Universität Siegen

Studienarbeit submitted
within the Master's Program
Masters in Mechatronics

Semi-supervised Adversarially Robust Distillation with Limited Labels

Submitted to Computer Vision Group

Universität Siegen

Author: Pavan Kumar Medarametla

(Matriculation No. 1633647)

Supervisor: Prof. Dr. Michael Möller

Secondary Supervisor: Dr. Kanchana Vaishnavi Gandikota

August 17, 2024

Thesis advisor: Prof. Dr. Michael Möller

Pavan Kumar Medarametla

Semi-supervised Adversarially Robust Distillation with Limited Labels

ABSTRACT

This research focuses on improving the resistance of deep learning models to adversarial attacks by using a semi-supervised learning method that requires only a small amount of labeled data. Conventional methods for training models to withstand such attacks typically need large datasets with labels, which can be hard to obtain. Our method uses a teacher-student model setup, where a teacher model that operates semi-supervised helps train a student model. This training allows the student model to learn robust features from both labeled data and additional data labeled by the teacher. We have developed and continuously improved a modified version of the TRADES loss function, which includes various elements to maintain a balance between robustness and accuracy on non-altered images. Through a series of experiments under conditions of limited labeled data, we found that our method enhances the model's resistance to adversarial attacks, while maintaining its clean accuracy.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Neural networks used for image classification, are susceptible to subtle perturbations that can alter their classification decisions despite being imperceptible to humans. These manipulated images, called as adversarial examples, can be created with varying levels of access to the model's details, posing significant security risks [2]. Adversarial examples represent a genuine threat as they can mislead classification models with ease. To counteract these threats, several adversarial learning methods have been developed [4, 5, 6].

Traditionally, building good prediction models requires extensive labeled data, which is often expensive and labor-intensive to gather. Conversely, acquiring unlabeled data is generally more feasible. Semi-supervised learning, which focuses on using unlabeled data alongside labeled data, offers a viable solution [7, 8]. Given the reliance of adversarial training on labeled data, combining it with semi-supervised learning techniques could enhance the reliability of deep learning models.

Our study aims to improve the robustness of models against adversarial attacks within a semi-supervised framework, especially when there is a scarcity of labeled data. We use a teacher-student model setup, a form of knowledge distillation, where the student model learns from pseudo-labels created by a teacher model trained with the FixMatch algorithm [1]. Additionally, we propose an enhanced version of the TRADES adversarial training method, tailored for situations with limited labeled data [3]. Our approach involves distinguishing between confident and less-confident samples based on the predictions from both the teacher and student models, employing different loss terms for these samples, and refining consistency checks between clean and adversarial examples. These strategic improvements are designed to enhance the training process, improving the model's performance and stability. As a result, our models make more reliable and consistent predictions. Through these tailored adjustments, our goal is to not only enhance the model's resistance against adversarial threats but also to improve its overall accuracy and reliability, ensuring effective performance even under adversarial conditions.

Chapters Outline: Chapter 2 starts with an overview of semi-supervised learning, discussing key concepts like consistency regularization and pseudo-labeling. We delve into the FixMatch algorithm and its role in semi-supervised learning, followed by a detailed look at adversarial examples, different adversarial attack methods such as FGSM

and PGD, and adversarial training techniques including TRADES. Chapter 3 describes our methodology, covering the generation of pseudo-labels using a teacher model trained with FixMatch, and details the training of the student model, which incorporates consistency regularization and a modified TRADES method to boost adversarial robustness. Chapter 4 outlines our experimental setup, including dataset choices, model architecture, and hyper-parameter configurations. We present our experimental results on CIFAR-10, showcasing the enhancements achieved through our approach. Chapter 5 summarizes our findings, discusses improvements in adversarial robustness, and outlines future research directions.

Chapter 2

Background

2.1 Semi-supervised Learning

Semi-supervised learning (SSL) serves as a hybrid approach positioned between the fully labeled datasets required by supervised learning and the label-free setting of unsupervised learning. It's especially useful in situations where labeling large amounts of data is either expensive or impractical. By utilizing a small set of labeled data with a larger pool of unlabeled data, SSL can enhance the accuracy and ability of models to generalize, effectively utilizing the data.

The main strategy of SSL is to utilize the unlabeled data to guide the learning process. The model uses this data to make initial predictions on unlabeled data, which are then treated as new data points, or pseudo-labels, to train the model further. This cycle repeats, allowing the model to continually refine its predictions, thus reducing its dependence on large sets of labeled data.

In summary, SSL offers a practical way to use both labeled and unlabeled data efficiently. It lessens the reliance on labeled data and enables learning from the broader dataset. This adaptability makes SSL a favored choice for enhancing model performance in situations where resources are limited.

2.2 Consistency regularization and Pseudo-Labeling

Consistency regularization is an important component of recent state-of-the-art semi-supervised learning (SSL) algorithms. This technique uses unlabeled data by assuming that the model should produce similar predictions for perturbed versions of the same image[9].

Pseudo-labeling is a semi-supervised learning strategy that enhances the effectiveness of deep learning models by incorporating unlabeled data. Initially, a model is trained exclusively with the labeled data at hand. Once trained, this model is then used to create labels for the unlabeled data, effectively treating these predicted labels as actual labels. This strategy has been thoroughly investigated and validated for its effectiveness

in numerous studies, such as those by Lee et al. [7] and Arazo et al. [10]. Furthermore, Min et al. [11] provide a theoretical foundation explaining the success of pseudo-labeling in semi-supervised scenarios. The technique has proven particularly effective in domains like computer vision and pattern recognition, consistently achieving top-tier results across various datasets.

2.3 FixMatch

FixMatch is a semi-supervised learning algorithm that uses both labeled and unlabeled data to improve model performance. The approach relies on the consistency regularization technique and combines it with pseudo-labeling to create a robust training framework [1]. The overall pipeline of FixMatch can be summarized in figure 2.1:

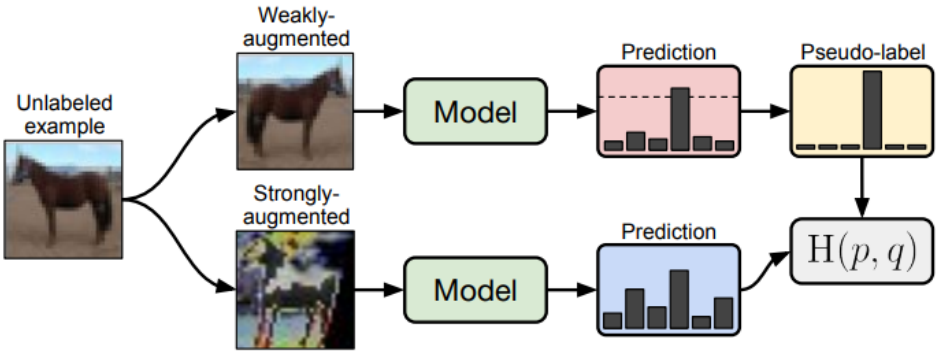

Figure 2.1: Diagram of FixMatch: A weakly augmented image is sent as input to the model to make a prediction. If that prediction is above a threshold, then aligns these predictions with those from a strongly-augmented version using cross-entropy loss, as shown in [1].

The training consists of the following steps:

1. **Supervised Learning:** The model is trained on labeled images using cross-entropy loss. This provides the foundation for the model's ability to make accurate predictions.
2. **Weak and Strong Augmentation:** For each unlabeled image, both weak and strong augmentations are applied. The weakly augmented image is passed through the model to obtain predictions.
3. **Pseudo-Label Generation:** The class with the highest predicted probability from the weakly augmented image is taken as the pseudo-label if it surpasses a predefined confidence threshold. This pseudo-label serves as the ground truth for the strongly augmented image.

4. **Consistency Regularization:** The strongly augmented image is then passed through the model, and its predictions are compared to the pseudo-label using cross-entropy loss. This encourages the model to produce consistent predictions for different augmentations of the same image.
5. **Training:** Batches of labeled and unlabeled images are prepared, with a hyperparameter μ determining the ratio of unlabeled to labeled images. The losses from the labeled and unlabeled images are combined, with the total loss for a batch being a weighted sum of the supervised and unsupervised losses.
6. **Similarity with Curriculum Learning:** As the training progresses, the model becomes more confident in its predictions. Initially, the model is trained mostly on labeled data because the predictions on unlabeled data do not meet the confidence threshold. Over time, as the model's confidence grows, more pseudo-labels are generated, incorporating more unlabeled data into the training process. This natural progression mirrors curriculum learning, where initially, the model trains on confidently labeled data and gradually includes more challenging examples as it becomes more adept.

Advantages of FixMatch By generating pseudo-labels for unlabeled data and enforcing consistency through strong augmentations, FixMatch effectively utilizes available data to enhance model performance. FixMatch incorporates a natural curriculum learning strategy, which enhances the model's ability to generalize to diverse and complex real-world scenarios. FixMatch provides a simple yet powerful framework for semi-supervised learning, requiring minimal additional hyperparameters and achieving state-of-the-art results on various benchmarks.

2.4 Adversarial Attacks

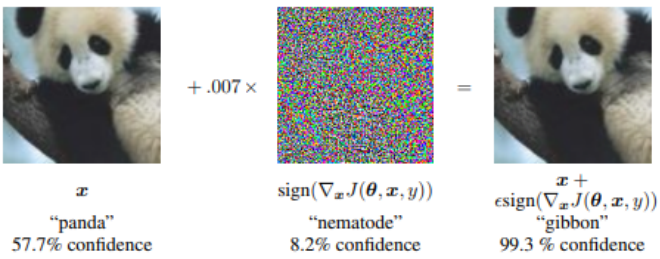

Figure 2.2: Example of adversarial example. Figure taken from [2]

Adversarial examples are created by intentionally introducing small changes, such as noise or other perturbations, to data samples [2]. These modifications are specifically designed to deceive the model into making incorrect predictions confidently. Classifying

these manipulated samples accurately remains a significant challenge for various machine learning techniques, especially neural networks. The impact of an adversarial example on a model is depicted in the figure 2.2, as shown in [2]. Essentially, adversarial examples are altered inputs crafted to confuse the model into errors. Even minor, meticulously designed perturbations can lead state-of-the-art classifiers to produce incorrect results [12, 13].

2.4.1 Creating Adversarial Examples

In deep learning, a model, typically a neural network denoted as h_θ, is designed to map input data X to output predictions across k classes. Here, θ encapsulates all trainable parameters like weights and biases. The objective during training is to minimize the loss function $\ell(h_\theta(x), y)$, where y is the true label for the input x. This loss quantifies the discrepancy between the model's prediction and the actual label.

The optimization task for a training dataset is given by:

$$\min_\theta \frac{1}{m} \sum_{i=1}^{m} \ell(h_\theta(x_i), y_i) \tag{2.1}$$

During each training iteration, parameters are updated using gradient descent, adjusting θ in the direction that reduces the batch loss:

$$\theta := \theta - \frac{\alpha}{|B|} \sum_{i \in B} \nabla_\theta \ell(h_\theta(x_i), y_i) \tag{2.2}$$

where B is a batch of training samples, and α is the learning rate.

Adversarial examples are crafted by subtly modifying the input data to maximize the prediction error. Unlike typical training that adjusts parameters to minimize loss, adversarial training involves tweaking the input to increase the loss, aiming to fool the model:

$$\max_{\delta \in \Delta} \ell(h_\theta(x + \delta), y) \tag{2.3}$$

Here, δ represents small perturbations within an allowable set Δ, ensuring the modified input $x + \delta$ remains visually similar to the original x. Commonly, Δ is chosen as an ℓ_∞ ball defined by $\Delta = \{\delta : \|\delta\|_\infty \leq \epsilon\}$, where ϵ is small enough to keep changes imperceptible.

This method reveals the vulnerability of neural networks to slight, often imperceptible changes to input data, exposing a significant challenge in achieving robustness. By understanding and mitigating these vulnerabilities, we enhance the model's reliability against adversarial attacks[2, 12].

2.4.2 The Fast Gradient Sign Method (FGSM)

The Fast Gradient Sign Method (FGSM) is a technique designed to generate adversarial examples. Consider a model h_θ and an input x paired with its true label y. The goal of FGSM is to perturb x by a small vector δ such that the perturbed input $x + \delta$ leads to an incorrect output from the model. The process involves calculating the gradient of the loss function with respect to the input x, which indicates the direction in which a slight change would increase the loss most significantly.

Initially, the gradient g is computed as:

$$g := \nabla_\delta \ell(h_\theta(x + \delta), y) \tag{2.4}$$

assuming that we start with $\delta = 0$, the gradient with respect to x is:

$$g := \nabla_x \ell(h_\theta(x), y) \tag{2.5}$$

To effectively use this gradient to increase the loss, δ is adjusted in the direction of g using a step size α:

$$\delta := \delta + \alpha g \tag{2.6}$$

This adjustment is then projected back onto the ℓ_∞ norm ball to ensure that the perturbations do not exceed a predefined threshold ϵ, ensuring they remain imperceptible:

$$\delta := \text{clip}(\alpha g, [-\epsilon, \epsilon]) \tag{2.7}$$

For large enough α, the direction of g becomes less significant, leading to an update that depends solely on the sign of g:

$$\delta := \epsilon \cdot \text{sign}(g) \tag{2.8}$$

This step simplifies the update process, focusing solely on the direction that increases the loss, regardless of the magnitude of the gradient. FGSM thus provides a straightforward and computationally efficient method to generate adversarial examples.

2.4.3 Projected Gradient Descent (PGD)

Projected Gradient Descent (PGD) is an iterative technique for generating adversarial examples. It builds on simpler approaches like the Fast Gradient Sign Method (FGSM) by using several small gradient steps. This enhancement generates more potent adversarial perturbations.

The procedure for PGD involves the following steps:

1. Initialize the perturbation δ to zero or a small random value within the allowable perturbation set.

2. For each iteration, compute the gradient of the loss with respect to the perturbed input $x + \delta$. This gradient indicates the direction in which the input should be altered to maximize the loss, thereby leading to a misclassification.
3. Update δ by moving it slightly in the direction of the gradient. This step size, denoted by α, is small to ensure the perturbation remains subtle.
4. Project δ back onto the perturbation set Δ to ensure it does not exceed the predefined bounds. For instance, if using the L_∞ norm, ensure the magnitude of δ does not exceed ϵ, which is the maximum allowed perturbation.

This method repeatedly adjusts δ, ensuring each adjustment does not push δ beyond the acceptable limits of ϵ, maintaining the adversarial example's closeness to the original input while still being effective at causing misclassification. The result is a more refined approach to adversarial training.

2.5 Adversarial Training

Adversarial training is recognized as one of the most effective methods for enhancing the resilience of deep learning models against adversarial attacks [13].

The core concept behind adversarial training involves the inclusion of adversarial examples into the training dataset during each cycle of model training. This practice enables models trained in this way to perform better when encountering adversarial attacks compared to models trained with standard methods. The effectiveness of adversarial training lies in its training approach, which familiarizes the model with perturbed inputs, thereby increasing its resistance to such disruptions.

The process of adversarial training can be described mathematically as a min-max optimization problem [6][14]:

$$\min_{\theta} \frac{1}{|S|} \sum_{(x,y)\in S} \max_{\|\delta\|\leq\epsilon} \ell(h_\theta(x+\delta), y) \tag{2.9}$$

In this formulation, the goal is to train the model, parameterized by θ, to minimize the loss ℓ over a set S of training examples (x, y). Concurrently, the model must withstand input perturbations δ that aim to maximize this loss, but are constrained by a perturbation limit ϵ. This dual objective forces the model to optimize for accuracy while contending with potential input perturbations designed to increase error, thereby enhancing the model's capabilities against such adversarial conditions.

2.6 TRADES: A Strategy for Balancing Accuracy and Robustness

TRADES (TRadeoff-inspired Adversarial DEfense via Surrogate-loss minimization) is a method designed to enhance the robustness of models against adversarial attacks while maintaining accuracy. The key to TRADES is its unique formulation of loss, which balances between the model's performance on regular examples and its resilience to adversarial perturbations.

The formulation of the TRADES loss function can be described as follows:

$$\min_{h} \mathbb{E}\left\{\ell(h(x), y) + \max_{x' \in B(x,\epsilon)} \frac{\ell(h(x), h(x'))}{\lambda}\right\} \tag{2.10}$$

Here, h represents the model, x is the input, y is the true label, x' denotes a perturbed version of x, and λ is a tuning parameter that helps balance the two terms of the loss function. The first term, $\ell(h(x), y)$, minimizes the error on the natural (unperturbed) inputs, ensuring the model performs well under normal conditions. The second term, $\max_{x' \in B(x,\epsilon)} \frac{\ell(h(x), h(x'))}{\lambda}$, acts as a regularizer that promotes model stability by encouraging consistent predictions across adversarial inputs. This term effectively pushes the model's decision boundary away from the data points to reduce the 'boundary error', which arises when data points lie too close to the decision boundary.

In practice, this adversarial training method modifies the traditional training approach by not only teaching the model to predict correctly but also to resist manipulation. The λ parameter is crucial as it dictates the trade-off between focusing on standard accuracy and enhancing adversarial robustness. Lower values of λ prioritize accuracy, while higher values strengthen robustness.

The visual representation of TRADES in action, depicted in figure 2.3, contrasts the decision boundaries formed by standard and adversarial training methods. The expanded decision boundary demonstrated by TRADES shows its effectiveness in creating a buffer zone around data points, which helps in mitigating the impact of adversarial attacks.

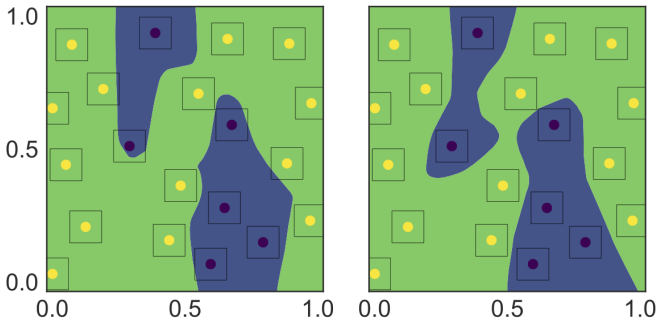

Figure 2.3: Illustration of decision boundaries: the left figure shows the boundary formed by natural training, while the right figure displays the expanded boundary due to TRADES adversarial training. The orange dotted line in the right figure indicates the original boundary from the left figure. Figure adapted from [3]

Chapter 3

Methodology

3.1 Adversarially Robust Distillation from Semi-supervised Teacher Model

During the initial training phase, the student model undergoes training using a dataset that consists of N_l labeled samples. This phase spans several warm-up epochs. Simultaneously, the Teacher model, which is trained using the FixMatch algorithm, generates pseudo-labels for N_u unlabeled data points. These pseudo-labels act as substitutes for true labels during the supervised training of the Student model. The dataset utilized for optimization thus comprises $N = N_l + N_u$ samples across various classes.

Initially, we employ the TRADES loss function, which is progressively modified by adding additional terms to enhance its effectiveness:

$$L(\theta) = L_{\text{nat}} + \lambda_1 L_{\text{rob}} \tag{3.1}$$

where:

- L_{nat} is the natural loss, computed as the cross-entropy between the true labels and the model's predictions for clean examples.
- L_{rob} is the robust loss. It employs the Kullback-Leibler divergence to assess and enhance the consistency between the probability distributions of natural and adversarial examples.

To further improve the model's resistance to adversarial attacks, we introduce an additional term, L_{adv}, to the TRADES loss function. This addition aims to ensure that the model not only minimizes the divergence between outputs for clean and adversarial examples but also accurately classifies the adversarial examples themselves.

$$L(\theta) = L_{\text{nat}} + \lambda_1 L_{\text{rob}} + L_{\text{adv}} \tag{3.2}$$

where:

- L_{adv} is the adversarial loss, calculated as the cross-entropy between the true labels and the model's predictions for adversarial examples.

3.1.1 Iterative learning with consistency in Predictions

We introduce a matching criterion to selectively guide the application of each term in Equation (3.2). This criterion ensures consistency between the predictions for clean images and their adversarial counterparts, meaning the model's output should remain the same for both an original image and its perturbed version. This matching process initially includes only those images whose predictions are consistent across both forms, enhancing the training focus on stable examples.

As training progresses, the model increasingly identifies and adapts to images that initially had label flips. This evolution allows more images to meet the consistency criterion over time, gradually expanding the scope of training data influenced by the adversarial robustness terms. This adaptive inclusion enhances model training, enabling it to develop robustness not just to initially consistent examples but also to those where consistency is achieved through iterative learning.

During each epoch, for every batch, images without label flips are selected for inclusion in both the first term (L_{nat}) and the last term (L_{adv}) of the loss function, targeting accurate classification of both clean and adversarial examples. Conversely, the robust loss term (L_{rob}) incorporates every image from the batch, regardless of their matching status. This inclusive approach ensures that all samples contribute to the model's ability to adjust to adversarial perturbations, thus enhancing overall robustness.

Important Note: Regardless of the matching criteria, labeled images from each batch are always included in the first (L_{nat}) and last (L_{adv}) loss terms, in addition to those images that meet the criteria. This approach addresses a crucial challenge during the early training epochs where only a few images may not exhibit label flips. Without a sufficient number of training examples, the model might struggle to learn effectively, potentially hindering the overall learning process. By consistently including labeled images in these terms, we ensure that the model has a stable and robust training foundation right from the start, facilitating better learning process.

3.1.2 Utilizing the Unmatched Images in Training

In this step, to better utilize the unmatched images (those with label flips), we introduce two additional loss terms: $L_{\text{KL-un}}$ and $L_{\text{KL-adv}}$. These terms represent the KL divergence between the model's predictions for unmatched natural and adversarial examples, respectively, and the soft labels provided by the teacher model. The purpose of incorporating these terms is to leverage the information from unmatched examples, ensuring that even when the initial predictions do not align, the model can still learn from the teacher's guidance. By aligning the model's predictions with the teacher's soft labels, these loss terms help improve the overall robustness and accuracy of the model, enabling it to handle both matched and unmatched examples more effectively.

The final loss function, shown in Equation (3.3), incorporates all five loss terms, with

λ_1, λ_2, and λ_3 being the respective weights of these terms.

$$L(\theta) = L_{\text{nat}} + \lambda_1 L_{\text{rob}} + L_{\text{adv}} + \lambda_2 L_{\text{KL-un}} + \lambda_3 L_{\text{KL-adv}} \tag{3.3}$$

- L_{nat} addresses the accurate classification of clean examples.
- L_{rob} ensures the consistency of outputs between natural and adversarial examples.
- L_{adv} focuses on the correct classification of adversarial examples.
- $L_{\text{KL-un}}$ and $L_{\text{KL-adv}}$ utilize the information from unmatched examples by aligning them with the teacher's soft labels, thus enhancing the overall robustness and accuracy of the model.

Chapter 4

Experiments and Results

4.1 Experimental Settings

4.1.1 Dataset

All tests are conducted using the CIFAR-10 dataset, which includes 50,000 training images and 10,000 test images. For our studies, we split the training data into two groups: labeled and unlabeled data, with the unlabeled group being substantially larger. In each training batch, there's an equal mix of labeled and unlabeled samples. Due to the limited size of the labeled data, these samples are repeatedly used across various batches. This repetitive use helps maintain a consistent ratio of labeled to unlabeled data in each batch, which aids in reducing confirmation bias [10] and ensures stable learning conditions by providing regular labeled data during the training process.

4.1.2 Model Selection

For the Teacher model, we've chosen WideResNet-28-5 for its robust performance and ability to generalize across different tasks, as noted in [15]. Its wide architecture allows it to capture detailed features more effectively than standard ResNet models, providing accurate pseudo-labels for the Student model's training. This setup is crucial for enhancing the Student model's learning capabilities from unlabeled data.

In contrast, the Student model uses a streamlined 13-layer Convolutional Neural Network (CNN) [16], which is designed for efficiency. Despite having fewer parameters than WideResNet-28-5, it includes advanced techniques like weight normalization [17], batch normalization [18], dropout [19], and Leaky ReLU activation to balance complexity with computational efficiency. This model choice is driven by the concept of knowledge distillation, where the complex Teacher model imparts its knowledge to the simpler Student model, allowing for faster training and reduced computational demands. This method enhances performance while minimizing operational costs, making the most of the resources available.

4.1.3 Hyper-Parameters Setting

The experiments are performed using Stochastic Gradient Descent (SGD) with momentum of 0.9 and weight decay of 1×10^{-4}. At the start, the learning rate is set to 0.01 for faster convergence and is drastically reduced using the ReduceLROnPlateau scheduler, which decreases the learning rate by a factor of 0.1 at different training epochs. Applying a dropout rate of 0.1 to the fully connected layers in the neural network during training helps prevent overfitting by randomly deactivating 10% of the neurons, promoting better generalization to unseen data.

We modified the TRADES loss by adding additional terms as described in the equation 3.3. PGD adversarial attack was performed on the test data to get the robust accuracy of the robustly trained model with carefully chosen hyper-parameters. The weight given to the KL divergence term, which quantifies the consistency between the outputs on natural and perturbed images, is set to 6. This parameter, denoted as λ_1 in equation 3.3. Furthermore, the weights λ_2 and λ_3 for additional KL divergence terms in the loss function are both set at 0.5

The step size for TRADES, which controls the size of perturbation added during optimization during each iteration, is set at 0.0078. During training the perturbation steps are fixed at 10 and for testing the steps are 20, determining the total number of iterations and the overall perturbations during optimization, and the epsilon value is set to 0.031, serving as a threshold to restrict the perturbations added to the input data. It's crucial to remember that TRADES adds perturbations to the images using the same iterative process with the hyper-parameters mentioned above.

For the experiments detailed below, during the training phase, adversarial examples were generated using KL-Divergence as the criterion. However, during testing, both KL divergence and cross entropy criteria were evaluated. The robust accuracy values for these criteria can be seen in the respective tables. For the graphs illustrating clean and robust accuracies with respect to number of epochs, we compute robust accuracies using KL divergence loss to generate adversarial examples.

4.1.4 Computational Setup and Implementation Details

All training procedures were conducted using two NVIDIA GeForce GTX 1080 Ti GPUs. The use of data parallelism was crucial in managing the computational demands and speeding up the training process. This setup not only allowed for efficient handling of large volumes of data but also ensured that the model training was rapid.

4.2 Performance of the Teacher Model (WideResnet 28-5)

The performance of the Teacher model on the CIFAR-10 dataset for different numbers of labeled samples (nl) is summarized in the table below:

CIFAR-10	Supervised(%)	FixMatch(%)
4000	81.82	95.87
500	73.56	91.20

Table 4.1: Accuracy of WideResnet 28-5 under different training conditions.

4.3 Experiment with Supervised Learning on CIFAR-10 Training Data with and without TRADES Adversarial Training

In this experiment, we employed a 13-layer CNN to train on the CIFAR-10 dataset. Our objective was to evaluate the effectiveness of adversarial training in improving the model's robust accuracy. The training was conducted both with and without adversarial methods, under the same hyperparameter settings described in the hyper-parameter section. Notably, during the adversarial training, we utilized 10 perturbation steps, whereas, in the testing phase, we increased this number to 20. The training was conducted over 100 epochs. As described in the methodology section, the loss function during adversarial training is given by Equation 3.2:

$$L(\theta) = L_{\text{nat}} + 6 \cdot L_{\text{rob}} + L_{\text{adv}} \tag{4.1}$$

Here, $\lambda = 6$ represents the weighting factor for the robust loss component.

4.3.1 Observations

A notable increase in robust accuracy highlights the effectiveness of adversarial training. Specifically:

- The robust accuracy increased significantly from 0% to 46.3% when adversarial training was introduced, demonstrating its importance in enhancing the model's ability to handle adversarial inputs.
- The decrease in clean accuracy from 92.46% to 76.84% suggests a trade-off, where increasing robustness to adversarial attacks slightly compromises performance on clean data.

Labeled Data	**Clean Accuracy (%)**	**Robust Accuracy (%)** *(criteria as CrossEntropy)*
50k clean data	92.46	0
50k with adversarial training	76.84	46.3

Table 4.2: Results of Supervised Learning with 50k train data with and without adversarial training on a 13-layer CNN network

4.4 Supervised Training of the Student Model with Labels from Teacher Model for Unlabeled Images

Initially, during the warm-up phase, the student model was trained on 500 labeled images in a supervised manner for 50 epochs. These 500 labeled images are the same images on which the teacher model was trained using the FixMatch semi-supervised learning algorithm. The loss function used in this warm-up phase is the same as described in Equation 3.2:

$$L(\theta) = L_{\text{nat}} + 6 \cdot L_{\text{rob}} + L_{\text{adv}} \tag{4.2}$$

After the warm-up phase training, the trained model checkpoint is saved and used in the main training phase. The dataset for the main training phase comprises both the initial 500 labeled images and an additional 49,500 unlabeled images. For the unlabeled images, the hard labels from the teacher model predictions were used as true labels for training the student model. The loss function used in this phase is the same as the one used in the warm-up phase. We further expanded the experiment to include a larger dataset of 4000 labeled images.

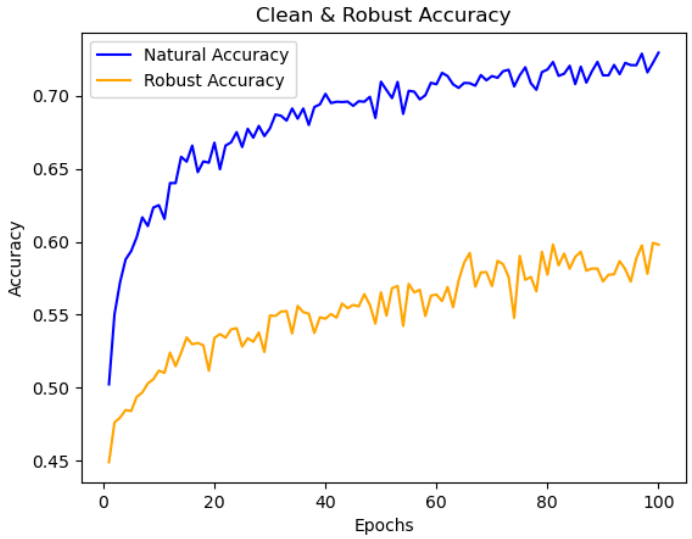

Figure 4.1: Graph for Clean and Robust accuracy for 500 labeled data

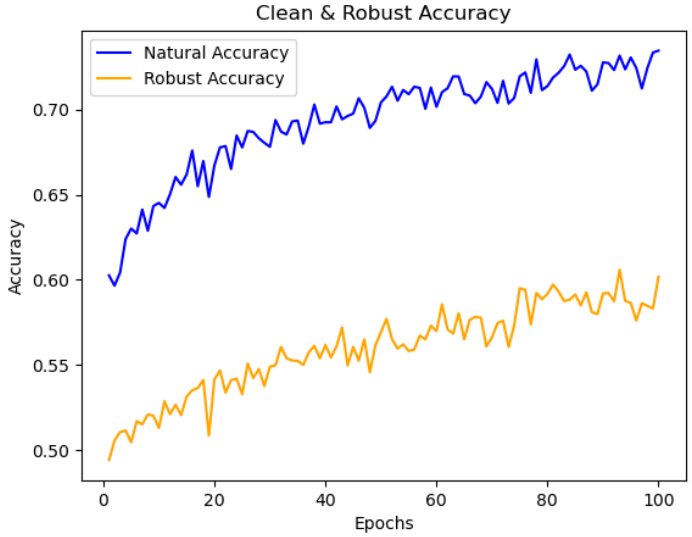

Figure 4.2: Graph for Clean and Robust accuracy for 4000 labeled data

Labeled Data	Clean Accuracy (%)	Robust Accuracy (%)	
		KL Divergence	Cross Entropy
500 labels	72.94	59.80	40.79
4000 labels	73.48	60.18	41.62

Table 4.3: Comparison of student model performance with different labeled data.

4.5 Refining Adversarial Training with Consistency Regularization

In this experiment, the warm-up phase training remains the same as described above. During the main phase training we enhance our training by incorporating the same loss function described in Equation 3.2 with a matching criteria. This criterion is meant to ensure consistency between the predictions of clean images and their adversarial counterparts. Within each batch of images, training is confined only to those images where the predictions for the clean and their perturbed versions align. Additionally, like in the above section, the labels for the unlabeled images are the hard labels from the teacher model trained on the FixMatch algorithm. This matching criterion, as described in section 3.1.1, promotes iterative learning with consistency in predictions.

$$L(\theta) = L_{\text{nat}} + 6 \cdot L_{\text{rob}} + L_{\text{adv}} \tag{4.3}$$

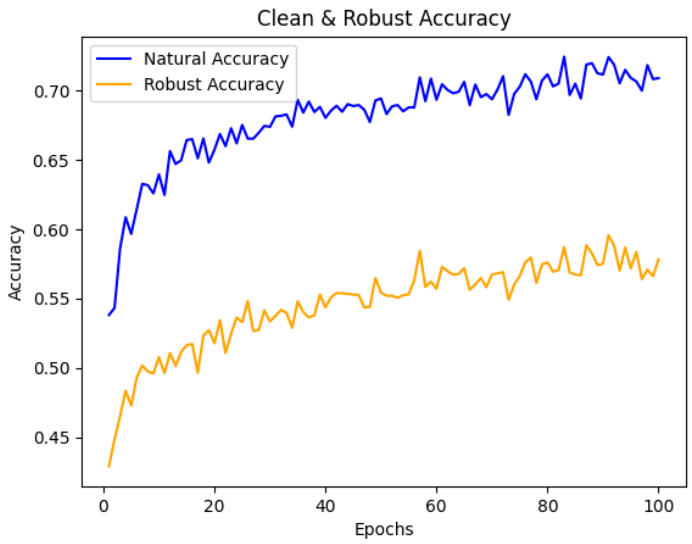

Figure 4.3: Graph of Clean and Robust Accuracy with Consistency Regularization for 500 Labeled Data

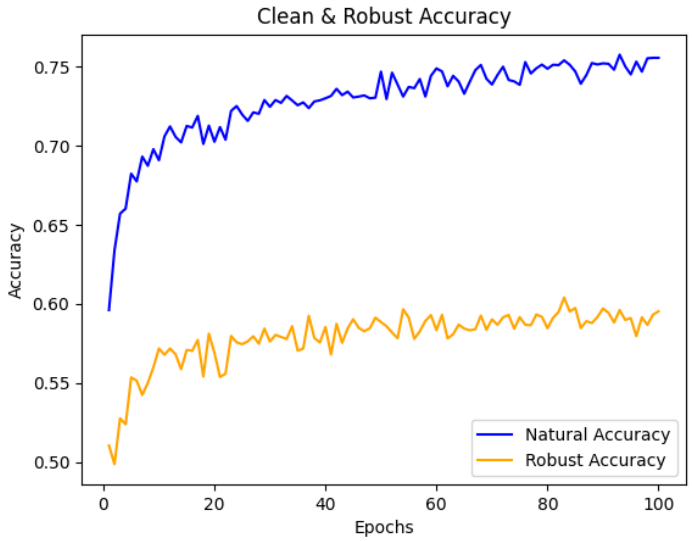

Figure 4.4: Graph of Clean and Robust Accuracy with Consistency Regularization for 4000 Labeled Data

Labeled Data	Clean Accuracy (%)	Robust Accuracy (%)	
		KL Divergence	Cross Entropy
500 labels	70.90	57.86	41.52
4000 labels	75.57	59.53	41.91

Table 4.4: Clean and Robust Accuracy Comparison for Different Sizes of Labeled Data with Consistency Regularization.

4.5.1 Summary

Based on the values from the tables 4.4 there isn't any significant increase in the accuracy values with matching criteria. But, from the graphs, we can infer that the matching criterion enhances model stability by reducing fluctuations in accuracy to some extent.

4.6 Integrating KL Divergence Loss Between Teacher Model's Soft Labels and Unmatched Indices of Clean Predictions

In the previous experiment, we did not utilize images with unmatched indices in the training process during an epoch. In this experiment, we have incorporated an additional term, $L_{\text{KL-un}}$, into our loss function. This new component is the KL divergence between the model's predictions for unmatched natural/clean images and the soft labels

provided by the teacher model. The warm-up training remains the same as in previous experiments, using the same loss function described in Equation 3.2. In the main phase training, we have integrated the matching criteria described in the previous experiment along with an additional loss term. The loss function is now defined as:

$$L(\theta) = L_{\text{nat}} + 6 \cdot L_{\text{rob}} + L_{\text{adv}} + 0.5 \cdot L_{\text{KL-un}} \tag{4.4}$$

This modification aims to enhance model training by effectively utilizing the information from unmatched index images for an epoch, potentially improving the model's generalization capabilities.

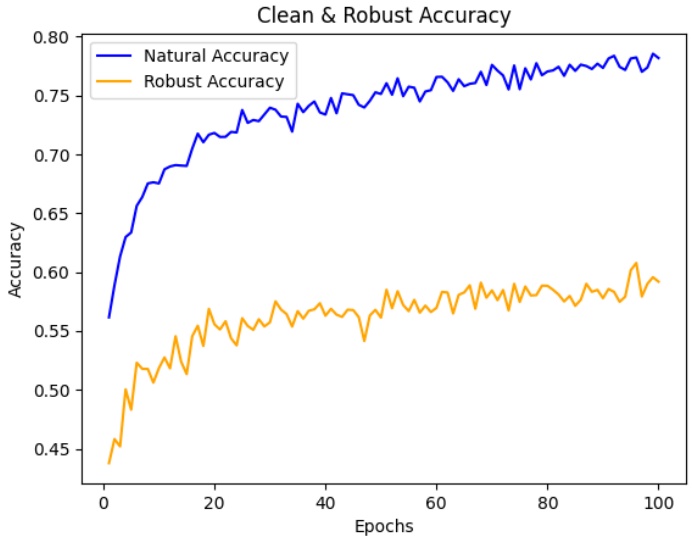

Figure 4.5: Graph of Clean and Robust Accuracy with KL Divergence Integration on clean predictions for 500 Labeled Data

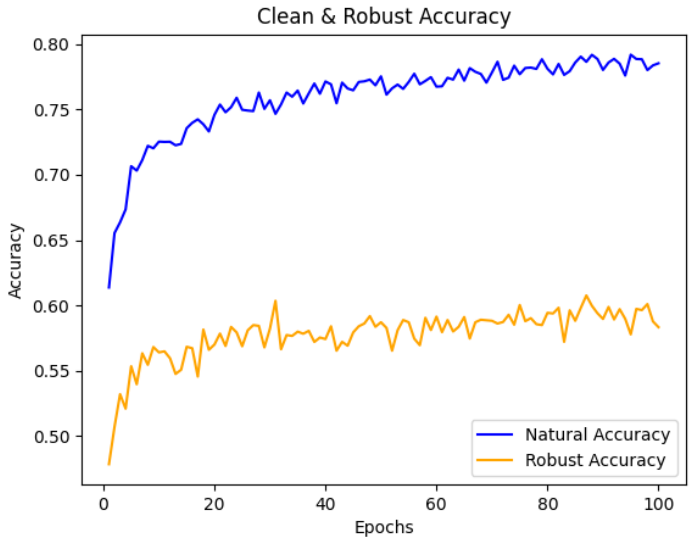

Figure 4.6: Graph of Clean and Robust Accuracy with KL Divergence Integration on clean predictions for 4000 Labeled Data

Labeled Data	**Clean Accuracy (%)**	**Robust Accuracy (%)**	
		KL Divergence	**Cross Entropy**
500 labels	78.18	59.19	41.76
4000 labels	78.54	58.33	40.86

Table 4.5: Clean and Robust Accuracy Metrics for Different Sizes of Labeled Data with KL Divergence of unmatched clean samples predictions

4.6.1 Observation

The graphs clearly demonstrate that the inclusion of the KL divergence term enhances the stability of the model's natural or clean accuracy across epochs. While there is no significant change in the robust accuracy values for both 500 and 4000 labeled data sets, there is a noticeable increase in the clean accuracy for both labeled data sizes. This confirms that the inclusion of the KL divergence term on clean predictions has significantly contributed to the enhancement of clean accuracy.

4.7 Integrating KL Divergence Loss Between Teacher Model's Soft Labels and Unmatched Indices of the Adversarial and Clean Predictions

The previous experiments— we have seen good results in stabilizing the training process by integrating a KL divergence term, $L_{\text{KL-un}}$, that aligns the model's clean predictions with the soft labels provided by the teacher model. This integration has effectively enhanced the model's performance on clean images. Building on this success, we have further expanded our approach by introducing another KL divergence term, $L_{\text{KL-adv}}$. This new addition specifically targets the alignment between the probability distributions of adversarial predictions and the teacher's soft labels. It utilizes unmatched adversarial examples that did not initially align with their clean counterparts or the teacher's predictions.

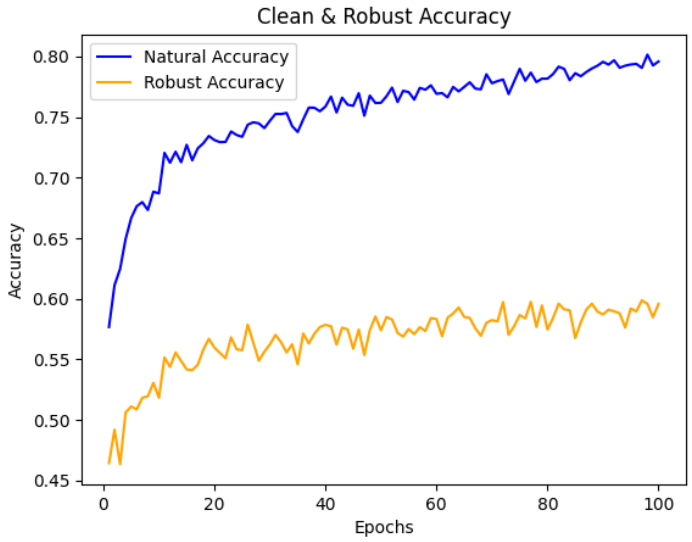

Figure 4.7: Graph of Clean and Robust Accuracy with KL Divergence Integration on adversarial predictions for 500 Labeled Data

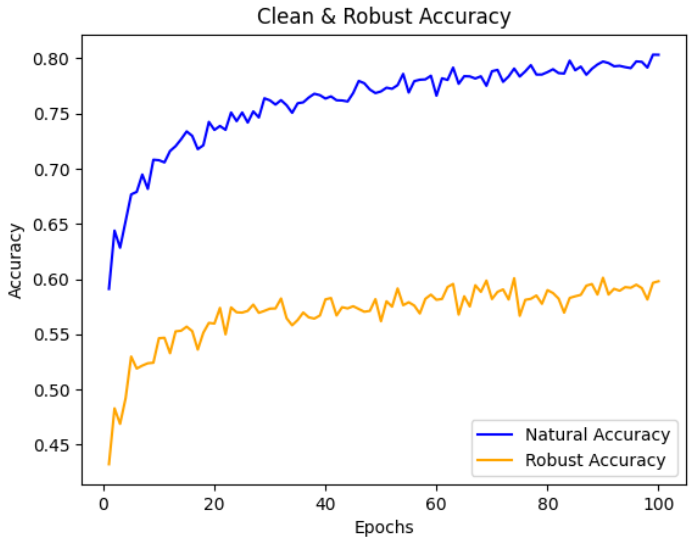

Figure 4.8: Graph of Clean and Robust Accuracy with KL Divergence Integration on adversarial predictions for 1000 Labeled Data

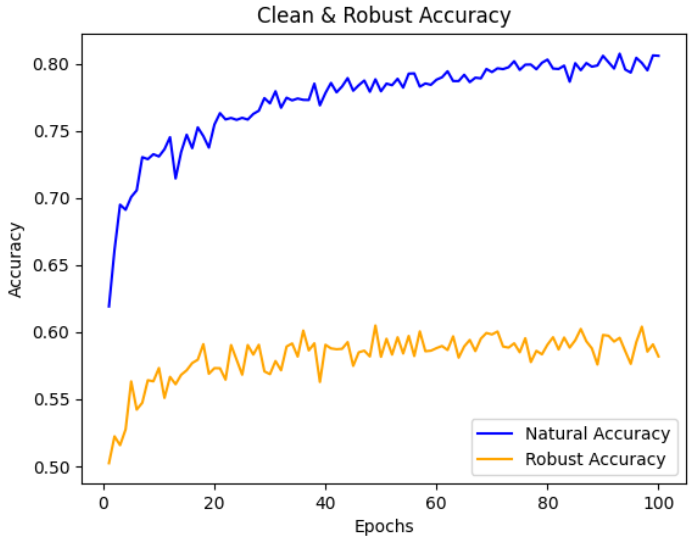

Figure 4.9: Graph of Clean and Robust Accuracy with KL Divergence Integration on adversarial predictions for 4000 Labeled Data

Labeled Data	Clean Accuracy (%)	Robust Accuracy (%)	
		KL Divergence	Cross Entropy
500 labels	79.59	59.58	40.20
1000 labels	80.33	59.81	41.22
4000 labels	80.62	59.09	41.37

Table 4.6: Clean and Robust Accuracy Metrics for Different Sizes of Labeled Data with KL Divergence of unmatched adversarial samples predictions

4.7.1 Observation

The graphs indicate that incorporating the $L_{\text{KL-adv}}$ term has led to a more stable performance trajectory, especially in terms of clean accuracy. Previous experiments demonstrated that the inclusion of a KL divergence term on clean predictions resulted in consistent accuracy values, regardless of the size of the labeled dataset. To further verify this consistency, we introduced an additional dataset size of 1000 labeled images. The results, as shown in the table, confirm that the accuracy values remain remarkably similar across different dataset sizes, reinforcing the effectiveness of the KL divergence term in stabilizing model performance.

4.8 Additional Matching Criterion in TRADES Loss Function

This experiment extends our exploration with an additional matching criterion that aligns not only with the predictions between clean and adversarial examples but also ensures consistency with the predictions provided by the teacher model. This dual consistency check—between model outputs and teacher predictions—aims to leverage the reliable pseudo-labels generated by the teacher.

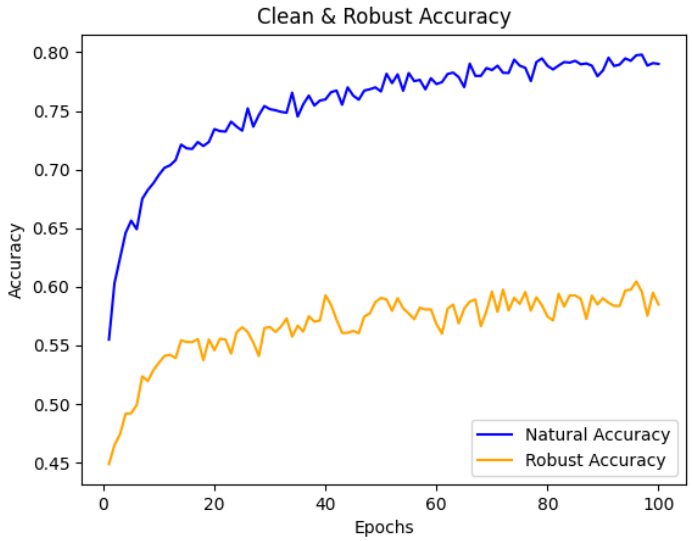

Figure 4.10: Graph of Clean and Robust Accuracy for 500 labeled data with consistency regularization and teacher model predictions as matching criteria

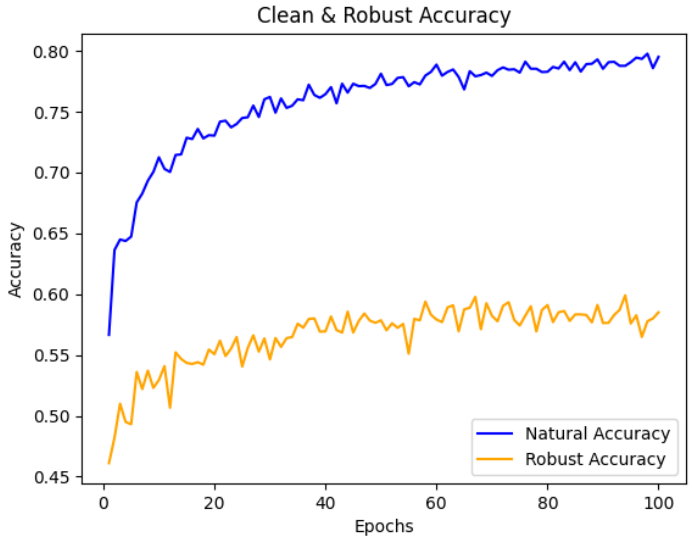

Figure 4.11: Graph of Clean and Robust Accuracy for 1000 labeled data with consistency regularization and teacher model predictions as matching criteria

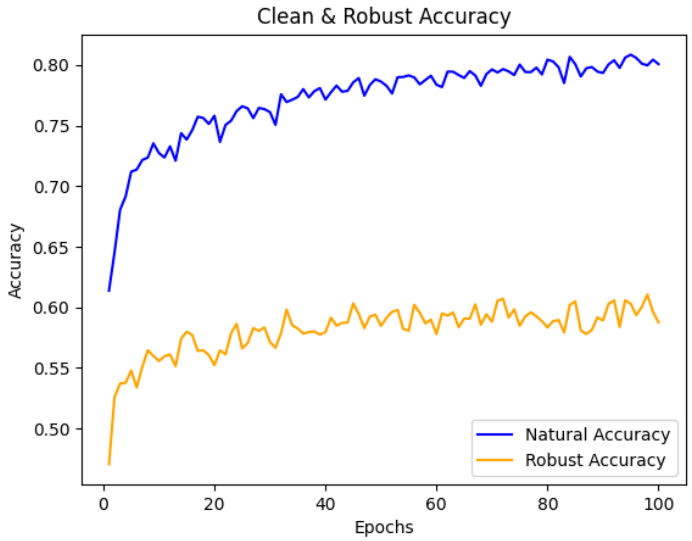

Figure 4.12: Graph of Clean and Robust Accuracy for 4000 labeled data with consistency regularization and teacher model predictions as matching criteria

Labeled Data	**Clean Accuracy (%)**	**Robust Accuracy (%)**	
		KL Divergence	**Cross Entropy**
500 labels	79.01	58.49	40.28
1000 labels	79.52	58.51	40.46
4000 labels	80.08	58.77	40.82

Table 4.7: Clean and Robust Accuracy Metrics for Different Sizes of Labeled Data with teacher model labels matching criteria

4.8.1 Observation

From the results, it is apparent that there is a slight decrease in both clean and robust accuracy values when compared to the experiment without this additional matching criterion. Although the drop is not substantial, it is clear that the introduction of this dual consistency check—aligning model outputs with teacher predictions—did not lead to improvements in clean and robust accuracies as anticipated. This could indicate that the added complexity of the matching criterion may have affected the selection of training samples in each epoch during training, potentially limiting the diversity of examples the model learns from and thus impacting its overall performance.

Chapter 5

Conclusion

This study has successfully demonstrated the potential of enhancing adversarial robustness in semi-supervised learning environments, particularly under the constraints of limited labeled data. By employing a teacher-student model framework, we were able to effectively utilize pseudo-labels derived from a teacher model trained via the FixMatch algorithm. Significant advancements were made by incorporating multiple loss terms, which strategically improved the balance between robustness and accuracy on clean data with limited labels.

The findings revealed notable increases in both clean and robust accuracies when incorporating multiple loss terms to enhance the performance. This enhancement in adversarial training, coupled with semi-supervised learning principles, proved crucial for boosting the model's resistance to adversarial attacks.

Looking ahead, future research should focus on applying these methods to larger datasets and varying neural network architectures to both corroborate and expand upon these findings. Additionally, refining the loss functions could offer more precise ways to manage the trade-offs between accuracy and robustness. This would be crucial for enhancing model stability, especially given the noticeable fluctuations in accuracy observed in the current study. Such improvements could lead to more reliable applications in security-sensitive domains.

Bibliography

[1] Kihyuk Sohn, David Berthelot, Chun-Liang Li, Zizhao Zhang, Nicholas Carlini, Ekin D. Cubuk, Alex Kurakin, Han Zhang, and Colin Raffel. Fixmatch: Simplifying semi-supervised learning with consistency and confidence. *CoRR*, abs/2001.07685, 2020. URL https://arxiv.org/abs/2001.07685.

[2] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[3] Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric Xing, Laurent El Ghaoui, and Michael Jordan. Theoretically principled trade-off between robustness and accuracy. In *International conference on machine learning*, pages 7472–7482. PMLR, 2019.

[4] Lu Liu and Robby T Tan. Certainty driven consistency loss on multi-teacher networks for semi-supervised learning. *Pattern Recognition*, 120:108140, 2021.

[5] Mehdi Sajjadi, Mehran Javanmardi, and Tolga Tasdizen. Regularization with stochastic transformations and perturbations for deep semi-supervised learning. *Advances in neural information processing systems*, 29, 2016.

[6] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

[7] Dong-Hyun Lee et al. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on challenges in representation learning, ICML*, volume 3, page 896. Atlanta, 2013.

[8] Ahmet Iscen, Giorgos Tolias, Yannis Avrithis, and Ondrej Chum. Label propagation for deep semi-supervised learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5070–5079, 2019.

[9] Philip Bachman, Ouais Alsharif, and Doina Precup. Learning with pseudo-ensembles, 2014. URL https://arxiv.org/abs/1412.4864.

[10] Eric Arazo, Diego Ortego, Paul Albert, Noel E O'Connor, and Kevin McGuinness. Pseudo-labeling and confirmation bias in deep semi-supervised learning. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2020.

[11] Zeping Min and Cheng Tai. Why pseudo label based algorithm is effective?–from the perspective of pseudo labeled data. *arXiv preprint arXiv:2211.10039*, 2022.

[12] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[13] Tao Bai, Jinqi Luo, Jun Zhao, Bihan Wen, and Qian Wang. Recent advances in adversarial training for adversarial robustness. *arXiv preprint arXiv:2102.01356*, 2021.

[14] Zico Kolter and Aleksander Madry. Adversarial robustness: Theory and practice. *Tutorial at NeurIPS*, page 3, 2018.

[15] Avital Oliver, Augustus Odena, Colin A Raffel, Ekin Dogus Cubuk, and Ian Goodfellow. Realistic evaluation of deep semi-supervised learning algorithms. *Advances in neural information processing systems*, 31, 2018.

[16] Ben Athiwaratkun, Marc Finzi, Pavel Izmailov, and Andrew Gordon Wilson. There are many consistent explanations of unlabeled data: Why you should average. *arXiv preprint arXiv:1806.05594*, 2018.

[17] Tim Salimans and Durk P Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. *Advances in neural information processing systems*, 29, 2016.

[18] Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry. How does batch normalization help optimization? *Advances in neural information processing systems*, 31, 2018.

[19] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.