tau = 0.5$	91.6	94.9	93.7	97.8	94.7	93.7	92.8	89.9	96.5	89.5
$\tau = 1$	92.2	94.9	93.8	97.8	94.8	93.3	92.6	90.2	96.6	90.0
$\tau = 10$	92.1	94.7	93.3	97.6	95.1	93.5	92.4	90.8	96.3	88.4
$\tau = 30$	91.7	94.4	93.5	97.4	95.0	93.8	90.6	89.5	95.9	88.7
$\tau = 50$	91.4	94.4	93.3	97.6	94.9	93.1	92.0	89.8	96.0	89.2
$\tau = 70$	91.2	94.3	92.8	97.4	94.4	93.2	92.4	89.6	96.3	87.5
$\tau = 100$	91.5	94.4	93.0	97.5	94.7	92.9	93.5	90.2	96.0	90.3

Table 3.8: Accuracy(%) under natural corruption CIFAR100-C dataset. Models use 'vit_small_patch16_224' as the backbone.

Temperature	Clean	**Noise**					**Blur**			
		Gauss.	Shot	Speckle	Impulse	Spatter	Defocus	Gauss.	Motion	Zoom
$\tau = 0.1$	89.7	55.3	62.1	61.9	57.8	80.0	82.0	78.7	74.9	79.1
$\tau = 0.5$	90.4	51.6	59.1	59.5	55.8	81.9	82.3	79.2	75.9	79.6
$\tau = 1$	90.4	50.7	59.5	59.8	63.7	82.0	82.6	79.2	75.7	80.2
$\tau = 10$	90.2	50.0	59.6	60.8	65.5	82.1	82.9	80.1	75.8	80.5
$\tau = 30$	89.4	48.3	58.0	59.4	62.0	80.9	81.8	78.5	75.1	79.6
$\tau = 50$	89.2	48.0	57.1	58.3	55.8	80.8	81.4	77.7	74.6	79.0
$\tau = 70$	89.3	47.9	56.8	57.9	63.3	80.8	81.7	78.3	74.8	79.4
$\tau = 100$	89.5	46.6	56.5	58.0	56.1	80.7	80.9	77.4	73.3	78.3
Temperature	**Average**	**Weather**				**Digital**				
		Snow	Fog	Brightness	Frost	Elastic	Pixelate	JPEG	Saturate	Contrast
$\tau = 0.1$	74.7	78.3	78.2	87.3	78.3	76.9	74.1	68.5	80.8	75.1
$\tau = 0.5$	74.9	79.2	78.7	88.4	80.0	78.1	76.2	69.5	81.5	75.4
$\tau = 1$	75.5	80.0	79.0	88.7	80.4	78.6	76.9	70.0	82.3	75.8
$\tau = 10$	75.5	79.6	79.5	88.2	79.4	78.4	76.0	69.2	81.8	75.7
$\tau = 30$	74.6	79.2	78.1	87.4	79.6	78.4	77.0	69.8	80.9	73.2
$\tau = 50$	73.8	78.1	77.5	87.1	78.6	77.7	77.1	69.4	80.6	73.5
$\tau = 70$	74.2	78.4	77.9	87.1	77.8	78.0	77.5	68.8	80.8	73.1
$\tau = 100$	73.47	78.9	77.0	87.5	78.8	77.5	76.9	68.8	80.6	72.7

Table 3.9: Clean and Adversarial accuracy(%) under different temperatures. Models use Resnet50 as the backbone and are trained on CIFAR10 dataset using SGD optimizer.

Temperature	Clean	PGD20	C&W-Linf	C&W-L2	APGD-CE	Square
$\tau = 0.1$	90.05	0	27.79	83.18	0	0.4
$\tau = 0.5$	94.17	0	16.03	29.28	0	0.29
$\tau = 1$	94.26	0	19.19	19.09	0	0.45
$\tau = 10$	95.41	0.56	39.79	3.14	0.02	0.54
$\tau = 30$	95.26	91.09	43.35	1.33	81.54	0.83
$\tau = 50$	94.92	93.04	36.13	1.02	83.26	0.63
$\tau = 70$	95.06	93.85	33.51	0.75	84.46	1.03
$\tau = 100$	95.15	94.29	35.43	0.53	84.63	2.95
$\tau = 150$	95.05	94.52	37.32	0.43	84.96	6.17

Table 3.10: Clean and Adversarial accuracy(%) under different temperatures. Models use Resnet50 as the backbone and are trained on CIFAR100 dataset using SGD optimizer.

Temperature	Clean	PGD20	C&W-Linf	C&W-L2	APGD-CE	Square
$\tau = 0.1$	70.39	0	14.32	62.92	0	0.46
$\tau = 0.5$	74.79	0	8.44	27.17	0	0.39
$\tau = 1$	74.58	0	11.26	24.95	0	0.48
$\tau = 10$	78.21	0.29	15.33	2.78	0.04	0.31
$\tau = 30$	78.27	68.47	18.81	1.04	48.61	0.49
$\tau = 50$	77.97	72.92	20.50	0.69	52.86	0.68
$\tau = 70$	77.20	73.49	21.66	0.46	53.34	0.81
$\tau = 100$	77.14	73.65	22.83	0.36	54.22	1.04
$\tau = 150$	73.15	66.27	20.59	0.37	47.12	1.48

Table 3.11: Clean and Adversarial accuracy(%) under different temperatures. Models use 'vit_small_patch16_224' as the backbone and are trained on CIFAR10 dataset using Adam optimizer.

Temperature	Clean	PGD20	C&W-Linf	C&W-L2	APGD-CE	Square
$\tau = 0.1$	98.45	0	26.13	56.89	0	0.21
$\tau = 0.5$	98.33	0	26.26	56.57	0.06	0.31
$\tau = 1$	98.29	0	31.69	58.63	0.21	2.69
$\tau = 10$	98.06	89.07	31.89	49.43	84.99	3.44
$\tau = 30$	98.23	97.10	38.21	55.15	96.83	6.71
$\tau = 50$	98.22	97.75	39.52	59.84	97.4	2.62
$\tau = 70$	98.03	97.72	39.02	67.49	97.57	5.27
$\tau = 100$	98.07	97.87	38.26	69.74	97.68	3.08

Table 3.12: Clean and Adversarial accuracy(%) under different temperatures. Models use 'vit_small_patch16_224' as the backbone and are trained on CIFAR100 dataset using Adam optimizer.

Temperature	Clean	PGD20	C&W-Linf	C&W-L2	APGD-CE	Square
$\tau = 0.1$	89.79	0	23.71	26.35	0	0.14
$\tau = 0.5$	90.53	0	29.25	31.58	0	0.43
$\tau = 1$	90.78	0	31.97	32.87	0	0.53
$\tau = 10$	89.94	58.71	34.96	20.37	44.09	0.81
$\tau = 30$	89.52	86.25	36.07	23.06	85.49	0.97
$\tau = 50$	89.28	87.29	33.64	25.98	87.27	0.72
$\tau = 70$	89.48	87.96	33.81	29.54	87.72	1.02
$\tau = 100$	89.13	86.99	31.84	32.9	87.04	0.53

Chapter 4

Low Dimension Distillation for Adversarial Robustness

4.1 Introduction

Since Szegedy et al. [52] first proposed adversarial examples, dimensionality has become one of the most important factors to consider for model's adversarial robustness. Goodfellow et al. [23] pointed out that with increasing input dimensions, models can exhibit more adversarial vulnerability in a linear model. Chen et al. [12] found that in low-dimensional input images, it is harder to find adversarial examples. Recent work [10] suggests that CNN-based networks tend to extract high-dimensional feature vectors, which may lead to redundant space for adversarial perturbation. Gilmer et al. [21] demonstrated that eliminating such redundancy may increase the adversarial accuracy of the model. Awasthi et al. [7] used the principal component analysis (PCA) method to create a projection matrix that projects data in the high-dimensional input space into a robust subspace. This subspace is smooth and, under small perturbations, maintains low variance and separable means. In subsequent work [6], they theoretically proved that PCA can find a robust subspace for both supervised and unsupervised tasks. The most recent work [22] found that as subspace dimensionality decreases, the average adversarial success rate also decreases.

Despite the insightful discovery in these prior studies, there are still some problems unsolved. First, these methods usually need auxiliary data to manually create a fixed projection matrix to reduce the dimension of the input data.

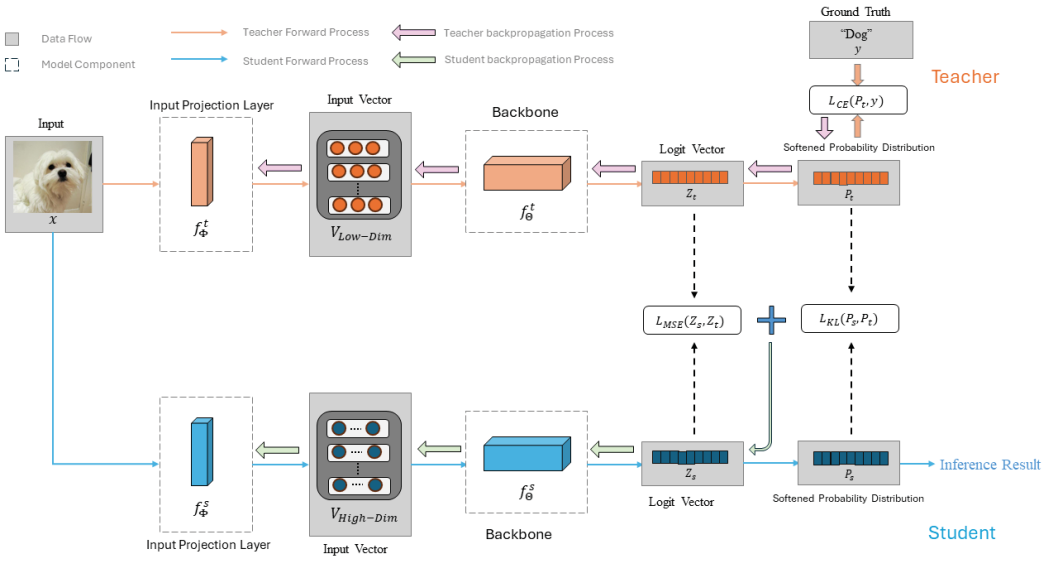


Figure 4.1: The diagram of our Low Dimension Distillation (LDD) Architecture.

Second, the low-dimension trained network tends to have weaker generalization ability on clean data compared to the high-dimension trained network.

To solve the first problem, we propose a generalized input layer in deep learning model for input data dimension reduction. It can be wither a pre-fixed down-sampling operation or a trainable layer with a large stride step. To address the second challenge, inspired by Zhang et al. [64]'s self-distillation method, we propose a distillation strategy to distill a low-dimension trained network into a high-dimension network to increase generalization ability of the model. This strategy encourages the student model with high-dimensional input preserving teacher model's robustness but with stronger generalization on clean data. In the experiment, our method achieves 43.45% robustness accuracy on the CIFAR-10 dataset using only clean images during the training process without any additional data.

4.2 Methodology

Figure 4.1 depicts the overall diagram of the Low Dimension Distillation (LDD) approach using a teacher-student architecture with online knowledge distillation. The primary distinction between the teacher and student models lies in their

input projection layers. Specifically, following the teacher's input projection layer, the input data is transformed into a low-dimensional vector, reducing information redundancy and facilitating robust feature learning. During model training, we employ cross-entropy loss to guide the teacher model to predict as accurately as possible. In contrast, the student model retains its input vector in a higher dimension and incorporates an online distillation mechanism to learn from the robust teacher model.

4.2.1 LDD Architecture

Our LDD model comprises a teacher-student pair. Each network, whether teacher or student, is structured with an architecture f consisting of two components: an input projection layer parameterized by Φ and a feature extraction backbone parameterized by Θ. Generally, the input project layer will be the input layers in a CNN network and the linear projection of flattened patches in a Vision Transformer. Given an input date x, we can obtain the corresponding input vector v by the function $f_\Phi : x \to v$. The second part of f is the backbone, which contains the encoder and the fully connected layers as a classifier. For any input vector v, we have a function $f_\Theta : v \to z$ that projects it onto a logit vector z. Therefore, our complete network can be represented as $f_{\Phi,\Theta} = f_\Phi \circ f_\Theta$.

In this chapter, we use subscript t and s to denote variables in the teacher model and student model. Therefore, the data flow of the teacher and student models are specified as follows. First, we pass x through f^t_Φ, obtaining an input vector $v_{Low-Dim}$. Then, f^t_Θ is used to process the input vector and predict a logit z_t. Finally, we use the log Softmax function σ to generate a softened probability p_t.

$$f^t_\Phi(x) = v_{Low-Dim}, \quad f^t_\Theta(v_{Low-Dim}) = z_t, \quad \sigma(z_t) = p_t \tag{4.1}$$

At the same time, a similar data flow occurs in the student network $f^s_{\Phi,\Theta}$:

$$f^s_\Phi(x) = v_{High-Dim}, \quad f^s_\Theta(v_{High-Dim}) = z_s, \quad \sigma(z_s) = p_s \tag{4.2}$$

The main architectural difference between the teacher and student models lies in their input projection layers. The teacher model includes an input

projection layer that transforms the input data into a relatively low-dimensional space. In contrast, the student model does not perform any special operations to reduce the dimensionality of the input vector. This design choice is motivated by the belief that the teacher model's reduction of input data redundancy decreases the likelihood of the model learning non-essential features. Conversely, by learning from the robust teacher model, the student model can leverage all the information in the original input sample to achieve a better generalization.

To realize the dimension reduction in the teacher model, we have 4 different implementations.

1. Direct Input Pooling (or input down-sampling): We add a pooling layer before the model's original input layer. This is the most straightforward method and can serve as a baseline in our method. However, this implementation has a significant drawback: the pooling layer cannot be trained, making it difficult to avoid losing some essential features during the down-sampling process.

2. Post-Input Pooling (or feature pooling): We can also add a pooling layer after the backbone's original input layer. This method shares the same problem as the previous one. It is still challenging to avoid missing some important features.

3. Trainable Input Layer with Large Stride for CNN: In this implementation, we increase stride of model's original input layer. This is the better method to reduce the dimensionality of the input feature vector. During the training process, this layer can automatically learn which features are essential to preserve and which are unnecessary to drop.

4. Modified Embedding Patch in Vision Transformer (ViT): In this implementation, we enlarge the patch size of the linear projection layer preceding the attention blocks in ViT. It share the same functionality as the CNN layer with a large stride step in CNN models. However, the primary challenge with this architectural modification is that the teacher

ViT model requires more time to converge, necessitating the use of a pre-trained model on a large dataset to expedite convergence.

In our experimentation section, we will show that all of the above implementation improves the model's robustness with different margins and the trainable ones achieve the best performance.

4.2.2 Online Distillation

We use a dataset D to train our teacher-student network. The training set contains n pairs of image-label pairs $\{x_i, y_i\}_n$ where $y_i \in \{1, \ldots, c\}$. In each update, we feed the network a minibatch $B = \{x, y\}_m$, which contains m pairs of image-label pairs. The student and teacher networks will be updated simultaneously in each update with the following loss functions, respectively.

Teacher Optimization: To optimize the teacher model, we use categorical cross-entropy loss L_{ce} between the predicted softened probability p_t and the true label y.

$$L_t = L_{ce}(p_t, y) = L_{ce}(\sigma(f^t_{\Phi,\Theta}(x)), y), \tag{4.3}$$

where $f^t_{\Phi,\Theta} = f^t_{\Phi} \circ f^t_{\Theta}$. When we update our network, we update both Φ and Θ together. We believe this can help our feature extractor obtain better features from x.

Student Optimization by Knowledge Distillation: In the original work of knowledge distillation [27], *Hinton et al.* use $L_{CE} + L_{KL}$ to formulate L_{KD}. It can be considered a combination of a classification loss function and an additional loss function that encourages the student to learning the distribution information of the teacher [61]. In our work, if we directly apply L_{CE} to the student network, it will encourage the student model to directly use the redundant information in input for classification, leading to adversarial vulnerability in the student network (which we want to avoid). In addition, *Taehyeon et al.* [32] found that the loss L_{KL} with low temperature focuses on learning the predictions of the teacher, and when it has a high temperature, it focuses on the teacher's distribution. They also point out that L_{MSE} can be

considered equivalent to L_{KL} with a sufficiently high temperature. Based on these prior studies, we propose a new distillation function in Equation 4.4.

$$L_s = L_{kL}(p_s, p_t) + \alpha L_{MSE}(z_s, z_t), \tag{4.4}$$

where α is the weight to balance the two terms. $L_{kL}(p_s, p_t)$ encourages the student to following the teacher's prediction, and $L_{MSE}(z_s, z_t)$ promotes the student sharing the same logit distribution of the robust teacher. In specific,

$$L_{KL} = KL(p_s, p_t) = L_{KL}(\sigma(f^s_{\Phi,\Theta}(x)), \sigma(f^t_{\Phi,\Theta}(x))) \tag{4.5}$$

$$L_{MSE} = (z_s - z_t)^2 = (f^s_{\Phi,\Theta}(x) - f^t_{\Phi,\Theta}(x))^2 \tag{4.6}$$

The pseudo-code of the training procedure of our Low Dimension Distillation is present in Algorithm 1.

Algorithm 1 Training procedure of our Low Dimension Distillation

Require: $f^t_\Theta, f^t_\Phi, f^s_\Theta, f^s_\Phi, \mathcal{D}$
for k `iteration` **do**
  while $\mathcal{B} \sim \mathcal{D}$ **do**
    $V_{Low-Dim} \leftarrow f^t_\Phi(x), Z_t \leftarrow f^t_\Theta(V_{Low-Dim}), P_t \leftarrow \sigma(Z_t)$
    $V_{High-Dim} \leftarrow f^s_\Phi(x), Z_s \leftarrow f^s_\Theta(V_{High-Dim}), P_s \leftarrow \sigma(Z_s)$
    $\mathcal{L}_t \leftarrow \mathcal{L}_{CE}(P_t, y)$
    $\mathcal{L}_s \leftarrow \mathcal{L}_{KL}(P_s, P_t) + \alpha \mathcal{L}_{MSE}(Z_s, Z_t)$
    $\{\Phi_t, \Theta_t\} \leftarrow \{\Phi_t, \Theta_t\} - \nabla_{\{\Phi_t,\Theta_t\}}\mathcal{L}_t$ ▷ Update Teacher
    $\{\Phi_s, \Theta_s\} \leftarrow \{\Phi_s, \Theta_s\} - \nabla_{\{\Phi_s,\Theta_s\}}\mathcal{L}_s$ ▷ Update Student
  end while
end for
return $f^s_{\Phi,\Theta}$

Discussion about Generalizability and Robustness

In recent work, *Shah et al.* [50] found that neural networks tend to use the simplest but not essentially robust features for classification, which could be background information or texture information. In subsequent work, *Singla et al.* [51] divided these features into two different types: spurious correlation features and over-emphasized features. Spurious correlation features are those features that usually appear with the target object but are not part of the

object. In Figure 4.2, a standard trained network tends to use the sea to determine whether the object in the image is a ship. Over-emphasized features are some small parts of the object that the network pays too much attention to for classification, which is not always correct. For example, the network might use the ear of a horse to determine whether the object is a horse (see Figure 4.2, horse column). *Shah et al.* [50] propose that these features usually lack robustness and are prone to attacks. They believe we need to help networks use complex predictive features, such as the semantics of the object, to improve robustness.

From Figure 4.2, we can see that the low-dimension teacher network tends to focus on the whole object rather than a small part of the object or the background. We believe this improvement is due to the limited number of features that can be extracted, forcing the network to use a large but rough perspective to look at the image. As seen in the teacher's heat map, it not only circles the entire object but also includes a lot of external field. Therefore, while the teacher network's robustness is increased, its generalization is reduced.

For the high-dimension student network, the increased dimension allows it to examine the image with a more detailed perspective. In these heat maps, the network includes almost no external field. Guided by the low-dimension teacher network, it can still focus on the whole object rather than a small part or the texture information.

4.3 Experimentation & Discussion

4.3.1 Experimental Setting

We use ResNet50, WideResNet, and Vision Transformer in our experiments. The ResNet50 and Vision Transformer are imported from TIMM, with the Vision Transformer pre-trained on ImageNet. WideResNet is sourced from RobustBench [13]. The depth and width of the WideResNet are 28 and 10, respectively, which are widely used in recent adversarial robustness research. The optimizer for the CNN networks is SGD, while Adam is used for the Vision Transformer. We also use the Cosine Annealing Learning Rate (CosineAn-

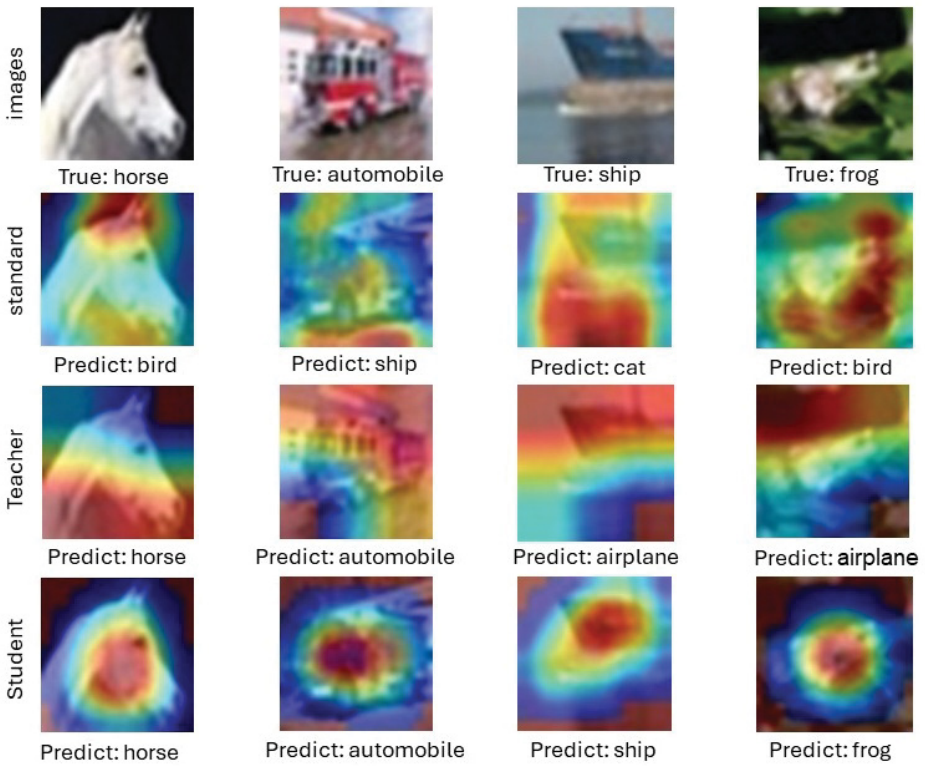


Figure 4.2: This diagram shows the heat maps of different method trained WideResnet. The heat maps are create by the GradCAM method. The teacher network is trained with 8x8 input vectors

nealingLR) scheduler and data normalization to improve performance. Unless otherwise specified, we also use auto-augmentation [57] on both the student and teacher networks to improve performance. All experiments were trained for 200 epochs.

The datasets we used in our experiments are CIFAR-10 and CIFAR-100. We also tested the CIFAR-10-C and CIFAR-100-C datasets to evaluate our performance on common corruptions and perturbations [26]. All attack methods are from TorchAttack; for PGD, we set ϵ to 8 and the number of steps to 20. AutoAttack uses the default settings. All experiments were performed on an RTX 3090.

WideResNet	Vanilla T-S		Pooling Image		Pooling feature		Large Stride Kernel	
Pooling ratio	-		4		4		-	
stride size	-		1		1		4	
	Teacher	Student	Teacher	Student	Teacher	Student	Teacher	Student
Clean	82.79	83.69	56.23	62.46	62.99	68.81	67.74	73.2
PGD	4.99	6.30	7	11.3	12.31	17.96	14.64	20.55

Table 4.1: Model performance under different dimension reduction implementations on CIFAR-100. The backbone is WideResNet.

4.3.2 Experimental Results

Dimension Reduction Implementations

In this experiment, we implement different dimensionality reduction to create feature vectors with the same low dimension in the teacher model. From Table 4.1, we observe that even if we directly down-sample the input image and feed it to the teacher network, it can still create a smooth feature space and improve model's adversarial robustness. Here, the vanilla T-S model in the table represents the conventional teacher-student distillation model with the identical architecture. Based on this experiment, we find that different dimension reduction implementations have varying feature extraction efficiencies.

Varying Latent Dimensions

Table 4.2 shows the performance of the different reduced dimensions. The feature space for all the student networks in this experiment is 32x32, which is the standard feature dimension for the Cifar100 dataset. The number in the first row of the table indicates the feature dimension that is used by the teacher network.

The results of this experiment demonstrate that with the decrease in the dimension of the feature vector, the ratio of robustness accuracy to clean accuracy gradually increases. This indicates that the lower the feature dimension, the smoother the feature space.

	32x32		16x16		8x8		4x4	
	Teacher	Student	Teacher	Student	Teacher	Student	Teacher	Student
Clean	82.79	83.69	78.41	80.34	67.74	73.2	49.32	56.44
PGD	4.99	6.3	10.58	14.03	14.64	20.55	14.23	20.58

Table 4.2: Illustrate performance of different target dimension in Cifar100 on Resnet50. 32x32, 16x16, 8x8, 4x4 means the dimension of the input vector of teacher network.

	KL	MSE	MSE+KL	KL+CE
clean	74.97	71.42	73.2	80.48
PGD	19.38	20.72	20.55	15.34

Table 4.3: Model performance under different knowledge distillation (KD) loss function

Distillation Loss Function

Table 4.3 reports the performance of the student network under different knowledge distillation loss functions. The results show that the $MSE + KL$ loss function is the most balanced, achieving the best average performance in terms of both robustness accuracy and clean accuracy.

Backbone architectures

To test adversarial robustness, we use one sixteenth (1/16) of the original feature dimension to train the teacher network. In the CNN network, we set the stride of the input layer to 4, which is originally 1 for CIFAR-10 and CIFAR-100. To maintain the same convergence speed, we keep the original overlap number in the CNN consistent by setting the kernel size to 12 and the padding size to 4. For the Vision Transformer, we use $vit\_base\_patch32\_224$ as the teacher and $vit\_base\_patch8\_224$ as the student network; the difference in input dimension is also exactly 16 times.

Table 4.4 shows the results of the experiment. Based on the results, we find that our method achieves great robustness accuracy across different models, datasets, and attack methods. Especially in CIFAR-10, WideResNet gains

Method		CIFAR10			CIFAR100		
		Clean	PGD20	AutoAttack	Clean	PGD20	AutoAttack
WideResnet	Baseline	97.11	7.50	6.06	82.79	4.99	4.02
	Ours	93.3	43.56	42.4	73.2	20.55	19.07
Resnet50	Baseline	95.520	16.72	14.57	77.13	8.65	7.66
	Ours	85.19	38.5	36.45	60.55	19.48	17.47
Vit	Baseline	94.06	20.39	19.51	76.49	12.33	11.66
	Ours	86.66	38.63	37.61	61.07	22.43	21.29
Madry et al. 2018	WideResnet	88.83	48.68	45.83	62.07	23.64	22.29

Table 4.4: Model performance and Robustness against Adversarial attacks (%)

43.56% robust accuracy, which is close to the classic adversarial training method by Madry et al. [41].

Robustness to Common Corruption

Base on Table 4.5, we find that common corruption achieves the best performance when the feature space is reduced to the 16**x**16 dimension. However, when the dimension is further reduced, even though the ratio of robustness accuracy to clean accuracy continues to increase, the absolute value begins to decrease. We believe that excessive reduction in dimensionality may negatively impact data extraction.

4.3.3 Extension: Finetune for Further Improvement

During our experiment, we found an interesting phenomenon. We used the teacher network to generate pseudo-labels for the input images. Since the low-dimension teacher network cannot achieve 99% training accuracy, the pseudo-labels contain some incorrect predictions. We then used these pseudo-labels to train a standard ResNet50 network with standard feature dimensions. As reported in Table 4.6, it shows great adversarial robustness.

We believe the nature of the robustness in the low-dimensional feature space is due to the exclusion of hard cases. This revisits the idea of learning easy cases

	CIFAR10		CIFAR100	
	Clean	-C	Clean	-C
Baseline	97.10	88.23	82.79	66.36
32x32	97.14	88.13	82.52	67.31
16x16	95.97	89.17	80.34	67.67
8x8	93.29	88.64	73.2	65.31

Table 4.5: Model performance and Robustness against Common Corruptions with different Teacher feature space. -C in the table represents the corresponding Common Corruptions and Perturbations set. 32x32, 16x16, 8x8 means the dimension of the input vector of teacher network.

	S_{iter1}	S_{iter2}	S_{iter3}
Clean	85	86.64	86.83
PGD200	38.86	39.77	40.73

Table 4.6: Illustrate performance of distillate pseudo label

first and then tackling harder cases later. We then added high-temperature cross-entropy loss with the true labels to the distillation loss. We found that if the temperature in the cross-entropy is sufficiently high, the network can learn the hard cases slowly. The S_{iter2} is the student network learning from this modified loss for 200 epochs, and $S_{iter3}$for 400 epochs.

Therefore, we consider that the neural network itself tends to converge to a global optimal point, which is smooth and robust. However, if the network focuses too much on the hard cases, it uses redundant information and converges to a local optimal point. We also found that convergence to a global optimal point is difficult and may take thousands of epochs to teach the neural network about the hard cases.

4.4 Conclusion

In this chapter, we study the relationship between the dimension of the input feature vector and adversarial vulnerability. We found that high-dimensional information in the input feature vector may act as strong noise, making the adversarial network vulnerable. Inspired by this discovery, we proposed a self-distillation noisy student training method that allows the network to gradually learn from such noise. This method helps neural networks focus on essential features. The results of extensive experiments show that this method has great performance and adaptability.

Chapter 5

Conclusion and Future Work

In this chapter, we summarize our content and contributions, discuss the results obtained, and suggest potential future work that could be developed based on this thesis.

5.1 Conclusion

Adversarial robustness has become one of the most important areas in the field of deep learning. The wider application of deep neural networks has increased the demand for the security of these networks. As one of the most efficient and powerful methods to measure the robustness of a model, adversarial examples have received widespread attention since they were first proposed in 2014. However, to date, no fast and effective defense method has been proposed to guard against these specially designed perturbations. Current methods either require extremely high time consumption or have very limited performance and are based on specific attack assumptions. In this thesis, we proposed two novel methods that could improve the adversarial robustness of neural networks without using adversarial examples.

In the first half of this thesis, we presented a novel perspective on the cross-entropy loss function to help DNNs achieve great adversarial performance against untargeted adversarial attacks with no additional time cost. We first theoretically analyzed how temperature affects the optimization process during training. We found that a high-temperature cross-entropy function guides the model to learn from different training samples equally. By increasing the

temperature during the training process, we found that neural networks achieve incredible performance against untargeted adversarial attacks. We believe this is due to the debiased optimization strategy. All non-target classes have nearly the same distance from the target class, which greatly reduces the gradient update strength. Experimental results show that our model achieves great success in defending against untargeted attacks.

In the second half of this dissertation, we presented our Low Dimension Distillation (LDD) method to enhancing the adversarial robustness of neural networks while preserving generalizability. By leveraging a teacher-student framework where a teacher model processes low-dimensional input data to enhance robustness, and a student model directly operates on original high-dimensional data, LDD achieves significant improvements in robust accuracy without the need for adversarial examples or additional training data. Through the distillation process, the student model effectively inherits the robustness imparted by the teacher model.

5.2 Future Work

In this section, we suggest possible directions for future research based on our work. First, "ensemble multiple models with different temperatures to defend against targeted attacks" is an interesting perspective for future study. In this thesis, we focused on using a high-temperature model to protect against untargeted attacks. However, for targeted attacks, a single model may not provide a complete solution. We believe that combining multiple models with different temperatures could be a great solution. This combination might be able to reduce the gradient strength between the true target class and the adversarial 'targeted' class.

Another interesting perspective for future research is "training deep neural networks with adaptive temperatures." According to our convergence analysis, we found that high temperatures can achieve debiased convergence, while low temperatures focus on hard cases. Although we believe that starting with hard cases is generally not ideal for learning, focusing on more difficult parts after

achieving a certain degree of convergence may lead to better results.

References

[1] A. Agarwala, J. Pennington, Y. N. Dauphin, and S. S. Schoenholz, "Temperature check: Theory and practice for training models with softmax-cross-entropy losses," *CoRR*, vol. abs/2010.07344, 2020. arXiv: 2010.07344.

[2] S. et al., "Label smoothing and logit squeezing: A replacement for adversarial training?" *arXiv preprint*, 2019.

[3] M. Alfarra, J. C. Pérez, A. Thabet, A. Bibi, P. H. Torr, and B. Ghanem, "Combating adversaries with anti-adversaries," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, 2022, pp. 5992–6000.

[4] M. Andriushchenko, F. Croce, N. Flammarion, and M. Hein, "Square attack: A query-efficient black-box adversarial attack via random search," in *European conference on computer vision*, Springer, 2020, pp. 484–501.

[5] F. Assion, P. Schlicht, F. Greßner, *et al.*, "The attack generator: A systematic approach towards constructing adversarial attacks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2019, pp. 0–0.

[6] P. Awasthi, V. Chatziafratis, X. Chen, and A. Vijayaraghavan, "Adversarially robust low dimensional representations," in *Conference on Learning Theory*, PMLR, 2021, pp. 237–325.

[7] P. Awasthi, H. Jain, A. S. Rawat, and A. Vijayaraghavan, "Adversarial robustness via robust low rank representations," *Advances in Neural Information Processing Systems*, vol. 33, pp. 11 391–11 403, 2020.

[8] C. M. Bishop and N. M. Nasrabadi, *Pattern recognition and machine learning*. Springer, 2006, vol. 4.

[9] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *2017 ieee symposium on security and privacy (sp)*, IEEE, 2017, pp. 39–57.

[10] S. Cascianelli, R. Bello-Cerezo, F. Bianconi, *et al.*, "Dimensionality reduction strategies for cnn-based classification of histopathological images," in *Intelligent Interactive Multimedia Systems and Services 2017 10*, Springer, 2018, pp. 21–30.

[11] N. Chattopadhyay, S. Chatterjee, and A. Chattopadhyay, "Robustness against adversarial attacks using dimensionality," in *International Conference on Security, Privacy, and Applied Cryptography Engineering*, Springer, 2021, pp. 226–241.

[12] P.-Y. Chen, H. Zhang, Y. Sharma, J. Yi, and C.-J. Hsieh, "Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models," in *Proceedings of the 10th ACM workshop on artificial intelligence and security*, 2017, pp. 15–26.

[13] F. Croce, M. Andriushchenko, V. Sehwag, *et al.*, "Robustbench: A standardized adversarial robustness benchmark," *arXiv preprint arXiv:2010.09670*, 2020.

[14] F. Croce and M. Hein, *Minimally distorted adversarial examples with a fast adaptive boundary attack*, 2020. arXiv: 1907.02044 [cs.LG].

[15] F. Croce and M. Hein, "Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks," in *International conference on machine learning*, PMLR, 2020, pp. 2206–2216.

[16] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," *CoRR*, vol. abs/1810.04805, 2018. arXiv: 1810.04805.

[17] E. Dohmatob, "Generalized no free lunch theorem for adversarial robustness," in *International Conference on Machine Learning*, PMLR, 2019, pp. 1646–1654.

[18] P. Domingos, "A few useful things to know about machine learning," *Communications of the ACM*, vol. 55, no. 10, pp. 78–87, 2012.

[19] L. Engstrom, A. Ilyas, and A. Athalye, "Evaluating and understanding the robustness of adversarial logit pairing," *arXiv preprint arXiv:1807.10272*, 2018.

[20] A. Fawzi, H. Fawzi, and O. Fawzi, "Adversarial vulnerability for any classifier," *Advances in neural information processing systems*, vol. 31, 2018.

[21] J. Gilmer, L. Metz, F. Faghri, *et al.*, "Adversarial spheres," *arXiv preprint arXiv:1801.02774*, 2018.

[22] C. Godfrey, H. Kvinge, E. Bishoff, *et al.*, "How many dimensions are required to find an adversarial example?" In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 2352–2359.

[23] I. J. Goodfellow, J. Shlens, and C. Szegedy, *Explaining and harnessing adversarial examples*, 2015. arXiv: 1412.6572 [stat.ML].

[24] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, "On calibration of modern neural networks," in *Proceedings of the 34th International Conference on Machine Learning*, D. Precup and Y. W. Teh, Eds., ser. Proceedings of Machine Learning Research, vol. 70, PMLR, Jun. 2017, pp. 1321–1330.

[25] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[26] D. Hendrycks and T. Dietterich, "Benchmarking neural network robustness to common corruptions and perturbations," *arXiv preprint arXiv:1903.12261*, 2019.

[27] G. Hinton, O. Vinyals, and J. Dean, *Distilling the knowledge in a neural network*, 2015. arXiv: 1503.02531 [stat.ML].

[28] P. Hou, J. Han, and X. Li, "Improving adversarial robustness with self-paced hard-class pair reweighting," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, 2023, pp. 14 883–14 891.

[29] S. Kanai, M. Yamada, S. Yamaguchi, H. Takahashi, and Y. Ida, "Constraining logits by bounded function for adversarial robustness," in *2021 International Joint Conference on Neural Networks (IJCNN)*, IEEE, 2021, pp. 1–8.

[30] G. J. Karanikas, "Adversarial robustness in high-dimensional deep learning," 2021.

[31] M. Khoury and D. Hadfield-Menell, "On the geometry of adversarial examples," *arXiv preprint arXiv:1811.00525*, 2018.

[32] T. Kim, J. Oh, N. Kim, S. Cho, and S.-Y. Yun, "Comparing kullback-leibler divergence and mean squared error loss in knowledge distillation," Aug. 2021, pp. 2628–2635. DOI: 10.24963/ijcai.2021/362.

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

[34] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, 2012.

[35] M. Kull, M. Perello Nieto, M. Kängsepp, T. Silva Filho, H. Song, and P. Flach, "Beyond temperature scaling: Obtaining well-calibrated multi-class probabilities with dirichlet calibration," in *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., vol. 32, Curran Associates, Inc., 2019.

[36] A. Kumar, S. Sarawagi, and U. Jain, "Trainable calibration measures for neural networks from kernel mean embeddings," in *Proceedings of the 35th International Conference on Machine Learning*, J. Dy and A. Krause, Eds., ser. Proceedings of Machine Learning Research, vol. 80, PMLR, Oct. 2018, pp. 2805–2814.

[37] A. Kurakin, I. J. Goodfellow, and S. Bengio, "Adversarial machine learning at scale," in *International Conference on Learning Representations*, 2017.

[38] B. Lakshminarayanan, A. Pritzel, and C. Blundell, "Simple and scalable predictive uncertainty estimation using deep ensembles," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, *et al.*, Eds., vol. 30, Curran Associates, Inc., 2017.

[39] Y. Le and X. Yang, "Tiny imagenet visual recognition challenge," *CS 231N*, vol. 7, no. 7, p. 3, 2015.

[40] M. Lee and D. Kim, "Robust evaluation of diffusion-based adversarial purification," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 134–144.

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

[42] M. Minderer, J. Djolonga, R. Romijnders, *et al.*, "Revisiting the calibration of modern neural networks," in *Advances in Neural Information Processing Systems*, M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, Eds., vol. 34, Curran Associates, Inc., 2021, pp. 15 682–15 694.

[43] A. van den Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," *CoRR*, vol. abs/1807.03748, 2018. arXiv: 1807.03748.

[44] N. Papernot, P. McDaniel, and I. Goodfellow, "Transferability in machine learning: From phenomena to black-box attacks using adversarial samples," *arXiv preprint arXiv:1605.07277*, 2016.

[45] G. Pereyra, G. Tucker, J. Chorowski, L. Kaiser, and G. Hinton, *Regularizing neural networks by penalizing confident output distributions*, 2017. [Online]. Available: https://openreview.net/forum?id=HkCjNI5ex.

[46] B. Prach and C. H. Lampert, "Almost-orthogonal layers for efficient general-purpose lipschitz networks," in *European Conference on Computer Vision*, Springer, 2022, pp. 350–365.

[47] Z. Qian, S. Zhang, K. Huang, Q. Wang, R. Zhang, and X. Yi, "Improving model robustness with latent distribution locally and globally," *arXiv preprint arXiv:2107.04401*, 2021.

[48] P. Samangouei, M. Kabkab, and R. Chellappa, "Defense-gan: Protecting classifiers against adversarial attacks using generative models," in *International Conference on Learning Representations*, 2018.

[49] B. Schölkopf, C. Burges, and A. Smola, *Advances in Kernel Methods - Support Vector Learning*. Cambridge, MA: MIT Press, 1999.

[50] H. Shah, K. Tamuly, A. Raghunathan, P. Jain, and P. Netrapalli, "The pitfalls of simplicity bias in neural networks," *Advances in Neural Information Processing Systems*, vol. 33, pp. 9573–9585, 2020.

[51] S. Singla, B. Nushi, S. Shah, E. Kamar, and E. Horvitz, "Understanding failures of deep networks via robust feature extraction," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 12 853–12 862.

[52] C. Szegedy, W. Zaremba, I. Sutskever, *et al.*, "Intriguing properties of neural networks," *arXiv preprint arXiv:1312.6199*, 2013.

[53] T. Tanay and L. Griffin, "A boundary tilting persepective on the phenomenon of adversarial examples," *arXiv preprint arXiv:1608.07690*, 2016.

[54] D. Tsipras, S. Santurkar, L. Engstrom, A. Turner, and A. Madry, "Robustness may be at odds with accuracy," *arXiv preprint arXiv:1805.12152*, 2018.

[55] L. Van der Maaten and G. Hinton, "Visualizing data using t-sne.," *Journal of machine learning research*, vol. 9, no. 11, 2008.

[56] F. Wang and H. Liu, "Understanding the behaviour of contrastive loss," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2021, pp. 2495–2504.

[57] J. Wang, L. Perez, *et al.*, "The effectiveness of data augmentation in image classification using deep learning," *Convolutional Neural Networks Vis. Recognit*, vol. 11, no. 2017, pp. 1–8, 2017.

[58] T. Wang and P. Isola, "Understanding contrastive representation learning through alignment and uniformity on the hypersphere," in *Proceedings of the 37th International Conference on Machine Learning*, H. D. III and A. Singh, Eds., ser. Proceedings of Machine Learning Research, vol. 119, PMLR, 13–18 Jul 2020, pp. 9929–9939.

[59] Z. Wang, T. Pang, C. Du, M. Lin, W. Liu, and S. Yan, "Better diffusion models further improve adversarial training," in *International Conference on Machine Learning*, PMLR, 2023, pp. 36 246–36 263.

[60] Z. Wu, Y. Xiong, S. X. Yu, and D. Lin, "Unsupervised feature learning via non-parametric instance discrimination," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2018.

[61] Z. Yang, Z. Li, Y. Gong, *et al.*, "Rethinking knowledge distillation via cross-entropy," *arXiv preprint arXiv:2208.10139*, 2022.

[62] J. Yoon, S. J. Hwang, and J. Lee, "Adversarial purification with score-based generative models," in *International Conference on Machine Learning*, PMLR, 2021, pp. 12 062–12 072.

[63] H. Zhang, Y. Yu, J. Jiao, E. Xing, L. E. Ghaoui, and M. Jordan, "Theoretically principled trade-off between robustness and accuracy," in *Proceedings of the 36th International Conference on Machine Learning*, K. Chaudhuri and R. Salakhutdinov, Eds., ser. Proceedings of Machine Learning Research, vol. 97, PMLR, Sep. 2019, pp. 7472–7482.

[64] L. Zhang, J. Song, A. Gao, J. Chen, C. Bao, and K. Ma, "Be your own teacher: Improve the performance of convolutional neural networks via self distillation," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 3713–3722.

[65] H. Zhao, X. Qi, X. Shen, J. Shi, and J. Jia, "Icnet for real-time semantic segmentation on high-resolution images," in *Proceedings of the European Conference on Computer Vision (ECCV)*, Sep. 2018.

[66] Y. Zhao, T. Pang, C. Du, *et al.*, "On evaluating adversarial robustness of large vision-language models," *Advances in Neural Information Processing Systems*, vol. 36, 2024.