REGULARIZING HARD EXAMPLES IMPROVES ROBUSTNESS

Anonymous authors
Paper under double-blind review

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 and that the memorization of hard examples is attributed to the significant increase in the difficulty of hard examples. We find that the increased difficulty of hard examples brings about the functioning of hard examples as label corrupted data in adversarial training, thereby leading to the memorization of those hard examples and deterioration of the robustness performance. Based upon these observations, we propose a new approach, difficulty proportional label smoothing (DPLS), to 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.

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 performance rendered by using more data is observably greater in adversarial settings than in standard settings (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 training 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.

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 set or continued by training on the entire training data[1]. We evaluated the test accuracy for clean 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 enlarged 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. Furthermore, we demonstrate that hard examples are fitted only through memorization in adversarial training and that pruning them from training improves robustness. Considering that the memorization of outliers or label corrupted data deteriorates generalization, we investigate the aforementioned demonstration by training a model with dataset which corrupts the labels of hard examples and comparing the performance. We find that, surprisingly, assigning random labels to hard examples does not change the robustness performance. It implies that hard examples in adversarial training function as label corrupted data and that it leads to memorization of hard examples, which accordingly degrades generalization.

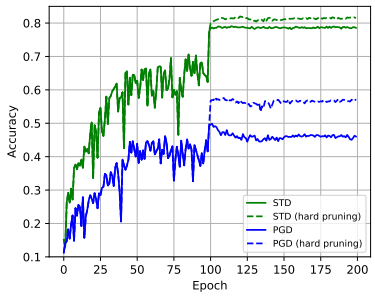


Figure 1: The test accuracy (for clean examples) of the standard (STD) models and the test accuracy (for adversarial examples) of the adversarial (PGD (Madry et al., 2017)) models trained on the subsets of CIFAR-10. On pruning hard examples from training, test accuracy increases both in the STD and PGD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

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 overconfidence and 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.

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:

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

In the aforementioned equation, θ 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 ϵ 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 online statistics-based discarding strategy (Liu et al., 2022).

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

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). 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 METHOD

3.1 THEORETICAL ANALYSIS ON 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 $f(\boldsymbol{x}) = \sigma(\boldsymbol{w}^\top \boldsymbol{x})$, 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 $\left\|\boldsymbol{w}^\top \boldsymbol{x}_1\right\| = d_1$, and the easy example is set as $(\boldsymbol{x}_2, y_2 = +1)$ where $\left\|\boldsymbol{w}^\top \boldsymbol{x}_2\right\| = d_2$ and $d_1 < d_2$. The differences between the gradient norm of the examples in standard and adversarial settings are as follows:

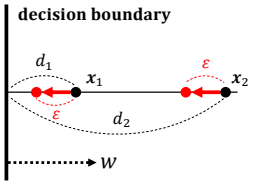


Figure 2: The schematic illustration of easy (x_2) and hard examples (x_1).

Theorem 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:*

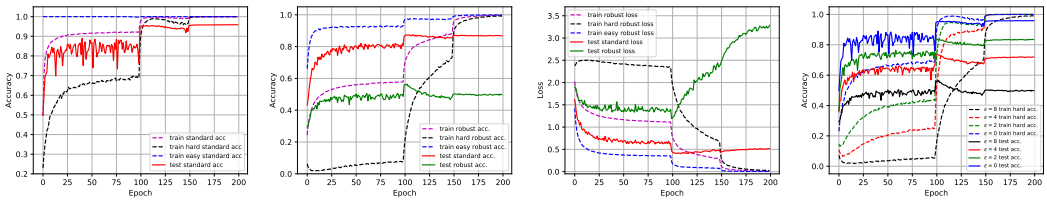
$$\begin{aligned} \mathcal{D}_{\text{std}}(\boldsymbol{x}_1, \boldsymbol{x}_2) &= \left\|(1 - \sigma(\boldsymbol{w}^\top \boldsymbol{x}_1))\boldsymbol{x}_1\right\| - \left\|(1 - \sigma(\boldsymbol{w}^\top \boldsymbol{x}_2))\boldsymbol{x}_2\right\| \\ \mathcal{D}_{\text{adv}}(\boldsymbol{x}_1, \boldsymbol{x}_2,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

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 corollary demonstrates that the difference between the loss gradient norm of hard and easy examples in adversarial training is more significant than that in standard training:

Corollary 1. *Let the training loss of the classifier almost converge to the minimum. Then, if $\boldsymbol{w}^\top \boldsymbol{x}_1 \geq 0$ and $0 < \epsilon < \frac{(d_1 + d_2)}{2}$, it satisfies*

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

We have reported the proof in Appendix B. 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 considerably 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.



(a) (b) (c) (d)

Figure 3: (a) and (b) shows the training and test accuracy curves in the STD and PGD models, respectively. (c) presents the training and test loss curves in the PGD model. (d) shows the training accuracy of hard examples and test accuracy of the models with different adversarial budgets ϵ.

3.2 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 in this section. We first demonstrate the theoretical analysis presented in Section 3.1. 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.

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.

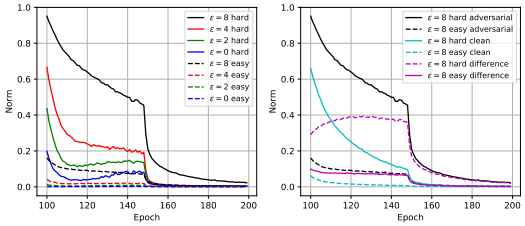
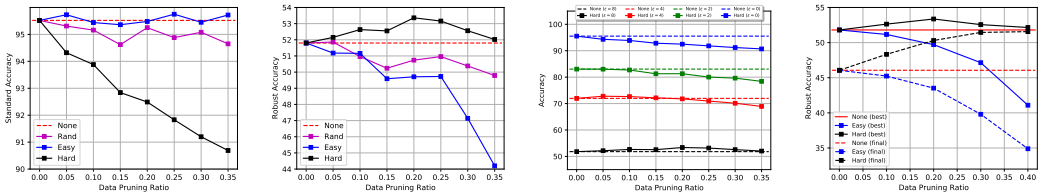


Figure 4: Average error vector norm (EL2N score (Paul et al., 2021)) of training examples.

As illustrated in Fig. 4 (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 ϵ 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.

Memorization 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. 3a and 3b 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. 3c illustrates the results of



(a) (b) (c) (d)

Figure 5: (a) and (b) show the accuracy of STD and PGD models where subset of training data is pruned. (c) shows the results that repeat the experiment of (a) and (b) for different adversarial budgets. (d) shows the best and final accuracy of the PGD pruning models.

cross-entropy loss in the PGD model, depicting the increasing gap between the train hard and test robust losses after the decay. These results suggest that the PGD model memorizes hard examples. Fig. 3d shows the training accuracy of hard examples and the test accuracy of the models with varying adversarial budget ϵ. 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 ϵ increases. This accordingly indicates that the memorization of hard examples may become severe in adversarial training.

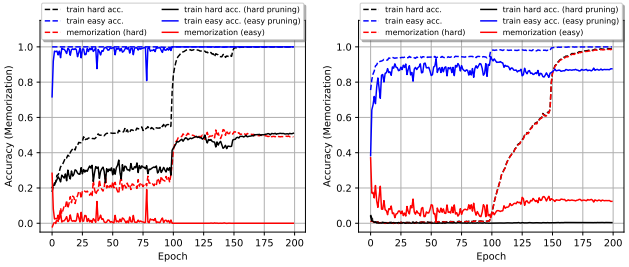


Figure 6: Memorization of easy and hard examples in the STD (left) and PGD (right) models.

Considering that determining the phenomenon as the memorization of hard examples is not sufficient, we approximately measure the memorization score defined in Feldman (2020), $\Pr_{f\sim A(D)}[\arg\max f(x_i) = y_i] - \Pr_{f\sim A(D^{\setminus i})}[\arg\max f(x_i) = y_i]$, where A denotes a training algorithm, D represents a dataset, and $D^{\setminus i}$ denotes the dataset with data (x_i, y_i) removed. Fig. 6 demonstrates the memorization (red lines) of easy and hard examples in the STD and PGD models. Observably, the PGD model does not fit hard examples before the learning rate decay and fits hard examples only through memorization after the decay; the memorization score of hard examples starts to increase after the decay and converges to the maximum score at the end. Thus, the PGD model memorizes hard examples, which implies that the memorization of hard examples deteriorates robustness. Details for the analysis of memorization are reported in Appendix C.4

While the result of the STD model in Fig 6 (left) shows a moderate increase in memorization for hard examples at the learning rate decay, the performance of the STD model increases, which is contrast to the result of the PGD model. Because even the easy examples present underfitting in the PGD model, as in Fig. 3b, the performance increase at the decay is the result of the fitting of underfitted data. Contrarily, the training accuracy of easy examples in the STD model is approximately 100%. This indicates that fitting hard examples increases the performance in standard training, and large portions of hard examples are fitted through memorization, like the result of the PGD model. Thus, the memorization of hard examples increases performance in standard settings but decreases performance in adversarial settings. We will address this difference between the memorization of hard examples in standard and adversarial settings in Section 3.3.

Pruning hard examples from training Before addressing the abovementioned phenomenon, as another rationalization that training hard examples decreases the performance in adversarial training, we demonstrate that pruning hard examples from training improves robustness. Because the memorization of hard examples is observed after the learning rate decay, we prune hard examples from around that training point. We prune hard, easy, and random example subsets from the training dataset of CIFAR-10. Figs. 5a and 5b show the best performance results of the STD and PGD pruning models. The result of the STD model is consistent with that reported by Paul et al. (2021); they noted that when data are normal, examples with high loss (e.g. EL2N score) are considered important. In this context, our results indicate that solely pruning hard examples reduces the accuracy plotted in Fig. 5a. However, the result of the PGD model shows the opposite trend. In Fig. 5b, pruning hard examples does not reduce the performance but rather improves the performance. Pruning easy examples decreases the performance, which does not affect accuracy in standard training.

The results of pruning easy examples may be a result of underfitting in the PGD model. Contrarily, hard examples exhibit underfitting before the decay and memorization after the decay (in Fig. 6) in both models; however, they yield different results for pruning of hard examples. Fig 5c demonstrates that the improvement by pruning hard examples decreases as the adversarial budget ϵ decreases, which implies that this improvement could be considered as the manifested property of adversarial training. The result in Fig. 5d reveals the property clearly; it shows the performances of the pruning models at the final checkpoint where the memorization score of hard examples is the maximum. Comparing the performance of the final checkpoint with the performance of the normal PGD model (red dotted line), the improvement by pruning hard examples is significant. Comparing the performances of the best and final checkpoints in each model, easy pruning models show large gaps; however, the gaps of hard pruning models decrease as the pruning ratio increases. These results also demonstrate that the memorization of hard examples deteriorates the robustness performance in adversarial training.

Label corrupted behavior of hard examples As reported by Paul et al. (2021), examples with corrupted labels exhibit the highest difficulty scores in addition to hard examples. Because memorizing examples with corrupted labels decreases performance, they demonstrated that pruning examples with highest score improved the accuracy when certain data are corrupted. Considering this demonstration, we hypothesize that the abovementioned difficulty amplification effect in Section 3.1 substantially increases the difficulty of hard examples, thereby transforming the hard examples into such examples of data with label corruption. To verify this, we perform shuffling test by assigning random labels to hard examples in standard and adversarial training.

Table 1: Performance comparison of models with the label corruption.

Dataset	STD	PGD	
		Best	Final
Normal	95.52	51.8	46.07
Corrupted	92.61	51.99	45.67
Pruned	93.88	52.63	48.81

Table 1 lists the test accuracy of the STD and PGD models trained with normal dataset (Normal), with dataset that the labels of the top 5k hard examples are corrupted (Corrupted), and with dataset that the same examples are pruend (Pruned). For the STD results, the normal model offers the best performance, and the pruned model naturally exhibits a higher performance than the corrupted model. However, the results for PGD indicate that the pruned model exhibits the best performance and that the normal model performance is similar to that of the corrupted model. The training accuracy of the corrupted model on hard examples is 0% before the learning rate decay and approximately 94% at the end, which is also similar result to that of the normal PGD model. This implies that, in contrast to standard training, the result in adversarial training is the same *regardless of whether the labels of hard examples are completely random or correct*. These findings support our hypothesis that the hard examples are transformed to examples that function as label corrupted data in adversarial training.

A possible reason for this phenomenon is the hypothesis that hard examples comprise features slightly correlated with the labels. Tsipras et al. (2018) noted that any feature slightly correlated with the label is advantageous, but the same feature becomes anti-correlated with the correct label in an adversarial setting. We can then assume the existence of certain examples comprising only features that are slightly correlated with the label. These examples are useful in a standard setting but are transformed into examples where all features are anti-correlated with the correct label in an adversarial setting. Training with only these examples assigns a zero or lower weight to the model parameters in the adversarial setting and exhibits performance as that of random prediction. We verify this by investigating the performance of the models trained with hard examples. The performances of the models trained using random 5k, top 5k easy, and top 5k hard examples are compared. We measure the standard (Std.) and robust (Rob.) accuracy for the PGD models. In Table 2, the STD model trained with hard examples exhibits a lower performance, suggesting that hard examples have a comparatively small correlation with labels. The PGD model trained with the hard examples exhibits a performance similar to that of the random prediction (10%) for both types of accuracy. This indicates that when a model is trained with normal dataset, it cannot fit hard examples without memorizing them in adversarial training because the model otherwise results in the random prediction when inferring these examples. These results support our hypothesis that hard examples comprise only slightly correlated features and become anti-correlated with the correct label in an adversarial setting. Here, the labels of hard examples are actually uncorrupted; however, the features of hard examples are corrupted in an adversarial setting, and these examples function as label corrupted data. Thus, we term these examples *feature corrupted data*.

Table 2: Performance of models trained with 5k examples.

Method	STD	PGD	
		Std.	Rob.
Rand	80.79	69.07	28.36
Easy	72.93	55.93	34.54
Hard	34.02	12.1	8.1

3.3 MITIGATION OF THE NEGATIVE EFFECT OF HARD EXAMPLES

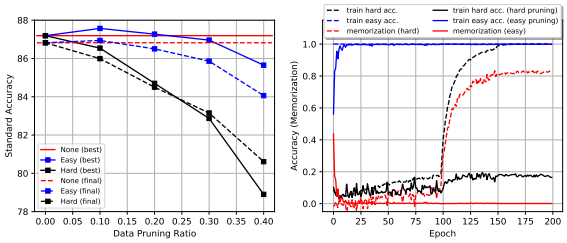


Figure 7: The standard accuracy of the PGD model at the best and final checkpoints (left). The memorization result of clean examples in the PGD model.

Although memorizing hard examples impairs the performance in an adversarial setting, simply pruning hard examples from training is not the optimal solution for the following reasons: (i) finding the optimal difficulty threshold for removing hard examples that deteriorate robustness is an intractable proposition, and (ii) hard examples are not incorrect data and do not function as feature corrupted data with respect to the standard accuracy. The results in Fig. 7 (left) show the standard accuracy of the PGD pruning models at the final and best performance checkpoints. This indicates that pruning hard examples in adversarial training decreases standard accuracy as the results of standard training in Fig. 5a, which is the opposite result of the previous demonstrations. With the observations in Figs. 3 and 6, we compared the memorization results of the STD and PGD models, where the memorization of hard examples increased the performance in the STD model but decreased the performance in the PGD model. In Feldman (2020), they noted that memorizing examples where each example is the sole representative of a subpopulation can improve the performance of a model that are trained with natural data distribution by contributing to inferring test data from a similar subpopulation, and our memorization results of the STD model are consistent with this.

In case of PGD, although the PGD model utilizes the loss for adversarial examples, it involves fitting a clean version of these example. Thus, memorizing an adversarial version of hard examples leads to memorizing the clean version of them, as shown in Fig. 7 (right), which is the memorization result of clean examples in the PGD model. We can then explain the increase in the standard accuracy and the decrease in the robust accuracy when pruning hard examples in adversarial training: the memorization of the same hard examples with *uncorrupted/corrupted* features. Hard examples are mostly transformed into feature corrupted data in an intense adversarial setting and function as label corrupted data, thereby decreasing the robust accuracy; however, from the perspective of standard accuracy, the memorization of hard examples improves the accuracy by memorizing uncorrupted labels of hard examples with uncorrupted features because memorizing the adversarial version of hard examples involves memorizing the clean version of them. Therefore, pruning hard examples is sufficient if considering only robustness; however, both standard and robust accuracy are generally considered important. Thus, instead of pruning hard examples whose difficulty exceeds the manually defined difficulty threshold, we propose a method to leverage hard examples in training by regularizing every example where the intensity of regularization is proportional to the difficulty.

Mitigating the negative effect of hard examples We extend the LS method (Szegedy et al., 2016) because it can control the training of each example by utilizing the LS factor. LS replaces a one-hot label with a mixture of the label and the uniform distribution by assigning the LS factor, λ, to the ground-truth class and distributes $1 - \lambda$ to the other classes equally. Rather than simply reweighting the loss scale of each example, it effectively regularizes the fitting of one-hot label (see Appendix D). The general LS method assigns an identical LS factor to every example, and thus, the intensity of regularization applied to every example is the same. We modified the LS method to ensure that the training of hard examples is regularized, but the training of other examples is less affected by regularization. We adopt a smoothing factor that is proportional to the difficulty of each example and control the intensity of regularization, thereby mitigating the memorization of hard examples, 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 in Appendix A.

Our method maintains the positive 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

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	51.8	TRADES	85.66	58.46	54.08
	+Pruning	86.44	56.18	52.56	+Pruning	84.07	58.04	54.64
	+LS	86.96	57.2	51.99	+LS	**86.32**	58.72	54.02
	+DPLS	**87.21**	**57.76**	**53.15**	+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	62.65	33.91	27.85	+LS	62.43	33.29	27.83
	+DPLS	62.43	**35.05**	**29.64**	+DPLS	**63.32**	**34.53**	**29.68**
SVHN	PGD	**92.96**	52.88	35.2	TRADES	91.84	59.14	47.46
	+Pruning	92.68	55.52	40.87	+Pruning	91.54	**60.27**	47.79
	+LS	92.65	66.2	38.44	+LS	92.27	59.65	46.37
	+DPLS	92.94	**76.53**	**42.01**	+DPLS	**92.33**	59.4	**49.05**

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.

4 Experiment

Implementation details We conducted experiments on CIFAR-10, CIFAR-100 (Krizhevsky et al., 2009), and SVHN (Netzer 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 & 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., 2022). Further details are presented in Appendix C.1.

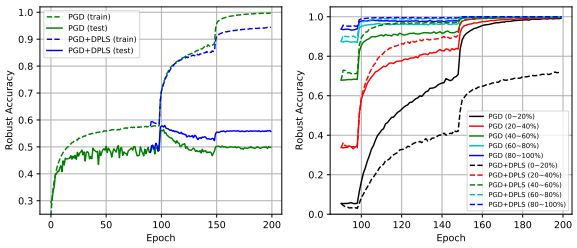


Figure 8: The training and test accuracy curves of the PGD and the PGD with DPLS model on CIFAR-10.

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 generalization. The performance improvement of the pruning models supports our analysis in Section 3.2, demonstrating 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. 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. 8 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 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

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	85.92	57.8	**53.71**

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

Method	PGD	TRADES
-	51.8	54.08
C-score	52.62	54.71
CE loss	52.57	54.45
EL2N	52.83	54.32
0-1 loss	**53.15**	**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**

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.

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 C.6. 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. This result indicates 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) (Carmon 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 (refer to Appendix A), further improvement is achievable by using the better difficulty measurement loss. Further experiments and ablation studies on DPLS can be found in Appendix C.6 and Appendix D.

5 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. The increased difficulty of hard examples in an adversarial setting transforms hard examples into feature corrupted data and decreases the performance. Thus, we proposed a method, DPLS, which 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.

REFERENCES

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*, pp. 11190–11201, 2019.

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.

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*, pp. 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*, pp. 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.

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.

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 (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 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.

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*, 2022.

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.

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.

Ludwig Schmidt, Shibani Santurkar, Dimitris Tsipras, Kunal Talwar, and Aleksander Madry. Adversarially robust generalization requires more data. In *Advances in Neural Information Processing Systems*, pp. 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*, pp. 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*, pp. 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*, pp. 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*, pp. 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*, pp. 25595–25610. PMLR, 2022.

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 (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 7472–7482, Long Beach, California, USA, 09–15 Jun 2019. PMLR. URL http://proceedings.mlr.press/v97/zhang19p.html. https://github.com/yaodongyu/TRADES.

A THE PROCEDURE OF DPLS

A.1 THE PROCEDURE OF DPLS IN ADVERSARIAL TRAINING

Algorithm 1 DPLS adversarial training

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

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

The procedure of difficulty proportional label smoothing (DPLS) is described in Algorithm 1. DPLS measures and saves the average difficulty of each example until the calculation epoch T. The difficulty of each example is calculated at the epoch T. Soft labels that are calculated using DPLS following Algorithm 2 replace the one-hot label of each example. DPLS does not include difficulty after the epoch T, and thus, the difficulty of each example is fixed at the epoch T. Because the soft labels are already biased to regularize each example according to the difficulty, the models trained using these soft labels can offer biased difficulty; thus, DPLS does not include difficulty after the calculation. Therefore, the subsequent training after the difficulty calculation is performed using the fixed soft labels.

A.2 THE PROCEDURE FOR THE CALCULATION OF DPLS

Algorithm 2 Pre-calculation of DPLS

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

Initialization:
  apply min-max normalization to difficulty:
  $\mathcal{S} \leftarrow \frac{\mathcal{S} - \min(\mathcal{S})}{\max(\mathcal{S}) - \min(\mathcal{S})}$
for $i = 1$ **to** N **do**
  $(\boldsymbol{x}_i, \boldsymbol{y}_i) \leftarrow D[i]$
  apply label smoothing that is proportional to difficulty:
  $\boldsymbol{y}_i \leftarrow LS(\boldsymbol{y}_i, 1 - (1-\lambda)\mathcal{S}[i], C)$
  $D[i] \leftarrow (\boldsymbol{x}_i, \boldsymbol{y}_i)$
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 $\boldsymbol{y}$
end function

The procedure of the calculation of DPLS is described in Algorithm 2. Because our method uses fixed soft labels after the calculation epoch T, our method can utilize the pre-calculated difficulty such as C-score (Jiang et al., 2021), which can be easily obtained from the official site. For the calculation, DPLS assigns the label smoothing factor 1 to the easiest example ($\mathcal{S}[i] = 0$) and assigns the factor λ to the most difficult example ($\mathcal{S}[i] = 1$).

B PROOFS

Theorem 2 (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:*

$$\begin{aligned} \mathcal{D}_{\text{std}}(\boldsymbol{x}_1, \boldsymbol{x}_2) &= \big\|(1-\sigma(\boldsymbol{w}^\top \boldsymbol{x}_1))\boldsymbol{x}_1\big\| - \big\|(1-\sigma(\boldsymbol{w}^\top \boldsymbol{x}_2))\boldsymbol{x}_2\big\| \\ \mathcal{D}_{\text{adv}}(\boldsymbol{x}_1, \boldsymbol{x}_2, \epsilon) &= \big\|(1-\sigma(\boldsymbol{w}^\top (\boldsymbol{x}_1+\boldsymbol{\delta})))(\boldsymbol{x}_1+\boldsymbol{\delta})\big\| - \big\|(1-\sigma(\boldsymbol{w}^\top (\boldsymbol{x}_2+\boldsymbol{\delta})))(\boldsymbol{x}_2+\boldsymbol{\delta})\big\|. \end{aligned} \tag{4}$$

Proof.

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

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

$$\frac{\partial \mathcal{L}(f(\boldsymbol{x}), y=+1)}{\partial \boldsymbol{w}} = -(1-\sigma(\boldsymbol{w}^\top \boldsymbol{x}))\boldsymbol{x}. \tag{5}$$

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

$$\left\|\frac{\partial \mathcal{L}(f(\boldsymbol{x}_1), y_1)}{\partial \boldsymbol{w}}\right\| - \left\|\frac{\partial \mathcal{L}(f(\boldsymbol{x}_2), y_2)}{\partial \boldsymbol{w}}\right\| = \big\|(1-\sigma(\boldsymbol{w}^\top \boldsymbol{x}_1))\boldsymbol{x}_1\big\| - \big\|(1-\sigma(\boldsymbol{w}^\top \boldsymbol{x}_2))\boldsymbol{x}_2\big\|. \tag{6}$$

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

$$\begin{aligned} &\left\|\frac{\partial \mathcal{L}(f(\boldsymbol{x}_1+\boldsymbol{\delta}), y_1)}{\partial \boldsymbol{w}}\right\| - \left\|\frac{\partial \mathcal{L}(f(\boldsymbol{x}_2+\boldsymbol{\delta}), y_2)}{\partial \boldsymbol{w}}\right\| \\ &\qquad = \big\|(1-\sigma(\boldsymbol{w}^\top (\boldsymbol{x}_1+\boldsymbol{\delta})))(\boldsymbol{x}_1+\boldsymbol{\delta})\big\| - \big\|(1-\sigma(\boldsymbol{w}^\top (\boldsymbol{x}_2+\boldsymbol{\delta})))(\boldsymbol{x}_2+\boldsymbol{\delta})\big\|. \end{aligned} \tag{7}$$

□

Corollary 2. *Let the training loss of the classifier almost converges to the minimum. Then, if $\boldsymbol{w}^\top \boldsymbol{x}_1 \geq 0$ and $0 < \epsilon < \frac{(d_1+d_2)}{2}$, it satisfies*

$$\mathcal{D}_{\text{std}}(\boldsymbol{x}_1, \boldsymbol{x}_2) < \mathcal{D}_{\text{adv}}(\boldsymbol{x}_1, \boldsymbol{x}_2, \epsilon). \tag{8}$$

Proof. Because the training loss of the classifier almost converges to the minimum, we can assume that the gradient norm of easy examples is smaller than the gradient norm of hard examples in both standard and adversarial training. Therefore, $\mathcal{D}_{\text{std}}(\boldsymbol{x}_1, \boldsymbol{x}_2) > 0$ and $\mathcal{D}_{\text{adv}}(\boldsymbol{x}_1, \boldsymbol{x}_2, \epsilon) > 0$. Then, we can rewrite Equation 9 as

$$\frac{\mathcal{D}_{\text{std}}(\boldsymbol{x}_1, \boldsymbol{x}_2)}{\mathcal{D}_{\text{adv}}(\boldsymbol{x}_1, \boldsymbol{x}_2, \epsilon)} < 1. \tag{9}$$

Let the prediction for x_1 is not incorrect, where $\boldsymbol{w}^\top \boldsymbol{x}_1 \geq 0$. 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

$$\frac{\mathcal{D}_{\text{std}}(\boldsymbol{x}_1, \boldsymbol{x}_2)}{\mathcal{D}_{\text{adv}}(\boldsymbol{x}_1, \boldsymbol{x}_2, \epsilon)} = \frac{\big\|(1-\sigma(\boldsymbol{w}^\top \boldsymbol{x}_1))\boldsymbol{x}_1\big\| - \big\|(1-\sigma(\boldsymbol{w}^\top \boldsymbol{x}_2))\boldsymbol{x}_2\big\|}{\|(1-\sigma(\boldsymbol{w}^\top(\boldsymbol{x}_1+\boldsymbol{\delta})))(\boldsymbol{x}_1+\boldsymbol{\delta})\| - \|(1-\sigma(\boldsymbol{w}^\top(\boldsymbol{x}_2+\boldsymbol{\delta})))(\boldsymbol{x}_2+\boldsymbol{\delta})\|} \tag{10}$$

$$= \frac{(1-\sigma(\boldsymbol{w}^\top \boldsymbol{x}_1))\|\boldsymbol{x}_1\| - (1-\sigma(\boldsymbol{w}^\top \boldsymbol{x}_2))\|\boldsymbol{x}_2\|}{(1-\sigma(\boldsymbol{w}^\top(\boldsymbol{x}_1+\boldsymbol{\delta})))\|(\boldsymbol{x}_1+\boldsymbol{\delta})\| - (1-\sigma(\boldsymbol{w}^\top(\boldsymbol{x}_2+\boldsymbol{\delta})))\|(\boldsymbol{x}_2+\boldsymbol{\delta})\|} \tag{11}$$

$$= \frac{(1-\sigma(\boldsymbol{w}^\top \boldsymbol{x}_1)) - (1-\sigma(\boldsymbol{w}^\top \boldsymbol{x}_2))}{(1-\sigma(\boldsymbol{w}^\top(\boldsymbol{x}_1+\boldsymbol{\delta}))) - (1-\sigma(\boldsymbol{w}^\top(\boldsymbol{x}_2+\boldsymbol{\delta})))} \tag{12}$$

$$= \frac{\sigma(-\boldsymbol{w}^\top \boldsymbol{x}_1) - \sigma(-\boldsymbol{w}^\top \boldsymbol{x}_2)}{\sigma(-\boldsymbol{w}^\top(\boldsymbol{x}_1+\boldsymbol{\delta})) - \sigma(-\boldsymbol{w}^\top(\boldsymbol{x}_2+\boldsymbol{\delta}))} \tag{13}$$

$$= \frac{\sigma(-d_1) - \sigma(-d_2)}{\sigma(-(d_1-\epsilon)) - \sigma(-(d_2-\epsilon))} \tag{14}$$

$$= \frac{\sigma(-d_1) - \sigma(-(d_1+(d_2-d_1)))}{\sigma(-(d_1-\epsilon)) - \sigma(-(d_1-\epsilon+(d_2-d_1)))}. \tag{15}$$

Here, it satisfies Equation 9 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,

$$\frac{d}{dx}\big(\sigma(-x) - \sigma(-(x+a))\big) = \frac{d}{dx}\left(\frac{1}{1+e^x} - \frac{1}{1+e^{x+a}}\right) \tag{16}$$

$$= e^x\left(\frac{e^a}{(1+e^{x+a})^2} - \frac{1}{(1+e^x)^2}\right) \tag{17}$$

$$= e^x\left(\frac{(e^a-1)(1-e^{2x+a})}{(e^x+1)^2(e^{x+a}+1)^2}\right) < 0. \tag{18}$$

Since $a > 0$, Equation 18 can be represented as $1 - e^{2x+a} < 0$. Then, we can rewrite Equation 18 as

$$x > -\frac{d_2-d_1}{2}. \tag{19}$$

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

$$\mathcal{D}_{\text{std}}(\boldsymbol{x}_1, \boldsymbol{x}_2) < \mathcal{D}_{\text{adv}}(\boldsymbol{x}_1, \boldsymbol{x}_2, \epsilon). \tag{20}$$

□

C Additional implementation and experimental details

C.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 32×32 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 32×32 pixels. SVHN is obtained from a very large set of images from urban areas in various countries by using Google Street View. The license of CIFAR datasets is unknown and SVHN is a non-cokmmercial dataset. Further details can be found in `https://paperswithcode.com/datasets`.

Implementation details We conducted experiments on CIFAR-10, CIFAR-100 (Krizhevsky et al., 2009), and SVHN (Netzer 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 & Komodakis, 2016) as the architecture of our models. The learning rates for CIFAR-10, CIFAR-100 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 ℓ_∞-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., 2022) for reliable evaluation. We used a single RTX 8000 GPU with CUDA11.6 and CuDNN7.6.5 in our experiments.

C.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 ℓ_∞-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).

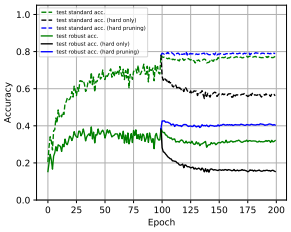


Figure 9: The test accuracy of PGD models trained with subsets of CIFAR-10.

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 excluded), (3) top 10k hard examples (hard only). We used 10k instead of 5k to prevent overfitting before pruning. In Fig. 9, 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. 9 are same with those of Fig. 4. 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).

C.3 Experiment details for comparing EL2N score

Figure 4 For the experiments in Fig. 4, we executed 200 training epochs on the CIFAR-10 dataset in each training. The other experimental details are summarized in C.1. The adversarial training time on the entire CIFAR-10 dataset is 47 hours for 200 epochs. The full results of Fig. 4 can be seen in Fig. 10. 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.

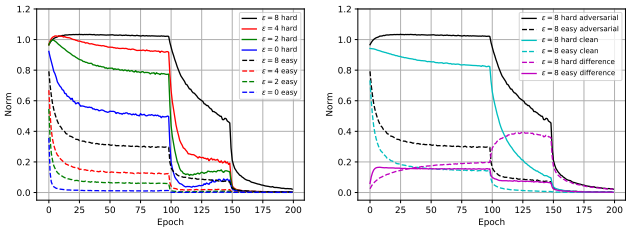
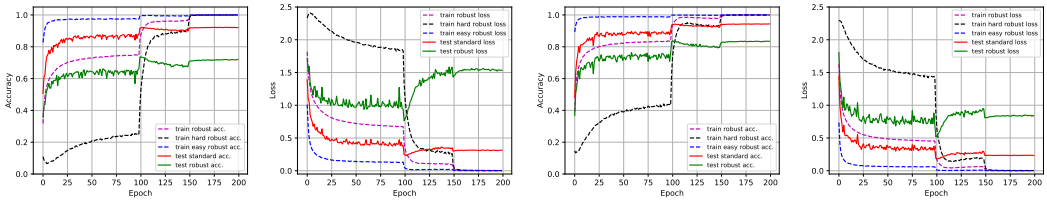


Figure 10: Average error vector norm (EL2N score) of hard and easy examples.

C.4 Experiment details and further experiments on memorization and pruning

Figure 3 We executed 200 training epochs on the CIFAR-10 dataset in each training. The other experimental details are summarized in C.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. 11. The robust overfitting phenomenon is observed to be severe as the adversarial budget of the training setting increases.



(a) (b) (c) (d)

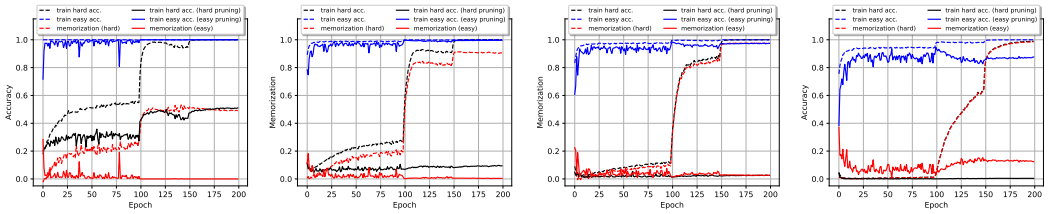
Figure 11: (a) shows the training and test accuracy in the PGD models with $\epsilon = 4$. (b) shows the loss curves in the PGD models with $\epsilon = 4$. (c) shows the training and test accuracy in the PGD models with $\epsilon = 2$. (d) shows the loss curves in the PGD models with $\epsilon = 2$.

Table 7: Performance comparison of models with subset pruned.

Method	STD	PGD
-	95.9	51.8
easy pruning	95.83	51.03
hard pruning	94.15	52.63

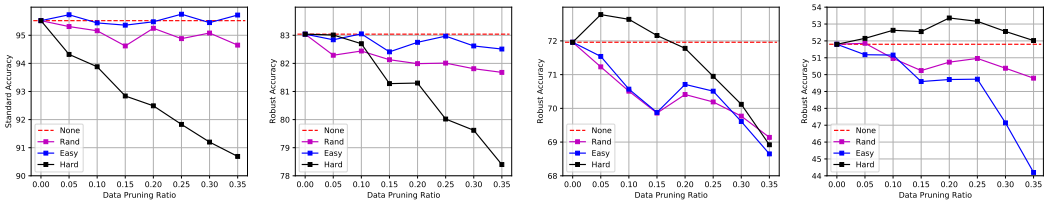
Figure 6 We executed 200 training epochs on the CIFAR-10 dataset in each training. The other experimental details are summarized in C.1. For the measurement of approximate memorization of easy and hard examples, we select top 5k easy and top 5k hard examples. We then trained the models with the dataset that prunes top 5k easy examples (easy pruning) and that prunes top 5k hard examples (hard pruning), respectively. Because it requires high computational complexity to compute the memorization score of each example in adversarial training, we used data subset as the unit of measurement instead of single data pair. Thus, *the memorization score is measured by the difference in the accuracy for easy/hard examples between the normal model and the model that easy/hard examples are pruned*. Since the performance of the hard pruning PGD model is 0% and the performance of the normal PGD model increase after the decay, the PGD model fits hard examples only through memorization. The test performance of each model is described in Table 7. It is expected that the measurement error of the memorization score caused by the difference in the performances of the models is not significant. In Fig. 6, the STD model also shows the comparatively high memorization score for hard examples after the learning rate decay. Memorization can cause overfitting; however, as indicated in Feldman (2020), memorization can improve accuracy on visually similar test examples, except outliers and examples with corrupted labels. Comparing the STD model at right after the learning rate decay epoch 100 where the memorization score of hard examples is high and the PGD model at around epoch 130 where the memorization score become similar with that of the STD model, the performance of the STD model increases, but the performance of the PGD model decreases. Because the training accuracy on the easy examples of the STD model are already saturated, it can be inferred that the increase in the performance is attributed to the increase in the fitting of hard examples. Thus, the memorization in the STD model improves the performance, which is the case of the improvement that caused by increase in the accuracy on visually similar test examples. Contrarily, the memorization in the PGD model deteriorates the performance, which is the except case of outliers and examples with

corrupted labels. We showed that hard examples in adversarial training functions like the examples with corrupted label through several experiments. The memorization results including experiments on adversarial training with $\epsilon = 4$ and $\epsilon = 2$ is described in Fig. 12.



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

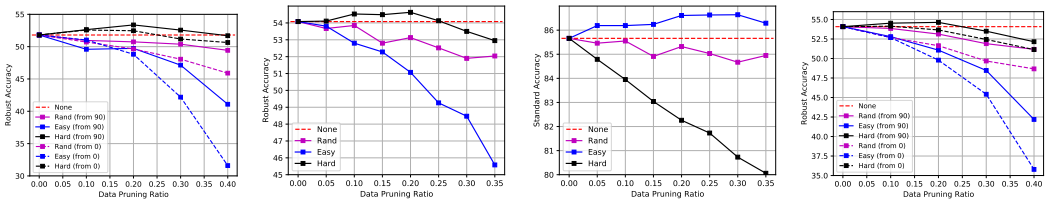
Figure 12: Memorization of easy and hard examples in each setting. Clean example results for a standard setting and adversarial example results for adversarial settings.



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

Figure 13: The performance of the models that prune subsets of training examples.

Figure 5 For the experiments in Fig. 5, We executed 110 training epochs on the CIFAR-10 dataset in each model. The other experimental details are summarized in C.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., 2022). The pruning results including experiments on adversarial training with $\epsilon = 4$ and $\epsilon = 2$ is described in Fig. 13. Fig. 14a depicts a comparison of the performance of the models that apply pruning at small epoch before the learning rate decay and at the beginning of the training. The trend is similar; however, the gap increases as the pruned examples are rendered easy. It is inferred that training of hard examples before the learning rate decay and memorization helps improve robustness generalization marginally. Figs. 14b and 14c illustrate the robust accuracy and the standard accuracy of the models that prune subsets of training examples in the TRADES models. Fig. 14d shows a comparison of the performance of the models that apply pruning at small epoch before the learning rate decay and at the beginning of the training in the TRADES models. The results of the TRADES models shows a similar trend to the results of the PGD models.



(a) (b) (c) (d)

Figure 14: (a) and (d) compares the accuracy of the PGD and TRADES models that apply pruning at epoch 90 and epoch 0, respectively. (b) and (c) show the robust accuracy and the standard accuracy of the models that prune subsets of training examples in the TRADES models, respectively.

C.5 EXPERIMENT DETAILS AND FURTHER EXPERIMENTS ON LABEL CORRUPTION

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 (hard)	82.31	49.32	45.67	82.11	50.64	46.15
Corrupted (rand)	82.27	43.86	38.64	83.0	47.89	42.92
Pruned	85.99	**52.36**	**48.81**	84.71	**53.44**	**48.78**

Table 1 For the experiments in Table 1, we executed 200 training epochs on the CIFAR-10 dataset in each model. The other experimental details are summarized in C.1. 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. We measured standard accuracy for the STD models and robust accuracy for the PGD models. The robust accuracy is evaluated using A^3 (Liu et al., 2022). Table 8 shows the final accuracy for the models trained with applying random labels to the top 5k hard examples or the 5k random 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. Because the corruption models which assign random labels to random examples show significantly low robust accuracy, it demonstrates that the models generally are vulnerable to label corruption; however, assigning random labels to hard examples does not change the robustness performance.

Table 9: Performance of models trained with 5k examples.

Method	STD		PGD ($\epsilon = 2$)		PGD ($\epsilon = 4$)		PGD ($\epsilon = 8$)	
	Std.	Rob.	Std.	Rob.	Std.	Rob.	Std.	Rob.
Rand	80.79	-	78.75	59.19	76.97	46.19	69.07	28.36
Easy	72.93	-	69.58	56.21	64.8	46.27	55.93	34.54
Hard	34.02	-	22.99	12.2	16.31	7.98	12.1	8.1

Table 2 For the experiments in Table 2, we executed 110 training epochs on the subsets of CIFAR-10 dataset in each model. The other experimental details are summarized in C.1. We measured standard accuracy for the STD models and measured both standad and robust accuracy for the PGD models. The robust accuracy is evaluated using A^3 (Liu et al., 2022). Table 9 shows the performance of the models trained with random 5k, top 5k easy, and top 5k hard examples as Table 2 with varying the adversarial budget ϵ. As the adversarial budget decreases, the standard accuracy of the models that prunes hard examples increases, which suggests that the number of anti-correlated features which were slightly correlated features decreases.

C.6 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 C.1. The adversarial training time on the CIFAR datasets is 26 hours for 110 epochs of the PGD models, and the training time on the SVHN dataset is 40 hours for 110 epochs of the PGD models. The adversarial training time on the CIFAR datasets is 37 hours for 110 epochs of the TRADES models, and the training time on the SVHN dataset is 57 hours for 110 epochs of the TRADES models. We mainly use 0-1 loss as difficulty measurement loss as in Section 3.2 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 and SVHN) and is 0.7 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 λ of the most difficult example for the DPLS models and the label smoothing factor for LS models

Table 10: The smoothing factor λ and the average of label smoothing factor for each model in Table 3 in the main paper.

Dataset	Model	Method	λ (smoothing factor)	Avg. smoothing
CIFAR10	PGD	LS	0.9	0.9
		DPLS	0.5	0.897
	TRADES	LS	0.9	0.9
		DPLS	0.5	0.893
CIFAR100	PGD	LS	0.7	0.7
		DPLS	0.3	0.693
	TRADES	LS	0.7	0.7
		DPLS	0.3	0.703
SVHN	PGD	LS	0.9	0.9
		DPLS	0.1	0.94
	TRADES	LS	0.9	0.9
		DPLS	0.1	0.927

Table 11: Performance of the models with $\epsilon = 2$ and $\epsilon = 4$ 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	91.78	72.6	71.96	TRADES	90.36	74.37	73.5
	+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	92.13	**74.68**	**73.53**	+DPLS	90.43	**75.38**	**74.47**

(a) $\epsilon = 4$

Dataset	Method	Std.	Robust (PGD)	Robust (A^3)	Method	Std.	Robust (PGD)	Robust (A^3)
CIFAR-10	PGD	94.17	83.2	83.04	TRADES	**93.32**	84.42	84.29
	+Pruning	92.95	82.73	82.7	+Pruning	92.08	83.84	83.69
	+LS	**94.56**	83.46	83.13	+LS	93.14	84.33	83.96
	+DPLS	94.35	**84.61**	**84.11**	+DPLS	92.94	**85.18**	**84.79**

(b) $\epsilon = 2$

in Table 3 of the main paper is summarized in Table 10. The results for $\epsilon = 2$ and $\epsilon = 4$ is listed in Table 11.

Figure 8 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 C.1.

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 C.1. We employed

Table 12: 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	86.69	57.02	52.62	86.6	59.0	54.71
CE loss	86.74	56.43	52.57	86.02	58.64	54.45
EL2N	86.53	56.95	52.83	**86.94**	58.94	54.32
0-1 loss	**87.21**	**57.76**	**53.15**	85.36	**59.33**	**55.19**

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 12.

Table 13: 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 14: 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 C.1. Table 13 and Table 14 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.

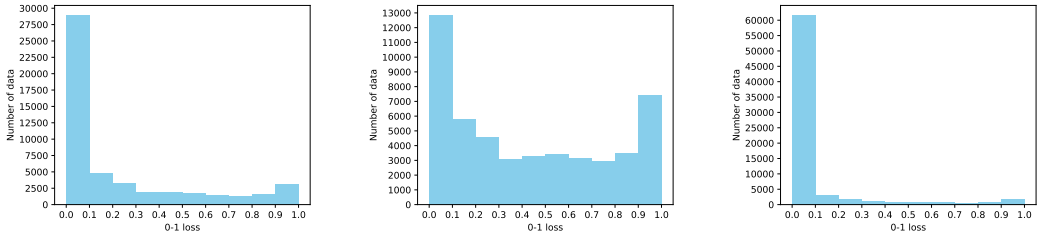
C.7 Other experiment details

Additional details of easy and hard examples Fig. 15 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. 16 shows the 0-1 loss distributions of the training datasets for CIFAR-10, CIFAR-100, 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. 17 shows easy and hard examples in the training dataset of CIFAR-10.

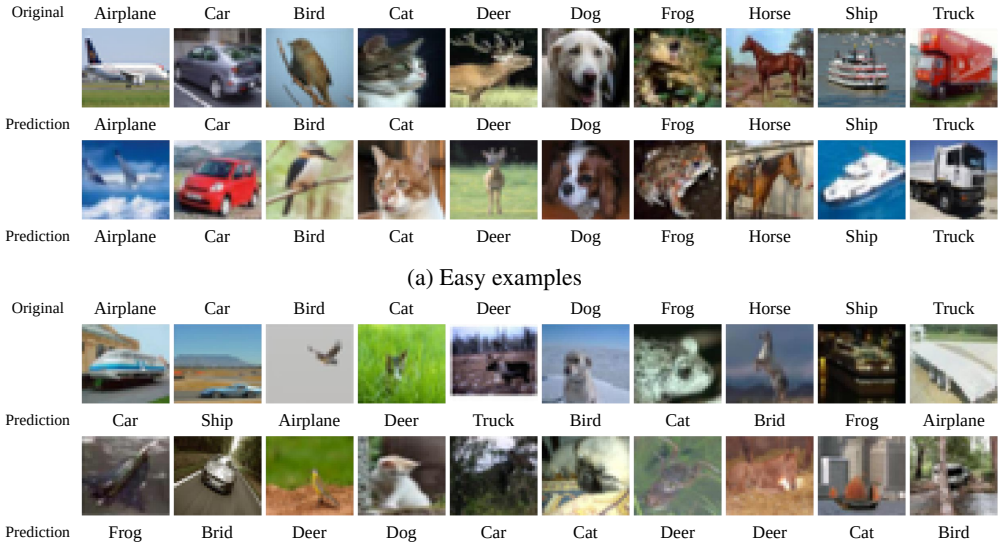
(a) Easy examples (b) Hard examples

Figure 15: 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 16: The 0-1 loss distributions of the each training dataset: CIFAR-10, CIFAR-100, and SVHN.



(a) Easy examples

(b) Hard examples

Figure 17: 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.

Table 15: Performance of DPLS according to the DPLS calculation epoch T.

T	PGD		TRADES	
	Std.	Rob.	Std.	Rob.
10	86.6	52.92	84.88	54.69
50	87.23	53.16	**85.56**	55.0
90	87.21	53.15	85.36	**55.19**
100	87.53	53.04	85.53	54.77

Table 16: 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	87.28	53.12	84.99	55.05
	50	87.22	53.11	84.89	55.11
	90	87.21	**53.15**	**85.36**	**55.19**
	100	**87.46**	53.04	85.26	54.93

D ADDITIONAL ABLATION STUDY

Ablation on the DPLS calculation epoch In Table 15, 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 16, 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 15 with the result in Table 16, 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 17: Performance of DPLS applying to the PGD model on CIFAR-10 according to the smoothing factor λ.

Model	λ (smoothing factor)	Avg. smoothing	Standard	Robust (PGD)	Robust (A^3)
PGD	-	1.0	87.19	56.44	51.8
	0.9	0.98	86.63	57.0	52.49
	0.7	0.939	**87.82**	56.89	52.48
	0.5	0.898	87.21	57.76	53.15
	0.3	0.857	87.05	57.93	53.2
	0.1	0.816	86.27	**58.44**	**53.92**

Ablation study on the factor of DPLS Table 17 and Table 18 show the performance of DPLS applying to the TRADES model on CIFAR-10 and CIFAR-100, 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

Table 18: Performance of DPLS applying to the TRADES model on CIFAR-10 according to the smoothing factor λ.

Model	λ (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**

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. Tab 19 lists the result of applying reweight and DPLS. In the results of both PGD and TRADES, DPLS shows higher standard accuracy. It is inferred that because the reweighting 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 19: 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
Reweight	86.1	57.32	53.2	84.63	58.54	54.88
DPLS	**87.21**	**57.76**	**53.15**	85.36	**59.33**	**55.19**

Table 20: Performance of exclusion 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 balancing the number of data in each class for exclusion 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 15. We conducted the ablation study on the effect of balancing the class distribution of the hard examples subset. Table 20 shows the results of the exclusion models of TRADES, which exclude 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 excluded data for each class (the number of excluded data for each class in balanced subset is 500). We extracted the balanced exclusion subset by selecting top 10% examples with high difficulty from the training subset of each class. It is observed that while the exclusion models without consideration of class imbalance improve robustness, the exclusion models with the balanced exclusion 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.

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 21. 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 15 and divide them into easy (class 0, 1, 7, 8, 9) and hard (class 2, 3, 4, 5, 6). From Table 21, 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 21: 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	**57.75**	70.32	**45.18**	**53.15**	**67.76**	**38.54**
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**