Journal of Machine Learning Research 26 (2025) 1-48 Submitted 12/22; Revised 11/24; Published 2/25

# Regularizing Hard Examples Improves Adversarial Robustness

**Hyungyu Lee** RUCY74@SNU.AC.KR
*Electrical and Computer Engineering*
*Interdisciplinary Program in Artificial Intelligence*
*Seoul National University, Seoul 08826, Republic of Korea*
**Saehyung Lee** HALO8218@SNU.AC.KR
*Electrical and Computer Engineering*
*Interdisciplinary Program in Artificial Intelligence*
*Seoul National University, Seoul 08826, Republic of Korea*
**Ho Bae*** HOBAE@EWHA.AC.KR
*Department of Cyber Security*
*Ewha Womans University, Seoul 03760, Republic of Korea*
**Sungroh Yoon*** SRYOON@SNU.AC.KR
*Electrical and Computer Engineering*
*Interdisciplinary Program in Artificial Intelligence*
*AIIS, ASRI, INMC, and ISRC, Seoul National University*
*Seoul National University, Seoul 08826, Republic of Korea*

**Editor:** Quanquan Gu

## Abstract

Recent studies have validated that pruning hard-to-learn examples from training improves the generalization performance of neural networks (NNs). In this study, we investigate this intriguing phenomenon—the negative effect of hard examples on generalization—in adversarial training. Particularly, we theoretically demonstrate that the increase in the difficulty of hard examples in adversarial training is significantly greater than the increase in the difficulty of easy examples. Furthermore, we verify that hard examples are only fitted through memorization of the label in adversarial training. We conduct both theoretical and empirical analyses of this memorization phenomenon, showing that pruning hard examples in adversarial training can enhance the model's robustness. However, the challenge remains in finding the optimal threshold for removing hard examples that degrade robustness performance. Based upon these observations, we propose a new approach, difficulty proportional label smoothing (DPLS), to adaptively mitigate the negative effect of hard examples, thereby improving the adversarial robustness of NNs. Notably, our experimental result indicates that our method can successfully leverage hard examples while circumventing the negative effect.

**Keywords:** adversarial robustness, adversarial examples, hard examples, memorization, robust overfitting

∗ Corresponding authors

## 1. Introduction

The structural regularities of classification datasets have been investigated in several studies (Jiang et al., 2021; Paul et al., 2021; Wu et al., 2018), and measures have been proposed for identifying the regularities of training samples through training statistics. For example, Jiang et al. (2021) discovered that the learning speed of training examples is strongly correlated with the structural regularities, i.e., regular examples are learned quickly, whereas irregular examples are learned slowly. Paul et al. (2021) measured the difficulty of a training example by using the loss gradient norm. Based on the proposed measures, the aforementioned studies clarified the relationship between example difficulty and generalization in the context of deep neural networks (DNNs). In particular, a compelling finding among their observations motivates our study: *memorizing hard-to-learn/irregular examples can deteriorate the generalization ability of DNNs.* In other words, hard-to-learn examples are unrepresentative outliers of a class or data with corrupted labels, and thus, these examples can result in the degradation of generalization. In this study, we explore this intriguing phenomenon—the negative effect of hard examples on generalization—in adversarial settings.

Adversarial training differs from standard training: it uses adversarial examples as the training data. Notably, adversarial training requires more data than standard training (Schmidt et al., 2018). Additionally, the improvement in generalization rendered by using more data is observably greater in an adversarial setting than in a standard setting (Lee et al., 2021; Gowal et al., 2021b). Therefore, recent studies on adversarial training have focused on approaches for using more data effectively to bridge the gap between train and test inferences of DNNs (Carmon et al., 2019; Gowal et al., 2021b). Nevertheless, in this study, we demonstrate that the improvement in generalization performance could be achieved by regularizing the use of data. Specifically, we demonstrate that the improvement achieved by regularizing the *training of hard examples* is greater in adversarial training.

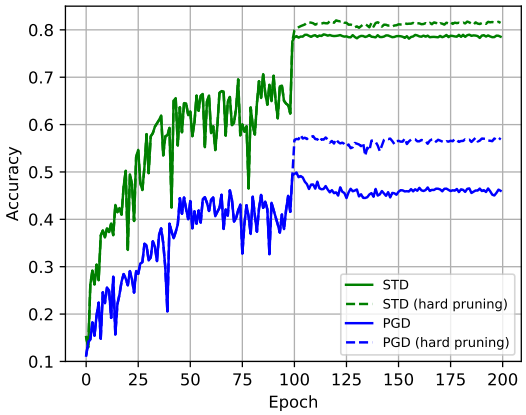



Figure 1: The test accuracy (for clean examples) of the standard (STD) models and the test accuracy (for adversarial examples) of the adversarial (PGD (Madry et al., 2017)) models trained on the subsets of CIFAR-10. On pruning hard examples from training, test accuracy increases both in the STD and PGD models, but the increase in the PGD model is greater than that in the STD model.

Fig. 1 illustrates the difference between the effect of pruning hard examples in standard training (denoted as STD) and adversarial training by using projected gradient descent (Madry et al., 2017) (denoted as PGD). After training models on subsets of CIFAR-10 (Krizhevsky et al., 2009) for 100 epochs, we either pruned hard examples from the training dataset or continued by training on the entire training data.* We evaluated the test accuracy for clean

*Example difficulty is measured by accumulating 0-1 loss along the training trajectory. For further details, refer to Appendix B.2.

examples in the STD models and the test accuracy for adversarial examples in the PGD models. Observably, pruning hard examples during training can improve the generalization performance, which is consistent with the previous study's result that training of hard examples can deteriorate generalization. However, the result indicates that the performance increase is greater in adversarial training. It can be inferred that the negative effect of hard examples on training is intensified by adversarial perturbations, and thus, the improvement by pruning hard examples is more substantial in adversarial training.

In this study, we verify that the cause of this result is the negative effect of hard examples in adversarial training. We theoretically prove that the difficulty of each example increases in an adversarial setting and that the increase is more significant for hard examples than for easy examples. In this context, we demonstrate that hard examples are fitted only through memorization in adversarial training and that pruning them from the training process improves robustness. Based on this, we conduct a theoretical analysis of the memorization phenomenon of hard examples, proving that including hard examples in adversarial training reduces robustness performance. To demonstrate this empirically, we perform analytical experiments on the assumptions and outcomes of the theoretical analysis, showing that the theoretical analysis aligns with actual results. Consequently, we show that hard examples negatively impact the model's robustness. However, the challenge remains that finding the optimal difficulty threshold for removing hard examples that deteriorate robustness is an intractable proposition.

To address this problem, we propose a regularization method, difficulty proportional label smoothing (DPLS), to mitigate the negative effect of hard examples. Label smoothing (LS) (Szegedy et al., 2016) regularizes the prediction of each example to prevent the memorization of the one-hot label. In this perspective, DPLS extends this LS approach to adopt a smoothing factor that is proportional to the difficulty of the example for regularizing the training of each example. Using the results of experiments on a variety of datasets and algorithms, we assess our proposed methodology and find that DPLS could successfully leverage hard examples while avoiding the negative effect.

In Section 2, we introduce the background and previous studies related to the topic addressed in this paper. In Section 3, we conduct a theoretical and empirical analysis of learning hard examples in adversarial training. In Section 4, we analyze approaches for effectively learning hard examples in adversarial training and propose a methodology. In Section 5, we demonstrate the effectiveness of the proposed methodology through experiments on various algorithms and datasets. Finally, in Section 6, we present the conclusions of the paper.

## 2. Related work

**Adversarial robustness** Adversarial training (Madry et al., 2017) employs adversarial examples as training data to improve robustness. For a dataset $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$, where $\boldsymbol{x}_i \in \mathbb{R}^d$ represents a clean data, and $y_i \in \{1, ..., C\}$ denotes its corresponding label of $C$

classes. Adversarial training can be formulated in an empirical risk minimization form featuring the following optimization:

$$\min_{\theta} \sum_{i=1}^{n} \max_{\delta \in S} \mathcal{L}(f(\boldsymbol{x_i} + \boldsymbol{\delta}), y_i; \theta). \tag{1}$$

Here, $\theta$ denotes the parameter of a model $f$, $\mathcal{L}$ represents the loss function, and $S = \{\boldsymbol{\delta} \in \mathbb{R}^d : ||\boldsymbol{\delta}|| \leq \epsilon\}$ where $\epsilon$ is an adversarial budget, which depicts an upper bound of adversarial perturbations. Another form of adversarial training is TRADES (Zhang et al., 2019), which controls the trade-off between robustness and standard accuracy by dividing training loss into the loss of clean examples and regularization for robustness. For adversarial attacks, fast gradient sign method (FGSM) is a general method for generating adversarial examples (Goodfellow et al., 2014). PGD is a multi-step version of FGSM (Madry et al., 2017). Adaptive auto attack ($A^3$) is an efficient and reliable attack which utilizes adaptive random starts and statistics-based discarding strategy (Liu et al., 2022b).

**Robust and non-robust feature** In Tsipras et al. (2018), certain data features were defined as moderately or slightly correlated with the data labels. These features are useful for improving performance when training a standard classifier, but they have the characteristic of reducing the classifier's performance because they can be manipulated by adversarial attacks. Such features are referred to as 'non-robust features.' In contrast, features that are not manipulated by adversarial attacks and maintain their correlation with the data labels are referred to as 'robust features.' Ilyas et al. (2019) formulated a mathematical definition of robust and non-robust features based on their correlation with the labels. Specifically, for a data pair $(x, y)$ from a dataset $\mathcal{D}$, a feature $g$ is defined as a $\rho$-useful feature if it satisfies $\mathbb{E}[y \cdot g(x)] \geq \rho$, where $\rho > 0$. Here, for an adversarial perturbation $\delta$, a useful feature $g$ is defined as a robust feature if it satisfies $\mathbb{E}\left[\inf_{\delta} y \cdot g(x + \delta)\right] \geq \rho$, and as a non-robust feature if it does not satisfy this condition. We conduct a theoretical analysis utilizing these concepts of robust and non-robust features as defined in previous studies.

**Overfitting and memorization in adversarial training** Wong et al. (2020) conducted analysis on overfitting in adversarial training, termed robust overfitting. Chen et al. (2020) utilized a smoothing method that combines knowledge distillation (Hinton et al., 2015) and stochastic weight average (Izmailov et al., 2018). Dong et al. (2022) also used a knowledge distillation method to mitigate overfitting caused by distribution shift and label noise. Yu et al. (2022) proposed a method to prevent overfitting of small-loss data combined with adversarial weight perturbation (Wu et al., 2020). In Huang et al. (2020); Liu et al. (2021); Dong et al. (2021), they used the exponential moving average of predictions as a label for training to prevent robust ovefitting. In particular, Dong et al. (2021) demonstrated that DNNs are sufficient to memorize training adversarial examples with random labels; reportedly the causes of robust overfitting may be the memorization of one-hot labels.

**Measurement on difficulty of examples** Paul et al. (2021) demonstrated that the norm of the error vector (EL2N score) can be used to identify important and difficult training data. They discovered that examples with high score tend to be hard to learn but important for improving the performance. However, pruning certain examples with the highest score can improve generalization performance, indicating that the highest score examples tend

to be unrepresentative. Jiang et al. (2021) proposed consistency score (C-score), which is the expected accuracy of each instance for the models trained with various data subsets excluding the given instance. While these scores can be utilized in several applications (Lee et al., 2022), we use these scores as the example difficulty in our study.

## 3. Analysis

In this section, we conduct a theoretical and empirical analysis of the effect of hard examples on training. Through this analysis, we reveal that in adversarial training, learning from hard examples primarily occurs through memorization. Next, we perform a theoretical analysis of the memorization of hard examples that occurs in adversarial training. Finally, we demonstrate the theoretical analysis through various empirical verifications of the memorization of hard examples.

### 3.1 Analysis on the effect of hard examples

We first demonstrate through a theoretical analysis of hard examples that training hard examples becomes more difficult as the adversarial budget in adversarial training increases, and we illustrate this theoretical analysis with experimental evidence. Next, we compare the learning patterns of hard examples in standard and adversarial training, confirming that in adversarial training, the learning of hard examples primarily occurs through memorization.

#### 3.1.1 Theoretical analysis of the effect of hard examples

We theoretically analyze the effect of hard examples in both standard and adversarial training. A simple model is designed to demonstrate the manner in which adversarial perturbations affect hard examples in adversarial training. We aim to design examples featuring different difficulties in learning. Specifically, we define a binary classification model $f(\boldsymbol{x}) = \sigma(\boldsymbol{w}^\top \boldsymbol{x} + b)$, which is a mapping function $f : \mathcal{X} \to \mathcal{Y}$ from input space $\mathcal{X}$ to output space $\mathcal{Y}$, where $(\boldsymbol{x}, y) \in \mathbb{R}^d \times \{\pm 1\}$ and $\sigma(\cdot)$ represents a sigmoid function. To ensure simplicity, we assume that $\|\boldsymbol{w}\| = 1$. Thereafter, we set the loss function as cross-entropy loss and define easy and hard examples by controlling the distance between each example and the decision boundary. The hard example is set as $(\boldsymbol{x}_1, y_1 = +1)$ where $\left\|\boldsymbol{w}^\top \boldsymbol{x}_1 + b\right\| = d_1$, and the easy example is set as $(\boldsymbol{x}_2, y_2 = +1)$ where $\left\|\boldsymbol{w}^\top \boldsymbol{x}_2 + b\right\| = d_2$ and $d_1 < d_2$. The differences between the gradient norm of the examples in standard and adversarial settings are as follows:

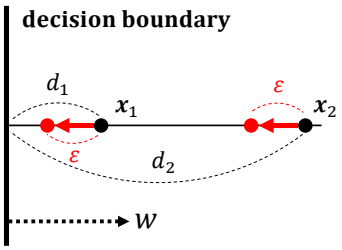



Figure 2: The schematic illustration of easy ($x_2$) and hard examples ($x_1$).

**Definition 1 (Gradient difference)** *The differences between the loss gradient norm on* $\boldsymbol{x}_1$ *and* $\boldsymbol{x}_2$ *in standard and adversarial training are defined as follows:*

$$\mathcal{D}_{\text{std}}(\boldsymbol{x}_1, \boldsymbol{x}_2) = \left\|(1 - \sigma(\boldsymbol{w}^\top \boldsymbol{x}_1 + b))\boldsymbol{x}_1\right\| - \left\|(1 - \sigma(\boldsymbol{w}^\top \boldsymbol{x}_2 + b))\boldsymbol{x}_2\right\|$$
$$\mathcal{D}_{\text{adv}}(\boldsymbol{x}_1, \boldsymbol{x}_2, \epsilon) = \left\|(1 - \sigma(\boldsymbol{w}^\top (\boldsymbol{x}_1 + \boldsymbol{\delta}) + b))(\boldsymbol{x}_1 + \boldsymbol{\delta})\right\| - \left\|(1 - \sigma(\boldsymbol{w}^\top (\boldsymbol{x}_2 + \boldsymbol{\delta}) + b))(\boldsymbol{x}_2 + \boldsymbol{\delta})\right\|. \tag{2}$$

Here, $\boldsymbol{\delta}$ represents an adversarial perturbation, where $\left\|\boldsymbol{w}^\top \boldsymbol{\delta}\right\| = \epsilon$. $\mathcal{D}_{\text{std}}$ denotes the difference between the loss gradient norm on inputs in standard training and $\mathcal{D}_{\text{adv}}$ denotes that in adversarial training. The following theorem and corollary demonstrate that the difference between the loss gradient norm of hard and easy examples in adversarial training is more significant than that in standard training:

**Theorem 2** *Let the gradient norm of easy examples is smaller than the gradient norm of hard examples in both standard and adversarial training. Then, if* $\boldsymbol{w}^\top \boldsymbol{x}_1 \geq 0$ *and* $0 < \epsilon < \frac{(d_1+d_2)}{2}$, *it satisfies*

$$\mathcal{D}_{\text{std}}(\boldsymbol{x}_1, \boldsymbol{x}_2) < \mathcal{D}_{\text{adv}}(\boldsymbol{x}_1, \boldsymbol{x}_2, \epsilon). \tag{3}$$

**Corollary 3** *The increase in gradient difference by an adversarial attack is proportional to the difficulty of the example and the size of the upper bound of adversarial perturbations. Thus, if* $\left\|\boldsymbol{w}^\top \boldsymbol{x}_3\right\| = d_3$ *and* $d_2 < d_3$, *it satisfies*

$$\mathcal{D}_{\text{adv}}(\boldsymbol{x}_1, \boldsymbol{x}_2, \epsilon) < \mathcal{D}_{\text{adv}}(\boldsymbol{x}_1, \boldsymbol{x}_3, \epsilon), \tag{4}$$

*and if* $\epsilon_1 < \epsilon_2$, *it satisfies*

$$\mathcal{D}_{\text{adv}}(\boldsymbol{x}_1, \boldsymbol{x}_2, \epsilon_1) < \mathcal{D}_{\text{adv}}(\boldsymbol{x}_1, \boldsymbol{x}_2, \epsilon_2). \tag{5}$$

We have reported the proof in Appendix A.1. In the model whose training loss is sufficiently converged, the outputs of easy examples are saturated, where $\sigma(\boldsymbol{w}^\top \boldsymbol{x}_2) \approx 1$. Thus, the loss gradient norm of easy examples in standard training is assumed to be similar to that in adversarial training. The assumption is validated through empirical analysis. The change in the gradient norm of hard examples through the addition of adversarial perturbation is larger than that of easy examples. According to Paul et al. (2021), the gradient norm can be used as the difficulty (importance) score. In other words, the difficulty increment of hard examples is more significant than that of easy examples.

#### 3.1.2 Empirical analysis of the effect of hard examples

The preliminary analysis verifies that the increase in the difficulty of hard examples is greater than the increase in that of easy examples in adversarial training, and we demonstrated that hard examples exerts a negative effect on generalization (Fig. 1). We accordingly hypothesize that a major factor deteriorating generalization in adversarial training is the large increase in the difficulty of hard examples. We empirically verify this hypothesis.

We select hard examples according to the average loss of the models along the training trajectory in accordance with the strategy reported by Jiang et al. (2021). The loss can be of any kind, which is a measure of the error of each example. We employ 0-1 loss as the measurement of difficulty. For a clear representation, the model trained with standard training is denoted as STD, and the model trained via adversarial training as PGD. We represent the accuracy for clean examples as standard accuracy and the accuracy for adversarial examples as robust accuracy.

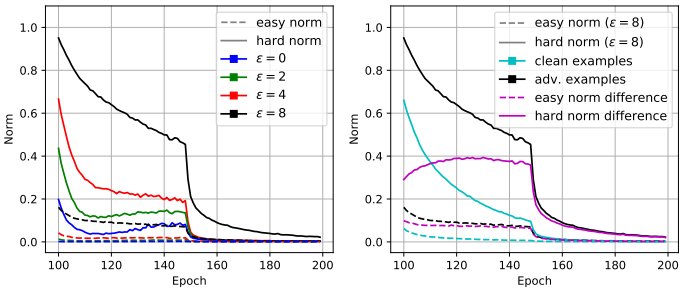


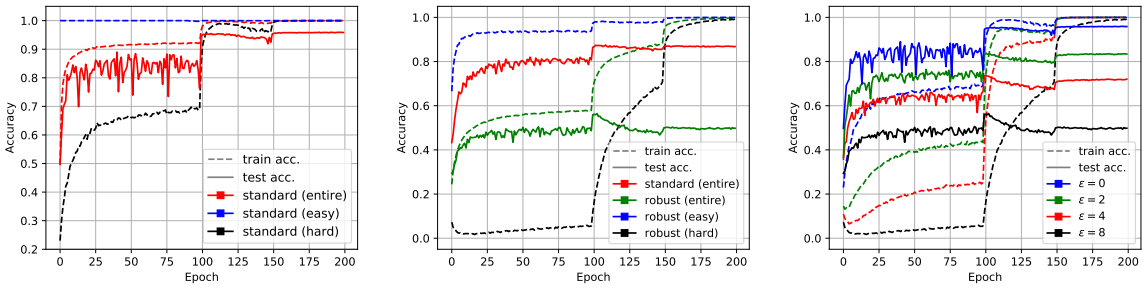

Figure 3: Average error vector norm (EL2N score (Paul et al., 2021)) of training examples.

**Comparing the difficulty of hard examples** We initially conduct an empirical study on the difficulty of hard examples by comparing the gradient norm in standard and adversarial training. Paul et al. (2021) reported that the average error vector norm (EL2N score) can approximate the expected training loss gradient norm of each example. Thus, we employ this to compare the gradient norm in both training. We select the top 10k hard and top 10k easy example subsets from the training dataset of CIFAR-10 and subsequently calculate the average error vector norm of examples in each subset during training, where the error vector norm of an input pair $(\boldsymbol{x}, y)$ is defined as $\mathbb{E}\|f(\boldsymbol{x}) - \boldsymbol{y}\|_2$ for the one-hot label vector $\boldsymbol{y}$. We train several models by varying the adversarial budget $\epsilon = 8$ (PGD), $4, 2$, and $0$ (STD), and the result is obtained after the learning rate decay epoch, which is the training point indicating that the model is sufficiently trained.

As illustrated in Fig. 3 (left), the average norm values of easy examples are small in all models, and the difference in the average norm of easy examples between models is also comparatively small. However, for the results of hard examples, the average norm of hard examples increases significantly as the adversarial budget $\epsilon$ increases. In the PGD model (right), greater difference is observed between clean and adversarial examples of hard examples than that of easy examples. This result indicates that the loss gradient norm of hard examples is rendered greater in adversarial training while that of easy examples does not, which demonstrates our theoretical analysis presented in Section 3.1.

**Comparing the training of hard examples** The difficulty of hard examples is significantly amplified in adversarial training. We therefore investigate whether hard examples are fitted normally in adversarial training. Figs. 4a and 4b show the test accuracy and training accuracy of the top 10k hard, top 10k easy, and entire training examples in the STD and PGD models. Observably, test accuracy decreases marginally following the learning rate decay (epoch 100) in the STD model. In contrast, the result of the PGD model indicates that when the training robust accuracy of hard examples starts to increase (at the decay epoch 100), the test robust accuracy starts to decrease rapidly. Moreover, the training accuracy of the easy examples remains unchanged after the decay, indicating that the decrease in the test performance is mostly attributed to the training of hard examples.

Fig. 4c shows the training accuracy of hard examples and the test accuracy of the models with varying adversarial budget $\epsilon$. The decrease in test accuracy and the increase in training accuracy of hard examples after the learning rate decay are observed to be severe as the $\epsilon$ increases. Therefore, given these characteristics in the learning patterns, we can regard this phenomenon as the memorization of hard examples that occurs in adversarial training, in contrast to standard training. Specifically, as the adversarial budget $\epsilon$ increases in adversarial training, it suggests that an increasing number of hard examples are being learned through memorization.




(a) (b) (c)

Figure 4: (a) and (b) show the training and test accuracy curves in the STD and PGD models, respectively. (c) shows the training accuracy of hard examples and test accuracy of the models with different adversarial budgets $\epsilon$.

**Pruning hard examples from training** We observed the phenomenon of hard examples being memorized after the learning rate decay, as shown in Fig. 4b. Next, we conducted experiments to verify whether the hard examples fitted through memorization actually contribute to the model's performance. To do this, as in Fig. 1, we trained the model after excluding the hard examples from the training dataset and then compared the performance. Specifically, we pruned the top 10k easy and top 10k hard examples from the training dataset of CIFAR-10, respectively.

Table 1: Performance of models with and without pruning 10k hard or easy examples.

| Model | Std. | Rob. |
|---|---|---|
| None | 87.19 | 51.80 |
| Easy pruning | 87.27 | 49.71 |
| Hard pruning | 84.70 | 53.36 |

Table 1 shows the standard and robust accuracies of the PGD (None) and PGD pruning models (Easy and Hard pruning). From the robust accuracy results, we can see that excluding easy examples from the training dataset results in the expected decrease in robust accuracy. In contrast, when hard examples are excluded, robust accuracy increases. This suggests that learning hard examples may reduce the model's robustness. However, when focusing on standard accuracy, we observe that excluding hard examples from the training process leads to a decrease in standard performance. Taking this into consideration, learning hard examples appears to increase standard accuracy but decrease robust accuracy. This characteristic resembles the robustness-accuracy trade-off observed by Tsipras et al. (2018). Therefore, it implies that the model may be learning non-robust features through the memorization of hard examples.

### 3.2 Theoretical analysis on the training of hard examples

In Section 3.1, we demonstrated that learning hard examples is challenging in adversarial training and observed that these hard examples are primarily learned through memorization. Subsequently, through hard example pruning experiments, we observed that the memorization of hard examples exhibits patterns similar to the learning of non-robust features. In this section, we conduct a theoretical analysis of this phenomenon in adversarial training. Specifically, we perform a theoretical analysis of the effects of learning through the memorization of hard examples in a binary classification task. We investigate the impact of

hard examples on the model's performance in adversarial training, and through this, we theoretically demonstrate that including hard examples in the training process leads to a decrease in robustness performance.

**Definition of data distributions in binary classification task** We first define the task for the theoretical analysis of learning through the memorization of hard examples. Following Tsipras et al. (2018) and Ilyas et al. (2019), we define a binary classification task consisting of the following input-output pair $(\boldsymbol{x}, y)$.

**Definition 4 (Normal example data distribution)**

$$y \overset{u.a.r.}{\sim} \{-1, +1\}, \qquad \boldsymbol{x} \in \mathbb{R}^{d+2} = \begin{cases} x_0 \sim \mathcal{N}(\eta y, 1), \\ x_{i \in \{1,\ldots,d\}} \overset{i.i.d.}{\sim} \mathcal{N}(\beta \eta y, 1), \\ x_{d+1} = 0, \end{cases} \tag{6}$$

where $\beta$ is a constant that satisfies $0 < \beta < 1$. Here, assuming $\beta\eta < \epsilon < \eta$, $x_0$ corresponds to a robust feature, while $x_{i \in \{1,\ldots,d\}}$ correspond to non-robust features. Additionally, we introduce a 'memorization feature' corresponding to $x_{d+1}$. The memorization feature represents a feature that exists only in specific data, specifically in hard examples, but does not exist in normal examples. If a model learns a memorization feature, it can be considered that the model has memorized it. We define that data sampled from the normal example data distribution does not exhibit memorization features. Therefore, we refer to examples without memorization features sampled from the distribution of Equation 6 as a 'normal example.' Here, the set of datasets that can be generated from this distribution is defined as $\mathcal{X}_{normal}$.

Next, we define the distribution of hard examples in the above binary classification task. In Section 3.1, we observed that hard examples are learned through memorization. Additionally, we observed that this memorization seems to involve the learning of non-robust features. Based on this, we make the following assumptions about hard examples.

**Assumption 1 (Properties of hard examples from the perspective of adversarial robustness)** *A hard example, difficult for the model to learn, consists only of non-robust and memorization features. The memorization feature functions as a robust feature during the training phase but acts as a non-robust feature for unseen data during the test phase.*

These assumptions describe hard examples that become more difficult to learn as the adversarial budget increases, aligning them with the properties of non-robust features. Their learning through memorization corresponds to *memorization features*, which act as robust features during training by avoiding zero assignment in adversarial training (Tsipras et al., 2018). However, during testing on unseen data, these features behave as non-robust features, increasing standard accuracy but decreasing robust accuracy. Based on these assumptions, we define the distribution of hard examples.

**Definition 5 (Hard example data distribution)**

$$y \overset{u.a.r.}{\sim} \{-1,+1\}, \quad \boldsymbol{x}_{hard} \in \mathbb{R}^{d+2} = \begin{cases} x_0 = 0, \\ x_{i\in\{1,\dots,d\}} \overset{i.i.d.}{\sim} \mathcal{N}(\beta\eta y, 1), \\ x_{d+1} = \begin{cases} \mathcal{N}(\eta y, 1), & \textit{training phase} \\ \mathcal{N}(\beta\eta y, 1), & \textit{test phase, unseen data} \end{cases} \end{cases} \tag{7}$$

The hard example data distribution is characterized by the robust feature $x_0$ taking a value of 0. The memorization feature, while being a non-robust feature, is defined to behave like a robust feature during training. We define the set of datasets that can be generated from this distribution as $\mathcal{X}_{hard}$.

**Analysis on the effect of training hard examples** Next, we compare the performance of a binary classification model trained on datasets sampled from the data distributions introduced above. To train the model, we sample $m$ normal examples to form the training dataset $\mathcal{X}^{train}_{normal} \subset \mathcal{X}_{normal}$ and sample $n$ hard examples to construct a hard example training set $\mathcal{X}^{train}_{hard} \subset \mathcal{X}_{hard}$. We then examine the case where the model is trained using the combined dataset $\mathcal{X}^{train}_{normal} \cup \mathcal{X}^{train}_{hard}$, consisting of $m+n$ data points. For simplicity, we redefine the model $f$ as $f = \text{sign}(\boldsymbol{w}^\top \boldsymbol{x} + b)$ and assume $\|\boldsymbol{x}\| \leq 1$. Here, the weight $\boldsymbol{w}_{adv}$ of the adversarially trained model $f_{adv}$ (with $\beta\eta < \epsilon < \eta$) is learned, and the model's standard and robust accuracies are expressed as follows.

$$\begin{cases} w_0 = \frac{m}{Z_{adv}(m+n)}, \\ w_{i\in\{1,\dots,d\}} = 0, \\ w_{d+1} = \frac{n}{Z_{adv}(m+n)}, \end{cases} \quad \begin{cases} Pr\big[f_{adv}(\boldsymbol{x}) = y\big] = Pr\left[\frac{1}{Z_{adv}}\left(\frac{m}{m+n}\cdot\mathcal{N}(\eta,1)\right) + by > 0\right], \\ Pr\big[f_{adv}(\boldsymbol{x}+\boldsymbol{\delta}) = y\big] = Pr\left[\frac{1}{Z_{adv}}\left(\frac{m}{m+n}\cdot\mathcal{N}(\eta-\epsilon,1)\right) + by > 0\right], \end{cases} \tag{8}$$

Here, $Z_{adv}$ is a constant that satisfies $\|\boldsymbol{w}\| = 1$. The model's performance is measured on a subset of the normal example data distribution, $\mathcal{X}^{test}_{normal} \subset (\mathcal{X}_{normal} \setminus \mathcal{X}^{train}_{normal})$, which does not include the training data. Since normal examples do not have memorization features, the results are as above. Then, based on Equation 8, the following theorem holds when $m > 0$.

**Theorem 6 (Degradation of normal test accuracy with increasing hard examples in training)** *For a model $f_{adv}$ adversarially trained on the combined dataset $\mathcal{X}^{train}_{normal} \cup \mathcal{X}^{train}_{hard}$, where $\mathcal{X}^{train}_{normal}$ consists of $m$ normal examples and $\mathcal{X}^{train}_{hard}$ consists of $n$ hard examples, both the standard accuracy $Pr\big[f_{adv}(\boldsymbol{x}) = y\big]$ and the robust accuracy $Pr\big[f_{adv}(\boldsymbol{x}+\boldsymbol{\delta}) = y\big]$ on the normal example test dataset $\mathcal{X}^{test}_{normal}$ are monotonically decreasing with respect to $n$.*

The above theorem demonstrates that including additional hard examples during training does not contribute to improving the performance on normal examples in adversarial training. Now, we similarly evaluate the models' performance on hard examples. The performance is measured on a subset of the hard example data distribution, $\mathcal{X}^{test}_{hard} \subset (\mathcal{X}_{hard} \setminus \mathcal{X}^{train}_{hard})$.

$$\begin{cases} Pr\big[f_{adv}(\boldsymbol{x}_{hard}) = y\big] = Pr\left[\frac{1}{Z_{adv}}\left(\frac{n}{m+n}\cdot\mathcal{N}(\beta\eta,1)\right) + by > 0\right], \\ Pr\big[f_{adv}(\boldsymbol{x}_{hard}+\boldsymbol{\delta}) = y\big] = Pr\left[\frac{1}{Z_{adv}}\left(\frac{n}{m+n}\cdot\mathcal{N}(\beta\eta-\epsilon,1)\right) + by > 0\right], \end{cases} \tag{9}$$

Hard examples do not contain robust features, and their performance is determined by the model weight corresponding to the memorization feature. Then, based on Equation 9, the following theorem holds when $n > 0$.

**Theorem 7 (Trade-off in hard test accuracy with increasing hard examples in training)** *For a model $f_{adv}$ adversarially trained on the combined dataset $\mathcal{X}^{train}_{normal} \cup \mathcal{X}^{train}_{hard}$, where $\mathcal{X}^{train}_{normal}$ consists of $m$ normal examples and $\mathcal{X}^{train}_{hard}$ consists of $n$ hard examples, the standard accuracy $Pr\big[f_{adv}(\boldsymbol{x}) = y\big]$ on the hard example test dataset $\mathcal{X}^{test}_{hard}$ is monotonically increasing with respect to $n$, but the robust accuracy $Pr\big[f_{adv}(\boldsymbol{x} + \boldsymbol{\delta}) = y\big]$ is monotonically decreasing with respect to $n$.*

To summarize Theorem 6 and Theorem 7, including hard examples in adversarial training can lead to either an increase or decrease in standard accuracy. This is because the performance on normal examples decreases, while the performance on hard examples increases. However, robust accuracy always decreases because performance declines across all types of examples. The derivation process of the equations introduced in this section, as well as the proofs of the theorems and corollaries, can be found in Appendix A.2.

### 3.3 Empirical verification on the training of hard examples

In this section, we conduct an empirical analysis to validate the assumptions and theorems discussed in Section 3.2. We first verify the assumptions regarding hard examples, and then we empirically examine the theorems.

#### 3.3.1 Empirical verification of the properties of hard examples

We proceed with the validation of Assumption 1. If the assumption holds, the properties of a model learning hard examples are characterized as follows.

1. **Memorization**: For the hard example dataset $\mathcal{X}^{train}_{hard}$, the model $f_{adv,hard}$, which was adversarially trained with $\mathcal{X}^{train}_{hard}$ included, exhibits high robust accuracy on $\mathcal{X}^{train}_{hard}$. However, the model $f_{adv,\setminus hard}$, which was adversarially trained without $\mathcal{X}^{train}_{hard}$, shows nearly 0% robust accuracy on unseen $\mathcal{X}^{train}_{hard}$.

2. **Infeasibility of adversarial training**: Standard training is possible with only hard examples, but adversarial training is not. Specifically, let $\bar{f}$ be the model trained solely on $\mathcal{X}^{train}_{hard}$. Then, $Pr\big[\bar{f}_{std}(\boldsymbol{x})\big] > Pr\big[\bar{f}_{adv}(\boldsymbol{x})\big] > \frac{1}{C}$, while $Pr\big[\bar{f}_{adv}(\boldsymbol{x} + \boldsymbol{\delta})\big] < \frac{1}{C}$, where $C$ is the number of class.

3. **Trade-off**: As the proportion of hard examples in the training dataset increases, the test robust accuracy decreases, while the test standard accuracy increases.

**Property 1: Memorization** Based on Assumption 1, when a hard example dataset $\mathcal{X}^{train}_{hard}$ is included in adversarial training, the memorization feature acts as a robust feature for the hard examples included in the training. As a result, even when adversarial attacks are applied to these hard examples, the memorization feature enables high robust accuracy on these examples. In contrast, if the same dataset $\mathcal{X}^{train}_{hard}$ is not included in adversarial

training, the memorization feature acts as a non-robust feature for these hard examples or may not have been learned at all. In this case, when adversarial attacks are applied to these unseen hard examples, the memorization feature cannot be utilized, leaving no available features for prediction, resulting in 0% accuracy. To illustrate this, we conduct the following experiment.

For the CIFAR-10 dataset, the hard example data subset $\mathcal{X}_{hard}^{train}$ is composed of the top 5k hard examples. Two models are trained: one including the top 5k hard examples ($f_{adv,hard}$) and the other excluding them ($f_{adv,\backslash hard}$). The dataset including the hard examples was trained using all 50k CIFAR-10 data points, while the dataset excluding the hard examples was trained using the remaining 45k data points after removing the top 5k hard examples. After training, the performance of both models was evaluated on the top 5k hard examples $\mathcal{X}_{hard}^{train}$. Additionally, for comparison with normal examples, a normal example data subset was composed of the top 5k easy examples, and the same experiments were conducted.

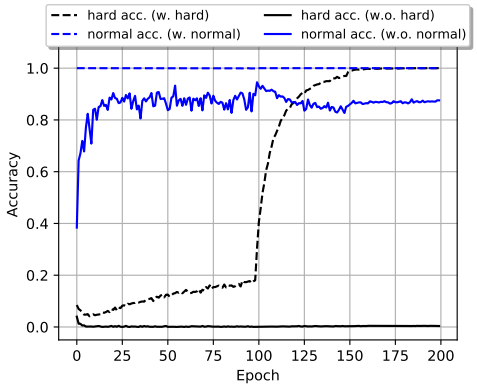



Figure 5: Comparison of the memorization property in adversarial training. Only hard examples exhibit the memorization property.

Fig. 5 compares the memorization property in adversarial training. Observably, normal examples show significant accuracy on data not included in the training, even if they were not part of the training process. In contrast, when hard examples are included in the training, the model can achieve up to 100% accuracy on these hard examples. However, when they are not included in the training, the model shows 0% accuracy. This demonstrates that, unlike normal examples, hard examples exhibit the memorization property. Additional details regarding Fig. 5 can be found in Appendix B.4.

**Property 2: Infeasibility of adversarial training** From the Assumption 1, when adversarial training is conducted using only hard examples, the model only learns the memorization feature; it leads to the robust accuracy that is lower than random prediction due to the memorization feature acting as a non-robust feature. Conversely, standard accuracy is higher than random prediction because of the same memorization feature. In contrast, standard training, which learns both non-robust and memorization features, will show higher standard accuracy compared to the adversarially trained model. To illustrate this, we conduct the following experiment.

We construct a hard example data subset using the top 10k hard examples and a normal example data subset using the top 10k easy examples. Then, instead of using the entire dataset, we train models using only these respective subsets through standard training and adversarial training to compare their performance. We measure the standard (Std.) and robust (Rob.) accuracy for the standard (STD) and adversarial (PGD) models.

Table 2: Performance of models trained with 10k examples.

| Method | STD | PGD | |
|---|---|---|---|
| | | Std. | Rob. |
| Normal | 78.46 | 61.63 | 39.56 |
| Hard | 77.60 | 19.92 | 8.68 |

In Table 2, the STD model trained with the hard example data subset exhibits performance comparable to that of the STD model trained with the easy example data subset. On the other hand, the PGD model trained with the hard example data subset shows lower standard accuracy than the STD model. As expected, in terms of robust accuracy, the model performs worse than random prediction (10%). Therefore, it can be concluded that standard training is possible with only hard examples, but adversarial training is not. Additional details can be found in Appendix B.4.

**Property 3: Trade-off** We have already demonstrated in Table 1 through the hard examples pruning experiment that learning hard examples exhibits the robustness-accuracy trade-off. To further support this, we additionally conduct a label corruption (shuffling) experiment, where the labels of hard examples are randomly changed. When label corruption is applied to hard examples, the corresponding weight of the memorization feature is learned incorrectly but does not become zero. Consequently, while this does not increase standard accuracy, the vulnerability of the non-robust feature remains, so unlike with pruning, robust accuracy is expected to remain unchanged rather than increase.

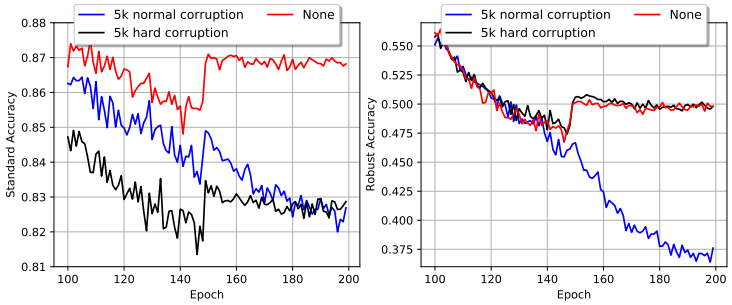
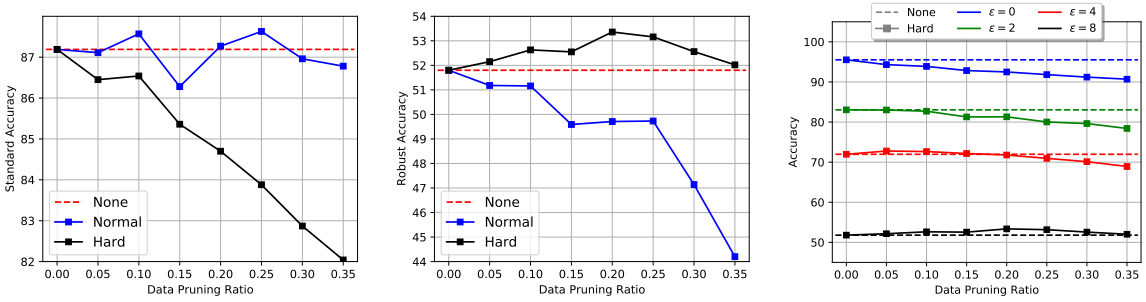



Figure 6: The standard and robust accuracies of the models whose labels of subsets are corrupted.

We construct a hard example data subset using the top 5k hard examples and a normal example data subset using the top 5k easy examples. We replace only the labels of data in each subset with random labels. We then conduct adversarial training on each of these datasets.

The models trained using the dataset with corrupted labels for normal and hard examples are shown in Fig. 6, along with their standard (left) and robust (right) accuracy curves. Examining the results for standard accuracy (left), it is evident that performance decreases for both normal and hard examples when label corruption is applied. In the results for robust accuracy (right), it is observed that while the performance of the models with corrupted normal examples consistently declines during training, the performance of the model with corrupted hard examples behaves similarly to that of the PGD model (None). This indicates that even when the labels of hard examples are corrupted, they do not significantly affect the robust accuracy of the adversarial training model. Additional details regarding Fig. 6 can be found in Appendix B.4.

Consequently, we empirically verify that the various characteristics that can arise in the learning of hard examples, as suggested by Assumption 1, appear as expected. Therefore, we confirm that Assumption 1—which posits that hard examples consist only of non-robust features and memorization features, with the memorization feature acting as a robust feature in the training data and as a non-robust feature in the test data—is supported to a certain degree.




(a) (b) (c)

Figure 7: (a) and (b) show the standard and robust accuracies of STD and PGD models where subset of training data is pruned. (c) shows the results that repeat the experiment (a) or (b) for different adversarial budgets.

### 3.3.2 Empirical verification of the training of hard examples

In this section, we conduct an empirical analysis of Theorem 6 and Theorem 7, which provide a theoretical analysis of the effect of increasing the number of hard examples in the training data during adversarial training. We empirically verify the theory through further experiments related to Fig. 1 and Table 1. We prune the top $n\%$ easy examples ($\subset \mathcal{X}_{normal}$) and the top $n\%$ hard examples ($\subset \mathcal{X}_{hard}$) from the training dataset of CIFAR-10, respectively ($0 \leq n \leq 35$). Figs. 7a and 7b show the standard and robust accuracies of the PGD pruning models.

In Fig. 7b, pruning hard examples improves the robust accuracy. This aligns with the theoretical analysis presented in Section 3.2, which suggests that the robustness of adversarial training models decreases as the number of hard examples in the training dataset increases. In contrast to hard examples, pruning normal examples leads to a decrease in performance. This outcome is likely due to the increased proportion of hard examples in the overall dataset.

Fig. 7c demonstrates that the improvement from pruning hard examples decreases as the adversarial budget $\epsilon$ decreases. Therefore, the performance improvement observed by pruning hard examples from the training dataset can be considered a characteristic of adversarial training.

Through Theorem 6 and Theorem 7, we observed that increasing the number of hard examples in the training dataset decreases the standard accuracy on normal examples but increases the standard accuracy on hard examples. Fig. 7a further confirms that the increase in standard accuracy is indeed predominant. To verify this, we conduct a theoretical analysis comparing the increase and decrease in standard accuracy as presented in Theorem 6 and Theorem 7. As noted in Corollary 2 in Appendix A.3, when the number of normal examples is sufficiently greater than the number of hard examples, increasing the number of hard examples in the training dataset leads to an increase in overall standard accuracy, as observed in Fig. 7a. Since typical datasets are assumed to have more normal examples than hard examples, this condition appears to be probable.

## 4. Method

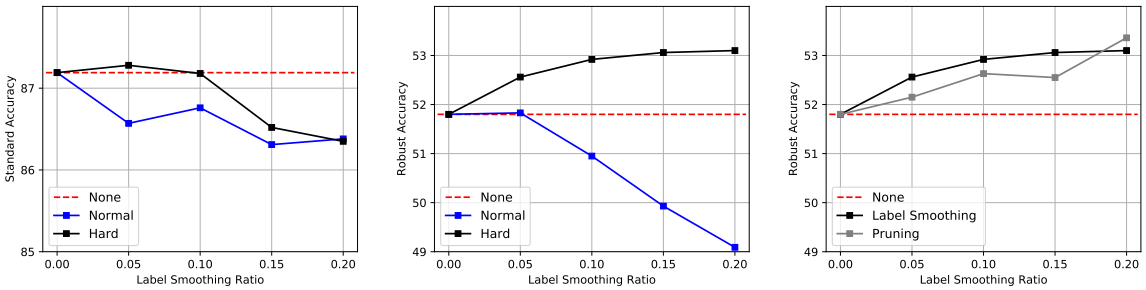
In this section, we explore approaches to mitigate the negative effects of learning hard examples. We evaluate the effectiveness of methodologies that regularize the learning of hard examples based on label smoothing (LS). Through this evaluation, we find that basic regularization techniques like label smoothing are inadequate for regularizing the learning of hard examples. To address this, we theoretically demonstrate that the appropriate direction for label smoothing regularization in the context of hard examples is to allocate the label smoothing factor in proportion to the difficulty of the hard examples. Accordingly, we propose Difficulty Proportional Label Smoothing (DPLS), a method that properly regularizes hard examples and enables effective adversarial training.

### 4.1 Mitigation of the negative effect of hard examples

Although training hard examples impairs the performance in adversarial training, simply pruning hard examples from training is not the optimal solution. In Section 3.2, we defined normal and hard examples as disjoint distributions, but in real data, this may not be the case. As shown in Fig. 7b, pruning the top 20% of hard examples improves performance, but beyond that point, performance starts to decline. Moreover, according to Fig. 7c, the point at which performance degradation begins varies depending on the adversarial budget $\epsilon$. These results suggest that finding the appropriate hard example pruning point depends on the type of dataset and the adversarial budget, making the task of identifying the optimal difficulty threshold for removing hard examples that deteriorate robustness an intractable proposition.

Thus, instead of pruning hard examples, we applied label smoothing (LS) (Szegedy et al., 2016) to subsets of the dataset to mitigate the negative effect of hard examples. Label smoothing (LS) is a technique aimed at reducing model overconfidence by assigning a value of $\lambda$ to the correct label while distributing the value $1-\lambda$ uniformly across all labels. Here, $\lambda$ represents the label smoothing factor and lies within the range $[\frac{1}{C}, 1]$, where $C$ is the number of class. Following the above pruning experiment, we applied LS ($\lambda = 0.9$) to the top $n\%$ of hard examples during training and then evaluated the model's performance.

Figures. 8b depict the test robust accuracy of the PGD LS models. As a result of pruning models, applying LS to the hard example subset increases robust accuracy, but applying LS to the normal example subset decreases performance. Fig. 8c shows that the increase in robust accuracy is similar to that observed with pruning. According to Fig. 8a, only marginal decreases are observed in standard accuracy. This implies that LS produces effects similar to pruning; however, the decrease in standard accuracy is less pronounced compared to pruning. Nevertheless, considering the performance drop observed when LS is applied to normal examples, it is expected that continuing to increase the proportion of hard examples to which LS is applied will eventually lead to a decrease in performance. Consequently, similar to pruning, determining the optimal difficulty threshold for applying LS to hard examples that negatively impact robustness is a challenging and complex task.




(a) (b) (c)

Figure 8: (a) the test standard accuracy and (b) robust accuracy of the PGD LS models. (c) the performance comparisons of the PGD LS and pruning models.

### 4.2 Our method to mitigate the negative effect of hard examples

Since it is difficult to accurately measure and define the difficulty of data, it is challenging to precisely determine whether a given data point is a hard example that does not contain robust features. We can expect that there will be data points on the boundary between hard and normal examples, depending on the difficulty of learning the data. Therefore, considering this difficulty, we define an input-label pair distribution that includes both hard and normal examples as follows.

**Definition 8 (Data distribution with difficulty)**

$$y \overset{u.a.r.}{\sim} \{-1,+1\}, \qquad \boldsymbol{x} \in \mathbb{R}^{d+2} = \begin{cases} x_0 = \mathcal{N}(\alpha\eta y, 1), \\ x_{i\in\{1,\dots,d\}} \overset{i.i.d.}{\sim} \mathcal{N}(\beta\eta y, 1), \\ x_{d+1} = \begin{cases} \mathcal{N}(\gamma\eta y, 1), & \textit{training phase} \\ \mathcal{N}(\beta\eta y, 1), & \textit{test phase, unseen data} \end{cases} \end{cases} \tag{10}$$

Here, $0 \leq \alpha \leq 1$ represents the difficulty of the data, with each data point having a different value. This reflects the difficulty through the coefficient of the robust feature, with lower difficulty corresponding to a higher value of $\alpha$. When $\alpha\eta \leq \epsilon$, the data point can be considered a hard example. $0 \leq \gamma \leq 1$ represents the coefficient of the memorization feature, which is inversely related to the robust feature coefficient $\alpha$; when $\gamma\eta > \epsilon$, the data point is classified as a hard example.

Next, we explore approaches to mitigate the negative effects of hard examples (data with low $\alpha$) when training a model using a dataset composed of data points with varying levels of difficulty. Unlike pruning, LS sacrifices standard accuracy less and also allows for adjusting the intensity of the method through the LS factor $\lambda$ ($\lambda \in [\frac{1}{C}, 1]$). Therefore, we explore an approach that adaptively mitigates the negative effects of hard examples by leveraging these characteristics of LS.

It is established that during adversarial training, the optimal weight configuration for achieving high robust accuracy is one where the weight values corresponding to features other than the robust feature are zero, i.e., $\boldsymbol{w}^* = (1, 0, 0, \dots)$. Therefore, when training

a model using a dataset composed of data with varying levels of difficulty, we explore an LS approach that aims to train the model's weight $\boldsymbol{w}$ to be as close as possible to $\boldsymbol{w}^*$. Let $\mathcal{X}_{difficulty}$ denote the set of datasets that can be generated by sampling from Equation 10, and let the training dataset be $\mathcal{X}^{train}_{difficulty} \subset \mathcal{X}_{difficulty}$. Then, the optimal LS factor for each data point that trains the model's weight $\boldsymbol{w}$ to be close to $\boldsymbol{w}^*$ during adversarial training with $\mathcal{X}^{train}_{difficulty}$ follows the theoretical principles outlined below.

**Theorem 9** *Let the robust feature coefficients of each data point in the dataset $\mathcal{X}^{train}_{difficulty}$, consisting of $k$ data points $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_k$, be denoted as $\alpha_1, \alpha_2, \ldots, \alpha_k$. Then, when performing adversarial training using $\mathcal{X}^{train}_{difficulty}$, the optimal LS factors $\lambda_1, \lambda_2, \ldots, \lambda_k$ for each training example are proportional to their respective robust feature coefficients $\alpha_1, \alpha_2, \ldots, \alpha_k$.*

The proof of Theorem 9 can be found in Appendix A.4. Theorem 9 demonstrates that applying LS in proportion to the robust feature coefficient of each data point during adversarial training is the optimal learning approach for achieving robustness. In other words, this suggests that the optimal LS approach for adversarial training is to decrease the LS factor as the difficulty of the data increases, making the intensity of smoothing regularization proportional to the difficulty. Therefore, based on Theorem 9, we propose a regularization approach that adjusts the LS factor for each example during training.

We modify the LS method to ensure that the training of hard examples is regularized, but the training of other examples is less affected. We use the LS factor that is proportional to the difficulty of each example and control the intensity of regularization, which we term as difficulty proportional label smoothing (DPLS). The difficulty is calculated by using the average loss of the training models along the training trajectory. We train the model and accumulate the difficulty until the difficulty calculation epoch $T$, and the training progresses with the application of DPLS using the calculated difficulty. We establish the LS factor of the easiest example as $\lambda = 1$ (no smoothing) and the factor of the most difficult example as $\lambda \in [\frac{1}{C}, 1)$, where $C$ represents the number of the class. The procedure of DPLS is described in Algorithms 1 and 2.

Our method maintains the training effect of each training example, but it avoids the negative effect of hard examples. We could utilize several loss types for difficulty measurement including cross-entropy loss, the norm of the error vector (EL2N score) (Paul et al., 2021), consistency score (C-score) (Jiang et al., 2021), and 0-1 loss. Applying DPLS via any error measure is an effective strategy for mitigating the negative effect of hard examples (refer to Table 5). However, in our experiments, as the 0-1 loss exhibits a higher performance than the other loss types, we mainly use the 0-1 loss as the difficulty measurement loss. Notably, our method calculates the difficulty using clean examples because the adversarial examples are different from training algorithms and training epochs.

**Algorithm 1** DPLS adversarial training

**Require:** Dataset $D$, model parameter $\boldsymbol{\theta}$, batch size $n$, difficulty calculation epoch $T$, total training epoch $K$, learning rate $\alpha$, difficulty loss storage $\mathcal{S}$

**for** $t = 1$ **to** $K$ **do**
  **for** batch **in** $D$ **do**
    batch and indices:
    $\{\boldsymbol{x}_i, \boldsymbol{y}_i\}_{i=1}^n, \mathcal{I} \sim D$
    compute or apply DPLS:
    **if** $t < T$ **then**
      $\mathcal{S}[\mathcal{I}] \leftarrow \{\mathcal{L}_{\text{difficulty}}(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^n$
    **else**
      $\{\boldsymbol{y}_i\}_{i=1}^n \leftarrow \text{DPLS}(\mathcal{S}[\mathcal{I}])$
    **end if**
    adversarial training:
    $\{\boldsymbol{\delta}_i\}_{i=1}^n \leftarrow \arg\max_{\boldsymbol{\delta}_i} \mathcal{L}_{\text{adv}}(\boldsymbol{x}_i + \boldsymbol{\delta}_i, \boldsymbol{y}_i)$
    $\mathcal{L} \leftarrow \frac{1}{n} \sum_{i=i}^n \mathcal{L}_{\text{adv}}(\boldsymbol{x}_i + \boldsymbol{\delta}_i, \boldsymbol{y}_i)$
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha \cdot \nabla_\theta \mathcal{L}$
  **end for**
**end for**
**Output:** robust classifier $f$

**Algorithm 2** Calculation of DPLS

**Require:** Dataset $D$, Dataset size $N$, number of class $C$, loss storage $\mathcal{S}$, DPLS factor $\lambda$

**Initialization:**
  apply min-max normalization:
  $\mathcal{S} \leftarrow \frac{\mathcal{S} - \min(\mathcal{S})}{\max(\mathcal{S}) - \min(\mathcal{S})}$
**for** $i = 1$ **to** $N$ **do**
  $(\boldsymbol{x}_i, \boldsymbol{y}_i) \leftarrow D[i]$
  apply LS using $\mathcal{S}$:
  $\boldsymbol{y}_i \leftarrow LS(\boldsymbol{y}_i, 1 - (1 - \lambda)\mathcal{S}[i], C)$
  $D[i] \leftarrow (\boldsymbol{x}_i, \boldsymbol{y}_i)$
**end for**
**function** $LS(\boldsymbol{y}, \lambda, C)$
  **for** $i = 1$ **to** $C$ **do**
    **if** $\boldsymbol{y}[i] = 1$ **then**
      $\boldsymbol{y}[i] \leftarrow \lambda$
    **else**
      $\boldsymbol{y}[i] \leftarrow \frac{1-\lambda}{C-1}$
    **end if**
  **end for**
  **return** $\boldsymbol{y}$

## 5. Experiment

**Implementation details** We conducted experiments on CIFAR-10, CIFAR-100 (Krizhevsky et al., 2009), SVHN (Netzer et al., 2011), and STL-10 (Coates et al., 2011) by employing PGD (Madry et al., 2017) and TRADES (Zhang et al., 2019) as the baseline adversarial training algorithms. We utilized WideResNet28-10 (Zagoruyko and Komodakis, 2016) as the architecture of our models. We set the LS factor of the most difficult example in DPLS considering the average difficulty score of each dataset and algorithm. We selected the difficulty calculation epoch as $T = 90$ for stability, which is 10 epochs before the first learning rate decay. To evaluate the robustness of the models, we employed the PGD and adaptive auto attack ($A^3$) (Liu et al., 2022b). Further details are presented in Appendix B.1.

**Improvement in robustness** We compared four mitigation methods: baseline (-), pruning hard examples (pruning), label smoothing (LS), and our proposed method (DPLS). We pruned the top 10% of examples with high difficulty for pruning models. For LS models, we applied the same LS factor with the average factor of DPLS. Thus, with the same LS budget, our method assigns a large portion of the budget to the hard examples and LS assigns it uniformly to every example. We trained the baseline model until epoch 90 and continued this training with the application each method. Table 3 shows that pruning hard examples and DPLS are effective for mitigating the effect of hard examples and improving the robustness

Table 3: Performance of the models for mitigation methods. The best results are indicated in bold.

| Dataset | Method | Std. | Robust (PGD) | Robust ($A^3$) | Method | Std. | Robust (PGD) | Robust ($A^3$) |
|---|---|---|---|---|---|---|---|---|
| CIFAR-10 | PGD | **87.19** | 56.44 | 51.80 | TRADES | 85.66 | 58.46 | 54.08 |
| | +Pruning | 86.44 | 56.18 | 52.56 | +Pruning | 84.07 | 58.04 | 54.64 |
| | +LS | 86.96 | 57.20 | 51.99 | +LS | **86.32** | 58.72 | 54.02 |
| | +DPLS | 87.13 | **58.43** | **54.03** | +DPLS | 85.36 | **59.33** | **55.19** |
| CIFAR-100 | PGD | 62.88 | 31.82 | 27.36 | TRADES | 60.78 | 33.19 | 28.05 |
| | +Pruning | 62.28 | 31.64 | 28.05 | +Pruning | 60.61 | 33.53 | 29.01 |
| | +LS | **63.02** | 32.26 | 27.67 | +LS | 62.43 | 33.29 | 27.83 |
| | +DPLS | 62.95 | **35.10** | **29.33** | +DPLS | **63.32** | **34.53** | **29.68** |
| SVHN | PGD | **92.96** | 52.88 | 35.20 | TRADES | 91.84 | 59.14 | 47.46 |
| | +Pruning | 92.68 | 55.52 | 40.87 | +Pruning | 91.54 | **60.27** | 47.79 |
| | +LS | 92.65 | 66.20 | 38.44 | +LS | 92.27 | 59.65 | 46.37 |
| | +DPLS | 92.94 | **76.53** | **42.01** | +DPLS | **92.33** | 59.40 | **49.05** |
| STL-10 | PGD | **66.95** | 36.49 | 33.12 | TRADES | 66.40 | 36.27 | 33.16 |
| | +Pruning | 66.75 | 35.27 | 33.15 | +Pruning | 65.39 | 37.39 | 34.91 |
| | +LS | 66.90 | 35.66 | 32.05 | +LS | **66.49** | 37.89 | 33.99 |
| | +DPLS | 66.16 | **38.27** | **34.75** | +DPLS | 65.63 | **39.94** | **36.06** |

(a) $\epsilon = 8$

| Dataset | Method | Std. | Robust (PGD) | Robust ($A^3$) | Method | Std. | Robust (PGD) | Robust ($A^3$) |
|---|---|---|---|---|---|---|---|---|
| CIFAR-10 | PGD | 91.78 | 72.60 | 71.96 | TRADES | 90.36 | 74.37 | 73.50 |
| | +Pruning | 91.24 | 73.03 | 72.64 | +Pruning | 89.12 | 73.76 | 73.32 |
| | +LS | **92.21** | 72.39 | 71.55 | +LS | **90.92** | 74.78 | 73.67 |
| | +DPLS | 92.07 | **74.90** | **74.01** | +DPLS | 90.43 | **75.38** | **74.47** |

(b) $\epsilon = 4$

| Dataset | Method | Std. | Robust (PGD) | Robust ($A^3$) | Method | Std. | Robust (PGD) | Robust ($A^3$) |
|---|---|---|---|---|---|---|---|---|
| CIFAR-10 | PGD | 94.17 | 83.20 | 83.04 | TRADES | **93.32** | 84.42 | 84.29 |
| | +Pruning | 92.95 | 82.73 | 82.70 | +Pruning | 92.08 | 83.84 | 83.69 |
| | +LS | **94.56** | 83.46 | 83.13 | +LS | 93.14 | 84.33 | 83.96 |
| | +DPLS | 94.06 | **84.87** | **84.51** | +DPLS | 92.94 | **85.18** | **84.79** |

(c) $\epsilon = 2$

generalization. The performance improvement of the pruning models demonstrates that the negative effect of hard examples appears in various datasets and algorithms. The results of LS models show no improvements in the robustness performance. Contrarily, the DPLS increases the robustness of the models across different training algorithms and datasets for all adversarial budgets $\epsilon$. The robustness performance difference between LS and DPLS indicates that regularization without considering the effect of each example on training is difficult to improve robust generalization.

Fig. 9 illustrates the training and test robust accuracy results (left) of the PGD and PGD with DPLS (denoted as DPLS) models and training accuracy of top N% (right) hard

Table 4: Performance comparison with other robust overfitting mitigation methods.

| Method | Std. | Robust (PGD) | Robust ($A^3$) |
|---|---|---|---|
| PGD | 87.19 | 56.44 | 51.8 |
| +SAT | 86.94 | 57.52 | 52.94 |
| +KD+SWA | **88.28** | 57.93 | 53.49 |
| +TE | 85.83 | 58.31 | 52.84 |
| +DPLS | 86.17 | 58.14 | 53.98 |
| +DPLS+SWA | 87.14 | **60.07** | **55.91** |

Table 5: Robust accuracy of DPLS according to various difficulty measurement.

| Method | PGD | TRADES |
|---|---|---|
| - | 51.8 | 54.08 |
| C-score | 52.9 | 54.71 |
| CE loss | 52.73 | 54.45 |
| EL2N | 53.38 | 54.32 |
| 0-1 loss | **54.03** | **55.19** |

Table 6: Performance improvements in other adversarial training algorithms.

| Method | Std. | Robust (PGD) | Robust ($A^3$) |
|---|---|---|---|
| AWP | **84.82** | 60.19 | 55.54 |
| +DPLS | 84.4 | **60.26** | **56.09** |
| MART | **83.9** | 59.05 | 52.19 |
| +DPLS | 83.81 | **59.54** | **53.18** |
| RST | **84.82** | 61.53 | 57.26 |
| +DPLS | 84.43 | **62.03** | **57.81** |

examples of the 50,000 training examples of CIFAR-10 (e.g. (0-20%) denotes the top 10k hard examples). Observably, comparing DPLS with PGD, the decrease in the training accuracy of the DPLS model coincides with the increase in the test accuracy. DPLS (right) prevents overfitting of hard examples and causes the fast convergence of comparatively easy examples. This result indicates that our method prevents the memorization of hard examples and mitigate the negative effect of hard examples, thereby leading to decrease in the gap between training and test accuracy by improving the generalization performance.

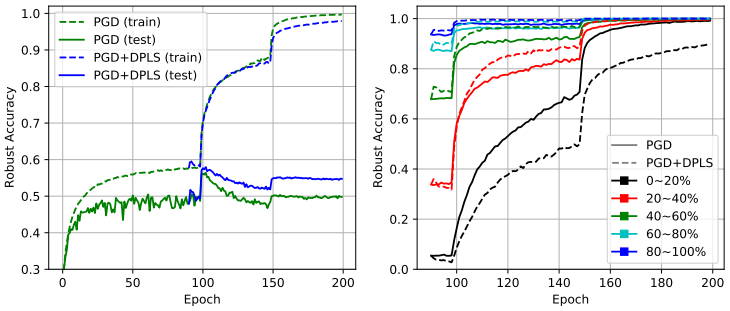



Figure 9: The training and test accuracy curves of the PGD and the PGD with DPLS model on CIFAR-10.

**Comparison with other mitigation methods** We compared DPLS with the previous robust overfitting mitigation methods: self-adaptive training (SAT) (Huang et al., 2020), knowledge distillation with stochastic weight averaging (KD+SWA) (Chen et al., 2020), and temporal ensemble (TE) (Dong et al., 2021). We trained the PGD model combined with each method and with total training epoch 200. The details on the settings of each method are described in Appendix B.5. The results listed in Table 4 indicate that all the methods are effective at mitigating robust overfitting. Among the methods, DPLS exhibits the highest improvement in the test robust accuracy against $A^3$. Additionally, because our method is orthogonal to SWA (Izmailov et al., 2018), we trained the model with the method combining DPLS and SWA. Further improvement is observed when combined with SWA. These results indicate that our method successfully mitigates the memorization of hard examples and prevents overfitting.

**Applying DPLS to other training algorithms** We applied our method to other adversarial training algorithms to verify that DPLS can consistently increase the robustness generalization performance independent of algorithms. We combined DPLS with three algorithms: adversarial weight perturbation (AWP) (Wu et al., 2020), misclassification aware adversarial training (MART) (Wang et al., 2019), and robust self training (RST) (Car-

mon et al., 2019). We trained each algorithm model combined with TRADES+DPLS and evaluated the trained models. Table 6 shows that all training algorithms combined with DPLS yield improvements in robustness. For the RST models, we calculated the difficulty of the unlabeled dataset as the ratio of number of training iterations to the number of right predictions. The result shows that DPLS successfully mitigates the negative effect of hard examples for the unlabeled dataset and improves the robust accuracy for the $A^3$ and PGD attacks.

**Various difficulty loss on DPLS** We compared the loss types of the difficulty for DPLS. Table 5 shows the performance of the TRADES models that apply DPLS with several loss functions: C-score, cross-entropy loss (CE loss), EL2N score, and 0-1 loss. All models in which DPLS was applied exhibited a better performance than the baseline models. The difficulty calculated from another model (e.g. C-score from STD) was also effective on mitigation. Any difficulty loss was effective, and DPLS with the 0-1 loss offered the best performance. The results indicate that the 0-1 loss leverages comparatively objective values because the 0-1 loss is less affected by scale or variance at each checkpoint of the model. As the calculation process can be detached from the training, further improvement is achievable by using the better difficulty measurement loss. Further experiments and ablation studies on DPLS can be found in Appendix B.5 and Appendix D.

## 6. Conclusion

In this study, we analyzed the effect of hard examples in adversarial training. We demonstrated that training hard examples can deteriorate the performance in adversarial training and verified that the cause is the memorization of hard examples. We accordingly conducted an analysis of the memorization phenomenon and observed that increasing the number of hard examples during adversarial training leads to a decrease in the model's robust accuracy. While we found that pruning hard examples from the training process enhances the model's robustness, the difficulty lies in identifying the optimal threshold for removing hard examples that weaken robustness performance. Thus, we proposed a method, DPLS, which adaptively mitigates the memorization of hard examples. Through experiments on various datasets and algorithms, we verified that our method could successfully leverage hard examples, thereby improving robustness.

## Acknowledgments

This work was supported by the National Research Foundation of Korea (NRF) grants funded by the Korea government (Ministry of Science and ICT, MSIT) (2022R1A3B1077720 and 2022R1A5A708390811), by the Institute of Information & Communications Technology Planning & Evaluation (IITP) grants funded by the Korea government (MSIT) (NO.RS-2021-II211343 for the Artificial Intelligence Graduate School Program at Seoul National University and RS-2024-004398199 for AI-Based Automated Vulnerability Detection and Safe Code Generation), by the Korea Evaluation Institute of Industrial Technology (KEIT) grants funded by the Korea government (MOTIE), and by the BK21 FOUR program of the

Education and Research Program for Future ICT Pioneers, Seoul National University in 2025.

## References

Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *International conference on machine learning*, pages 274–283. PMLR, 2018.

Yair Carmon, Aditi Raghunathan, Ludwig Schmidt, John C Duchi, and Percy S Liang. Unlabeled data improves adversarial robustness. In *Advances in Neural Information Processing Systems*, pages 11190–11201, 2019.

Jinghui Chen and Quanquan Gu. Rays: A ray searching method for hard-label adversarial attack. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1739–1747, 2020.

Tianlong Chen, Zhenyu Zhang, Sijia Liu, Shiyu Chang, and Zhangyang Wang. Robust overfitting may be mitigated by properly learned smoothening. In *International Conference on Learning Representations*, 2020.

Adam Coates, Andrew Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 215–223. JMLR Workshop and Conference Proceedings, 2011.

Francesco Croce and Matthias Hein. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. *arXiv preprint arXiv:2003.01690*, 2020.

Francesco Croce, Maksym Andriushchenko, Vikash Sehwag, Edoardo Debenedetti, Nicolas Flammarion, Mung Chiang, Prateek Mittal, and Matthias Hein. Robustbench: a standardized adversarial robustness benchmark. *arXiv preprint arXiv:2010.09670*, 2020.

Terrance DeVries and Graham W Taylor. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*, 2017.

Chengyu Dong, Liyuan Liu, and Jingbo Shang. Label noise in adversarial training: A novel perspective to study robust overfitting. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022. URL `https://openreview.net/forum?id=9_O9mTLYJQp`.

Yinpeng Dong, Ke Xu, Xiao Yang, Tianyu Pang, Zhijie Deng, Hang Su, and Jun Zhu. Exploring memorization in adversarial training. *arXiv preprint arXiv:2106.01606*, 2021.

Vitaly Feldman. Does learning require memorization? a short tale about a long tail. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, pages 954–959, 2020.

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

Sven Gowal, Sylvestre-Alvise Rebuffi, Olivia Wiles, Florian Stimberg, Dan Andrei Calian, and Timothy A Mann. Improving robustness using generated data. *Advances in Neural Information Processing Systems*, 34, 2021a.

Sven Gowal, Sylvestre-Alvise Rebuffi, Olivia Wiles, Florian Stimberg, Dan Andrei Calian, and Timothy A Mann. Improving robustness using generated data. *Advances in Neural Information Processing Systems*, 34, 2021b.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

Geoffrey Hinton, Oriol Vinyals, Jeff Dean, et al. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2(7), 2015.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

Lang Huang, Chao Zhang, and Hongyang Zhang. Self-adaptive training: beyond empirical risk minimization. *Advances in neural information processing systems*, 33:19365–19376, 2020.

Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Logan Engstrom, Brandon Tran, and Aleksander Madry. Adversarial examples are not bugs, they are features. *arXiv preprint arXiv:1905.02175*, 2019.

Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. Averaging weights leads to wider optima and better generalization. *arXiv preprint arXiv:1803.05407*, 2018.

Linxi Jiang, Xingjun Ma, Zejia Weng, James Bailey, and Yu-Gang Jiang. Imbalanced gradients: A new cause of overestimated adversarial robustness. 2020.

Ziheng Jiang, Chiyuan Zhang, Kunal Talwar, and Michael C Mozer. Characterizing structural regularities of labeled data in overparameterized models. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 5034–5044. PMLR, 18–24 Jul 2021. URL `https://proceedings.mlr.press/v139/jiang21k.html`.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

Saehyung Lee, Changhwa Park, Hyungyu Lee, Jihun Yi, Jonghyun Lee, and Sungroh Yoon. Removing undesirable feature contributions using out-of-distribution data. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=eIHYL6fpbkA`.

Saehyung Lee, Sanghyuk Chun, Sangwon Jung, Sangdoo Yun, and Sungroh Yoon. Dataset condensation with contrastive signals. *arXiv preprint arXiv:2202.02916*, 2022.

Chen Liu, Zhichao Huang, Mathieu Salzmann, Tong Zhang, and Sabine Süsstrunk. On the impact of hard adversarial instances on overfitting in adversarial training. *arXiv preprint arXiv:2112.07324*, 2021.

Chen Liu, Zhichao Huang, Mathieu Salzmann, Tong Zhang, and Sabine Süsstrunk. On the impact of hard adversarial instances on overfitting in adversarial training, 2022a. URL `https://openreview.net/forum?id=hbGV3vzMPzG`.

Ye Liu, Yaya Cheng, Lianli Gao, Xianglong Liu, Qilong Zhang, and Jingkuan Song. Practical evaluation of adversarial robustness via adaptive auto attack. *arXiv preprint arXiv:2203.05154*, 2022b.

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.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. 2011.

Weili Nie, Brandon Guo, Yujia Huang, Chaowei Xiao, Arash Vahdat, and Animashree Anandkumar. Diffusion models for adversarial purification. In *International Conference on Machine Learning*, pages 16805–16827. PMLR, 2022.

Tianyu Pang, Xiao Yang, Yinpeng Dong, Hang Su, and Jun Zhu. Bag of tricks for adversarial training. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=Xb8xvrtB8Ce`.

Mansheej Paul, Surya Ganguli, and Gintare Karolina Dziugaite. Deep learning on a data diet: Finding important examples early in training. *Advances in Neural Information Processing Systems*, 34, 2021.

Sylvestre-Alvise Rebuffi, Sven Gowal, Dan Andrei Calian, Florian Stimberg, Olivia Wiles, and Timothy A Mann. Data augmentation can improve robustness. *Advances in Neural Information Processing Systems*, 34:29935–29948, 2021.

Ludwig Schmidt, Shibani Santurkar, Dimitris Tsipras, Kunal Talwar, and Aleksander Madry. Adversarially robust generalization requires more data. In *Advances in Neural Information Processing Systems*, pages 5014–5026, 2018.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

Antonio Torralba, Rob Fergus, and William T Freeman. 80 million tiny images: A large data set for nonparametric object and scene recognition. *IEEE transactions on pattern analysis and machine intelligence*, 30(11):1958–1970, 2008.

Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. Robustness may be at odds with accuracy. *arXiv preprint arXiv:1805.12152*, 2018.

Yisen Wang, Difan Zou, Jinfeng Yi, James Bailey, Xingjun Ma, and Quanquan Gu. Improving adversarial robustness requires revisiting misclassified examples. In *International Conference on Learning Representations*, 2019.

Eric Wong, Leslie Rice, and Zico Kolter. Overfitting in adversarially robust deep learning. In *Proceedings of Machine Learning and Systems 2020*, pages 5304–5315. 2020.

Dongxian Wu, Shu-Tao Xia, and Yisen Wang. Adversarial weight perturbation helps robust generalization. *Advances in Neural Information Processing Systems*, 33:2958–2969, 2020.

Zuxuan Wu, Tushar Nagarajan, Abhishek Kumar, Steven Rennie, Larry S Davis, Kristen Grauman, and Rogerio Feris. Blockdrop: Dynamic inference paths in residual networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8817–8826, 2018.

Han Xu, Xiaorui Liu, Yaxin Li, Anil Jain, and Jiliang Tang. To be robust or to be fair: Towards fairness in adversarial training. In *International Conference on Machine Learning*, pages 11492–11501. PMLR, 2021.

Chaojian Yu, Bo Han, Li Shen, Jun Yu, Chen Gong, Mingming Gong, and Tongliang Liu. Understanding robust overfitting of adversarial training and beyond. In *International Conference on Machine Learning*, pages 25595–25610. PMLR, 2022.

Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 6023–6032, 2019.

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

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

Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations*, 2018.

## Appendix A. Proofs

### A.1 Section 3.1 proofs

**Definition 1 (Gradient difference)** *The differences between the loss gradient norm on $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ in standard and adversarial training are defined as follows:*

$$\mathcal{D}_{\text{std}}(\boldsymbol{x}_1, \boldsymbol{x}_2) = \Big\|(1-\sigma(\boldsymbol{w}^\top \boldsymbol{x}_1 + b))\boldsymbol{x}_1\Big\| - \Big\|(1-\sigma(\boldsymbol{w}^\top \boldsymbol{x}_2 + b))\boldsymbol{x}_2\Big\|$$
$$\mathcal{D}_{\text{adv}}(\boldsymbol{x}_1, \boldsymbol{x}_2, \epsilon) = \Big\|(1-\sigma(\boldsymbol{w}^\top (\boldsymbol{x}_1+\boldsymbol{\delta}) + b))(\boldsymbol{x}_1+\boldsymbol{\delta})\Big\| - \Big\|(1-\sigma(\boldsymbol{w}^\top (\boldsymbol{x}_2+\boldsymbol{\delta}) + b))(\boldsymbol{x}_2+\boldsymbol{\delta})\Big\|. \tag{11}$$

**Proof**

$$\begin{aligned}\frac{\partial \mathcal{L}(f(\boldsymbol{x}), y)}{\partial \boldsymbol{w}} &= -\frac{\partial}{\partial \boldsymbol{w}}\big(y \log f(\boldsymbol{x}) + (1-y)\log(1-f(\boldsymbol{x}))\big) \\ &= -\frac{\partial}{\partial \boldsymbol{w}}\big(y \log \sigma(\boldsymbol{w}^\top \boldsymbol{x} + b) + (1-y)\log\Big(1-\sigma(\boldsymbol{w}^\top \boldsymbol{x} + b)\Big)\big) \\ &= -\big(y(1-\sigma(\boldsymbol{w}^\top \boldsymbol{x} + b)) + (1-y)(-\sigma(\boldsymbol{w}^\top \boldsymbol{x} + b))\big)\boldsymbol{x}.\end{aligned}$$

Here, $y = +1$ for $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$

$$\frac{\partial \mathcal{L}(f(\boldsymbol{x}), y=+1)}{\partial \boldsymbol{w}} = -(1-\sigma(\boldsymbol{w}^\top \boldsymbol{x} + b))\boldsymbol{x}. \tag{12}$$

The gradient norm difference between $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ in standard training is represented as:

$$\left\|\frac{\partial \mathcal{L}(f(\boldsymbol{x}_1), y_1)}{\partial \boldsymbol{w}}\right\| - \left\|\frac{\partial \mathcal{L}(f(\boldsymbol{x}_2), y_2)}{\partial \boldsymbol{w}}\right\| = \Big\|(1-\sigma(\boldsymbol{w}^\top \boldsymbol{x}_1 + b))\boldsymbol{x}_1\Big\| - \Big\|(1-\sigma(\boldsymbol{w}^\top \boldsymbol{x}_2 + b))\boldsymbol{x}_2\Big\|. \tag{13}$$

Thus, the gradient norm difference between the adversarial examples is represented as:

$$\begin{aligned}&\left\|\frac{\partial \mathcal{L}(f(\boldsymbol{x}_1+\boldsymbol{\delta}), y_1)}{\partial \boldsymbol{w}}\right\| - \left\|\frac{\partial \mathcal{L}(f(\boldsymbol{x}_2+\boldsymbol{\delta}), y_2)}{\partial \boldsymbol{w}}\right\| \\ &\quad = \Big\|(1-\sigma(\boldsymbol{w}^\top (\boldsymbol{x}_1+\boldsymbol{\delta}) + b))(\boldsymbol{x}_1+\boldsymbol{\delta})\Big\| - \Big\|(1-\sigma(\boldsymbol{w}^\top (\boldsymbol{x}_2+\boldsymbol{\delta}) + b))(\boldsymbol{x}_2+\boldsymbol{\delta})\Big\|. \end{aligned} \tag{14}$$

■

**Theorem 1** *Let the gradient norm of easy examples is smaller than the gradient norm of hard examples in both standard and adversarial training. Then, if $\boldsymbol{w}^\top \boldsymbol{x}_1 + b \geq 0$ and $0 < \epsilon < \frac{(d_1+d_2)}{2}$, it satisfies*

$$\mathcal{D}_{\text{std}}(\boldsymbol{x}_1, \boldsymbol{x}_2) < \mathcal{D}_{\text{adv}}(\boldsymbol{x}_1, \boldsymbol{x}_2, \epsilon). \tag{15}$$

**Proof**

Because the gradient norm of easy examples is smaller than the gradient norm of hard examples in both standard and adversarial training, $\mathcal{D}_{\text{std}}(\boldsymbol{x}_1, \boldsymbol{x}_2) > 0$ and $\mathcal{D}_{\text{adv}}(\boldsymbol{x}_1, \boldsymbol{x}_2, \epsilon) > 0$. Then, we can rewrite Equation 16 as

$$\frac{\mathcal{D}_{\text{std}}(\boldsymbol{x}_1, \boldsymbol{x}_2)}{\mathcal{D}_{\text{adv}}(\boldsymbol{x}_1, \boldsymbol{x}_2, \epsilon)} < 1. \tag{16}$$

Let the prediction for $x_1$ is not incorrect, where $\boldsymbol{w}^\top \boldsymbol{x}_1 + b \geq 0$. Here, we assume the norm of every input $x$ that is defined in the input space $\mathcal{X}$ is almost the same. Then, the above inequality is represented as

$$\frac{\mathcal{D}_{\text{std}}(\boldsymbol{x}_1, \boldsymbol{x}_2)}{\mathcal{D}_{\text{adv}}(\boldsymbol{x}_1, \boldsymbol{x}_2, \epsilon)} = \frac{\big\|(1-\sigma(\boldsymbol{w}^\top \boldsymbol{x}_1 + b))\boldsymbol{x}_1\big\| - \big\|(1-\sigma(\boldsymbol{w}^\top \boldsymbol{x}_2 + b))\boldsymbol{x}_2\big\|}{\|(1-\sigma(\boldsymbol{w}^\top (\boldsymbol{x}_1+\boldsymbol{\delta}) + b))(\boldsymbol{x}_1+\boldsymbol{\delta})\| - \|(1-\sigma(\boldsymbol{w}^\top (\boldsymbol{x}_2+\boldsymbol{\delta}) + b))(\boldsymbol{x}_2+\boldsymbol{\delta})\|} \tag{17}$$

$$= \frac{(1-\sigma(\boldsymbol{w}^\top \boldsymbol{x}_1 + b))\|\boldsymbol{x}_1\| - (1-\sigma(\boldsymbol{w}^\top \boldsymbol{x}_2 + b))\|\boldsymbol{x}_2\|}{(1-\sigma(\boldsymbol{w}^\top (\boldsymbol{x}_1+\boldsymbol{\delta}) + b))\|(\boldsymbol{x}_1+\boldsymbol{\delta})\| - (1-\sigma(\boldsymbol{w}^\top (\boldsymbol{x}_2+\boldsymbol{\delta}) + b))\|(\boldsymbol{x}_2+\boldsymbol{\delta})\|} \tag{18}$$

$$= \frac{(1-\sigma(\boldsymbol{w}^\top \boldsymbol{x}_1 + b)) - (1-\sigma(\boldsymbol{w}^\top \boldsymbol{x}_2 + b))}{(1-\sigma(\boldsymbol{w}^\top (\boldsymbol{x}_1+\boldsymbol{\delta}) + b)) - (1-\sigma(\boldsymbol{w}^\top (\boldsymbol{x}_2+\boldsymbol{\delta}) + b))} \tag{19}$$

$$= \frac{\sigma(-\boldsymbol{w}^\top \boldsymbol{x}_1 - b) - \sigma(-\boldsymbol{w}^\top \boldsymbol{x}_2 - b)}{\sigma(-\boldsymbol{w}^\top (\boldsymbol{x}_1+\boldsymbol{\delta}) - b) - \sigma(-\boldsymbol{w}^\top (\boldsymbol{x}_2+\boldsymbol{\delta}) - b)} \tag{20}$$

$$= \frac{\sigma(-d_1) - \sigma(-d_2)}{\sigma(-(d_1-\epsilon)) - \sigma(-(d_2-\epsilon))} \tag{21}$$

$$= \frac{\sigma(-d_1) - \sigma(-(d_1+(d_2-d_1)))}{\sigma(-(d_1-\epsilon)) - \sigma(-(d_1-\epsilon+(d_2-d_1)))}. \tag{22}$$

Here, it satisfies Equation 16 if the function $h(x) = \sigma(-x) - \sigma(-(x+(d_2-d_1)))$ is monotonically decreasing for input $x \geq d_1 - \epsilon$, where $\epsilon > 0$. Let $a$ denote $(d_2 - d_1)$, where $a = (d_2 - d_1)$. Then,

$$\frac{d}{dx}\big(\sigma(-x) - \sigma(-(x+a))\big) = \frac{d}{dx}\left(\frac{1}{1+e^x} - \frac{1}{1+e^{x+a}}\right) \tag{23}$$

$$= e^x\left(\frac{e^a}{(1+e^{x+a})^2} - \frac{1}{(1+e^x)^2}\right) \tag{24}$$

$$= e^x\left(\frac{(e^a-1)(1-e^{2x+a})}{(e^x+1)^2(e^{x+a}+1)^2}\right) < 0. \tag{25}$$

Since $a > 0$, Equation 25 can be represented as $1 - e^{2x+a} < 0$. Then, we can rewrite Equation 25 as

$$x > -\frac{d_2 - d_1}{2}. \tag{26}$$

Thus, the fuctnion $h(x) = \sigma(-x) - \sigma(-(x+(d_2-d_1)))$ is monotonically decreasing for $x > -\frac{d_2-d_1}{2}$, and here, the minimum value of $x$ is $d_1 - \epsilon$. Therefore, if $d_1 - \epsilon > -\frac{d_2-d_1}{2}$, which can be rewritten as $\epsilon < \frac{d_1+d_2}{2}$, it satisfies

$$\mathcal{D}_{\text{std}}(\boldsymbol{x}_1, \boldsymbol{x}_2) < \mathcal{D}_{\text{adv}}(\boldsymbol{x}_1, \boldsymbol{x}_2, \epsilon). \tag{27}$$

■

**Corollary 1** *The increase in gradient difference by an adversarial attack is proportional to the difficulty of the example and the size of the upper bound of adversarial perturbations.*

*Thus, if* $\left\|\boldsymbol{w}^\top \boldsymbol{x}_3\right\| = d_3$ *and* $d_2 < d_3$*, it satisfies*

$$\mathcal{D}_{\text{adv}}(\boldsymbol{x}_1, \boldsymbol{x}_2, \epsilon) < \mathcal{D}_{\text{adv}}(\boldsymbol{x}_1, \boldsymbol{x}_3, \epsilon), \tag{28}$$

*and if* $\epsilon_1 < \epsilon_2$*, it satisfies*

$$\mathcal{D}_{\text{adv}}(\boldsymbol{x}_1, \boldsymbol{x}_2, \epsilon_1) < \mathcal{D}_{\text{adv}}(\boldsymbol{x}_1, \boldsymbol{x}_2, \epsilon_2). \tag{29}$$

**Proof** In the equation 23 of the above proof, it satisfies Equation 28 if the function $h(x) = \sigma(-(d_1 - \epsilon)) - \sigma(-(d_1 - \epsilon + (x - d_1)))$ is monotonically increasing for input $x > d_1$. Then,

$$\frac{d}{dx}\big(\sigma(-(d_1 - \epsilon)) - \sigma(-(d_1 - \epsilon + (x - d_1)))\big) = \frac{d}{dx}\left( - \frac{1}{1 + e^{x - d_1}}\right) \tag{30}$$

$$= \left(\frac{e^{x-d_1}}{(1 + e^{x-d_1})^2}\right) > 0. \tag{31}$$

In the same way, it satisfies Equation 29 if the function $h(x) = \sigma(-(d_1 - \epsilon)) - \sigma(-(d_1 - \epsilon + (d_2 - d_1)))$ is monotonically increasing for input $\epsilon \geq 0$. Let $x$ denote $\epsilon - d_1$ and $a$ denote $d_2 - d_1$. Then, if $\epsilon - d_1 < \frac{d_2 - d_1}{2}$, which can be rewritten as $\epsilon < \frac{d_1 + d_2}{2}$, it satisfies

$$\frac{d}{dx}\big(\sigma(x) - \sigma(x - a)\big)\big) = \frac{d}{dx}\left(\frac{1}{1 + e^{-x}} - \frac{1}{1 + e^{-x+a}}\right) \tag{32}$$

$$= e^{x-a}\left(\frac{(e^a - 1)(1 - e^{2x-a})}{(e^x + 1)^2(e^{x-a} + 1)^2}\right) > 0. \tag{33}$$

■

### A.2 Section 3.2 proofs

Next, we proceed with the proofs of Theorems 6 and 7. Before the proof, we will examine the derivation process of the adversarial training model weight $w_{adv}$ in Equation 8, which represents the weights of models trained with $m$ normal examples and $n$ hard examples. Following Tsipras et al. (2018), we utilize a soft-margin SVM classifier. The optimization problem for the model is as follows.

$$\begin{aligned} &\min_{\boldsymbol{w}} \mathbb{E}\big[\max(0, 1 - y\boldsymbol{w}^\top \boldsymbol{x})\big] \\ &\text{subject to } \ \|\boldsymbol{w}\| = 1 \end{aligned} \tag{34}$$

Given that $\|\boldsymbol{x}\| \leq 1$ and $\beta\eta < \epsilon < \eta$, the weight learning process for the adversarial training model is as follows.

$$\mathbb{E}\big[\max(0, 1 - y\boldsymbol{w}^\top(\boldsymbol{x} + \boldsymbol{\delta}))\big] = \mathbb{E}\big[1 - y\boldsymbol{w}^\top(\boldsymbol{x} + \boldsymbol{\delta})\big] \tag{35}$$

$$= \mathbb{E}\big[1 - w_0\mathcal{N}(\eta - \epsilon, 1) - \sum_{i=1}^{d} w_i\mathcal{N}(\text{sign}(w_i) \cdot (\beta\eta - \epsilon), 1) - w_{d+1}\mathcal{N}(\eta - \epsilon, 1)\big] \tag{36}$$

$$= 1 - \frac{m}{m+n}w_0(\eta - \epsilon) - \sum_{i=1}^{d} w_i\text{sign}(w_i) \cdot (\beta\eta - \epsilon) - \frac{n}{m+n}w_{d+1}(\eta - \epsilon) \tag{37}$$

According to the above equation, in adversarial training, the optimal solution is to assign a value of 0 to the weights corresponding to the non-robust features, which is consistent with what was demonstrated by Tsipras et al. (2018). Therefore, when $w_{i\in 1,\dots,d} = 0$, Equation 34 can be simplified as follows.

$$\begin{aligned} \min_{\boldsymbol{w}} -\frac{m}{m+n} w_0(\eta-\epsilon) - \frac{n}{m+n} w_{d+1}(\eta-\epsilon) \\ \text{subject to } \; w_0^2 + w_{d+1}^2 = 1 \end{aligned} \tag{38}$$

Using the Lagrange multiplier method, we define the above equation, and the partial derivatives with respect to each term are as follows.

$$\mathcal{L}(w_0, w_{d+1}, \mu) = -\frac{m}{m+n} w_0(\eta-\epsilon) - \frac{n}{m+n} w_{d+1}(\eta-\epsilon) + \mu(w_0^2 + w_{d+1}^2 - 1) \tag{39}$$

$$\frac{\partial \mathcal{L}}{\partial w_0} = -\frac{m}{m+n}(\eta-\epsilon) + 2\mu w_0 = 0 \tag{40}$$

$$\frac{\partial \mathcal{L}}{\partial w_{d+1}} = -\frac{n}{m+n}(\eta-\epsilon) + 2\mu w_{d+1} = 0 \tag{41}$$

$$\frac{\partial \mathcal{L}}{\partial \mu} = w_0^2 + w_{d+1}^2 - 1 = 0 \tag{42}$$

Given that $-\frac{m}{m+n}(\eta-\epsilon) = a_0$ and $-\frac{n}{m+n}(\eta-\epsilon) = a_1$, we can rewrite the equations as $a_0 = -2\mu w_0$ and $a_1 = -2\mu w_{d+1}$. Dividing the two equations gives $\frac{a_0}{a_1} = \frac{w_0}{w_{d+1}}$, so $w_{d+1} = \frac{a_1}{a_0} w_0$. Substituting this into $w_0^2 + w_{d+1}^2 - 1 = 0$ gives $w_0^2 = \frac{a_0^2}{a_0^2+a_1^2}$ and $w_{d+1}^2 = \frac{a_1^2}{a_0^2+a_1^2}$. Therefore, to obtain the minimum values, $w_0 = -\frac{a_0}{\sqrt{a_0^2+a_1^2}}$ and $w_{d+1} = -\frac{a_1}{\sqrt{a_0^2+a_1^2}}$. Dividing $a_0$ and $a_1$ by $-(\eta-\epsilon)$ gives $w_0 = \frac{1}{Z_{adv}}\frac{m}{m+n}$ and $w_{d+1} = \frac{1}{Z_{adv}}\frac{n}{m+n}$.

Next, we proceed with the proof of Theorem 2.

**Theorem 2** *For a model $f_{adv}$ adversarially trained on the combined dataset $\mathcal{X}^{train}_{normal} \cup \mathcal{X}^{train}_{hard}$, where $\mathcal{X}^{train}_{normal}$ consists of $m$ normal examples and $\mathcal{X}^{train}_{hard}$ consists of $n$ hard examples, both the standard accuracy $Pr\big[f_{adv}(\boldsymbol{x}) = y\big]$ and the robust accuracy $Pr\big[f_{adv}(\boldsymbol{x}+\boldsymbol{\delta}) = y\big]$ on the normal example test dataset $\mathcal{X}^{test}_{normal}$ are monotonically decreasing with respect to $n$.*

**Proof** First, the standard accuracy $\Pr\big[f_{adv}(\boldsymbol{x}) = y\big]$ of $f_{adv}$ on $\mathcal{X}^{test}_{normal}$ is expressed as follows. Here, the bias $b$ is assumed to be $b \neq 0$, and it is also assumed that the proportions of the two classes in the test data are equal.

$$\Pr\big[f_{adv}(\boldsymbol{x}) = y\big] = \Pr\big[y(\boldsymbol{w}^\top \boldsymbol{x}) + b > 0\big] \tag{43}$$

$$= \Pr\left[\frac{1}{Z_{adv}}\left(\frac{m}{m+n}\cdot\mathcal{N}(\eta,1)\right) + y\cdot b > 0\right] \tag{44}$$

$$= \Pr\left[\frac{m}{\sqrt{m^2+n^2}}\cdot\mathcal{N}(\eta,1) + y\cdot b > 0\right] \tag{45}$$

$$= \Pr\left[\mathcal{N}(0,1) > -y\cdot\frac{b\sqrt{m^2+n^2}}{m} - \eta\right] \tag{46}$$

$$= \frac{1}{2}\Pr\left[\mathcal{N}(0,1) > -\frac{b\sqrt{m^2+n^2}}{m} - \eta\right] + \frac{1}{2}\Pr\left[\mathcal{N}(0,1) > \frac{b\sqrt{m^2+n^2}}{m} - \eta\right] \tag{47}$$

Let the CDF $\Phi(\cdot)$ represent the cumulative distribution function, and the PDF $\phi(\cdot)$ represent the probability density function. Additionally, for readability, we substitute $g(n) = \frac{b\sqrt{m^2+n^2}}{m}$. Then,

$$\Pr\big[f_{adv}(\boldsymbol{x}) = y\big] \overset{\Phi(\cdot)\text{ is CDF}}{=} \frac{1}{2}\Big(2 - \Phi(-g(n)-\eta) - \Phi(g(n)-\eta)\Big) \tag{48}$$

$$= \frac{1}{2}\Big(1 + \Phi(g(n)+\eta) - \Phi(g(n)-\eta)\Big) \tag{49}$$

$$\underset{\text{derivative w.r.t. } n}{\overset{\phi(\cdot)\text{ is PDF}}{\longrightarrow}} \frac{1}{2}\Big(\phi(g(n)+\eta)\cdot g'(n) - \phi(g(n)-\eta)\cdot g'(n)\Big) \tag{50}$$

$$= \frac{g'(n)}{2\sqrt{2\pi}}\left(\exp\left(-\frac{(g(n)+\eta)^2}{2}\right) - \exp\left(-\frac{(g(n)-\eta)^2}{2}\right)\right) \leq 0 \tag{51}$$

When $b < 0$, $g'(n) \leq 0$ and $-\big(g(n)+\eta\big)^2 > -\big(g(n)-\eta\big)^2$, so the above result is negative. Similarly, when $b > 0$, $g'(n) \geq 0$ and $-\big(g(n)+\eta\big)^2 < -\big(g(n)-\eta\big)^2$, so the above result is also negative. Consequently, with respect to the size $n$ of the hard example training dataset, the standard accuracy $\Pr\big[f_{adv}(\boldsymbol{x}) = y\big]$ of $f_{adv}$ on the normal example test dataset $\mathcal{X}^{test}_{normal}$ is monotonically decreasing as $n$ increases.

For robust accuracy $\Pr\big[f_{adv}(\boldsymbol{x}+\boldsymbol{\delta}) = y\big]$, since $\eta - \epsilon > 0$, replacing $\eta$ with $\eta - \epsilon$ in the above proof regarding standard accuracy does not change the outcome of the proof. Therefore, similar to standard accuracy, the robust accuracy $\Pr\big[f_{adv}(\boldsymbol{x}+\boldsymbol{\delta}) = y\big]$ is also monotonically decreasing with respect to $n$. ■

**Theorem 3** *For a model $f_{adv}$ adversarially trained on the combined dataset $\mathcal{X}^{train}_{normal} \cup \mathcal{X}^{train}_{hard}$, where $\mathcal{X}^{train}_{normal}$ consists of $m$ normal examples and $\mathcal{X}^{train}_{hard}$ consists of $n$ hard examples, the standard accuracy $Pr\big[f_{adv}(\boldsymbol{x}) = y\big]$ on the hard example test dataset $\mathcal{X}^{test}_{hard}$ is monotonically increasing with respect to $n$, but the robust accuracy $Pr\big[f_{adv}(\boldsymbol{x}+\boldsymbol{\delta}) = y\big]$ is monotonically decreasing with respect to $n$.*

**Proof**

First, the standard accuracy $\Pr\big[f_{adv}(\boldsymbol{x}) = y\big]$ of $f_{adv}$ on $\mathcal{X}_{hard}^{test}$ is expressed as follows. As before, it is assumed that the bias $b$ is $b \neq 0$, and that the two classes in the test data are equally represented.

$$\Pr\big[f_{adv}(\boldsymbol{x}) = y\big] = \Pr\big[y(\boldsymbol{w}^\top \boldsymbol{x}) + b > 0\big] \tag{52}$$

$$= \Pr\left[\frac{1}{Z_{adv}}\left(\frac{n}{m+n} \cdot \mathcal{N}(\beta\eta, 1)\right) + y \cdot b > 0\right] \tag{53}$$

$$= \Pr\left[\frac{n}{\sqrt{m^2+n^2}} \cdot \mathcal{N}(\beta\eta, 1) + y \cdot b > 0\right] \tag{54}$$

$$= \Pr\left[\mathcal{N}(0,1) > -y \cdot \frac{b\sqrt{m^2+n^2}}{n} - \beta\eta\right] \tag{55}$$

$$= \frac{1}{2}\Pr\left[\mathcal{N}(0,1) > -\frac{b\sqrt{m^2+n^2}}{n} - \beta\eta\right] + \frac{1}{2}\Pr\left[\mathcal{N}(0,1) > \frac{b\sqrt{m^2+n^2}}{n} - \beta\eta\right] \tag{56}$$

As with the above, let the CDF $\Phi(\cdot)$ represent the cumulative distribution function, and the PDF $\phi(\cdot)$ represent the probability density function. Additionally, we substitute $g(n) = \frac{b\sqrt{m^2+n^2}}{n}$. Then,

$$\Pr\big[f_{adv}(\boldsymbol{x}) = y\big] \overset{\Phi(\cdot) \text{ is CDF}}{=} \frac{1}{2}\Big(2 - \Phi(-g(n) - \beta\eta) - \Phi(g(n) - \beta\eta)\Big) \tag{57}$$

$$= \frac{1}{2}\Big(1 + \Phi(g(n) + \beta\eta) - \Phi(g(n) - \beta\eta)\Big) \tag{58}$$

$$\underset{\text{derivative w.r.t. } n}{\overset{\phi(\cdot) \text{ is PDF}}{\rightarrow}} \frac{1}{2}\Big(\phi(g(n) + \beta\eta) \cdot g'(n) - \phi(g(n) - \beta\eta) \cdot g'(n)\Big) \tag{59}$$

$$= \frac{g'(n)}{2\sqrt{2\pi}}\left(\exp\left(-\frac{(g(n)+\beta\eta)^2}{2}\right) - \exp\left(-\frac{(g(n)-\beta\eta)^2}{2}\right)\right) \geq 0 \tag{60}$$

$$\tag{61}$$

The CDF $\Phi(\cdot)$ represents the cumulative distribution function, and the PDF $\phi(\cdot)$ represents the probability density function. For readability, we substitute $g(n) = \frac{b\sqrt{m^2+n^2}}{n}$. Since the sign of $g'(n)$ in this result is opposite to that in the proof of Theorem 2, the outcome is also opposite. When $b < 0$, $g'(n) \geq 0$ and $-\big(g(n)+\eta\big)^2 > -\big(g(n)-\eta\big)^2$, so the result is positive. Similarly, when $b > 0$, $g'(n) \leq 0$ and $-\big(g(n)+\eta\big)^2 < -\big(g(n)-\eta\big)^2$, so the result is also positive. Consequently, with respect to the size $n$ of the hard example training dataset, the standard accuracy $\Pr\big[f_{adv}(\boldsymbol{x}) = y\big]$ of $f_{adv}$ on the hard example test dataset $\mathcal{X}_{hard}^{test}$ is monotonically increasing with $n$.

For robust accuracy $\Pr\big[f_{adv}(\boldsymbol{x}+\boldsymbol{\delta}) = y\big]$, since $\eta - \epsilon < 0$, replacing $\eta$ with $\eta - \epsilon$ in the proof for standard accuracy results in $-\big(g(n)+\eta\big)^2$ and $\big(g(n)-\eta\big)^2$ being replaced by $-\big(g(n)+\eta-\epsilon\big)^2$ and $\big(g(n)-\eta+\epsilon\big)^2$, respectively, which reverses the order of the two terms. Consequently, the result is the opposite of that for standard accuracy, and the robust accuracy $\Pr\big[f_{adv}(\boldsymbol{x}+\boldsymbol{\delta}) = y\big]$ is monotonically decreasing with $n$.

■

### A.3 Section 3.3 proofs

Next, by comparing the relative increases and decreases in standard accuracy in Theorem 2 and Theorem 3, we can derive the following corollary.

**Corollary 2** *For a model $f_{adv}$ adversarially trained on the combined dataset $\mathcal{X}_{normal}^{train} \cup \mathcal{X}_{hard}^{train}$, where $\mathcal{X}_{normal}^{train}$ consists of $m$ normal examples and $\mathcal{X}_{hard}^{train}$ consists of $n$ hard examples, the standard accuracy $Pr\big[f_{adv}(\boldsymbol{x}) = y\big]$ on the overall test dataset $\mathcal{X}_{normal}^{test} \cup \mathcal{X}_{hard}^{est}$ is monotonically increasing with respect to $n$, if $n^3 \leq \beta m^3$.*

**Proof**

To compare the expressions for the increase and decrease in standard accuracy derived from the proofs of the two theorems above, we approximate each value using a first-order Taylor expansion. The results are as follows.

$$\begin{aligned}
&\frac{g'(n)}{2\sqrt{2\pi}}\left(\exp\left(-\frac{(g(n)+\eta)^2}{2}\right)-\exp\left(-\frac{(g(n)-\eta)^2}{2}\right)\right)=\frac{g'(n)}{2\sqrt{2\pi}}(-2g(n)\eta), \quad \text{(normal)}\\
&\frac{g'(n)}{2\sqrt{2\pi}}\left(\exp\left(-\frac{(g(n)+\beta\eta)^2}{2}\right)-\exp\left(-\frac{(g(n)-\beta\eta)^2}{2}\right)\right)=\frac{g'(n)}{2\sqrt{2\pi}}(-2g(n)\beta\eta), \text{ (hard)}
\end{aligned} \tag{62}$$

Here, we examine the sign of the sum of the two terms above. It is assumed that the ratio of normal examples to hard examples in the test phase is $m : n$. Let $g(n)$ for hard examples be denoted as $g_{hard}(n)$, which can be expressed as follows.

$$m \cdot \frac{g'(n)}{2\sqrt{2\pi}}(-2g(n)\eta) + n \cdot \frac{g'_{hard}(n)}{2\sqrt{2\pi}}(-2g_{hard}(n)\beta\eta) = \frac{1}{2\sqrt{2\pi}}\left(-\frac{2nb^2\eta}{m}+\frac{2m^2b^2\beta\eta}{n^2}\right) \tag{63}$$

$$= \frac{b^2\eta}{\sqrt{2\pi}mn^2}\big(-n^3+\beta m^3\big) \tag{64}$$

Therefore, if $n^3 \leq \beta m^3$, increasing the number of hard examples $n$ in the training dataset will lead to an increase in standard accuracy, whereas in the opposite case, it will decrease. Since datasets collected under normal circumstances usually have a higher proportion of normal examples compared to hard examples, it can generally be assumed that the condition $n^3 \leq \beta m^3$ is satisfied. Consequently, increasing the number of hard examples $n$ in the training dataset will typically result in an increase in standard accuracy.

■

### A.4 Section 4.2 proofs

We proceed with the proof of the following theory discussed in Section 4.1.

**Theorem 4** *Let the robust feature coefficients of each data point in the dataset $\mathcal{X}^{train}_{difficulty}$, consisting of k data points $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_k$, be denoted as $\alpha_1, \alpha_2, \ldots, \alpha_k$. Then, when performing adversarial training using $\mathcal{X}^{train}_{difficulty}$, the optimal LS factors $\lambda_1, \lambda_2, \ldots, \lambda_k$ for each training example are proportional to their respective robust feature coefficients $\alpha_1, \alpha_2, \ldots, \alpha_k$.*

**Proof**

First, the weight of the adversarially trained model using the data given in Theorem 4 is derived through the same process as the weight $\boldsymbol{w}_a dv$ of the above $f_{adv}$, and is given as follows.

$$\boldsymbol{w} = \frac{1}{Z}\begin{cases} w_0 = \sum_{i=1}^{k} -(2\lambda_i - 1)(\alpha_i \eta - \epsilon) \\ w_{i \in 1,\ldots,k} = 0 \\ w_{d+1} = \sum_{i=1}^{k} -(2\lambda_i - 1)(\gamma_i \eta - \epsilon) \end{cases} \tag{65}$$

Here, $Z$ is a normalization term tuned to satisfy $\|\boldsymbol{w}\| = 1$. To ensure that the weight parameter $\boldsymbol{w}$ is close to $\boldsymbol{w}^* = (1, 0, ..., 0)$, the LS factor is assigned in such a way that $w_0$ increases and $w_{d+1}$ decreases. Therefore, the optimization equation to be solved can be expressed as follows.

$$\begin{aligned} &\min_{\boldsymbol{\lambda}} w_{d+1} - w_0 \\ &\text{subject to } \ 0.5 \leq \lambda_i \leq 1, \ i = 1, ..., k \end{aligned} \tag{66}$$

Here, $\boldsymbol{\lambda} = (\lambda_0, ..., \lambda_k)$. The optimization objective can be expressed in the form of a vector operation with respect to $\boldsymbol{\lambda}$ as follows.

$$w_{d+1} - w_0 = 2\eta \boldsymbol{\lambda}^\top (\boldsymbol{\alpha} - \boldsymbol{\gamma}) - \eta (\boldsymbol{\alpha} - \boldsymbol{\gamma})^\top \mathbf{1} \tag{67}$$

Therefore, the optimization objective is minimized when $\lambda$ is proportional to $(\alpha - \gamma)$. Here, let $c_1, c_2 > 0$ be constants. Since $\gamma$ is inversely proportional to $\alpha$, it can be expressed as $\gamma = -c_1 \alpha$. Consequently, $\lambda$ satisfies $\lambda = \min(0.5, \max(1, c_2(1 + c_1)\alpha))$. Therefore, the optimal LS factor can ultimately be said to be proportional to the robust feature coefficient. ■

## Appendix B. Additional implementation and experimental details

### B.1 Implementation details

**Datasets** CIFAR-10 (Krizhevsky et al., 2009) consists of 50,000 training images and 10,000 test images with 10 classes. CIFAR-100 (Krizhevsky et al., 2009) consists of 50,000 training

images and 10,000 test images with 100 classes. CIFAR-10 and CIFAR-100 images have sizes of $32 \times 32$ pixels. The CIFAR datasets are subsets of the 80 million tiny images dataset (Torralba et al., 2008), and the 80 million tiny images dataset contains images downloaded from seven independent image search engines: Altavista, Ask, Flickr, Cydral, Google, Picsearch, and Webshots. SVHN (Netzer et al., 2011) consists of 73,257 training images and 26,032 test images with 10 classes. SVHN images have sizes of $32 \times 32$ pixels. SVHN is obtained from a very large set of images from urban areas in various countries by using Google Street View. STL-10 (Coates et al., 2011) consists of 5,000 training images and 8,000 test images with 10 classes. Additionally, it provides 100,000 unlabeled images to support unsupervised learning. STL-10 images have sizes of $96 \times 96$ pixels, which is larger than the typical $32 \times 32$ images found in datasets like CIFAR-10 (Krizhevsky et al., 2009). In our experiments, we resized the STL-10 dataset images to $64 \times 64$ pixels for model training. Further details can be found in `https://paperswithcode.com/datasets`.

**Implementation details** We conducted experiments on CIFAR-10, CIFAR-100 (Krizhevsky et al., 2009), SVHN (Netzer et al., 2011), and STL-10 (Coates et al., 2011). We used PGD (Madry et al., 2017) and TRADES (Zhang et al., 2019) as the baseline adversarial training algorithms. We used WideResNet28-10 (Zagoruyko and Komodakis, 2016) as the architecture of our models. The learning rates for CIFAR-10, CIFAR-100, and STL-10 are set to 0.1 and 0.01 for SVHN, and the decay at 100 and 105 of the total training epoch 110 with decay factor 0.1 following Pang et al. (2021). We additionally conducted experiments with the learning rate decay at 50% and 75% of the total training epoch 200 with decay factor 0.1 following Madry et al. (2017). We used stochastic gradient descent optimizer with the weight decay factor 5e-4 and the momentum 0.9. The upper bounds of adversarial perturbation were set to 0.031 ($\epsilon = 8$), 0.0155 ($\epsilon = 4$), and 0.00775 ($\epsilon = 2$), and the step-size of training adversarial examples of each model were set to one fourth of the $\ell_\infty$-bound of each model with 10 steps. To evaluate robustness of the models, we used the 10-step and 20-step PGD attacks, and adaptive auto attack ($A^3$) (Liu et al., 2022b) for reliable evaluation. We used a single RTX 8000 GPU with CUDA11.6 and CuDNN7.6.5 in our experiments.

### B.2 Experiment details and futher experiments on subset training

**Figure 1** For the experiments in Fig. 1, we executed 200 training epochs on the CIFAR-10 dataset in both standard and adversarial training. We constituted datasets for both standard and adversarial (Madry et al., 2017) training by selecting 5k hard and 5k random examples from the training set, where the difficulty of data is measured by accumulating 0-1 loss along the training trajectory of the models trained with standard and adversarial training until epoch 90. Considering that the size of the dataset is small and to prevent overfitting before the learning rate decay in both standard and adversarial training, we used the ResNet-18 (He et al., 2016) architecture. The $\ell_\infty$-bound and step-size of adversarial examples were set to 0.0155, and 0.0035, respectively, with 5 steps. The adversarial training time on the 5k+5k subset of the CIFAR-10 dataset is 2 hours for 200 epochs. The adversarial training time on the 5k subset (hard excluded) of the CIFAR-10 dataset is 0.5 hours for 100 epochs (after the learning rate decay epoch 100).

For further experiments in adversarial training, we constituted a new dataset as in Fig. 1, with only using top 10k hard and 10k random examples (total 20k). We trained the PGD

model until learning rate decay, then progress training for the models with the dataset: (1) entire examples, (2) 10k random examples (hard pruning), (3) top 10k hard examples (hard only). We used 10k instead of 5k to prevent overfitting before pruning. In Fig. 10, the models trained with the dataset that includes hard examples show decrease in test robust accuracy; however, the model trained with the dataset that hard examples are pruned shows increases in test robust accuracy. Thus, pruning hard examples improves robustness, indicating that hard examples degrades the performance in adversarial training. The implementation details for Fig. 10 are same with those of Fig. 3. The adversarial training time on the 10k+10k subset of the CIFAR-10 dataset is 20 hours for 200 epochs. The adversarial training time on the 10k subset (hard excluded) of the CIFAR-10 dataset is 5 hours for 100 epochs (after learning rate decay epoch 100).

### B.3 Experiment details for comparing el2n score

**Figure 3** For the experiments in Fig. 3, we executed 200 training epochs on the CIFAR-10 dataset in each training. The other experimental details are summarized in Appendix B.1. The adversarial training time on the entire CIFAR-10 dataset is 47 hours for 200 epochs. The full results of Fig. 3 can be seen in Fig. 11. For the results of the PGD model before the learning rate decay, because hard examples are difficult to learn also in the perspective of standard accuracy with high learning rate, the result shows large EL2N score for hard clean examples. After the learning rate decay, where the model starts to fit hard examples, the difficulty difference between clean and adversarial examples of hard examples starts to increase.

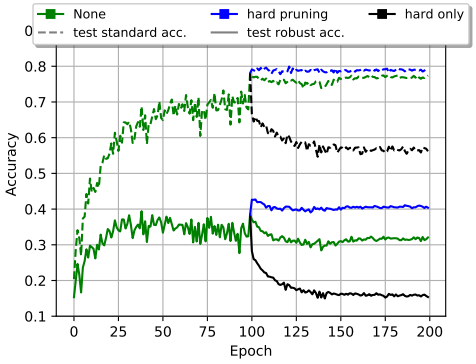



Figure 10: Test accuracy of PGD models trained with subsets of CIFAR-10.

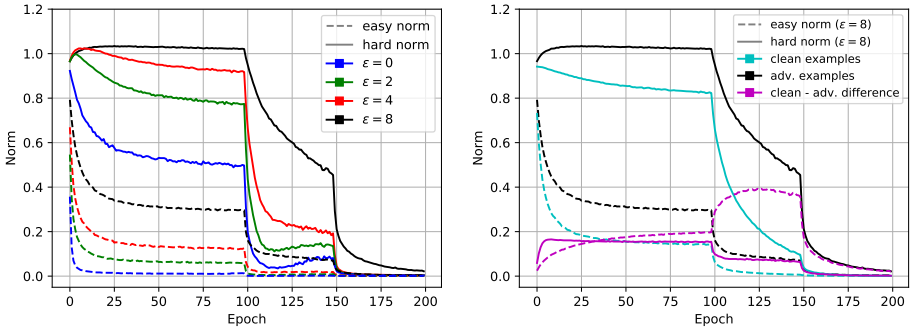
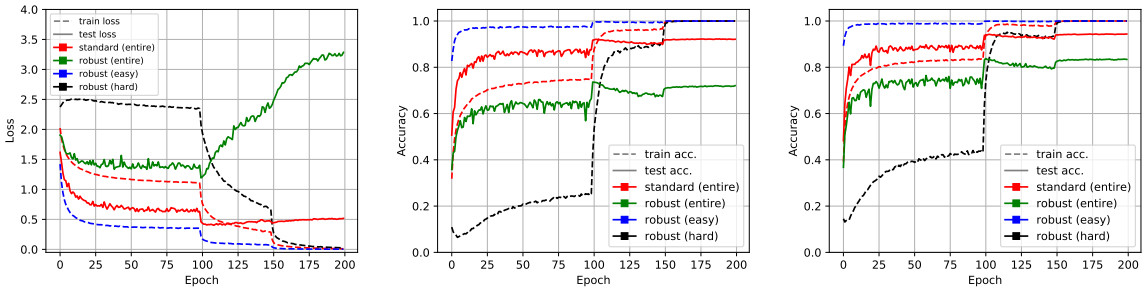



Figure 11: Average error vector norm (EL2N score) of hard and easy examples.

### B.4 Experiment details and further experiments on other analyses

**Figure 4** We executed 200 training epochs on the CIFAR-10 dataset in each training. The other experimental details are summarized in Appendix B.1. The hard and easy examples are selected according to the difficulty of each model. The training and test accuracy results

and loss results for the PGD model with $\epsilon = 4$ and $\epsilon = 2$ can be seen from Fig. 12. The robust overfitting phenomenon is observed to be severe as the adversarial budget of the training setting increases.



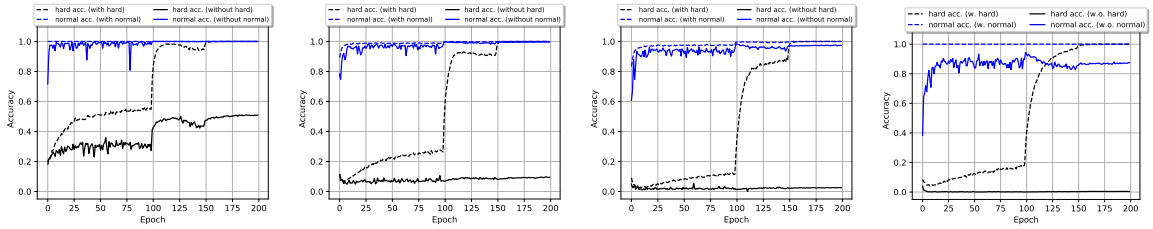

(a) (b) (c)

Figure 12: (a) presents the training and test loss curves in the PGD model. (b) shows the training and test accuracy in the PGD models with $\epsilon = 4$. (c) shows the training and test accuracy in the PGD models with $\epsilon = 2$.

**Figure 5** We executed 200 training epochs on the CIFAR-10 dataset in each training. The other experimental details are summarized in Appendix B.1. We select top 5k easy and top 5k hard examples. We then trained the models with the dataset that prunes top 5k easy examples and that prunes top 5k hard examples, respectively. The memorization results including experiments on adversarial training with $\epsilon = 4$, $\epsilon = 2$, and $\epsilon = 0$ (STD) are described in Fig. 13.

These results illustrate the necessity of a memorization feature. To explain this, let us consider the possibility that the results in Fig. 5 could be explained solely by the presence of robust and non-robust features, without the need for a memorization feature. First, if we assume that hard examples possess robust features, then it should be possible to predict the adversarial examples of hard examples through the training of the remaining 45k data points of CIFAR-10 in Fig. 5, even if these hard examples are not included in the adversarial training. However, since the model trained without the hard examples shows 0% accuracy on them, this assumption contradicts the idea that hard examples contain robust features. Second, let us assume that hard examples are composed solely of non-robust features. In this case, the train accuracy for these examples should be 0%. However, Fig. 5 shows 100% train accuracy, which contradicts this assumption. Therefore, the assumption that the results can be explained solely by the presence of robust and non-robust features without a memorization feature is difficult to support. To explain the results in Fig. 5, the existence of a feature that "acts as a robust feature during training but as a non-robust feature during testing"—in other words, a memorization feature—is necessary.

**Table 2** For the experiments in Table 2, we executed 110 training epochs on the CIFAR-10 dataset in each model. The other experimental details are summarized in Appendix B.1. Table 7 shows the performance of the models trained with top 10k easy and top 10k hard examples as Table 2 with varying the adversarial budget $\epsilon$. The robust accuracy is evaluated using $A^3$ (Liu et al., 2022b). As the adversarial budget decreases, the standard accuracy




(a) $\epsilon = 0$ (b) $\epsilon = 2$ (c) $\epsilon = 4$ (d) $\epsilon = 8$

Figure 13: Memorization of easy and hard examples in each setting. Clean example results for a standard setting and adversarial example results for adversarial settings.

of the models that prunes hard examples increases, which suggests that the number of anti-correlated features which were slightly correlated features decreases.

Table 7: Performance of models trained with 10k examples.

| Method | STD | | PGD ($\epsilon = 2$) | | PGD ($\epsilon = 4$) | | PGD ($\epsilon = 8$) | |
|---|---|---|---|---|---|---|---|---|
| | Std. | Rob. | Std. | Rob. | Std. | Rob. | Std. | Rob. |
| Easy | 78.46 | - | 75.66 | 62.00 | 69.65 | 51.51 | 61.63 | 39.56 |
| Hard | 77.60 | - | 45.47 | 22.54 | 32.17 | 9.04 | 19.92 | 8.68 |

**Figure 6** We assigned random labels to the hard examples for the corruption model from the beginning of the training, and we pruned the hard examples from the beginning of the training for the pruning model. The robust accuracy is evaluated using $A^3$ (Liu et al., 2022b). Table 8 shows the final accuracy for the models trained with applying random labels to the top 5k hard examples. We compared the results with the performance of the normal models and the models that prunes top 5k hard examples for PGD and TRADES. Observably, assigning random labels to hard examples does not change the robustness performance.

Table 8: Performance comparison of the final accuracy for models with label corruption.

| Dataset | PGD | | | TRADES | | |
|---|---|---|---|---|---|---|
| | Std. | Robust (PGD) | Robust ($A^3$) | Std. | Robust (PGD) | Robust ($A^3$) |
| Normal | **86.82** | 50.25 | 46.07 | **84.83** | 51.43 | 46.69 |
| Corrupted | 82.31 | 49.32 | 45.67 | 82.11 | 50.64 | 46.15 |
| Pruned | 85.99 | **52.36** | **48.81** | 84.71 | **53.44** | **48.78** |

**Distinguishing hard examples** In Section 3.2, we assumed properties of hard examples and conducted a theoretical analysis on the impact of learning hard examples on model performance. Based on this theoretical analysis, we can distinguish hard examples using the following metrics:

1. $\Pr_{f\sim A(D_{adv})}[\arg\max f(x_i) = y_i] - \Pr_{f\sim A(D_{adv}\setminus\{i\})}[\arg\max f(x_i) = y_i]$
2. $\mathbb{E}_{(x,y)\sim D_{std}}\left[Pr\left(f_{std}(x; A(D_{std})) = y\right) - Pr\left(f_{std}(x; A(D_{std}\setminus\{i\})) = y\right)\right]$
   $-\mathbb{E}_{(x,y)\sim D_{adv}}\left[Pr\left(f_{adv}(x; A(D_{adv})) = y\right) - Pr\left(f_{adv}(x; A(D_{adv}\setminus\{i\})) = y\right)\right]$
3. $\mathbb{E}_{(x,y)\sim D_{std}}\left[Pr\left(f_{adv}(x; A(D_{adv})) = y\right) - Pr\left(f_{adv}(x; A(D_{adv}\setminus\{i\})) = y\right)\right]$
   $-\mathbb{E}_{(x,y)\sim D_{adv}}\left[Pr\left(f_{adv}(x; A(D_{adv})) = y\right) - Pr\left(f_{adv}(x; A(D_{adv}\setminus\{i\})) = y\right)\right]$

In all the above metrics, a higher value indicates a stronger representation of the properties presented in Assumption 1. The first metric is the same as the memorization score (Feldman, 2020), which reflects the difference in performance for a data point $(x_i, y_i)$ when it is included in training versus when it is not. The second metric compares the difference for the data point $(x_i, y_i)$ between standard training (with and without inclusion in training) and adversarial training (with and without inclusion). The third metric measures the trade-off $(x_i, y_i)$ by comparing the standard accuracy and adversarial accuracy in adversarial training when the data point is included versus when it is excluded from training.

While the metrics above can identify data with high values as hard examples, it is challenging to train every model with and without each data point for comparison. Therefore, as in Section 3.1, we conducted analyses in Section 3.3 using hard examples identified through 0-1 loss to check whether data that is difficult to learn actually exhibits the properties assumed for hard examples in Assumption 1.

**Figure 7** For the experiments in Fig. 7, We executed 110 training epochs on the CIFAR-10 dataset in each model. The other experimental details are summarized in Appendix B.1. We trained the baseline models until epoch 90, which is the 10 epoch before the learning rate decay for the stability of the training, then we continued by the training using the dataset with the subset pruned as the experiments in Fig. 1. We evaluated the performance of the best checkpoint in each model using $A^3$ (Liu et al., 2022b).

### B.5 Experiment details and ablation studies of DPLS

**Table 3** We executed 110 training epochs on the CIFAR-10 dataset in adversarial training. The other experimental details are summarized in Appendix B.1. We mainly use 0-1 loss as difficulty measurement loss for DPLS. We set the label smoothing factor of the most difficult example in DPLS considering the average difficulty score of each dataset and algorithm. We tuned the factor such that the average label smoothing factor is between 0.8 and 0.9 in 10-class datasets (CIFAR-10, SVHN, and STL-10) and is between 0.7 and 0.9 in a 100-class dataset (CIFAR-100). We selected difficulty calculation epoch as $T = 90$, which is 10 epochs before the first learning rate decay. The label smoothing factor hyperparameter $\lambda$ of the most difficult example for the DPLS models and the label smoothing factor for LS models in Table 3 of the main paper is summarized in Table 9. The PGD model in SVHN with label smoothing methods shows a result similar to gradient masking effect (Athalye et al., 2018). We applied DPLS only to the labels of outer maximization loss to reduce the effect. However, it is considered as the properties of the SVHN dataset with PGD and label smoothing because the experiments on other datasets and algorithms do not show gradient masking. Furthermore, the results of our method still show improvement also against $A^3$ for SVHN by 6.81%p, which could ignore gradient masking and evaluate true robustness. It means that our method increases the true robustness of the model also for SVHN.

Table 9: The smoothing factor $\lambda$ and the average of label smoothing factor for each model in Table 3 in the main paper.

| Dataset | Model | Method | $\lambda$ (smoothing factor) | Avg. smoothing |
|---|---|---|---|---|
| CIFAR-10 | PGD | LS | 0.900 | 0.900 |
| | | DPLS | 0.500 | 0.897 |
| | TRADES | LS | 0.900 | 0.900 |
| | | DPLS | 0.500 | 0.893 |
| CIFAR-100 | PGD | LS | 0.900 | 0.900 |
| | | DPLS | 0.700 | 0.862 |
| | TRADES | LS | 0.700 | 0.700 |
| | | DPLS | 0.300 | 0.703 |
| SVHN | PGD | LS | 0.900 | 0.900 |
| | | DPLS | 0.100 | 0.940 |
| | TRADES | LS | 0.900 | 0.900 |
| | | DPLS | 0.100 | 0.927 |
| STL-10 | PGD | LS | 0.800 | 0.800 |
| | | DPLS | 0.500 | 0.788 |
| | TRADES | LS | 0.800 | 0.800 |
| | | DPLS | 0.500 | 0.797 |

**Figure 9 and Table 4** For the experiments of previous robust overfitting mitigation methods, we used the code uploaded in the official Github by the author of each method. The initial learning rate was set to 0.1, and the learning rate decay was applied at the learning rate decay scheduling of each method with total training epochs 200 and with a decay factor of 0.1. The learning rate decay was applied at 50 and 150 epochs for the method of knowledge distillation with stochastic weight averaging (KD+SWA) (Chen et al., 2020) and the learning rate decay was applied at 100 and 150 epochs for the other methods (the baseline model, our method (DPLS), self-adpative training (SAT) (Huang et al., 2020), and temporal ensemble (TE) (Dong et al., 2021)). We executed 200 training epochs on the CIFAR-10 dataset in adversarial training. The other experimental details are summarized in Appendix B.1.

Table 10: Performance of DPLS according to various difficulty measurement losses.

| Method | PGD | | | TRADES | | |
|---|---|---|---|---|---|---|
| | Std. | Robust (PGD) | Robust ($A^3$) | Std. | Robust (PGD) | Robust ($A^3$) |
| - | 87.19 | 56.44 | 51.8 | 85.66 | 58.46 | 54.08 |
| C-score | **88.05** | 57.58 | 52.9 | 86.6 | 59.0 | 54.71 |
| CE loss | 87.51 | 57.47 | 52.73 | 86.02 | 58.64 | 54.45 |
| EL2N | 87.27 | 58.89 | 53.38 | **86.94** | 58.94 | 54.32 |
| 0-1 loss | 87.13 | **58.43** | **54.03** | 85.36 | **59.33** | **55.19** |

**Table 5** For the experiments in Table 5, we executed 110 training epochs on the subsets of CIFAR-10 dataset in each model. The other experimental details are summarized in Appendix B.1. We employed the pre-calculated C-score (Jiang et al., 2021) from a standard model, and the difficulty from the other losses was calculated during training according to the procedure described in Algorithm 1. The detailed results of the performance of DPLS according to various difficulty losses are listed in Table 10.

Table 11: Applying DPLS to MART (Wang et al., 2019). DPLS model denotes the model that calculate the difficulty of training examples for DPLS.

| Method | DPLS model | Standard | Robust ($A^3$) |
|---|---|---|---|
| MART | - | **83.9** | 52.19 |
| | MART | 83.18 | 52.84 |
| | TRADES | 83.81 | **53.2** |

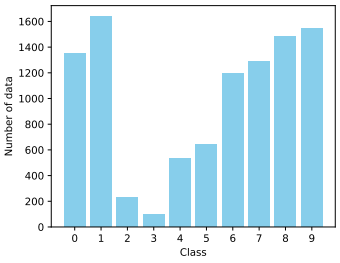
Table 12: Applying DPLS to RST (Carmon et al., 2019). DPLS dataset denotes the dataset to which DPLS is applied.

| Method | DPLS dataset | Standard | Robust ($A^3$) |
|---|---|---|---|
| RST | - | 84.82 | 57.26 |
| | Sup | 84.36 | 57.35 |
| | Unsup | **85.2** | 57.62 |
| | Sup+Unsup | 84.43 | **57.81** |

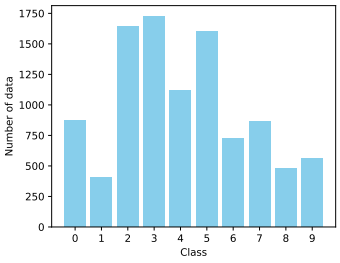
**Table 6** For the experiments of applying DPLS to other adversarial training algorithms, we used the code uploaded in the official Github by the author of each method. We executed 110 training epochs on the CIFAR-10 dataset in adversarial training. The other experimental details are summarized in Appendix B.1. Table 11 and Table 12 show the experimental details of applying DPLS to misclassification aware adversarial training (MART) (Wang et al., 2019) and robust self training (RST) (Carmon et al., 2019). We use the 1,000k extra data generated by a denoising diffusion probabilistic model (DDPM) (Ho et al., 2020) as in Gowal et al. (2021a). For the MART models, we used the difficulty score calculated from the TRADES model because MART uses weighted loss considering difficulty (ground-truth class confidence) of each sample, in which the trained model by MART can be biased and miscalculate the difficulty of each sample. The MART model combined with DPLS where the difficulty is calculated from TRADES shows higher performance than the other models. It can be inferred that calculating accurate difficulty score is important, but applying DPLS with approximate difficulty is also effective at improving robustness. For the RST models, we applied DPLS to both labeled and unlabeled dataset (Sup+Unsup), only the labeled dataset (Sup), and only the unlabeled dataset (Unsup). We calculated difficulty of the unlabeled dataset as the ratio of the number of training to the number of right predictions. For the DPLS (Sup) model, it is inferred that the model is trained to depend on the unlabeled dataset because of regularization only on the labeled data. For the DPLS (Unsup) model, although the difficulty of the unlabeled dataset is not calculated from the full trajectory, the DPLS successfully mitigates the negative effect of hard examples for the unlabeled dataset.

### B.6 Other experiment details

**Additional details of easy and hard examples** Fig. 14 shows the class distributions of 10k easy and 10k hard examples in the training dataset of CIFAR-10 selected by using accumulated 0-1 loss difficulty along the training trajectory of TRADES Zhang et al. (2019). Fig. 15 shows the 0-1 loss distributions of the training datasets for CIFAR-10, CIFAR-100,

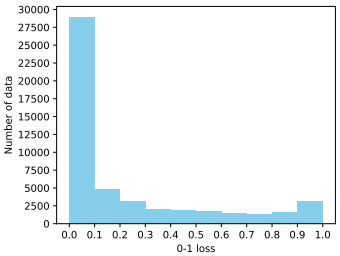



(a) Easy examples

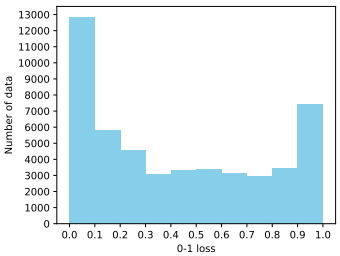



(b) Hard examples

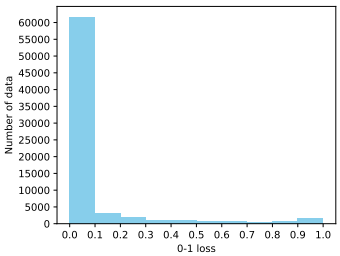
Figure 14: The class distributions of 10k easy and 10k hard examples in CIFAR-10 selected by using accumulated 0-1 loss difficulty.




(a) CIFAR-10




(b) CIFAR-100



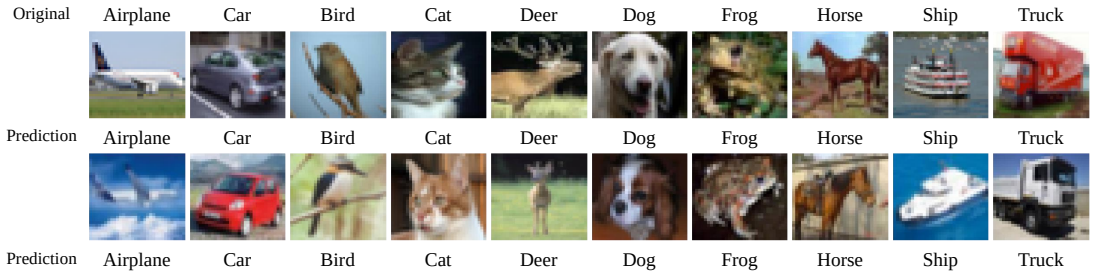

(c) SVHN

Figure 15: The 0-1 loss distributions of the each training dataset: CIFAR-10, CIFAR-100, and SVHN.

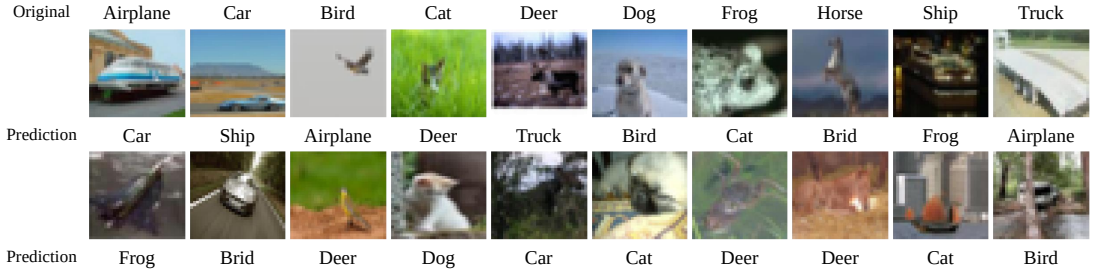
and SVHN, which are calculated with the training trajectory of TRADES. The value of 0-1 loss is normalized by using min-max normalization. Fig. 16 shows easy and hard examples in the training dataset of CIFAR-10.

## Appendix C. Related work

Wong et al. (2020) conducted analysis on overfitting in adversarial training, which is termed as robust overfitting. They applied various data augmentation and regularziation methods to reduce the generalization gap between training and test robust accuracy, concluding that the effect of early stopping can be matched by other methods. Chen et al. (2020) utilized smoothing of both logits and weights in adversarial training. They applied knowledge distillation (Hinton et al., 2015) to logits instead of label smoothing (Szegedy et al., 2016) with robust-trained and standard-trained models as teacher models and used stochastic weight average (Izmailov et al., 2018) as a weight smoothing method by averaging weights in another model along the training trajectory. They demonstrated that the smoothing methods are effective at mitigating robust overfitting. Rebuffi et al. (2021) combined data augmentation methods and weight averaging and significantly improved robust accuracy. They noted that weight averaging is effective when robust accuracy between training iterations is maintained. They experimented on several heuristics-driven augmentation such as cutout (DeVries and Taylor, 2017), mixup (Zhang et al., 2018), and cutmix (Yun et al., 2019). Dong et al.




(a) Easy examples




(b) Hard examples

Figure 16: Easy and hard examples in CIFAR-10 Krizhevsky et al. (2009) selected by using accumulated 0-1 loss difficulty. Original denotes the ground-truth class of images and prediction denotes the most frequently predicted class of images at each epoch of training until the calculation epoch.

(2021) demonstrated that DNNs are sufficient to memorize training adversarial examples with random labels. They showed that the memorization of hard training examples leads to decrease in training loss but increase in test error, noting that the causes of robust overfitting may be the memorization of one-hot labels. Specifically, they showed that training adversarial examples with completely random labels are possible when the training algorithm is TRADES (Zhang et al., 2019). For the PGD (Madry et al., 2017) models, when the label noise ratio exceeds a certain threshold, the generalization gap between train and test accuracy becomes zero and the model exhibits a performance similar to that of the random prediction.

In Liu et al. (2022a), they utilize average training loss as difficulty measurement and analyze overfitting mitigation methods in adversarial training. However, instead of using difficulty score to improve the performance of adversarial training, they proposed Adpative Target and Reweight by using moving average predictions as in Huang et al. (2020); Dong et al. (2021) and ground truth prediction as in Wang et al. (2019). Because the performance for standard adversarial training is not provided in the paper, but the performance only for fast adversarial training and fine-tuning with extra data is provided, we trained and evaluated their models for the comparison. The results of Table 13 shows that Reweight is not applicable to the standard adversarial training and Adaptive Target shows the same performance as SAT Huang et al. (2020) because it is simply a linear combination of SAT and one-hot label. In Liu et al. (2022a), they constituted a subset dataset in a balanced form between classes. However, the difficulty of classes can be imbalanced, and selecting the

Table 13: Performance comparison with the methods in Liu et al. (2021). The models trained using the code in Liu et al. (2021) are denoted as (*).

| Method | Std. | Robust (PGD) | Robust ($A^3$) |
|---|---|---|---|
| PGD | **87.19** | 56.44 | 51.8 |
| +DPLS | 86.17 | **58.14** | **53.98** |
| PGD (*) | 86.75 | 56.24 | 51.81 |
| +Reweight (*) | 25.91 | 22.16 | 21.93 |
| +Adaptive Target (*) | 86.04 | 58.21 | 52.87 |
| +Reweight+Adaptive Target (*) | 30.49 | 25.81 | 25.6 |

most difficult data considering the balance between classes can result in a deviation from correct observations. As noted in Xu et al. (2021), adversarial training differently increases the robustness of each class in a dataset, so the performance imbalance between classes becomes more significant in adversarial training. Thus, it can be negligible in standard training to balance the number of each class in training example subsets, but it can be non-trivial in adversarial training. The distributions of hard and easy example subsets in our paper are plotted in Fig 14. The distributions of easy and hard example subsets are highly class-imbalanced. The comparison study on the pruning balanced or imbalanced training subsets with different difficulty losses can be seen in Table 19, where the models that prune imbalanced subsets consistently show higher performance than the models that prune balanced subsets. Furthermore, we showed that our method does not only exploit the performance increase of the easy classes, but increases the both performance of easy and hard classes as shown in Table 20.

Table 14: Performance of DPLS according to the DPLS calculation epoch $T$.

| $T$ | PGD | | TRADES | |
|---|---|---|---|---|
| | Std. | Rob. | Std. | Rob. |
| 10 | 84.13 | 53.34 | 84.88 | 54.69 |
| 50 | 86.33 | **54.05** | **85.56** | 55.0 |
| 90 | **87.13** | 54.03 | 85.36 | **55.19** |
| 100 | 86.94 | 53.8 | 85.53 | 54.77 |

Table 15: Performance of DPLS according to the re-initialized epoch, where DPLS is pre-calculated by using DPLS of the calculation epoch 90.

| $T$ | Reinit. | PGD | | TRADES | |
|---|---|---|---|---|---|
| | | Std. | Rob. | Std. | Rob. |
| 90 | 0 | 86.62 | 53.69 | 84.99 | 55.05 |
| | 50 | 86.68 | **54.09** | 84.89 | 55.11 |
| | 90 | **87.13** | 54.03 | **85.36** | **55.19** |
| | 100 | 86.76 | 53.76 | 85.26 | 54.93 |

## Appendix D. Additional ablation study

**Ablation on the DPLS calculation epoch** In Table 14, we conducted experiments varying the calculation epoch of DPLS. The results show that applying DPLS at any epoch

is effective at increasing robustness. However, the results show that the improvement is slightly higher when using the difficulty calculated until higher epoch. Additionally, when DPLS is applied after the learning rate decay, the result shows that the increase in robustness is diminished, but it still shows better performance than the baseline model. Thus, the application of DPLS before memorization of hard examples can maximize the mitigation effect of the method, which indicates the overfitting prevention effect of DPLS.

**Ablation on the re-initialization of training by using DPLS** In Table 15, we conducted ablation study varying the training re-initialization epoch with pre-calculated DPLS by using the difficulty of that calculated until the training epoch 90. There are small differences between the results. Comparing the result in Table 14 with the result in Table 15, it indicates that the difficulty score is important for DPLS and the applying checkpoint is less important after fitting training examples and before memorization of hard examples. However, all models still show the improvement in the robustness performance of the model as in the above DPLS calculation epoch ablation study.

Table 16: Performance of DPLS applying to the PGD model on CIFAR-10 according to the smoothing factor $\lambda$.

| Model | $\lambda$ (smoothing factor) | Avg. smoothing | Standard | Robust (PGD) | Robust ($A^3$) |
|---|---|---|---|---|---|
| PGD | - | 1.0 | 87.19 | 56.44 | 51.8 |
| | 0.9 | 0.98 | **87.73** | 56.8 | 52.37 |
| | 0.7 | 0.939 | 87.25 | 57.58 | 53.17 |
| | 0.5 | 0.898 | 87.13 | 58.43 | 54.03 |
| | 0.3 | 0.857 | 86.09 | **58.51** | **54.29** |
| | 0.1 | 0.816 | 85.2 | 58.36 | 53.92 |

Table 17: Performance of DPLS applying to the TRADES model on CIFAR-10 according to the smoothing factor $\lambda$.

| Model | $\lambda$ (smoothing factor) | Avg. smoothing | Standard | Robust (PGD) | Robust ($A^3$) |
|---|---|---|---|---|---|
| TRADES | - | 1.0 | **85.66** | 58.46 | 54.08 |
| | 0.9 | 0.979 | 85.49 | 58.74 | 54.39 |
| | 0.7 | 0.935 | 85.41 | 59.08 | 54.8 |
| | 0.5 | 0.893 | 85.36 | **59.33** | 55.19 |
| | 0.3 | 0.85 | 84.95 | 59.29 | 55.22 |
| | 0.1 | 0.807 | 84.06 | 59.25 | **55.38** |

**Ablation study on the factor of DPLS** Table 16 and Table 17 show the performance of DPLS applying to the PGD and TRADES models on CIFAR-10, respectively. We calculated

the average value for the smoothing factor of DPLS. In both results, it is observed that applying DPLS with the factor of high value further increases the robustness performance.

**Ablation study on the regularization of DPLS** We used an LS approach for mitigating the effect of hard examples, instead of simply reweighting the loss of hard examples. Pruning hard examples corresponds to assigning zero values to the loss of hard examples, so we compare the performance of DPLS with the reweighting method. We applied the difficulty of each example to the loss to reweight the loss of each example, which is the same difficulty with DPLS. Because TRADES utilized the robustness loss without labels, we applied DPLS to cross-entropy loss for clean examples and reweighted the robust loss with the same difficulty; thus, the difference between DPLS and the reweighting method in TRADES is the difference of application of the method to the loss for clean examples. Table18 lists the result of applying difficulty proportional reweight (DPR) and DPLS. In the results of both PGD and TRADES, DPLS shows higher standard accuracy. It is inferred that because the DPR method still ensures that the predictions are identical to one-hot labels and that the method focuses on the memorization of examples, the method improves robustness and deteriorates accuracy depending on robustness-accuracy trade-off. In contrast, instead of memorizing hard examples with one-hot label, our method takes advantage of hard examples by allowing soft labels and learns distributions that smoothly include hard examples, which results in improving robustness with decreasing the reduction of standard performance.

Table 18: Performance comparison of DPLS with reweighting methods.

| Method | PGD | | | TRADES | | |
|---|---|---|---|---|---|---|
| | Std. | Robust (PGD) | Robust ($A^3$) | Std. | Robust (PGD) | Robust ($A^3$) |
| - | **87.19** | 56.44 | 51.8 | **85.66** | 58.46 | 54.08 |
| DPR | 86.1 | 57.32 | 53.2 | 84.63 | 58.54 | 54.88 |
| DPLS | 87.13 | **58.43** | **54.03** | 85.36 | **59.33** | **55.19** |

**Study on balancing the number of data in each class for pruning** In our experiments, we selected easy and hard examples without consideration of class imbalance. The class distributions of 10k easy and 10k hard examples in our experiment are shown in Figure 14. We conducted the ablation study on the effect of balancing the class distribution of the hard examples subset. Table 19 shows the results of the pruning models of TRADES, which pruned top 10% examples with high difficulty from the training dataset of CIFAR-10. We experimented with varying the difficulty measurement loss and the balance of the number of pruned data for each class (the number of pruned data for each class in balanced subset is 500). We extracted the balanced pruning subset by selecting top 10% examples with high difficulty from the training subset of each class. It is observed that while the pruning models without consideration of class imbalance improve robustness, the pruning models with the balanced pruning subset show the similar robustness performance with that of the baseline model. It indicates that when selecting hard examples which have the negative effect to training, it is less effective for the improvement of the robustness performance to consider the balance of the number of data for each class.

Table 19: Performance of pruning models according to various difficulty losses with and without balancing the excluded data number of each class.

| Model | Method | Balancing | Standard | Robust (PGD) | Robust (CW) | Robust ($A^3$) |
|---|---|---|---|---|---|---|
| TRADES | - | - | **85.66** | 58.46 | 56.53 | 54.08 |
| | C-score | ○ | 85.2 | 57.87 | 56.72 | 53.97 |
| | C-score | × | 85.35 | **58.52** | 56.82 | 54.25 |
| | CE loss | ○ | 83.7 | 57.49 | 56.76 | 54.01 |
| | CE loss | × | 83.82 | 58.37 | **57.31** | 54.45 |
| | EL2N | ○ | 83.8 | 57.76 | 56.89 | 54.07 |
| | EL2N | × | 83.25 | 57.86 | 56.9 | 54.28 |
| | 0-1 loss | ○ | 83.81 | 57.84 | 56.76 | 54.22 |
| | 0-1 loss | × | 84.07 | 58.04 | 57.4 | **54.64** |

**Study on the robust fairness** In Xu et al. (2021), they indicated that adversarial training differently increases robustness of each class in a dataset, which effectively increases the robustness of easy classes but not effectively increases the robustness of hard classes. They noted that the performance imbalance between classes, which is termed as the robustness fairness problem, becomes more significant in adversarial training. Because DPLS regularizes hard examples, it can be assumed that the increase in the robustness performance is attributed to the increase in the performance of easy classes. Thus, we compared the robustness performance of easy and hard classes in Table 20. We divide the class of CIFAR-10 into 5 easy and 5 hard class subsets. We referred to the class distributions of easy and hard examples from Figure 14 and divide them into easy (class 0, 1, 7, 8, 9) and hard (class 2, 3, 4, 5, 6). From Table 20, it is observed that the performance increase is attributed to the increase in the robustness improvement of both the easy and hard class subsets in the DPLS models. It indicates that our method does not only exploit the performance increase of the easy classes, but increases the both performance of easy and hard classes.

Table 20: Robustness performance against adversarial attacks for easy class and hard class subsets of CIFAR-10.

| Model | Method | PGD | PGD (easy) | PGD (hard) | $A^3$ | $A^3$ (easy) | $A^3$ (hard) |
|---|---|---|---|---|---|---|---|
| PGD | - | 56.54 | 70.5 | 42.58 | 51.8 | 67.62 | 35.98 |
| | DPLS | **58.49** | **71.82** | **45.16** | **54.05** | **68.66** | **39.44** |
| TRADES | - | 58.52 | 71.42 | 45.62 | 54.08 | 68.64 | 39.52 |
| | DPLS | **59.32** | **72.9** | **45.74** | **55.19** | **70.22** | **40.16** |

Table 21: Performance evaluation of DPLS using MD (Jiang et al., 2020) and RayS (Chen and Gu, 2020) attacks.

| Method | PGD | | | | TRADES | | | |
|---|---|---|---|---|---|---|---|---|
| | Std. | Robust (MD) | Robust (RayS) | Robust ($A^3$) | Std. | Robust (MD) | Robust (RayS) | Robust ($A^3$) |
| - | **87.19** | 51.93 | 61.13 | 51.80 | **85.66** | 54.17 | 60.84 | 54.08 |
| DPLS | 87.13 | **54.07** | **62.14** | **54.03** | 85.36 | **55.25** | **62.11** | **55.19** |

## Appendix E. Additional comparison study

**Additional comparative study using existing evaluation attack methods** We primarily evaluated our models against AutoAttack (Croce and Hein, 2020) because it provides an accurate assessment without giving a false sense of security due to issues like gradient obfuscation. Since the MD ensemble attack (Jiang et al., 2020), which includes margin decomposition (MD), appears to be a variant of AutoAttack, we also measured the CIFAR-10 model's performance against this attack. The results are presented in Table 21. As shown in the table, the MD ensemble attack yields performance results and patterns very similar to those of AutoAttack, demonstrating that our method, DPLS, is also robust against the MD attack.

Furthermore, we evaluated our models against RayS (Chen and Gu, 2020), an attack method that can reveal vulnerabilities leading to a false sense of security. The results, also presented in Table 21, indicate that our methodology, DPLS, is robust against RayS and shows improved performance compared to the baseline model.

Table 22: Performance evaluation of DPLS combined with Diffpure (Nie et al., 2022) on RobustBench (Croce et al., 2020)

| Dataset | PGD | | TRADES | |
|---|---|---|---|---|
| | Std. | Robust ($A^3$) | Std. | Robust ($A^3$) |
| Baseline | **88.28** | 53.32 | 86.13 | 56.05 |
| Baseline + DPLS | 86.72 | 56.25 | 85.15 | 56.84 |
| Baseline + DiffPure | **88.28** | 66.87 | **87.11** | 66.80 |
| Baseline + DPLS + DiffPure | 87.50 | **68.61** | 86.72 | **71.09** |

**Additional comparative study combining existing defense methods** DiffPure (Nie et al., 2022) is an image purification method that leverages pre-trained diffusion-based models. Since the classifier model operates independently of the image purification method, it can be combined with any adversarial training-based approach. We applied the DiffPure method to two models: one trained using baseline adversarial training methods without DPLS, and another trained using adversarial training methods with DPLS. In the CIFAR-10 setting of

DiffPure, we measured the performance by replacing the classifier model and evaluated its robustness using RobustBench (Croce et al., 2020).

As shown in Table 22, we compared the performance of adding only DiffPure to the PGD and TRADES baseline models with the performance when both DiffPure and DPLS are added together. We observed that the models with DPLS added achieved relatively higher performance. Since DiffPure operates independently of the classifier model to enhance adversarial defense performance, utilizing a model trained with DPLS—which is comparatively more robust—appears to result in better defense performance.