Journal of Machine Learning Research 25 (2024) 1-46 Submitted 8/22; Revised 5/23; Published 12/24

On the Impact of Hard Adversarial Instances on Overfitting in Adversarial Training

Chen Liu[1] † * CHEN.LIU@CITYU.EDU.HK

Zhichao Huang[2] ZHICHAO.HUANG@CONNECT.UST.HK

Mathieu Salzmann[3] MATHIEU.SALZMANN@EPFL.CH

Tong Zhang[4] ‡ TONGZHANG@TONGZHANG-ML.ORG

Sabine Süsstrunk[3] SABINE.SUSSTRUNK@EPFL.CH

[1] *Department of Computer Science, City University of Hong Kong*
83 Tai Chee Ave, Kowloon Tong, Hong Kong, China
[2] *Department of Mathematics, Hong Kong University of Science and Technology*
Clear Water Bay, Hong Kong, China
[3] *School of Computer and Communication Sciences, École Polytechnique Fédérale de Lausanne*
Rte Cantonale, 1015 Lausanne, Switzerland
[4] *Siebel School of Computing and Data Science, University of Illinois Urbana-Champaign*
201 N Goodwin Ave, Urbana, IL 61801, USA
† *Most of the work was done when Chen Liu was with École Polytechnique Fédérale de Lausanne.*
‡ *The work was done when Tong Zhang was with Hong Kong University of Science and Technology.*
∗ *Corresponding author*

Editor: Pradeep Ravikumar

Abstract

Adversarial training is a popular method to robustify models against adversarial attacks. However, it exhibits much more severe overfitting than training on clean inputs. In this work, we investigate this phenomenon from the perspective of training instances, i.e., training input-target pairs. Based on a quantitative metric measuring the relative difficulty of an instance in the training set, we analyze the model's behavior on training instances of different difficulty levels. This lets us demonstrate that the decay in generalization performance of adversarial training is a result of fitting hard adversarial instances. We theoretically verify our observations for both linear and general nonlinear models, proving that models trained on hard instances have worse generalization performance than ones trained on easy instances, and that this generalization gap increases with the size of the adversarial budget. Finally, we investigate solutions to mitigate adversarial overfitting in several scenarios, including fast adversarial training and fine-tuning a pretrained model with additional data. Our results demonstrate that using training data adaptively improves the model's robustness.

Keywords: Robustness, overfitting, adversarial training, deep learning, optimization.

1. Introduction

The existence of adversarial examples (Szegedy et al., 2014) causes serious safety concerns when deploying modern deep learning models. For example, for classification tasks, imperceptible perturbations of the input instance can fool state-of-the-art classifiers. Many strategies to obtain models that are robust against adversarial attacks have been proposed (Buckman et al., 2018; Dhillon et al., 2018; Ma et al., 2018; Samangouei et al., 2018; Pang et al., 2019, 2020; Xiao et al., 2020), but most of them have been found to be ineffective in the presence of adaptive attacks (Athalye et al., 2018; Croce and Hein, 2020b; Tramer et al., 2020; Croce and Hein, 2021). Ultimately, this leaves adversarial training (Madry et al., 2018) and its variants (Alayrac et al., 2019; Carmon et al., 2019; Hendrycks et al., 2019; Kumari et al., 2019; Zhang et al., 2019a; Gowal et al., 2020; Wu et al., 2020; Gowal et al., 2021; Jiang et al., 2023; Wang et al., 2023; Cui, 2024; Zhong et al., 2024) as the most effective and popular approaches to construct robust models. Unfortunately, adversarial training yields much worse performance on the test data than vanilla training. In particular, it strongly suffers from overfitting (Rice et al., 2020), with the model's performance decaying significantly on the test set in the later phase of adversarial training. Because modern deep neural networks have sufficient capacity to fit the training data perfectly, even under adversarial attacks, overfitting remains one of the primary challenges for improving model robustness on the test data. While the overfitting issue can be mitigated by early stopping (Rice et al., 2020) or model smoothing (Chen et al., 2021b), the reason behind the overfitting of adversarial training remains poorly understood.

In this paper, we study this phenomenon from the perspective of training instances, i.e., training input-target pairs. We first introduce a quantitative metric, based on the percentile of the instance's loss objective, to measure the relative difficulty of an instance within a training set. Then, we analyze the model's behavior, such as its loss and intermediate activations, on training instances of different difficulty levels. This lets us discover that the model's generalization performance decays significantly when it fits the hard adversarial instances in the later training phase.

To more rigorously study this phenomenon, we conduct theoretical analyses on both linear and nonlinear models. For linear models, we study logistic regression on a Gaussian mixture model, in which we can calculate the analytical expression of the model parameters upon convergence and thus the robust test accuracy. Our theorem demonstrates that adversarial training on harder instances leads to larger generalization gaps. Furthermore, the difference in robust accuracy between the models trained by the hard instances and the ones trained by the easy instances increases with the size of the adversarial budget. In the case of nonlinear models, we derive the lower bound of the model's Lipschitz constant when the model is well fit to the training instances under adversarial attacks. This bound increases with the difficulty level of the training instances and the size of the adversarial budget. Since a larger Lipschitz constant indicates a higher adversarial vulnerability (Ruan et al., 2018; Weng et al., 2018a,b), our theoretical analysis confirms our empirical observations.

Our empirical and theoretical analyses indicate that avoiding fitting the hard training instances can mitigate adversarial overfitting. We therefore study this in three different scenarios: standard adversarial training, fast adversarial training and adversarial fine-tuning with additional training data. We show that existing approaches that successfully mitigate adversarial overfitting (Balaji et al., 2019; Chen et al., 2021b; Huang et al., 2020) implicitly avoid fitting the hard adversarial input-target pairs, by either adaptive inputs or adaptive targets. By contrast, the methods that focus on fitting hard adversarial (Zhang et al., 2021) instances are not truly robust under adaptive attacks (Hitaj et al., 2021).

Contributions. Our contributions are as follows: 1) Based on a quantitative metric of instance difficulty, we show that fitting hard adversarial instances leads to degraded generalization performance in adversarial training. 2) We conduct rigorous theoretical analyses on both linear and nonlinear models. For linear models, we show analytically that models trained on harder instances have larger robust test error than the ones trained on easy instances; the gap increases with the size of the adversarial budget. For nonlinear models, we derive a lower bound of the model's Lipschitz constant. It increases with the difficulty of the training instances and the size of the adversarial budget, indicating that both factors exacerbate adversarial overfitting. 3) We show that existing approaches to mitigating adversarial overfiting implicitly avoid fitting hard adversarial instances.

Notation and terminology. In this paper, $\boldsymbol{x}$ and $\boldsymbol{x}'$ are the clean input and its adversarial counterpart. We use $f_{\boldsymbol{w}}$ to represent a model parameterized by $\boldsymbol{w}$ and omit the subscript $\boldsymbol{w}$ unless ambiguous. $\boldsymbol{o} = f_{\boldsymbol{w}}(\boldsymbol{x})$ and $\boldsymbol{o}' = f_{\boldsymbol{w}}(\boldsymbol{x}')$ are the model's output of the clean input and the adversarial input. $\mathcal{L}_{\boldsymbol{w}}(\boldsymbol{x}, \boldsymbol{y})$ and $\mathcal{L}_{\boldsymbol{w}}(\boldsymbol{x}', \boldsymbol{y})$ represent the loss of the clean and adversarial instances, receptively, in which we sometimes omit $\boldsymbol{w}$ and $\boldsymbol{y}$ for notation simplicity. We use $\|\boldsymbol{w}\|$ and $\|\mathbf{X}\|$ to represent the l_2 norm of the vector $\boldsymbol{w}$ and the spectral norm of the matrix $\mathbf{X}$, respectively. *sign* is an elementwise function which returns $+1$ for positive elements, -1 for negative elements and 0 for 0. $\mathbf{1}_y$ is the one-hot vector with only the y-th dimension being 1. The term *adversarial budget* refers to the allowable perturbations applied to the input instance. It is characterized by l_p norm and the size ϵ as a set $\mathcal{S}^{(p)}(\epsilon) = \{\Delta | \|\Delta\|_p \leq \epsilon\}$. A notation table is provided in Appendix A.

Based on the notations above, given the training set $\mathcal{D}$, the robust learning problem can be formulated as the following min-max optimization problem. Unless explicitly stated, we usually omit y in the loss function for notation simplicity.

$$\min_{\boldsymbol{w}} \mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{D}} \max_{\Delta \in \mathcal{S}^{(p)}(\epsilon)} \mathcal{L}_{\boldsymbol{w}}(\boldsymbol{x} + \Delta, y) \tag{1}$$

In this paper, *vanilla training* refers to training on the clean inputs, and *vanilla adversarial training* to the adversarial training method in Madry et al. (2018). *RN18* and *WRN34* are the 18-layer ResNet (He et al., 2016) and the 34-layer WideResNet (Zagoruyko and Komodakis, 2016) with the width factor 10 used in Madry et al. (2018) and Wong et al. (2020), respectively. To avoid confusion with the general term *overfitting*, which refers to the gap between the training error and the test error, we use the term *adversarial overfitting* to indicate the phenomenon where the robust error on the test set significantly increases in the late phase of training. Adversarial overfitting often results in a significant generalization gap, because the model's robust error on the training set decreases during training, an increase in robust test error indicates that the model is not effectively generalizing to new data.

The code to reproduce the results of this paper is publicly available on Github[1].

1. https://github.com/IVRL/RobustOverfit-HardInstance.git

2. Related Work

We concentrate on white-box attacks, where the attacker has access to the model parameters. Such attacks are usually based on first-order information and stronger than black-box attacks (Andriushchenko et al., 2020; Dong et al., 2018). For example, the *fast gradient sign method (FGSM)* (Goodfellow et al., 2014) perturbs the input based on its gradient's sign. The *iterative fast gradient sign method (IFGSM)* (Kurakin et al., 2016) iteratively runs FGSM using a smaller step size and projects the perturbation to the adversarial budget after each iteration. On top of IFGSM, *projected gradient descent (PGD)* (Madry et al., 2018) uses random initialization and restarts to boost the strength of the attack.

It is challenging to defend models against adversarial examples. Some early defense methods (Pang et al., 2019, 2020; Xiao et al., 2020) are shown to utilize obfuscated gradients (Athalye et al., 2018), which means they can only tackle some specific types of attacks instead of achieving true robustness. Models trained by these methods are vulnerable to stronger adaptive attacks Athalye et al. (2018); Croce and Hein (2020b); Tramer et al. (2020); Croce and Hein (2021). In contrast, several works have designed training algorithms to obtain *provably* robust models (Raghunathan et al., 2018; Wong and Kolter, 2018; Cohen et al., 2019; Gowal et al., 2019; Salman et al., 2019). Unfortunately, these methods either do not generalize to modern network architectures or have a prohibitively large computational complexity. As a consequence, adversarial training (Madry et al., 2018) and its variants (Alayrac et al., 2019; Carmon et al., 2019; Hendrycks et al., 2019; Kumari et al., 2019; Zhang et al., 2019a; Gowal et al., 2020; Wu et al., 2020; Gowal et al., 2021; Jiang et al., 2023; Wang et al., 2023; Cui, 2024; Zhong et al., 2024) have become the de facto approach to obtain robust models in practice. In essence, these methods generate adversarial examples, usually using PGD, and use them to optimize the model parameters.

While effective, adversarial training is more challenging than vanilla training. It was shown to require larger models (Xie and Yuille, 2020) and to exhibit a poorer convergence behavior (Liu et al., 2020). Furthermore, as observed in Rice et al. (2020), it suffers from *adversarial overfitting*: the robust accuracy on the test set significantly decreases in the late adversarial training phase. Rice et al. (2020) thus proposed to perform early stopping based on a separate validation set to improve the generalization performance in adversarial training. Furthermore, Chen et al. (2021b) introduced logit smoothing and weight smoothing strategies to reduce adversarial overfitting. In parallel to this, several techniques to improve the model's robust test accuracy were proposed (Wang et al., 2020; Wu et al., 2020; Zhang et al., 2021), but without solving the adversarial overfitting issue. By contrast, other works (Balaji et al., 2019; Huang et al., 2020) were empirically shown to mitigate adversarial overfitting but without providing any explanations as to how this phenomenon was addressed.

In addition to adversarial training, there are some previous works studying the training dynamics and generalization properties of vanilla training (Neyshabur et al., 2017; Zhang et al., 2017; Toneva et al., 2018; Swayamdipta et al., 2020). Unlike adversarial training, models usually have pretty good generalization performance (Bartlett et al., 2020; Li et al., 2021; Kou et al., 2023) despite over-parameterization, which are usually the cases of deep neural networks. This phenomenon is called *benign overfitting.* There are some works connecting benign overfitting with adversarial robustness. Bubeck and Sellke (2021) theoretically proves that at least $\Omega(nm)$ trainable parameters are needed for interpolating n m-dimensional instances. Sanyal et al. (2020) studies the overparameterization regime in the context of label noise, and demonstrates that label noise in the training data dramatically hurts adversarial robustness.

In this paper, we study the causes of adversarial overfitting from both an empirical and a theoretical point of view. We address how adversarial perturbations affect the generalization properties of deep neural networks. We also identify the reasons why prior attempts (Balaji et al., 2019; Chen et al., 2021a; Huang et al., 2020) successfully mitigate it.

3. A Metric for Instance Difficulty

Parametric models are trained to minimize a loss objective based on several input-target pairs called training set, and are then evaluated on a held-out set called test set. By comparing the loss value of each instance, we can understand which ones, in either the training or the test set, are more difficult for the model to fit. Therefore, our metric for instance difficulty is based on an instance's loss during the training process.

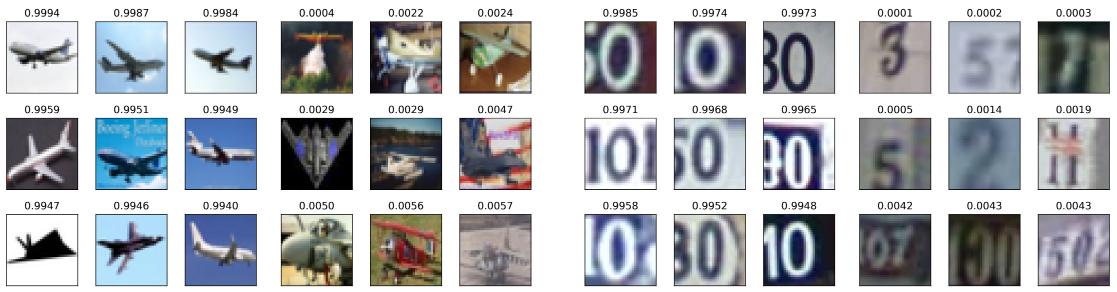
To this end, considering that we train the model for M epochs, we use $\{\boldsymbol{w}_i\}_{i=1}^M$ to represent the model parameters in each epoch. In addition, we introduce the perturbation algorithm $\mathcal{A}$ and use $\mathcal{A}(\boldsymbol{x}, \boldsymbol{w})$ to denote the adversarial examples of the input $\boldsymbol{x}$ given the model parameters $\boldsymbol{w}$. In vanilla training, $\mathcal{A}_{clean}$ does not perturb the input, i.e., $\mathcal{A}_{clean}(\boldsymbol{x}, \boldsymbol{w}) = \boldsymbol{x}$; in adversarial training in (Madry et al., 2018), $\mathcal{A}_{PGD}(\boldsymbol{x}, \boldsymbol{w})$ is the adversarial example of $\boldsymbol{x}$ generated by PGD. Under this notation, the average loss $\overline{\mathcal{L}}$ is calculated as $\overline{\mathcal{L}}(\boldsymbol{x}, \mathcal{A}) := \frac{1}{M}\sum_{i=1}^M \mathcal{L}_{\boldsymbol{w}_i}(\mathcal{A}(\boldsymbol{x}, \boldsymbol{w}_i), y)$, where the loss function $\mathcal{L}$ is defined in Equation (1). We then study the relative difficulty level of an instance within a finite set, and define the difficulty function d of an instance $\boldsymbol{x}$ within a set $\mathcal{D}$ for the perturbation algorithm $\mathcal{A}$ as

$$d(\boldsymbol{x}, \mathcal{A}) = \mathbb{P}(\overline{\mathcal{L}}(\boldsymbol{x}, \mathcal{A}) > \overline{\mathcal{L}}(\widetilde{\boldsymbol{x}}, \mathcal{A}) | \widetilde{\boldsymbol{x}} \sim U(\mathcal{D})) + \frac{1}{2}\mathbb{P}(\overline{\mathcal{L}}(\boldsymbol{x}, \mathcal{A}) = \overline{\mathcal{L}}(\widetilde{\boldsymbol{x}}, \mathcal{A}) | \widetilde{\boldsymbol{x}} \sim U(\mathcal{D})) , \tag{2}$$

where $\widetilde{\boldsymbol{x}} \sim U(\mathcal{D})$ indicates that $\widetilde{\boldsymbol{x}}$ is uniformly sampled from the finite set $\mathcal{D}$. $d(\boldsymbol{x}, \mathcal{A})$ is defined based on the model, the attack algorithm $\mathcal{A}$ and the set $\mathcal{D}$. Since $d(\boldsymbol{x}, \mathcal{A})$ denotes the relative difficulty, it is a bounded function, close to 1 for the hardest instances and close to 0 for the easiest ones.

We discuss the motivation for and properties of $d(\boldsymbol{x}, \mathcal{A})$ in Appendix D.1. In particular, in Appendix D.1, we demonstrate that the difficulty function d mainly depends on the original data $\boldsymbol{x}$ and the perturbation algorithm $\mathcal{A}$; the model architecture and the training duration have negligible effects on d. Therefore, we use $\boldsymbol{x}$ and $\mathcal{A}$ as the parameters of the function d, and omit the others for notation simplicity. In other words, $d(\boldsymbol{x}, \mathcal{A})$ can represent the difficulty of $\boldsymbol{x}$ within a set under a specific type of attack $\mathcal{A}$.

We show some of the easiest and hardest examples according to our metric in adversarial training in Figure 1, which indicates that our metric aligns well with human perception. The easiest instances are visually highly similar, with consistent and typical features of the corresponding category. By contrast, the hardest ones are much more diverse and with non-typical visual features. Some of them are ambiguous or even incorrectly labeled.



(a) Easy@CIFAR10. (b) Hard@CIFAR10. (c) Easy@SVHN. (d) Hard@SVHN.

Figure 1: Some examples of the easiest and the hardest instances in CIFAR10 (Krizhevsky et al., 2009) and SVHN (Netzer et al., 2011) datasets. We pick some examples from the "plane" category in CIFAR10 and "0" category in SVHN. The number on top of each image indicates the corresponding value of the difficulty function

In the remainder of this paper, we use the difficulty metric as defined by Equation (2), which not only aligns well with human perception but also is straightforward, easy to obtain, and facilitates our theoretical analysis. Although other instance difficulty metrics have been proposed, such as the ones in Baldock et al. (2021); Paul et al. (2021) based on margins to the decision boundary, comparing them with our metric is subjective and out of the scope of this work. We focus on using the difficulty metric as a tool to analyze the adversarial overfitting phenomenon. In the following sections, we study how easy and hard training instances affect adversarial overfitting.

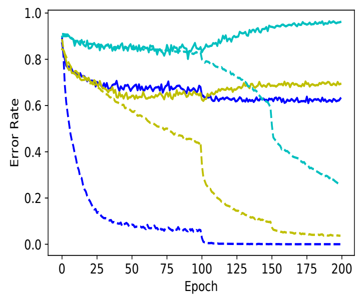
4. Hard Instances Lead to Overfitting

We empirically study how easy and hard instances impact the performance of adversarial training, with a focus on the adversarial overfitting phenomenon. Unless otherwise mentioned, we use the general experimental settings in Appendix C.1.

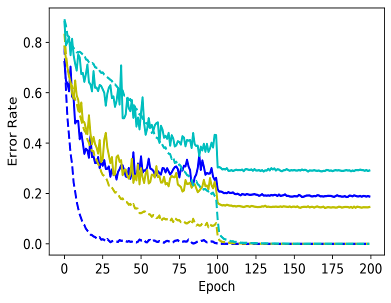
4.1 Using a Subset of Training Data

We start by training RN18 models for 200 epochs using either the 10000 easiest, random or hardest instances of the CIFAR10 training set via either vanilla training, FGSM or PGD adversarial training. For FGSM and PGD adversarial training, the adversarial budget is based on the l_∞ norm and $\epsilon = 8/255$. Note that the instance's difficulty is defined based on Equation (2) with the same perturbations as in training. The perturbations of vanilla training are considered to be zero. In addition, we enforce the training subsets to be class-balanced. For example, the easiest 10000 instances consist of the easiest 1000 instances in each class. We provide the learning curves under different perturbations in Figure 2.

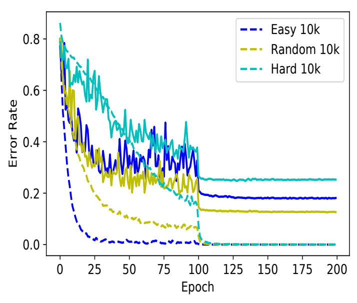
For PGD adversarial training, in Figure 2a, while we observe adversarial overfitting as in Rice et al. (2020) when using the random instances, no such phenomenon occurs when using the easiest instances: the performance on the test set does not degrade during training.



(a) PGD Adversarial Training.

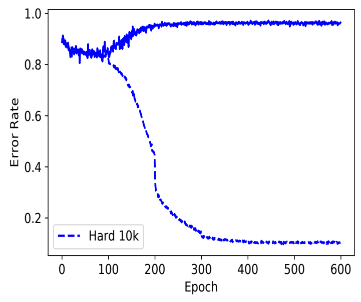


(b) FGSM Adversarial Training.

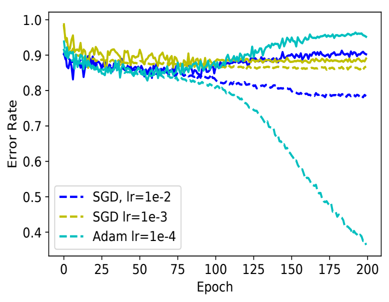


(c) Vanilla Training.

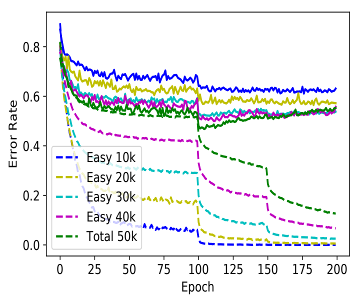
Figure 2: Learning curves obtained by training on the 10000 easiest, random and hardest instances of CIFAR10 under different scenarios. The training error (dashed lines) is the error on the selected instances, and the test error (solid lines) is the error on the whole test set. The y-axis of each subfigure indicates the training or test error under the corresponding perturbation, so the error rates of different subfigures are not comparable.



(a) Longer Duration.



(b) Different Optimizers.



(c) Hard Instance Removal.

Figure 3: (a) The training error (dashed line) and the test error (solid line) when we conduct adversarial training on the 10000 hardest training instances for more epochs until convergence. (b) The learning curves of training on the 10000 hardest training instances when we use a different optimizer, including different learning rates and a different algorithm. (c) The learning curves on the training (dash lines) and the test (solid lines) sets when we remove the hardest training instances.

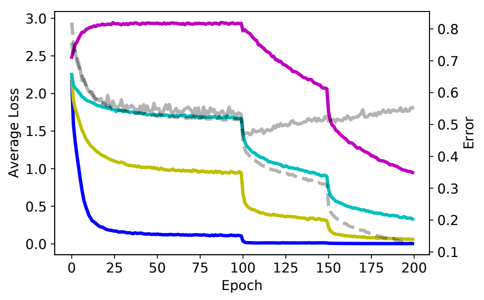
However, PGD adversarial training fails and suffers more severe overfitting when using the hardest instances. Note that this failure is not due to improper optimization. In Figure 3a and 3b, we use longer training duration and different optimizers to conduct PGD adversarial training on the hardest training instances, but the models' performance on the test set are always near trivial. All these phenomena indicate that the cause of overfitting is fitting the hard adversarial instances generated by PGD.

By contrast, FGSM adversarial training and vanilla training (Figure 2b, 2c) do not suffer from severe adversarial overfitting. In these cases, the models trained with the hardest instances also achieve non-trivial test accuracy. Furthermore, the gaps in robust test accuracy between the models trained by easy instances and by hard ones are much smaller. Since vanilla training can be considered as PGD adversarial training with $\epsilon = 0$, FGSM adversarial training does not yield truly robust models (Madry et al., 2018); the observations in Figure 2 indicate that adversarial overfitting happens when we aim to obtain models robust against an adversarial budget of a large size ϵ.

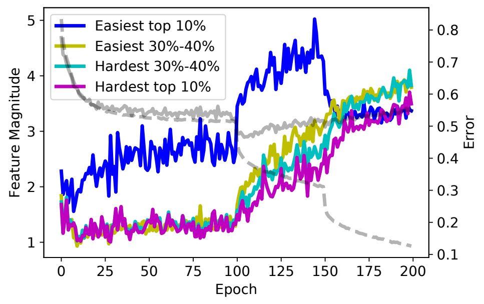
In Appendix D.3, we perform additional and comprehensive experiments, evidencing that our conclusions hold for various difficulty metrics, datasets and values of ϵ, and for an adversarial budget based on the l_2 norm. Specifically, we show that more severe adversarial overfitting happens when the size of the adversarial budget ϵ increases. That is to say, in term of model generalization, fitting hard training instances is more harmful when we are training against stronger perturbations.

Despite harmful, the experiments discussed below show that simply removing hard instances is not the optimal choice. In Figure 3c, we conduct PGD adversarial training using increasingly more training instances, starting with the easiest ones. The learning curves on the test set indicate that the models can still benefit from more data, but only when combined with early stopping used in (Rice et al., 2020). It means that the hard instances can still benefit adversarial training, but need to be utilized in an adaptive manner.

4.2 Using the Whole Training Set



(a)



(b)

Figure 4: Analysis on the groups $\mathcal{G}_0$, $\mathcal{G}_3$, $\mathcal{G}_6$ and $\mathcal{G}_9$ in the training set. The right vertical axis corresponds to the training (dashed grey line) and test (solid grey line) error under adversarial attacks for both plots. **Left plot:** The left vertical axis represents the average loss of different groups. **Right plot:** The left vertical axis represents the average l_2 norm of features extracted during training for different groups.

Let us now turn to the more standard setting where we train the model with the entire training set. To nonetheless analyze the influence of instance difficulty in this scenario, we divide the training set $\mathcal{D}$ into 10 non-overlapping groups $\{\mathcal{G}_i\}_{i=0}^{9}$, with $\mathcal{G}_i = \{\boldsymbol{x} \in \mathcal{D} | 0.1 \times i \leq d(\boldsymbol{x}, \mathcal{A}_{PGD}) < 0.1 \times (i+1)\}$, where $d(\boldsymbol{x}, \mathcal{A}_{PGD})$ is the difficulty of $\boldsymbol{x}$ based on PGD attacks. That is, $\mathcal{G}_0$ is the easiest group, whereas $\mathcal{G}_9$ is the hardest one. We then train a RN18 model on the entire CIFAR10 training set by PGD adversarial training and monitor the training behavior of the different groups. In particular, in Figure 4a, we plot the average loss of the instances in the groups $\mathcal{G}_0$, $\mathcal{G}_3$, $\mathcal{G}_6$ and $\mathcal{G}_9$. The results show that, in the early training stages, the model first fits the easy instances, as evidenced by the average loss of group $\mathcal{G}_0$ decreasing much faster than that of the other groups. By contrast, in the late training phase, the model tries to fit the more difficult instances, with the average loss of groups $\mathcal{G}_9$ and $\mathcal{G}_6$ decreasing much faster than that of the other groups. In this period, however, the robust test error (solid grey line) increases, which indicates that adversarial overfitting arises from the model's attempt to fit the hard adversarial instances.

In addition to average losses, inspired by Ilyas et al. (2019), which showed that the penultimate layer's activations of a robust model correspond to its *robust features* that cannot be misaligned by adversarial attacks, we monitor the group-wise average magnitudes of the penultimate layer's activations. As shown in Figure 4b, the model first focuses on extracting robust features for the easy instances, as evidenced by the comparatively large activations of the instances in $\mathcal{G}_0$. In the late phase of training, the norm of the activations of the hard instances increases significantly, bridging the gap between easy and hard instances. This further indicates that the model focuses more on the hard instances in the later phase, at which point it starts overfitting.

5. Theoretical Analysis

The empirical study in Section 4 indicates that adversarial overfitting arises from fitting hard adversarial training instances. We now study this relationship from a theoretical viewpoint. We start with a linear model: the logistic regression model on a Gaussian Mixture Model. In this toy example, the adversarial examples and the corresponding loss function have analytical expressions, allowing us to precisely draw the relationship between the instance difficulty and the generalization performance. We then generalize our analysis to general nonlinear models and use the models' Lipschitz constant as a proxy for their robustness on the test set. Our conclusions are consistent with the empirical study.

We use $\{\boldsymbol{x}_i, y_i\}_{i=1}^n$ to represent the training data, and $(\mathbf{X}, \boldsymbol{y})$ as its matrix form. $\{\boldsymbol{x}'_i, y_i\}_{i=1}^n$ and $(\mathbf{X}', \boldsymbol{y})$ are their adversarial counterparts. Here, $\boldsymbol{x}_i \in \mathbb{R}^m$, $y_i \in \{-1, +1\}$, $\mathbf{X} \in \mathbb{R}^{n \times m}$ and $\boldsymbol{y} \in \{-1, +1\}^n$. Note that these adversarial examples are generated based on the model parameters $\boldsymbol{w}$ to maximize the loss objective, so they depend on the model parameters $\boldsymbol{w}$ and are generated on the fly during training, which is consistent with adversarial training in practice. For simplicity, we do not explicitly represent this dependence in the notation.

The notation is summarized in Table 4 of Appendix A.

5.1 Linear Models

We study the logistic regression model under an l_2 norm based adversarial budget. In this case, the model is parameterized by $\boldsymbol{w} \in \mathbb{R}^m$ and outputs $sign(\boldsymbol{w}^T \boldsymbol{x}'_i)$ given the adversarial example $\boldsymbol{x}'_i$ of the input $\boldsymbol{x}_i$. The loss function for this instance is $\frac{1}{1+e^{y_i \boldsymbol{w}^T \boldsymbol{x}'_i}}$. We assume over-parameterization, which means $n < m$.

The following theorem shows that, under mild assumptions, the parameters of the adversarially trained model converge to the l_2 max-margin direction of the training data.

Theorem 1. *For a dataset $\{\boldsymbol{x}_i, y_i\}_{i=1}^n$ that is linearly separable under the adversarial budget $\mathcal{S}^{(2)}(\epsilon)$, any initial point $\boldsymbol{w}_0$ and step size $\alpha \leq 2\|\mathbf{X}\|^{-2}$, the gradient descent $\boldsymbol{w}_{u+1} = \boldsymbol{w}_u - \alpha \nabla_{\boldsymbol{w}} \mathcal{L}_{\boldsymbol{w}_u}(\mathbf{X}')$ converges asymptotically to the l_2 max-margin vector of the training data. That is,*

$$\lim_{u\to\infty} \frac{\boldsymbol{w}_u}{\|\boldsymbol{w}_u\|} = \frac{\widehat{\boldsymbol{w}}}{\|\widehat{\boldsymbol{w}}\|}, \text{ where } \widehat{\boldsymbol{w}} = \arg\min_{\boldsymbol{w}} \|\boldsymbol{w}\| \quad s.t. \ \forall i \in \{1, 2, ..., n\}, \ \boldsymbol{w}^T \boldsymbol{x}_i \geq 1 \ . \tag{3}$$

The proof is in Appendix B.1. Theorem 1 extends the conclusion in Soudry et al. (2018), which only studies the non-adversarial case. It also indicates that the optimal parameters are only determined by the support vectors of the training data, which are the ones with the smallest margin. According to the loss function, the smallest margin means the largest loss values and thus the hardest training instances based on our definition in Section 3.

To further study how the training instances' difficulty influences the model's generalization performance, we assume that the data points are drawn from a K-mode Gaussian mixture model (GMM). Specifically, the k-th component has a probability p_k of being sampled and is formulated as:

$$\boldsymbol{x}_i \sim \mathcal{N}(y_i r_k \boldsymbol{\eta}, \mathbf{I}) \tag{4}$$

Here, $\boldsymbol{\eta} \in \mathbb{R}^m$ is the unit vector indicating the direction of the mean for each mode, and $r_k \in \mathbb{R}^+$ controls the average distance between the positive and negative instances. The mean values of all modes in this GMM are colinear, so r_k indicates the difficulty of instances sampled from the k-th component. In Appendix D.2, we demonstrate the strong correlation of r_k and the difficulty defined in Section 3.

Without the loss of generality, we assume that $r_1 < r_2 < ... < r_{K-1} < r_K$. Same as in Section 4.1, we consider models trained with the subsets of the training data, e.g., n instances from the l-th component. $l = 1$ then indicates training on the hardest examples, while $l = K$ means using the easiest. In matrix form, we have $\mathbf{X} = r_l \boldsymbol{y}\boldsymbol{\eta}^T + \mathbf{Q}$ for the instances sampled from the l-th component, where the rows of noise matrix $\mathbf{Q}$ are sampled from $\mathcal{N}(\mathbf{0}, \mathbf{I})$.

Although the max-margin direction in Equation (3), where the parameters converge based on Theorem 1, does not have an analytical expression, the results in Wang and Thrampoulidis (2020) indicate that, in the over-parameterization regime and when the training data is sampled from a GMM, the max-margin direction is the min-norm interpolation of the data with high probability. Since the latter has an analytical form given by $\mathbf{X}^T(\mathbf{X}\mathbf{X}^T)^{-1}\boldsymbol{y}$, we can then calculate the exact generalization performance of the trained model as stated in the following theorem.

Theorem 2. *If a logistic regression model is adversarially trained on n separable training instances sampled from the l-th component of the GMM described in (4), $\{p_k\}_{k=1}^K$ are the probabilities of sampling from the k-th component of the GMM; when $\frac{m}{n \log n}$ is sufficiently large[2], then with probability $1 - O(\frac{1}{n})$, the expected adversarial test error $\mathcal{R}$ under the adversarial budget $\mathcal{S}^{(2)}(\epsilon)$, which is a function of r_l and ϵ, on the whole GMM described in*

2. Specifically, m and n need to satisfy $m > 10n \log n + n - 1$ and $m > Cnr_l\sqrt{\log 2n}\|\boldsymbol{\eta}\|$. The constant C is derived in the proof of Theorem 1 in Wang and Thrampoulidis (2020).

(4) is given by

$$\begin{aligned}\mathcal{R}(r_l, \epsilon) &= \sum_{k=1}^{K} p_k \Phi\left(r_k g(r_l) - \epsilon\right) \\ \text{where } g(r_l) &= (C_1 - \frac{1}{C_2 r_l^2 + o(r_l^2)})^{\frac{1}{2}},\ C_1, C_2 \geq 0.\end{aligned} \tag{5}$$

C_1, C_2 are non-negative numbers independent of ϵ and r_l. The function Φ is defined as $\Phi(x) = \mathbb{P}(Z > x),\ Z \sim \mathcal{N}(0, 1)$.

We defer the proof of Theorem 2 to Appendix B.2, in which we calculate the *exact* expression of $\mathcal{R}(r_l, \epsilon)$, C_1, C_2, and show that C_1, C_2 are positive numbers almost surely. Since C_1 and C_2 are independent of r_l, and $\Phi(x)$ is a monotonically decreasing function, we conclude that the robust test error $\mathcal{R}(r_l, \epsilon)$ becomes smaller when r_l increases. Since the training set is separable, our results indicate that when the training instances become easier, the corresponding generalization error under adversarial attack becomes smaller.

Theorem 2 holds for any ϵ as long as the training data is separable under the corresponding adversarial budget. The following corollary shows that the difference in the robust test error between models trained with easy instances and the ones with hard ones increases when ϵ becomes larger, i.e., under a larger adversarial budget.

Corollary 3. *Under the conditions of Theorem 2 and the definition of $\mathcal{R}$ in Equation (5), if $\epsilon_1 < \epsilon_2$, then we have $\forall\ 0 \leq i < j \leq K, \mathcal{R}(r_i, \epsilon_1) - \mathcal{R}(r_j, \epsilon_1) < \mathcal{R}(r_i, \epsilon_2) - \mathcal{R}(r_j, \epsilon_2)$.*

The proof is in Appendix B.3. $\mathcal{R}(r_i, \epsilon) - \mathcal{R}(r_j, \epsilon)$ is the gap in robust accuracy between the models trained on the easy instances and the ones on the hard instances under the adversarial budget $\mathcal{S}^{(2)}(\epsilon)$. Corollary 3 shows that such a gap increases with the size of the adversarial budget. This indicates that, compared with training on the clean inputs, i.e., $\epsilon = 0$, the generalization performance of adversarial training, i.e., $\epsilon > 0$, is more sensitive to the difficulty of the training instances. Furthermore, overfitting in adversarial training becomes increasingly severe as ϵ becomes larger. This is consistent with our empirical observations in Figures 2, 13, 14.

5.2 General Nonlinear Models

In this section, we study the binary classification problem using a general nonlinear model. We consider a model with b parameters, i.e., $\boldsymbol{w} \in \mathbb{R}^b$. Without loss of generality, we assume the output of the function $f_{\boldsymbol{w}}$ to lies in $[-1, +1]$. Similarly to the K-mode Gaussian mixture model studied in the linear case, we assume the data distribution to be a composition of K sub-distributions. Furthermore, each of these distributions are isoperimetric.

Assumption 4. *The data distribution μ is a composition of K c-isoperimetric distributions on $\mathbb{R}^m$, each of which has a positive conditional variance. That is, $\mu = \sum_{k=1}^{K} \alpha_k \mu_k$, where $\alpha_k > 0$ and $\sum_{k=1}^{K} \alpha_k = 1$. We define $\sigma_k^2 = \mathbb{E}_{\mu_k}[Var[y|\boldsymbol{x}]]$, and without loss of generality assume that $\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_K > 0$. Furthermore, given any L-Lipschitz function $f_{\boldsymbol{w}}$,* *i.e.,* $\forall \boldsymbol{x}_1, \boldsymbol{x}_2, \|f_{\boldsymbol{w}}(\boldsymbol{x}_1) - f_{\boldsymbol{w}}(\boldsymbol{x}_2)\| \leq L\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|$, *we have the following inequality satisfied* $\forall k \in \{1, ..., K\}$

$$\mathbb{P}(\boldsymbol{x} \sim \mu_k, \|f_{\boldsymbol{w}}(\boldsymbol{x}) - \mathbb{E}_{\mu_k}(f_{\boldsymbol{w}})\| \geq t) \leq 2e^{-\frac{mt^2}{2cL^2}} \ . \tag{6}$$

This is a benign assumption; the data distribution is a mixture of K components and each of them contains samples from a sub-Gaussian distribution. These components correspond to training instances of different difficulty levels measured by the conditional variance. This is because the conditional variance σ_k^2 is the expected test error of a well-trained model (Bubeck and Sellke, 2021). Subsets with large σ_k^2 have higher loss and the difficulty function defined by the average training loss

We now study the properties of the model $f_{\boldsymbol{w}}$ under adversarial attacks.

Definition 5. *Given the dataset* $\{\boldsymbol{x}_i, y_i\}_{i=1}^n$, *the model* $f_{\boldsymbol{w}}$, *the adversarial budget* $\mathcal{S}^{(p)}(\epsilon)$ *and a positive constant* C, *we define the function* $h(C, \epsilon)$ *as:*

$$\begin{aligned} h(C,\epsilon) &= \min_{\boldsymbol{w} \in \mathcal{T}(C,\epsilon)} \min_i h_{i,\boldsymbol{w}}(\epsilon) \\ \text{where } \mathcal{T}(C,\epsilon) &= \left\{ \boldsymbol{w} \middle| \frac{1}{n}\sum_{i=1}^n (f_{\boldsymbol{w}}(\boldsymbol{x}_i') - y_i)^2 \leq C \right\} \ , \\ h_{i,\boldsymbol{w}}(\epsilon) &= \max \zeta, \ s.t. \ [f_{\boldsymbol{w}}(\boldsymbol{x}_i) - \zeta, f_{\boldsymbol{w}}(\boldsymbol{x}_i) + \zeta] \subset \left\{ f_{\boldsymbol{w}}(\boldsymbol{x}_i + \Delta) \middle| \Delta \in \mathcal{S}^{(p)}(\epsilon) \right\} . \end{aligned} \tag{7}$$

Here, $\boldsymbol{x}_i'$ *is the adversarial example of* $\boldsymbol{x}_i$. *We omit the superscript* (p) *for notation simplicity.*

By definition, $h_{i,\boldsymbol{w}}(\epsilon) \geq 0$ depicts the bandwidth ζ of the model's output range in the domain of the adversarial budget on a training instance. $\mathcal{T}(C, \epsilon)$ represents the set of well-trained models whose adversarial training loss is smaller than C. Therefore, $h(C, \epsilon)$ is the minimum bandwidth among such well-trained models. The following lemma demonstrates monotonicity properties of the function h.

Lemma 6. $\forall C, \epsilon_1 < \epsilon_2$, $h(C, \epsilon_1) \leq h(C, \epsilon_2)$; $\forall \epsilon, C_1 < C_2$, $h(C_1, \epsilon) \geq h(C_2, \epsilon)$.

Based on the definitions of $\mathcal{T}$ and $h_{i,\boldsymbol{w}}$, and for a fixed value of C, we have $\forall \epsilon_1 < \epsilon_2$, $h_{i,\boldsymbol{w}}(\epsilon_1) \leq h_{i,\boldsymbol{w}}(\epsilon_2)$ and $\mathcal{T}(C, \epsilon_2) \subset \mathcal{T}(C, \epsilon_1)$. As a result, $\forall \epsilon_1 < \epsilon_2$, $h(C, \epsilon_1) \leq h(C, \epsilon_2)$. In addition, since $\forall C_1 < C_2$, $\mathcal{T}(C_1, \epsilon) \subset \mathcal{T}(C_2, \epsilon)$ for a fixed value of ϵ, we have $\forall C_1 < C_2$, $h(C_1, \epsilon) \geq h(C_2, \epsilon)$. That is to say, $h(C, \epsilon)$ is a monotonically non-decreasing function on ϵ and a monotonically non-increasing function on C. In practice, when $f_{\boldsymbol{w}}$ represents a deep neural network, $h(C, \epsilon)$ increases with ϵ almost surely, because the attack algorithm usually generates adversarial examples at the boundary of the adversarial budget. Based on the monotonicity properties of h, We then state our main theorem below.

Theorem 7. *Given* n *training pairs* $\{\boldsymbol{x}_i, y_i\}_{i=1}^n$ *sampled from the* l*-th component* μ_l *of the distribution in Assumption 4, the parametric model* $f_{\boldsymbol{w}}$, *the adversarial budget* $\mathcal{S}^{(p)}(\epsilon)$ *and the corresponding function* h *defined in Definition 5, we assume that the model* $f_{\boldsymbol{w}}$ *is in the function space* $\mathcal{F} = \{f_{\boldsymbol{w}}, \boldsymbol{w} \in \mathcal{W}\}$ *with* $\mathcal{W} \subset \mathbb{R}^b$ *having a finite diameter* $diam(\mathcal{W}) \leq W$ *and,* $\forall \boldsymbol{w}_1, \boldsymbol{w}_2 \in \mathcal{W}, \|f_{\boldsymbol{w}_1} - f_{\boldsymbol{w}_2}\|_\infty \leq J\|\boldsymbol{w}_1 - \boldsymbol{w}_2\|_\infty$. *We train the model* $f_{\boldsymbol{w}}$ *adversarially* *using these n data points. Let $\boldsymbol{x}'_i$ be the adversarial example of the data point $\boldsymbol{x}_i$, i.e., $\boldsymbol{x}'_i = \arg\max_{\boldsymbol{x}_{adv}} (f_{\boldsymbol{w}}(\boldsymbol{x}_{adv}) - y_i)^2$ s.t. $\|\boldsymbol{x}_{adv} - \boldsymbol{x}_i\|_p \leq \epsilon$. $\forall \delta \in (0,1)$, if we have $\frac{1}{n}\sum_{i=1}^{n}(f_{\boldsymbol{w}}(\boldsymbol{x}'_i) - y_i)^2 = C$ and $\gamma := \sigma_l^2 + h^2(C,\epsilon) - C \geq 0$, then with probability at least $1-\delta$, the Lipschitz constant of $f_{\boldsymbol{w}}$ is lower bounded as*

$$Lip(f_{\boldsymbol{w}}) \geq \frac{\gamma}{2^7}\sqrt{\frac{nm}{c\left(b\log(4WJ\gamma^{-1}) - \log(\delta/2 - 2e^{-2^{-11}n\gamma^2})\right)}}\,, \tag{8}$$

$Lip(f_{\boldsymbol{w}})$ is the Lipschitz constant of $f_{\boldsymbol{w}}$: $\forall \boldsymbol{x}_1, \boldsymbol{x}_2$, $\|f_{\boldsymbol{w}}(\boldsymbol{x}_1) - f_{\boldsymbol{w}}(\boldsymbol{x}_2)\| \leq Lip(f_{\boldsymbol{w}})\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|$.

The proof is deferred to Appendix B.4. Theorem 7 extends the results in Bubeck and Sellke (2021) to the case of adversarial training. The Lipschitz constant is widely used to bound a model's adversarial vulnerability on the test set (Ruan et al., 2018; Weng et al., 2018a,b); larger Lipschitz constants indicate higher adversarial vulnerability on the test set. Note that modern deep neural network models typically have millions of parameters, so $b \gg \max\{c, m, n\}$. In this case, we can approximate the lower bound (8) by $Lip(f_{\boldsymbol{w}}) \gtrsim \frac{\gamma}{2^7}\sqrt{\frac{nm}{bc\log(4WJ\gamma^{-1})}}$, and the right hand side increases with γ.

Lemma 6 indicates that γ monotonically increases with the decrease of C, and Theorem 7 assumes $\gamma > 0$, so the conclusion of Theorem 7 is based on a sufficient small adversarial training loss C. That is to say, our theorem is applicable when the model is well fit to the adversarial training instances, i.e., small adversarial training loss, which is exactly when adversarial overfitting occurs. By contrast, there is usually no adversarial overfitting with large adversarial training loss when adversarial training does not or cannot fit the training set. For example, when ϵ is too large for adversarial training to converge, we will obtain a constant classifier as indicated in Liu et al. (2020). While the model has a high robust test error, the adversarial overfitting does not happen in this case.

Theorem 7 is applicable to any l_p norm based adversarial budget based on the definition of $h(C,\epsilon)$. Since $\gamma := \sigma_l^2 + h^2(C,\epsilon) - C$, we can conclude that the Lipschitz upper bound and thus the adversarial vulnerability on the test set is affected by three factors: it increases when σ_l , ϵ increase or C decreases. We elaborate the conclusion in the following paragraphs.

First, as the training processes, the adversarial training loss C becomes smaller, and correspondingly $h(C,\epsilon)$ becomes bigger based on Lemma 6. Therefore, $\gamma = \sigma_l^2 + h^2(C,\epsilon) - C$ increases and the condition $\gamma \geq 0$ will be satisfied in the late phase of adversarial training. In this context, as $\gamma \geq 0$ increases during this period, the Lipschitz lower bound also increases based on (8), indicating a higher adversarial test loss. In summary, in the final stages of training, which ensure that $\gamma \geq 0$, the training loss C decreases while the test loss increases. As a result, the generalization gap increases.

Second, with fixed C, i.e., the adversarial training loss is fixed, and the generalization gap is indicated by the adversarial test loss, represented by the Lipschitz lower bound in (8). When ϵ is fixed, the Lipschitz lower bound increases with the increase of σ_l. That is, under the same adversarial budget, the generalization gap increases with the instances' difficulty, measured by σ_l in our theorem. When σ_l is fixed, the Lipschitz lower bound increases with the increase of ϵ. Therefore, using the same training instances, the generalization gap increases with the size of the adversarial budget, measured by ϵ.

Finally, Theorem 7 discusses the case where the model is trained on samples from one components of the data distribution, i.e., a subset of the training set. This is exactly the case of Section 4.1. Furthermore, we can utilize Theorem 7 to analyze the cases when the model is trained on samples from the entire data distribution, which consists from K components. Similarly to the analysis in Section 4.2, we calculate the training loss $\{C_i\}_{i=1}^K$ for all K components. Correspondingly, we can define the function $h_i(C, \epsilon)$ same as in Definition 5 except that it is based on, instead of all training instances, the training instances sampled from the i-th component from the data distribution. Based on this, we define $\gamma_i := \sigma_i^2 + h_i^2(C_i, \epsilon) - C_i$ for $i \in \{1, 2, ..., K\}$. We can then utilize Theorem 7 for training samples from each distribution component and then obtain the lower bound of the model's Lipschitz constant. Formally, we have the following:

Corollary 8. *Given the assumptions of Theorem 7, except that the training data is sampled from all K components and contains n_i training instances from the i-th component, $\{C_i\}_{i=1}^K$, $\{h_i\}_{i=1}^K$, $\{\gamma_i\}_{i=1}^K$ defined for each components of the data distribution, then with probability at least $1-\delta$, the Lipschitz constant of $f_{\boldsymbol{w}}$ is lower bounded as*

$$Lip(f_{\boldsymbol{w}}) \geq \max \left\{ \frac{\gamma_i}{2^7} \sqrt{\frac{n_i m}{c\left(b \log(4WJ\gamma_i^{-1}) - \log(\delta/2 - 2e^{-2^{-11}n\gamma_i^2})\right)}} \middle| \gamma_i \geq 0 \right\} \tag{9}$$

Corollary 8 is straightforward from Theorem 7: We calculate the Lipschitz lower bound based on the adversarial training loss of each component as long as it is valid, i.e., $\gamma_i \geq 0$. The formal proof is provided in Appendix B.5. Corollary 8 indicates the Lipschitz lower bound of the model when it is trained on the whole training distribution consisting of instances of different difficulty levels. Similarly to the analysis of Theorem 8, the value of γ_i for each component of the data distribution increases as the training processes. That is to say, the size of the set $\{i|\gamma_i \geq 0\}$ increases during training, i.e., there are more and more numbers fed to the max operator in (9). In addition, the Lipschitz lower bound derived by the training instances from each components monotonically increases during training. Combining these two points together, we conclude that the Lipschitz lower bound provided by (9) monotonically increases during training, indicating more and more severe overfitting. As in Theorem 7, the Lipschitz lower bound also increases with the increase of ϵ, indicating that using a larger adversarial budget in adversarial training suffers more from overfitting.

In the early phase of adversarial training, the difference in the adversarial training loss for easy and hard instances is large. That is, the value of C_i dominates the calculation of γ_i. In this stage, the Lipschitz lower bound in (9) is dominated by the easy instances, because for hard instances sampled from the i-th component, C_i is huge and the corresponding γ_i does not satisfy the condition $\gamma_i \geq 0$. However, in the late phase of adversarial training, the adversarial training loss for all training instances is close to 0. As a result, the value of σ_i dominates the calculation of γ_i. In this stage, the Lipschitz lower bound in (9) is dominated by the hard instances, because $\forall i, C_i \simeq 0$, and γ_i increase with the increase of σ_i.

Corollary 8 explains the phenomena shown in Section 4.2 and confirms that fitting hard adversarial instances is harmful for the generalization performance.

5.3 Numerical Simulation

We conduct numerical simulation to confirm the validity of Theorem 7 in our settings. To this end, we use the CIFAR10 dataset and an RN18 network architecture. However, calculating the Lipschitz constant of a deep neural network is NP-hard (Scaman and Virmaux, 2018), exactly calculating the Lipschitz constant (Jordan and Dimakis, 2020) is so far infeasible for modern deep neural networks. Instead, we therefore estimate the upper bound of the Lipschitz constant numerically, as in (Scaman and Virmaux, 2018).

Value of ϵ	Lipschitz in l_∞ Cases ($\times 10^4$)		
	2/255	4/255	8/255
Easy10K	5.91 ± 0.00	6.06 ± 0.00	14.54 ± 0.02
Random10K	28.98 ± 0.03	79.96 ± 0.16	93.63 ± 0.34
Hard10K	72.42 ± 0.48	117.60 ± 2.18	567.24 ± 0.59
Value of ϵ	**Lipschitz in l_2 Cases ($\times 10^4$)**		
	0.50	0.75	1.00
Easy10K	3.34 ± 0.01	3.67 ± 0.00	3.91 ± 0.00
Random10K	30.01 ± 0.08	31.28 ± 0.04	39.34 ± 0.08
Hard10K	60.62 ± 0.07	80.06 ± 0.16	77.55 ± 0.61

Table 1: Upper bound of the Lipschitz constant under different settings of the adversarial budget when the model is adversarially trained for the easiest, random, or the hardest 10000 instances of the training set. The experiments are run 20 times, and we report both the mean and the standard deviation in the form of "mean ± standard deviation".

Table 1 provides the upper bound of the Lipschitz constant of models trained by different subsets of the training data and different adversarial budget. Due to the stochasticity introduced by the algorithm of Scaman and Virmaux (2018), we run it 20 times and report the average and standard deviation; we observed that the standard deviation is negligible compared with the magnitude of the mean value. Based on the results in Table 1, it is clear that the models adversarially trained on the hard training instances have a much larger Lipschitz constant than the ones trained on the easy instances.

Figure 5 depicts the curves of the Lipschitz upper bound when the model is adversarially trained by the easiest, random, the hardest 10000 instances or the whole training set. The adversarial budget is based on the l_∞ norm with $\epsilon = 8/255$. We can clearly see that, as training progresses, the Lipschitz upper bound increases in all cases. Furthermore, in the last phase of training, the Lipschitz estimation of the models adversarially trained on hard instances is bigger than the ones on easy instances. These results are consistent with Theorem 7. In addition, when we conduct adversarial training on the whole training set, the Lipschitz bound is close to the one trained on the easy instances in the early phase of training, while it is close to the one trained on hard instances in the late phase. This observations is consistent with what Corollary 8 indicates.

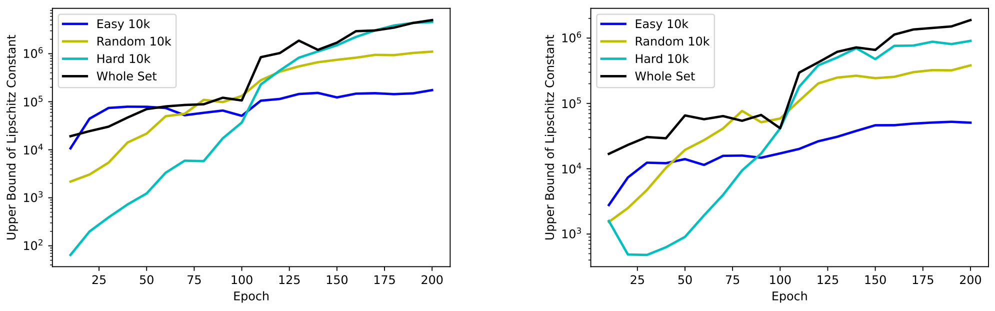


Figure 5: Curves of the Lipschitz upper bound when the model is adversarially trained by the easiest, random, the hardest 10000 instances or the whole training set. The y-axis is in log-scale. Left: the adversarial budget is based on the l_∞ norm with $\epsilon = 8/255$. Right: the adversarial budget is based on the l_2 norm with $\epsilon = 1$.

6. Case Study and Discussion

Our empirical and theoretical analyses indicate that fitting hard adversarial leads to adversarial overfitting. In this section, we first review existing approaches to mitigating overfitting in adversarial training. Specifically, we show that they implicitly avoid fitting hard adversarial instances, which provides an explanation for their success. We also show that the methods that encourage fitting hard adversarial instances fail to yield truly robust models.

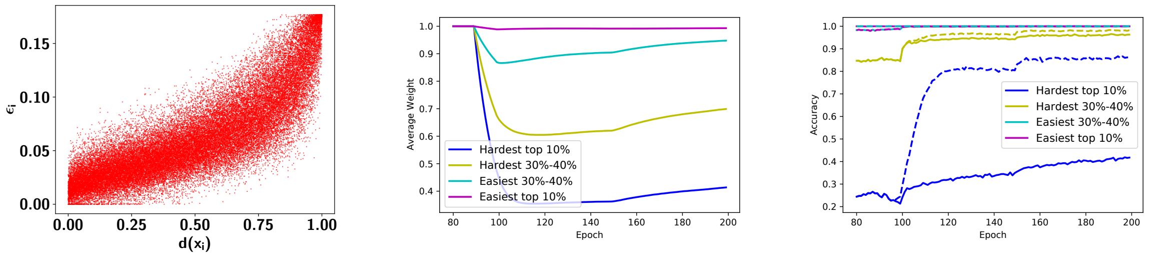
We believe that our discovery is broadly applicable to different tasks aiming to achieve adversarial robustness against a norm-based adversarial budget. In this regard, we study the cases of fast adversarial training and adversarial fine-tuning with additional training data. Our results indicate that avoiding to fit hard adversarial instances also improves the performance in these cases. More detailed discussions are deferred to Appendix D.4.

6.1 Standard Adversarial Training: A New Perspective on Existing Methods

Existing methods aiming to mitigate adversarial overfitting can be generally divided into two categories: those that use adaptive inputs, such as Balaji et al. (2019), and those that rely on adaptive targets, such as Chen et al. (2021b); Huang et al. (2020). We show below that both categories implicitly aim to prevent the model from fitting hard input-target pairs.

We use instance-wise adversarial training (IAT) (Balaji et al., 2019) and self-adaptive training (SAT) (Huang et al., 2020) as examples of these two categories. IAT uses an instance-adaptive adversarial budget during training. It adaptively adjusts ϵ, the size of the adversarial budget, for each training instance. SAT uses self-supervised adaptive targets instead of the ground truth during training. We run both algorithms using the settings in their original papers, except that we train the model for 200 epochs for a consistent comparison. The details are provided in Appendix D.4.

Let us study how these algorithms adaptively use instances of different difficulty levels. For IAT, we plot the relationship between the instance difficulty $d(\boldsymbol{x}_i)$ and its adaptive adversarial budget's size ϵ_i in Figure 6a, which shows a high correlation (0.884) between



(a) (b) (c)

Figure 6: Results of our case study. The model is always an RN18 and the target adversarial budget's size $\epsilon = 8/255$. (a) Relationship between instance difficulty $d(\boldsymbol{x}_i)$ and its adversarial budget size in IAT for the CIFAR10 training set. (b) Average weights of different groups in the CIFAR10 training set during training in SAT. The warmup period is 90 epochs, and SAT is enabled after that. (c) Training accuracy of different groups on the CIFAR10 training set during training in SAT. The solid lines and the dashed lines represent the accuracy on the ground truth and on the adaptive targets, respectively. The warmup period is 90 epochs, and SAT is enabled after that.

them. Specifically, we find that the hard instances are assigned smaller adversarial budgets for training, which indicates that IAT prevents the model from fitting the hard adversarial instances. For SAT, we show the average weights assigned to the instances in each group of $\{\mathcal{G}_i\}_{i=0}^{9}$ during training in Figure 6b. The hard instances are clearly assigned much smaller weights to calculate the loss, which indicates that they are downplayed during training. We also provide the average accuracy of each group during training in Figure 6c, given both the ground truth or the adaptive target.[3] We observe that the hard instances have much higher accuracy on their adaptive targets compared with the ground truth, while such a difference is much smaller for the easy instances. Our results thus indicate that the adaptive targets used by SAT are much easier to fit, which avoids having to directly fit the hard adversarial input-target pairs.

In addition to IAT and SAT, other methods have introduced regularization terms to mitigate adversarial overfitting, such as Zhang et al. (2019b) and Chen et al. (2021b). These regularization terms calculate the distance between the adversarial output logits and their anchor points. The anchor points are the adaptive targets, and can be the clean output logits in Zhang et al. (2019b) or a teacher network's outputs in Chen et al. (2021b). The regularizers used in these methods encourage the adversarial output logits to be closer to the anchor points other than to the ground truth for the hard instances. In other words, these methods also use adaptive targets to avoid fitting the hard input-target pairs.

In contrast to the methods above, Zhang et al. (2021) proposed an instance-adaptive reweighting strategy which assigns larger weights to the training instances that PGD breaks in fewer iterations. In other words, this approach assigns larger weights to the hard adversarial instances, which contrasts with what our analysis revealed. As a matter of fact, this method was recently shown to be vulnerable to adaptive attacks (Hitaj et al., 2021).

3. For the adaptive target $\boldsymbol{t}$, the prediction $\boldsymbol{o}$ is considered correct if and only if $\arg\max_i \boldsymbol{t}_i = \arg\max_i \boldsymbol{o}_i$.

6.2 Alternative Training Scenarios

We believe that our findings can be applied to improve the generalization performance of robust models in various situations. In this regard, we conduct preliminary analyses on two examples: fast adversarial training and fine-tuning a pre-trained model using additional data. In these examples, we show consistent observations with standard adversarial training. Our focus in this section is to showcase the general applicability of our findings rather than proposing entirely new algorithms. Our results below demonstrate that avoiding fitting hard adversarial instances can consistently mitigate overfitting and improve models' robustness in various scenarios.

6.2.1 Fast Adversarial Training

Adversarial training in Madry et al. (2018) introduces a significant computational overhead. Thus it is desirable to accelerate this method. This section studies how adaptive training based on the instances' difficulty mitigates overfitting and improves fast adversarial training. Specfically, our experiments in this section are based on adversarial training with transferable adversarial examples (ATTA in Zheng et al. (2020)), which stores the adversarial perturbation for each training instance as an initial point for the next epoch.

First, we use a reweighting scheme to assign lower weights to hard instances when calculating the loss objective: each training instance is assigned a weight equal to the adversarial output probability of the true label. Then this weight is normalized to ensure that the weights in a mini-batch sum to 1. Note that our reweighting scheme is based on the adversarial output instead of the clean output, because the adversarial output probability will also be used to calculate the loss objective. As a result, the computational overhead of the reweighting scheme is negligible.

In addition to reweighting, we adapt SAT (Huang et al., 2020) to fast adversarial training and quantitatively study how adaptive targets for hard adversarial training instances mitigate adversarial overfitting. For each training instance $(\boldsymbol{x}, y)$, we maintain an adaptive moving average target $\widetilde{\boldsymbol{t}}$. $\widetilde{\boldsymbol{t}}$ is updated in an exponential averaging manner for each epoch: $\widetilde{\boldsymbol{t}} \leftarrow \rho\widetilde{\boldsymbol{t}} + (1-\rho)\boldsymbol{o}'$ where ρ is the momentum factor and $\boldsymbol{o}'$ is the logit of the adversarial input $\boldsymbol{x}'$. Like the reweighting scheme, compared with Huang et al. (2020), we use the adversarial output $\boldsymbol{o}'$ instead of the clean output $\boldsymbol{o}$ to avoid computational overhead. The final adaptive target we use is $\boldsymbol{t} = \beta\mathbf{1}_y + (1-\beta)\widetilde{\boldsymbol{t}}$ and thus the loss objective is $\mathcal{L}_{\boldsymbol{w}}(\boldsymbol{x}', \boldsymbol{t})$. The factor β controls how "adaptive" our target is: $\beta = 0$ yields a fully adaptive moving average target $\widetilde{\boldsymbol{t}}$ and $\beta = 1$ yields a one-hot target $\mathbf{1}_y$. We provide the pseudocode as Algorithm 1.

Our experiment is on CIFAR10 and use l_∞ norm based adversarial budget with $\epsilon = 8/255$, the standard setting where most fast adversarial training algorithms are benchmarked Croce et al. (2020). Unless specified, we use the same settings as in Zheng et al. (2020). we train the model for 38 epochs, the learning rate is 0.1 on the first 30 epochs, it decays to 0.01 in the next 6 epochs and further decays to 0.001 in the last 2 epochs. We evaluate the model's robust accuracy on the test set by AutoAttack Croce and Hein (2020b), the popular and reliable attack for evaluation. More hyper-parameter details are deferred to Appendix C.2

Algorithm 1 One epoch of the accelerated adversarial training.

```
Input: training data $\mathcal{D}$, model $f$, batch size $B$, PGD step size $\alpha$, adversarial budget $\mathcal{S}^{(p)}(\epsilon)$, coefficient $\rho$, $\beta$.
for Sample a mini-batch $\{\boldsymbol{x}_i, y_i\}_{i=1}^B \sim \mathcal{D}$ do
    $\forall i$, obtain the initial perturbation $\Delta_i$ as in Zheng et al. (2020).
    $\forall i$, one step PGD update: $\Delta_i \leftarrow \Pi_{\mathcal{S}^{(p)}(\epsilon)}\left[\Delta_i + \alpha sign(\nabla_{\Delta_i}\mathcal{L}_\theta(\boldsymbol{x}_i + \Delta_i, y_i)\right])$.
    $\forall i$, update the cached adversarial perturbation $\Delta_i$ as in Zheng et al. (2020).
    if use reweight then
        $\forall i$, weight $w_i = \mathrm{softmax}[f(\boldsymbol{x}_i + \Delta_i)]_{y_i}$
    else
        $\forall i$, weight $w_i = 1$
    end if
    $\forall i$, query the adaptive target $\tilde{\boldsymbol{t}}_i$ and update: $\tilde{\boldsymbol{t}}_i \leftarrow \rho\tilde{\boldsymbol{t}}_i + (1-\rho)softmax[f(\boldsymbol{x}_i + \Delta_i)]$.
    $\forall i$, the final adaptive target $\boldsymbol{t}_i = \beta\mathbf{1}_{y_i} + (1-\beta)\tilde{\boldsymbol{t}}_i$
    Calculate the loss $\frac{1}{\sum_i^B w_i}\sum_i^B w_i\mathcal{L}_\theta(\boldsymbol{x}_i + \Delta_i, \boldsymbol{t}_i)$ and update the parameters.
end for
```

Method	Model	Epochs	Complexity	AutoAttack(%)
Shafahi et al. (2019)	WRN34	200	2	41.17
Wong et al. (2020)	RN18	15	4	43.21
Zheng et al. (2020)	WRN34	38	4	44.48
Zhang et al. (2019a)	WRN34	105	3	44.83
Chen et al. (2021a)	WRN34	100	7	51.12
Reweighting (Ours)	WRN34	38	4	46.15
Adaptive Target (Ours)	WRN34	38	4	51.17

Table 2: Comparison between different accelerated adversarial training methods in robust test accuracy against AutoAttack (AA). The baseline results are from RobustBench. *Complexity* shows the total number of forward passes and backward passes in one mini-batch update.

The results are provided in Table 2, where the results of the baseline methods are taken from RobustBench Croce et al. (2020). We also report the number of epochs and the number of forward and backward passes in a mini-batch update of each method. The product of these two values indicates the training complexity. We can clearly see that both reweighting and adaptive targets improve the performance on top of ATTA Zheng et al. (2020). Note that our method based on adaptive targets achieve the best performance while needing only 1/4 of the training time of Chen et al. (2021a), the strongest baseline. Wong et al. (2020) is the only baseline consuming less training time than ours, but its performance is much worse than ours; it suffers from catastrophic overfitting when using a WRN34 model.

We also conduct ablation study in the context of fast adversarial training. In Figure 7, we plot the learning curves for different values of β in Algorithm 1, we also compare the learning curves of ATTA with and without reweighting. Lower the value of β is, more weights assigned to the adaptive part of the target: $\beta = 0$ means we directly utilize the moving average target as the final target, $\beta = 1$ means we use the one-hot groundtruth label. In the left part of Figure 7, the generalization gap decreases with the decrease in β. That is to say, the adaptive target can indeed improve the generalization performance. In addition, the right part of Figure 7 confirm that the reweighting scheme can prevent adversarial overfitting and decrease the generalization gap.

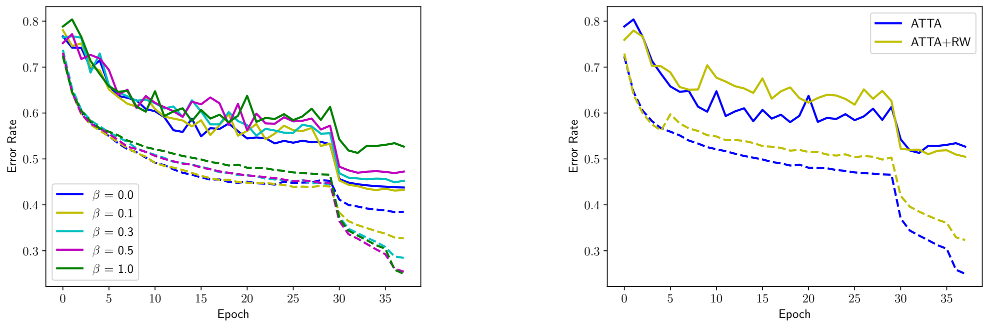


Figure 7: The learning curves of Algorithm 1 when we use different values of β (left) or compare the performance with and without reweighting (right). The solid curve and the dashed curve represent the robust test error and the robust training error, respectively.

6.2.2 Adversarial Fine-tuning with Additional Data

In this section, we study fine-tuning an adversarially pretrained model using additional training data. We observe that adversarial overfitting occurs when using a small learning rate in Section 4. Since we also use a small learning rate to conduct adversarial fine-tuning with additional data, it is important to address the adversarial overfitting issue in this context. While additional training data was shown to be beneficial in Alayrac et al. (2019); Carmon et al. (2019), we demonstrate that letting the model adaptively fit the easy and hard instances of the additional data further improve the performance.

We conduct experiments on both CIFAR10 and SVHN, using WRN34 and RN18 models, respectively. The model is fine-tuned for either 1 epoch or 5 epochs, which means that each additional training instance is used either 5 times or only once. This is because we observed the performance of vanilla adversarial training to start decaying after 5 epochs. As such, methods requiring many epochs such as Balaji et al. (2019) and Huang et al. (2020) are not applicable here. More hyper-parameter details are deferred to Appendix C.2.

Our first technique, reweighting, is the same as in the previous section. In addition to reweighting, we can also add a KL regularization term measuring the KL divergence between the output probability of the clean instance and of the adversarial instance. The KL term encourages the adversarial output to be close to the clean one. In other words, the clean output probability serves as the adaptive target. For hard instances, the clean and adversarial inputs are usually both misclassified. Therefore, the clean outputs of these instances constitute simpler targets compared with the ground-truth labels. Ultimately, the loss objective of a mini-batch $\{\boldsymbol{x}_i\}_{i=1}^B$ used for fine-tuning is expressed as $\mathcal{L}_{FT}(\{\boldsymbol{x}_i\}_{i=1}^B) =$

$\sum_{i=1}^{B} w_i \left[\mathcal{L}_{\boldsymbol{w}}(\boldsymbol{x}'_i) + \lambda KL(\boldsymbol{o}_i || \boldsymbol{o}'_i)\right]$ where w_i is the adaptive weight when we use re-weighting, or $1/B$ otherwise. λ is 6 when using the regularization term and 0 otherwise.

We use reweighting and KL regularization to fine-tune the model. Results in Table 3 clearly show that both techniques benefit the performance of the finetuned model. This shows that avoiding fitting hard adversarial examples helps to improve the generalization performance in adversarial fine-tuning with additional training data.

Duration	Method	AutoAttack(%)	Duration	Method	AutoAttack(%)
WRN34 on CIFAR10, $\epsilon = 8/255$			**RN18 on SVHN,** $\epsilon = 0.02$		
No Fine Tuning		52.01	No Fine Tuning		67.77
1 Epoch	Vanilla AT	54.11	1 Epoch	Vanilla AT	70.81
	RW	54.69		RW	70.83
	KL	54.73		KL	72.29
	RW + KL	54.69		RW + KL	72.53
5 Epoch	Vanilla AT	55.49	5 Epoch	Vanilla AT	72.18
	RW	56.41		RW	72.72
	KL	56.55		KL	73.17
	RW + KL	56.99		RW + KL	73.35

Table 3: Robust accuracy of fine-tuned models against AutoAttack(AA). We conduct ablation study on both reweighting (RW) and KL regularization (KL).

7. Conclusion

We have investigated *adversarial overfitting* from the perspective of training instances' difficulty. By introducing a quantitative metric to measure the instance difficulty, we have shown that a model's generalization performance under adversarial attacks degrades during the later phase of training as the model fits the hard adversarial instances. We have conducted theoretical analyses on both linear and nonlinear models. On an over-parameterized logistic regression model, we have shown that training on harder adversarial instances leads to poorer generalization performance. We have also proven that the performance of adversarial training is more sensitive to hard instances than vanilla training. On general nonlinear models, we have shown that the lower bound of a well-trained model's Lipschitz constant increases when trained with more difficult instances. Finally, we have shown that existing approaches to mitigating adversarial overfitting implicitly avoid fitting hard adversarial instances. We believe that our findings shed some light on adversarial training, and will allow the community to design new algorithms and improve robustness in diverse applications.

Acknowledgment

Part of this work is supported by National Natural Science Foundation of China (NSFC Project No. 62306250) and CityU APRC Project (Project No. 9610614).

Appendix A. Notation

$\mathcal{A}$	Section 3	Perturbation method.
b	Section 5.2	The number of parameters in a general nonlinear model.
c	Assumption 4, Section 5.2	The coefficient in isoperimetry.
C	Section 5.2	The mean squared error on the adversarial training set.
d	Equation 2, Section 3	The function representing the difficulty metric.
$\mathcal{D}$	Section 3	The data set.
$f_{\boldsymbol{w}}$	Section 1	The model parameterized by $\boldsymbol{w}$.
$\mathcal{F}$	Theorem 7, Section 5.2	The function space of the model.
$\mathcal{G}$	Section 4.2	Groups of the training set divided by instance difficulty.
h	Definition 5, Section 5.2	The bandwidth of the model's output range.
J	Theorem 7, Section 5.2	The Lipschitz constant of $f_{\boldsymbol{w}}$ w.r.t $\boldsymbol{w}$.
K	Section 5	The number of components in the data distribution.
l	Section 5	The component index where the training data is sampled.
L	Assumption 4, Section 5.2	The Lipschitz constant of $f_{\boldsymbol{w}}$ w.r.t the input.
$\mathcal{L}$	Section 1	The loss function.
m	Section 5	Dimension of the input data.
M	Section 3	The number of total training epochs.
n	Section 5	The number of training instances.
$\boldsymbol{o}$, $\boldsymbol{o}'$	Section 6	Model's output of the clean and the adversarial input.
p	Section 1	Shape of the adversarial budget.
p_k	Section 5.1	The probability of k-th component in the GMM model.
r	Equation 4, Section 5.1	The coefficient in the GMM model.
$\mathcal{R}$	Theorem 2, Section 5.1	The robust test error.
$\boldsymbol{t}$, $\widetilde{\boldsymbol{t}}$	Section 6.2	The adaptive target and the moving average target.
$\boldsymbol{w}$	Section 5	Model parameters.
W	Theorem 7, Section 5.2	The diameter upper bound of the parameter space.
$\mathcal{W}$	Theorem 7, Section 5.2	The space of model parameters.
$\boldsymbol{x}, \boldsymbol{x}', \mathbf{X}$	Section 1 & Section 5	Clean input, adversarial input and its matrix form.
y, $\boldsymbol{y}$	Section 1 & Section 5	Label and its vector form.
α	Algorithm 1	The step size of the adversarial attacks.
β	Section 6.2	The coefficient controlling how adaptive the target is.
γ	Theorem 7, Section 5.2	The non-negative variable introduced in Theorem 7.
δ	Theorem 7, Section 5.2	The probability introduced in Theorem 7.
ϵ	Section 1	The size of the adversarial budget.
$\boldsymbol{\eta}$	Equation 4, Section 5.1	The direction of the mean of each GMM's component.
ρ	Section 6.2	The momentum calculating the moving average target.
μ_l, μ_l	Assumption 4, Section 5.2	Data distribution and its l-th component.
σ	Assumption 4, Section 5.2	The conditional variance of the data distribution.

Table 4: The notation in this paper. In addition to what they represent, we provide the section of their definition or first appearance.

Appendix B. Proofs in Theoretical Analysis

B.1 Proof of Theorem 1

Similar to Soudry et al. (2018), we can assume all instances are positive without the loss of generality, this is because we can always redefine $y_i \boldsymbol{x}_i$ as the input. In this regard, the loss to optimize in a logistic regression model under the adversarial budget $\mathcal{S}^{(2)}(\epsilon)$ is:

$$\mathcal{L}_{\boldsymbol{w}}(\mathbf{X}) = \sum_{i=1}^{n} l(\boldsymbol{w}^T \boldsymbol{x}_i - \epsilon\|\boldsymbol{w}\|) \tag{10}$$

Here $l(\cdot)$ is the logistic function: $l(x) = \frac{1}{1+e^{-x}}$. We use $\mathbf{X} \in \mathbb{R}^{n\times m}$ to represent the training set as said in Section 5, then the loss function $\mathcal{L}(\boldsymbol{w})$ is $\|\mathbf{X}\|^2$-smooth, where $\|\mathbf{X}\|^2$ is the maximal singular value of $\mathbf{X}$. Since function $\mathcal{L}_{\boldsymbol{w}}$ is convex on $\boldsymbol{w}$, so gradient descent of step size smaller than $2\|\mathbf{X}\|^{-2}$ will asymptotically converge to the global infimum of the function $\mathcal{L}_{\boldsymbol{w}}$ on $\boldsymbol{w}$.

Before proving Theorem 1, we first introduce the following lemma:

Lemma 9. *Consider the max-margin vector $\widehat{\boldsymbol{w}}$ of the vanilla case defined in Equation (3), we then introduce the max margin vector $\widehat{\boldsymbol{w}'}$ defined under the adversarial attack of budget $\mathcal{S}^{(2)}(\epsilon)$ as follows:*

$$\widehat{\boldsymbol{w}'} = \arg\min_{\boldsymbol{w}} \|\boldsymbol{w}\| \quad s.t.\ \forall i \in \{1,2,...,n\},\ \boldsymbol{w}^T\boldsymbol{x}_i - \epsilon\|\boldsymbol{w}\| \geq 1 \tag{11}$$

Then we have $\widehat{\boldsymbol{w}'}$ is collinear with $\widehat{\boldsymbol{w}}$, i.e., $\frac{\widehat{\boldsymbol{w}'}}{\|\widehat{\boldsymbol{w}'}\|} = \frac{\widehat{\boldsymbol{w}}}{\|\widehat{\boldsymbol{w}}\|}$

Proof We show that $\widehat{\boldsymbol{w}} = \frac{1}{1+\epsilon\|\widehat{\boldsymbol{w}'}\|}\widehat{\boldsymbol{w}'}$ and prove it by contraction.

Let's assume $\exists \boldsymbol{v},\ s.t.\ \|\boldsymbol{v}\| < \frac{\|\widehat{\boldsymbol{w}'}\|}{1+\epsilon\|\widehat{\boldsymbol{w}'}\|}$ and $\forall i \in \{1,2,...,n\},\ \boldsymbol{v}^T\boldsymbol{x}_i \geq 1$, then we can consider $\boldsymbol{v}' = (1 + \|\widehat{\boldsymbol{w}'}\|)\boldsymbol{v}$. The l_2 norm of $\boldsymbol{v}'$ is smaller than that of $\widehat{\boldsymbol{w}'}$, and we have

$$\forall i \in \{1,2,...,n\}, \boldsymbol{v}'^T\boldsymbol{x}_i - \epsilon\|\boldsymbol{v}'\| = (1+\epsilon\|\widehat{\boldsymbol{w}'}\|)\boldsymbol{v}^T\boldsymbol{x}_i - \epsilon\|\boldsymbol{v}'\| > (1+\epsilon\|\widehat{\boldsymbol{w}'}\|) - \epsilon\|\widehat{\boldsymbol{w}'}\| = 1 \tag{12}$$

Inequality 12 shows we can construct a vector $\boldsymbol{v}'$ whose l_2 norm is smaller than $\widehat{\boldsymbol{w}'}$ and satisfying the condition (11), this contracts with the optimality of $\widehat{\boldsymbol{w}'}$. Therefore, there is no solution of condition (3) whose norm is smaller than $\frac{\|\widehat{\boldsymbol{w}'}\|}{1+\epsilon\|\widehat{\boldsymbol{w}'}\|}$.

On the other hand, $\frac{1}{1+\epsilon\|\widehat{\boldsymbol{w}'}\|}\widehat{\boldsymbol{w}'}$ satisfies the condition (3) and its l_2 norm is $\frac{\|\widehat{\boldsymbol{w}'}\|}{1+\epsilon\|\widehat{\boldsymbol{w}'}\|}$. As a result, we have $\widehat{\boldsymbol{w}} = \frac{1}{1+\epsilon\|\widehat{\boldsymbol{w}'}\|}\widehat{\boldsymbol{w}'}$. That means $\widehat{\boldsymbol{w}}$ and $\widehat{\boldsymbol{w}'}$ are collinear. ■

With Lemma 9, Theorem 1 is more straightforward, whose proof is shown below. Regarding the convergence analysis of the logistic regression model in non-adversarial cases, we encourage the readers to find more details in Ji and Telgarsky (2019); Soudry et al. (2018).

Proof Theorem 1 in Ji and Telgarsky (2019) and Theorem 3 in Soudry et al. (2018) proves the convergence of the direction of the logistic regression parameters in different cases. In this regard, we can let $\boldsymbol{w}_\infty = \lim_{u\to\infty} \frac{\boldsymbol{w}(u)}{\|\boldsymbol{w}(u)\|}$. That is to say, for sufficiently large u, the direction of the parameter $\boldsymbol{w}(u)$ can be considered fixed. As a result, the adversarial perturbations of each data instance $\boldsymbol{x}_i$ is fixed, i.e., $\epsilon\boldsymbol{w}_\infty$.

We can then apply the conclusion of Theorem 3 in Soudry et al. (2018) here, the only difference is the data points are $\{\boldsymbol{x}_i - \epsilon\boldsymbol{w}_\infty\}_{i=1}^n$. Therefore, the parameter $\boldsymbol{w}(u)$ will converge to the l_2 max margin of the dataset $\{\boldsymbol{x}_i - \epsilon\boldsymbol{w}_\infty\}_{i=1}^n$. When $t \to \infty$, we have $\boldsymbol{w}(u)^T(\boldsymbol{x}_i - \epsilon\boldsymbol{w}_\infty) = \boldsymbol{w}(u)^T\boldsymbol{x}_i - \epsilon\|\boldsymbol{w}(u)\|$. This is exactly the adversarial max margin condition in (11). Based on Lemma 9, we have $\lim_{u\to\infty} \frac{\boldsymbol{w}(u)}{\|\boldsymbol{w}(u)\|} = \frac{\widehat{\boldsymbol{w}'}}{\|\widehat{\boldsymbol{w}'}\|} = \frac{\widehat{\boldsymbol{w}}}{\|\widehat{\boldsymbol{w}}\|}$ ■

B.2 Proof of Theorem 2

Given the parameter $\boldsymbol{w}$ of the logistic regression model, we can first calculate the robust error for the k-th component of the GMM model defined in (4).

Lemma 10. *The 0-1 classification error of a linear classifier $\boldsymbol{w}$ under the adversarial attack of the budget $\mathcal{S}^{(2)}(\epsilon)$ for the k-th component of the GMM model defined in (4) is:*

$$\widehat{\mathcal{R}}_k(\epsilon) = \Phi(\frac{r_k\boldsymbol{w}^T\boldsymbol{\eta}}{\|\boldsymbol{w}\|} - \epsilon) \tag{13}$$

where $\Phi(x) = \mathbb{P}(Z > x), Z \sim \mathcal{N}(0,1)$.

Proof For a random drawn data instance $(\boldsymbol{x}, y)$, the adversarial perturbation is $-y\epsilon\frac{\boldsymbol{w}}{\|\boldsymbol{w}\|}$. Let's decompose $\boldsymbol{x}$ as $r_k y\boldsymbol{\eta} + \boldsymbol{z}$, where $\boldsymbol{z} \sim \mathcal{N}(0, \mathbf{I})$. Then, we have

$$\begin{aligned}\widehat{\mathcal{R}}_k(\epsilon) &= \mathbb{P}(y\boldsymbol{w}^T(\boldsymbol{x} - y\epsilon\frac{\boldsymbol{w}}{\|\boldsymbol{w}\|}) < 0) = \mathbb{P}(y\boldsymbol{w}^T(r_k y\boldsymbol{\eta} + \boldsymbol{z} - y\epsilon\frac{\boldsymbol{w}}{\|\boldsymbol{w}\|}) < 0) \\ &= \mathbb{P}(-y\boldsymbol{w}^T\boldsymbol{z} > r_k\boldsymbol{w}^T\boldsymbol{\eta} - \epsilon\|\boldsymbol{w}\|)\end{aligned} \tag{14}$$

Since $\boldsymbol{z} \sim \mathcal{N}(0, \mathbf{I})$, we have $-y\boldsymbol{w}^T\boldsymbol{z} \sim \mathcal{N}(0, (-y\boldsymbol{w}^T)^T(-y\boldsymbol{w}^T)) = \mathcal{N}(0, \boldsymbol{w}^T\boldsymbol{w})$. Furthermore $\frac{-y\boldsymbol{w}^T\boldsymbol{z}}{\|\boldsymbol{w}\|} \sim \mathcal{N}(0,1)$, and we can further simplify $\widehat{\mathcal{R}}_k(\epsilon)$ as follows:

$$\widehat{\mathcal{R}}_k(\epsilon) = \mathbb{P}(\frac{-y\boldsymbol{w}^T\boldsymbol{z}}{\|\boldsymbol{w}\|} > \frac{r_k\boldsymbol{w}^T\boldsymbol{\eta}}{\|\boldsymbol{w}\|} - \epsilon) = \Phi(\frac{r_k\boldsymbol{w}^T\boldsymbol{\eta}}{\|\boldsymbol{w}\|} - \epsilon) \tag{15}$$

■

With Lemma 10, we can straightforwardly calculate the robust error for all components of the GMM model defined in (4):

$$\widehat{\mathcal{R}}(\epsilon) = \sum_{k=1}^K p_k\Phi(\frac{r_k\boldsymbol{w}^T\boldsymbol{\eta}}{\|\boldsymbol{w}\|} - \epsilon) \tag{16}$$

On the other hand, Theorem 1 indicates the parameter $\boldsymbol{w}$ will converge to the l_2 max margin. However, for arbitrary training set, we do not have the closed form of $\boldsymbol{w}$, which is a barrier for the further analysis. Nevertheless, results from Wang and Thrampoulidis (2020) indicates in the over-parameterization regime, the parameter $\boldsymbol{w}$ will converge to min-norm interpolation of the data with high probability.

Lemma 11. *(Directly from Theorem 1 in Wang and Thrampoulidis (2020)) Assume n training instances drawn from the l-th mode of the described distribution in (4) and each of them is a m-dimensional vector. If $\frac{m}{n \log n}$ is sufficiently large*[4]*, then the l_2 max margin vector in Equation (3) will be the same as the solution of the min-norm interpolation described below with probability at least $(1 - O(\frac{1}{n}))$.*

$$\bar{\boldsymbol{w}} = \arg\min_{\boldsymbol{w}} \|\boldsymbol{w}\| \quad s.t. \; \forall i \in \{1, 2, ..., n\}, \; y_i = \boldsymbol{w}^T \boldsymbol{x}_i \tag{17}$$

Since the min-norm interpolation has a closed solution $\bar{\boldsymbol{w}} = \mathbf{X}^T(\mathbf{X}\mathbf{X}^T)^{-1}\boldsymbol{y}$, Lemma 11 will greatly facilitate the calculation of $\mathbb{R}(\boldsymbol{w})$ in Theorem 2. To simplify the notation, we first define the following variables.

$$\mathbf{U} = \mathbf{Q}\mathbf{Q}^T, \; \boldsymbol{d} = \mathbf{Q}\boldsymbol{\eta}, \; s = \boldsymbol{y}^T\mathbf{U}^{-1}\boldsymbol{y}, \; t = \boldsymbol{d}\mathbf{U}^{-1}\boldsymbol{d}, \; v = \boldsymbol{y}^T\mathbf{U}^{-1}\boldsymbol{d} \tag{18}$$

The proof of Theorem 2 is then presented below.

Proof Based on (16), the key is to simplify the term $\frac{\boldsymbol{w}^T\boldsymbol{\eta}}{\|\boldsymbol{w}\|}$, let's denote it by A, then we have:

$$A^2 = \frac{\boldsymbol{\eta}^T\boldsymbol{w}\boldsymbol{w}^T\boldsymbol{\eta}}{\boldsymbol{w}^T\boldsymbol{w}} = \frac{(\boldsymbol{y}^T(\mathbf{X}\mathbf{X}^T)^{-1}\mathbf{X}\boldsymbol{\eta})^2}{\boldsymbol{y}^T(\mathbf{X}\mathbf{X}^T)^{-1}\boldsymbol{y}} \tag{19}$$

The key challenge here is to calculate the term $(\mathbf{X}\mathbf{X}^T)^{-1}$ where $\mathbf{X} = r_l\boldsymbol{y}\boldsymbol{\eta}^T + Q$. Here we utilize Lemma 3 of Wang and Thrampoulidis (2020) and Woodbury identity Horn and Johnson (2012), we have:

$$\boldsymbol{y}^T(\mathbf{X}\mathbf{X})^{-1} = \boldsymbol{y}^T\mathbf{U}^{-1} - \frac{(r_l^2 s\|\boldsymbol{\eta}\|^2 + r_l^2 v^2 + r_l v - r_l^2 st)\boldsymbol{y}^T + r_l s\boldsymbol{d}^T}{r_l^2 s(\|\boldsymbol{\eta}\|^2 - t) + (r_l v + 1)^2}\mathbf{U}^{-1} \tag{20}$$

Here, s, t, v, $\mathbf{U}$ and $\boldsymbol{d}$ are defined in Equation (18). The scalar divisor comes from the matrix inverse calculation. Base of Equation (20), we can then calculate $\boldsymbol{y}^T(\mathbf{X}\mathbf{X}^T)^{-1}\boldsymbol{y}$ and $\boldsymbol{y}^T(\mathbf{X}\mathbf{X}^T)^{-1}\mathbf{X}\boldsymbol{\eta}$.

$$\begin{aligned}\boldsymbol{y}^T(\mathbf{X}\mathbf{X}^T)^{-1}\boldsymbol{y} &= s - \frac{(r_l^2 s\|\boldsymbol{\eta}\|^2 + r_l^2 v^2 + r_l v - r_l^2 st)s + r_l sv}{r_l^2 s(\|\boldsymbol{\eta}\|^2 - t) + (r_l v + 1)^2} \\ &= \frac{s}{r_l^2 s(\|\boldsymbol{\eta}\|^2 - t) + (r_l v + 1)^2}\end{aligned} \tag{21}$$

4. Specifically, m and n need to satisfy $m > 10n \log n + n - 1$ and $m > Cnr_l\sqrt{\log 2n}\|\boldsymbol{\eta}\|$. The constant C is derived in the proof of Theorem 1 in Wang and Thrampoulidis (2020).

$$\begin{aligned}\boldsymbol{y}^T(\mathbf{X}\mathbf{X}^T)^{-1}\mathbf{X}\boldsymbol{\eta} &= \boldsymbol{y}^T(\mathbf{X}\mathbf{X}^T)^{-1}(r_l\boldsymbol{y}\boldsymbol{\eta}^T + Q)\boldsymbol{\eta} \\ &= r_l\|\boldsymbol{\eta}\|^2\boldsymbol{y}^T(\mathbf{X}\mathbf{X}^T)^{-1}\boldsymbol{y} + \boldsymbol{y}^T(\mathbf{X}\mathbf{X}^T)^{-1}\boldsymbol{d} \\ &= \frac{r_l s(\|\boldsymbol{\eta}\|^2 - t) + r_l v^2 + v}{r_l^2 s(\|\boldsymbol{\eta}\|^2 - t) + (r_l v + 1)^2}\end{aligned} \tag{22}$$

Plug Equation (21) and (22) into (19), we have:

$$\begin{aligned}A^2 &= \frac{\left(r_l s(\|\boldsymbol{\eta}\|^2 - t) + r_l v^2 + v\right)^2}{s\left(r_l^2 s(\|\boldsymbol{\eta}\|^2 - t) + (r_l v + 1)^2\right)} \\ &= \frac{s(\|\boldsymbol{\eta}\|^2 - t) + v^2}{s} - \frac{\|\boldsymbol{\eta}\|^2 - t}{r_l^2 s(\|\boldsymbol{\eta}\|^2 - t) + (r_l v + 1)^2} \\ &= \frac{s(\|\boldsymbol{\eta}\|^2 - t) + v^2}{s} - \frac{1}{\left(\frac{s(\|\boldsymbol{\eta}\|^2 - t) + v^2}{\|\boldsymbol{\eta}\|^2 - t}\right) r_l^2 + \frac{2v}{\|\boldsymbol{\eta}\|^2 - t} r_l + \frac{1}{\|\boldsymbol{\eta}\|^2 - t}}\end{aligned} \tag{23}$$

Plug (23) into (16), we then obtain the robust error on all components of the GMM defined in (4):

$$\begin{gathered}\mathcal{R}(r_l, \epsilon) = \sum_{k=1}^{K} p_k \Phi\left(r_k g(r_l) - \epsilon\right),\ g(r_l) = (C_1 - \frac{1}{C_2 r_l^2 + C_3})^{\frac{1}{2}} \\ C_1 = \frac{s(\|\boldsymbol{\eta}\|^2 - t) + v^2}{s},\ C_2 = \frac{s(\|\boldsymbol{\eta}\|^2 - t) + v^2}{\|\boldsymbol{\eta}\|^2 - t},\ C_3 = \frac{2v}{\|\boldsymbol{\eta}\|^2 - t} r_l + \frac{1}{\|\boldsymbol{\eta}\|^2 - t}.\end{gathered} \tag{24}$$

We study the sign of C_1 and C_2. Consider $\mathbf{U} = \mathbf{Q}\mathbf{Q}^T$ is a positive semidefinite matrix, so $s = \boldsymbol{y}\mathbf{U}^{-1}\boldsymbol{y}^T \geq 0$. In addition, we have $\|\boldsymbol{\eta}\|^2 - t = \boldsymbol{\eta}^T\left(\mathbf{I} - (\mathbf{Q}\mathbf{Q}^T)^{-1}\right)\boldsymbol{\eta}$. Since $\mathbf{I} - (\mathbf{Q}\mathbf{Q}^T)^{-1} = (\mathbf{I} - (\mathbf{Q}\mathbf{Q}^T)^{-1})^T(\mathbf{I} - (\mathbf{Q}\mathbf{Q}^T)^{-1})$ is a positive semidefinite matrix, we can obtain $\mathbf{I} - (\mathbf{Q}\mathbf{Q}^T)^{-1}$ is also a positive semidefinite matrix. As a result, C_1 and C_2 are both non-negative.

■

B.3 Proof of Corollary 3

To prove Corollary 3, we first prove the following lemma:

Lemma 12. *Under the condition of Theorem 2 and $\mathcal{R}$ in Equation (5), $\frac{\partial \mathcal{R}(r_l, \epsilon)}{\partial r_l}$ is negative and monotonically decreases with ϵ.*

Proof

Based on Equation (24), we have:

$$\frac{\partial \mathcal{R}(r_l, \epsilon)}{\partial r_l} = \sum_{k=1}^{K} p_k \Phi'(r_k g(r_l) - \epsilon) \frac{\partial g(r_l)}{\partial r_l} \tag{25}$$

Since the training data is separable, we have $\forall k, r_k \boldsymbol{w}^T \boldsymbol{\eta} - \epsilon \|\boldsymbol{w}\| > 0$, which is equivalent to the following:

$$\forall k, r_k g(r_l) - \epsilon > 0 \tag{26}$$

First, p_k is a positive number by definition. Consider function $\Phi(x)$ monotonically decrease with x and is convex when $x > 0$, so $\forall k, \Phi'(r_k g(r_l) - \epsilon)$ is negative and decreases with ϵ. In addition, $g(r_l)$ increases with r_l and is independent on ϵ, so $\frac{\partial g(r_l)}{\partial r_l}$ can be considered as a positive constant. Therefore, $\frac{\partial \mathcal{R}(r_l, \epsilon)}{\partial r_l}$ is negative and monotonically decreases with ϵ.

■

Now, we are ready to prove Corollary 3:

Proof

We subtract the left hand side from the right hand side in the inequality of Corollary 3:

$$\begin{aligned}
[\mathcal{R}(r_j, \epsilon_1) - \mathcal{R}(r_i, \epsilon_1)] - [\mathcal{R}(r_j, \epsilon_2) - \mathcal{R}(r_i, \epsilon_2)] &= \int_{r_i}^{r_j} \frac{\partial \mathcal{R}(r_l, \epsilon_1)}{\partial r_l} d_{r_l} - \int_{r_i}^{r_j} \frac{\partial \mathcal{R}(r_l, \epsilon_2)}{\partial r_l} d_{r_l} \\
&= \int_{r_i}^{r_j} \left[\frac{\partial \mathcal{R}(r_l, \epsilon_1)}{\partial r_l} - \frac{\partial \mathcal{R}(r_l, \epsilon_2)}{\partial r_l} \right] d_{r_l} \\
&> 0
\end{aligned} \tag{27}$$

The last inequality is based on the conditions $r_j > r_i$, $\epsilon_2 > \epsilon_1$ as well as Lemma 12, they jointly indicate $\left[\frac{\partial \mathcal{R}(r_l, \epsilon_1)}{\partial r_l} - \frac{\partial \mathcal{R}(r_l, \epsilon_2)}{\partial r_l} \right]$ is always positive. We reorganize (27) and obtain $\mathcal{R}(r_i, \epsilon_1) - \mathcal{R}(r_j, \epsilon_1) < \mathcal{R}(r_i, \epsilon_2) - \mathcal{R}(r_j, \epsilon_2)$.

■

B.4 Proof of Theorem 7

We start with the following lemma.

Lemma 13. *Given the assumptions of Theorem 7, we define $g(\boldsymbol{x}) = \mathbb{E}(y|\boldsymbol{x})$, $z(\boldsymbol{x}) = y - g(\boldsymbol{x})$ and consider $\gamma = \sigma_l^2 + h^2(C, \epsilon) - C$, then the following inequality holds.*

$$\begin{aligned}
&\forall a \in (0,1), \mathbb{P}(\exists f_{\boldsymbol{w}} \in \mathcal{F} : \frac{1}{n}\sum_{i=1}^{n}(y_i - f_{\boldsymbol{w}}(\boldsymbol{x}_i'))^2 \leq C) \\
&\leq 2e^{-\frac{na^2\gamma^2}{8}} + \mathbb{P}(\exists f_{\boldsymbol{w}} \in \mathcal{F} : \frac{1}{n}\sum_{i=1}^{n} f_{\boldsymbol{w}}(\boldsymbol{x}_i) z(\boldsymbol{x}_i) \geq \frac{1}{2}(1-3a)\gamma)
\end{aligned} \tag{28}$$

Proof Given the definition of $h(C, \epsilon)$, we have:

$$\begin{aligned}
(y_i - f_{\boldsymbol{w}}(\boldsymbol{x}_i'))^2 &= [(y_i - f_{\boldsymbol{w}}(\boldsymbol{x}_i)) + (f_{\boldsymbol{w}}(\boldsymbol{x}_i) - f_{\boldsymbol{w}}(\boldsymbol{x}_i'))]^2 \\
&\geq (y_i - f_{\boldsymbol{w}}(\boldsymbol{x}_i))^2 + (f_{\boldsymbol{w}}(\boldsymbol{x}_i) - f_{\boldsymbol{w}}(\boldsymbol{x}_i'))^2 \\
&\geq (y_i - f_{\boldsymbol{w}}(\boldsymbol{x}_i))^2 + h^2(C, \epsilon)
\end{aligned} \tag{29}$$

For the first inequality, $\boldsymbol{x}_i'$ is the adversarial example which tries to maximize the loss objective, $y_i \in \{-1, +1\}$ and the range of $f_{\boldsymbol{w}}$ is $[-1, +1]$, so $\langle y_i - f_{\boldsymbol{w}}(\boldsymbol{x}_i), f_{\boldsymbol{w}}(\boldsymbol{x}_i) - f_{\boldsymbol{w}}(\boldsymbol{x}_i')\rangle \geq 0$. The second inequality is based on the definition of $h^2(C, \epsilon)$ in Definition 5. As a result, we can simplify the left hand side of (28) as follows:

$$\mathbb{P}(\exists f_{\boldsymbol{w}} \in \mathcal{F} : \frac{1}{n}\sum_{i=1}^{n}(y_i - f_{\boldsymbol{w}}(\boldsymbol{x}_i'))^2 \leq C) \leq \mathbb{P}(\exists f_{\boldsymbol{w}} \in \mathcal{F} : \frac{1}{n}\sum_{i=1}^{n}(y_i - f_{\boldsymbol{w}}(\boldsymbol{x}_i))^2 \leq C - h^2(C, \epsilon)) \tag{30}$$

We consider the sequence $\{z(\boldsymbol{x}_i)\}_{i=1}^n$, it is i.i.d with $\mathbb{E}_{\mu_l}(z(\boldsymbol{x})^2) = \mathbb{E}_{\mu_l}[Var(y|\boldsymbol{x})] = \sigma_l^2$. Since the range of the prediction is $[-1, +1]$, so $z^2(\boldsymbol{x}) \in [0, 4]$. Then, we have the following inequality by Hoeffding's inequality Hoeffding (1994).

$$\forall a \in (0,1), \mathbb{P}(\frac{1}{n}\sum_{i=1}^{n} z^2(\boldsymbol{x}_i) \leq \sigma_l^2 - a\gamma) \leq e^{-\frac{na^2\gamma^2}{8}} \tag{31}$$

Similarly, we consider the sequence $\{z(\boldsymbol{x}_i)g(\boldsymbol{x}_i)\}_{i=1}^n$, the following inequality holds based on the Hoeffding's inequality and the fact $\mathbb{E}(z(\boldsymbol{x})g(\boldsymbol{x})) = 0$, $z(\boldsymbol{x})g(\boldsymbol{x}) \in [-2, +2]$.

$$\forall a \in (0,1), \mathbb{P}(\frac{1}{n}\sum_{i=1}^{n} z(\boldsymbol{x}_i)g(\boldsymbol{x}_i) \leq a\gamma) \leq e^{-\frac{na^2\gamma^2}{8}} \tag{32}$$

Now we study the right hand side of (30):

$$\begin{aligned}
\frac{1}{n}\sum_{i=1}^{n}(y_i - f_{\boldsymbol{w}}(\boldsymbol{x}_i))^2 &= \frac{1}{n}\sum_{i=1}^{n}\left(z^2(\boldsymbol{x}_i) + (g(\boldsymbol{x}_i) - f_{\boldsymbol{w}}(\boldsymbol{x}_i))^2 + 2z(\boldsymbol{x}_i)(g(\boldsymbol{x}_i) - f_{\boldsymbol{w}}(\boldsymbol{x}_i))\right) \\
&\geq \frac{1}{n}\sum_{i=1}^{n}\left(z^2(\boldsymbol{x}_i) + 2z(\boldsymbol{x}_i)g(\boldsymbol{x}_i) - 2z(\boldsymbol{x}_i)f_{\boldsymbol{w}}(\boldsymbol{x}_i)\right)
\end{aligned} \tag{33}$$

Consider the following reasoning:

$$\begin{cases} \frac{1}{n}\sum_{i=1}^{n}(y_i - f_{\boldsymbol{w}}(\boldsymbol{x}_i))^2 \leq C - h^2(C,\epsilon) = \sigma_l^2 - \gamma \\ \frac{1}{n}\sum_{i=1}^{n} z^2(\boldsymbol{x}_i) \geq \sigma_l^2 - a\gamma \\ \frac{1}{n}\sum_{i=1}^{n} z(\boldsymbol{x}_i)g(\boldsymbol{x}_i) \geq -a\gamma \end{cases} \Longrightarrow \frac{1}{n}\sum_{i=1}^{n} z(\boldsymbol{x}_i)f_{\boldsymbol{w}}(\boldsymbol{x}_i) \geq \frac{1}{2}(1-3a)\gamma \tag{34}$$

As a result, we have:

$$\begin{aligned} &\mathbb{P}(\exists f_{\boldsymbol{w}} \in \mathcal{F} : \frac{1}{n}\sum_{i=1}^{n}(y_i - f_{\boldsymbol{w}}(\boldsymbol{x}_i) \leq C - h^2(C,\epsilon))) \\ \leq &\mathbb{P}(\exists f_{\boldsymbol{w}} \in \mathcal{F} : \frac{1}{n}\sum_{i=1}^{n} z^2(\boldsymbol{x}_i) \leq \sigma_l^2 - a\gamma) + \mathbb{P}(\exists f_{\boldsymbol{w}} \in \mathcal{F} : \frac{1}{n}\sum_{i=1}^{n} z(\boldsymbol{x}_i)g(\boldsymbol{x}_i) \geq -a\gamma) + \\ &\mathbb{P}(\exists f_{\boldsymbol{w}} \in \mathcal{F} : \frac{1}{n}\sum_{i=1}^{n} z(\boldsymbol{x}_i)f_{\boldsymbol{w}}(\boldsymbol{x}_i) \geq \frac{1}{2}(1-3a)\gamma) \\ \leq &2e^{-\frac{na^2\gamma^2}{8}} + \mathbb{P}(\exists f_{\boldsymbol{w}} \in \mathcal{F} : \frac{1}{n}\sum_{i=1}^{n} z(\boldsymbol{x}_i)f_{\boldsymbol{w}}(\boldsymbol{x}_i) \geq \frac{1}{2}(1-3a)\gamma) \end{aligned} \tag{35}$$

The first inequality is based on the reasoning of (34). The second inequality is based on (31) and (32).

Based on the inequality (30) and (35), we conclude the proof.

■

To further simplify the right hand side of (28), $\mathbb{P}(\exists f_{\boldsymbol{w}} \in \mathcal{F} : \frac{1}{n}\sum_{i=1}^{n} z(\boldsymbol{x}_i)f_{\boldsymbol{w}}(\boldsymbol{x}_i) \geq \frac{1}{2}(1-3a)\gamma)$ needs to be bounded, and this is solved by the following lemma.

Lemma 14. *Given the assumptions of Theorem 7 and the definition of $g(\boldsymbol{x})$, $z(\boldsymbol{x})$ in Lemma 13, then the following inequality holds.*

$$\begin{aligned} &\forall a \in (0,1), a_1 > 0, a_2 > 0 \text{ and } a_1 + a_2 = \frac{1}{2}(1-3a), \\ &\mathbb{P}(\exists f_{\boldsymbol{w}} \in \mathcal{F} : \frac{1}{n}\sum_{i=1}^{n} z(\boldsymbol{x}_i)f_{\boldsymbol{w}}(\boldsymbol{x}_i) \geq \frac{1}{2}(1-3a)\gamma) \leq 2|\mathcal{F}|e^{-\frac{nm}{144cL^2}a_1^2\gamma^2} + 2e^{-\frac{n}{8}a_2^2\gamma^2} \end{aligned} \tag{36}$$

Proof We recall that the data points $\{\boldsymbol{x}_i, y_i\}_{i=1}^n$ are sampled from the distribution μ_l, which is c-isoperimetric. For any L-Lipschitz function f, we have:

$$\forall t, \mathbb{P}[|f_{\boldsymbol{w}}(\boldsymbol{x}) - \mathbb{E}_{\mu_l}(f_{\boldsymbol{w}})| \geq t] \leq 2e^{-\frac{mt^2}{2cL^2}} \tag{37}$$

Since $z(\boldsymbol{x}) = y - g(\boldsymbol{x}) \in [-2, +2]$, we can then bound $\mathbb{P}[z(\boldsymbol{x})(f_{\boldsymbol{w}}(\boldsymbol{x}) - \mathbb{E}_{\mu_l}(f_{\boldsymbol{w}})) \geq t]$:

$$\begin{aligned}\forall t, \mathbb{P}[z(\boldsymbol{x})(f_{\boldsymbol{w}}(\boldsymbol{x}) - \mathbb{E}_{\mu_l}(f_{\boldsymbol{w}})) \geq t] &\leq \mathbb{P}[|z(\boldsymbol{x})(f_{\boldsymbol{w}}(\boldsymbol{x}) - \mathbb{E}_{\mu_l}(f_{\boldsymbol{w}}))| \geq t] \\ &\leq \mathbb{P}[|(f_{\boldsymbol{w}}(\boldsymbol{x}) - \mathbb{E}_{\mu_l}(f_{\boldsymbol{w}}))| \geq \frac{t}{2}] \leq 2e^{-\frac{mt^2}{8cL^2}}\end{aligned} \tag{38}$$

Here we utilize the proposition in Vershynin (2018); Van Handel (2014)[5], which claims *if* $\{X_i\}_{i=1}^n$ *are independent variables and all* C*-subgaussian, then* $\frac{1}{\sqrt{n}}\sum_{i=1}^n X_i$ *is* $18C$*-subgaussian.* Therefore, we have:

$$\forall t, \mathbb{P}[\frac{1}{\sqrt{n}}\sum_{i=1}^n z(\boldsymbol{x}_i)(f_{\boldsymbol{w}}(\boldsymbol{x}_i) - \mathbb{E}_{\mu_l}(f_{\boldsymbol{w}})) \geq t] \leq 2e^{-\frac{mt^2}{144cL^2}} \tag{39}$$

Let $t = a_1\gamma\sqrt{n}$, then we have:

$$\mathbb{P}[\frac{1}{n}\sum_{i=1}^n z(\boldsymbol{x}_i)(f_{\boldsymbol{w}}(\boldsymbol{x}_i) - \mathbb{E}_{\mu_l}(f)) \geq a_1\gamma] \leq 2e^{-\frac{nm}{144cL^2}a_1^2\gamma^2} \tag{40}$$

In addition, we can bound $\mathbb{P}[\frac{1}{n}\sum_{i=1}^n z(\boldsymbol{x}_i)\mathbb{E}_{\mu_l}(f_{\boldsymbol{w}}) \geq a_2\gamma]$ by:

$$\mathbb{P}[\exists f_{\boldsymbol{w}} \in \mathcal{F} : \frac{1}{n}\sum_{i=1}^n z(\boldsymbol{x}_i)\mathbb{E}_{\mu_l}(f_{\boldsymbol{w}}) \geq a_2\gamma] \leq \mathbb{P}[\frac{1}{n}\sum_{i=1}^n |z(\boldsymbol{x}_i)| \geq a_2\gamma] \leq 2e^{-\frac{n}{8}a_2^2\gamma^2} \tag{41}$$

The first inequality is based on the fact $\mathbb{E}_{\mu_l}(f_{\boldsymbol{w}}) \in [-1, +1]$; the second inequality is based on Hoeffding's inequality.

Now, we are ready to bound the probability $\mathbb{P}(\exists f_{\boldsymbol{w}} \in \mathcal{F} : \frac{1}{n}\sum_{i=1}^n z(\boldsymbol{x}_i)f_{\boldsymbol{w}}(\boldsymbol{x}_i) \geq \frac{1}{2}(1-3a)\gamma)$.

$$\begin{aligned}&\mathbb{P}(\exists f_{\boldsymbol{w}} \in \mathcal{F} : \frac{1}{n}\sum_{i=1}^n z(\boldsymbol{x}_i)f_{\boldsymbol{w}}(\boldsymbol{x}_i) \geq \frac{1}{2}(1-3a)\gamma) \\ \leq&\mathbb{P}[\exists f_{\boldsymbol{w}} \in \mathcal{F} : \frac{1}{n}\sum_{i=1}^n z(\boldsymbol{x}_i)(f_{\boldsymbol{w}}(\boldsymbol{x}_i) - \mathbb{E}_{\mu_l}(f)) \geq a_1\gamma] + \mathbb{P}[\exists f_{\boldsymbol{w}} \in \mathcal{F} : \frac{1}{n}\sum_{i=1}^n z(\boldsymbol{x}_i)\mathbb{E}_{\mu_l}(f_{\boldsymbol{w}}) \geq a_2\gamma] \\ \leq&2|\mathcal{F}|e^{-\frac{nm}{144cL^2}a_1^2\gamma^2} + 2e^{-\frac{n}{8}a_2^2\gamma^2}\end{aligned} \tag{42}$$

The first inequality is based on the fact $a_1 + a_2 = \frac{1}{2}(1-3a)$; the second inequality is based on the Boole's inequality Boole (1847), inequality (40) and (41).

■

To simplify the constant notation, we let $a = \frac{1}{8}$, $a_1 = \frac{3}{16}$ and $a_2 = \frac{1}{8}$. We plug this into the inequality (28) and (36), then:

5. Proposition 2.6.1 in Vershynin (2018) and Exercise 3.1 in Van Handel (2014)

$$\mathbb{P}(\exists f_{\boldsymbol{w}} \in \mathcal{F} : \frac{1}{n}\sum_{i=1}^{n}(y_i - f_{\boldsymbol{w}}(\boldsymbol{x}'_i))^2 \leq C) \leq 4e^{-\frac{n\gamma^2}{2^9}} + 2|\mathcal{F}|e^{-\frac{nm\gamma^2}{2^{12}cL^2}} \tag{43}$$

Now we turn to the proof of Theorem 7.

Proof

We let $\mathcal{F}_L = \{f_{\boldsymbol{w}} | \boldsymbol{w} \in \mathcal{W}, Lip(f_{\boldsymbol{w}}) \leq L\}$, $\mathcal{F}_\gamma = \{f_{\boldsymbol{w}} | \boldsymbol{w} \in \mathcal{W}, \boldsymbol{w} = \frac{\gamma}{4J} \odot \boldsymbol{z}, \boldsymbol{z} \in \mathbb{Z}^b\}$ and $\mathcal{F}_{\gamma,L} = \mathcal{F}_\gamma \cap \mathcal{F}_L$. Correspondingly, we let $\mathcal{W}_L = \{\boldsymbol{w} | \boldsymbol{w} \in \mathcal{W}, Lip(f_{\boldsymbol{w}}) \leq L\}$, $\mathcal{W}_\gamma = \{\boldsymbol{w} | \boldsymbol{w} \in \mathcal{W}, \boldsymbol{w} = \frac{\gamma}{4J} \odot \boldsymbol{z}, \boldsymbol{z} \in \mathbb{Z}^b\}$ and $\mathcal{W}_{\gamma,L} = \mathcal{W}_\gamma \cap \mathcal{W}_L$. Because the diameter of $\mathcal{W}$ is W, we have $|\mathcal{F}_{\gamma,L}| \leq |\mathcal{F}_\gamma| \leq \left(\frac{4WJ}{\gamma}\right)^b$. Here, $\odot$ means the element-wise multiplication.

Note that the inequality (43) is valid for any values of C as long as it satisfies $\gamma \geq 0$. Based on this, we apply the substitution $\begin{cases} C \leftarrow C + \frac{1}{2}\gamma \\ \gamma \leftarrow \frac{1}{2}\gamma \end{cases}$, then:

$$\begin{aligned}\mathbb{P}(\exists f_{\boldsymbol{w}} \in \mathcal{F}_{\gamma,L} : \frac{1}{n}\sum_{i=1}^{n}(y_i - f_{\boldsymbol{w}}(\boldsymbol{x}'_i))^2 \leq C + \frac{1}{2}\gamma) &\leq 4e^{-\frac{n\gamma^2}{2^{11}}} + 2|\mathcal{F}|e^{-\frac{nm\gamma^2}{2^{14}cL^2}} \\ &\leq 4e^{-\frac{n\gamma^2}{2^{11}}} + 2e^{b\log(\frac{4WJ}{\gamma}) - \frac{nm\gamma^2}{2^{14}cL^2}}\end{aligned} \tag{44}$$

Based on the definition of $\mathcal{W}_{\gamma,L}$, we can conclude that $\forall \boldsymbol{w}_1 \in \mathcal{W}_L, \exists \boldsymbol{w}_2 \in \mathcal{W}_{\gamma,L}\ s.t. \|\boldsymbol{w}_1 - \boldsymbol{w}_2\|_\infty \leq \frac{\gamma}{8J}$. Therefore, $\forall f_{\boldsymbol{w}_1} \in \mathcal{F}_L, \exists f_{\boldsymbol{w}_2} \in \mathcal{F}_{\gamma,L} s.t. \|f_{\boldsymbol{w}_1} - f_{\boldsymbol{w}_2}\|_\infty \leq \frac{\gamma}{8}$. Let choose such $f_{\boldsymbol{w}_2} \in \mathcal{F}_{\gamma,L}$ given an arbitrary $f_{\boldsymbol{w}_1} \in \mathcal{F}_L$, then:

$$\begin{aligned}(y - f_{\boldsymbol{w}_1}(\boldsymbol{x}))^2 &= (y - f_{\boldsymbol{w}_2}(\boldsymbol{x}))^2 + (2y - f_{\boldsymbol{w}_1}(\boldsymbol{x}) - f_{\boldsymbol{w}_2}(\boldsymbol{x}))(f_{\boldsymbol{w}_2}(\boldsymbol{x}) - f_{\boldsymbol{w}_1}(\boldsymbol{x})) \\ &\geq (y - f_{\boldsymbol{w}_2}(\boldsymbol{x}))^2 - \frac{\gamma}{8}|(2y - f_{\boldsymbol{w}_1}(\boldsymbol{x}) - f_{\boldsymbol{w}_2}(\boldsymbol{x}))| \\ &\geq (y - f_{\boldsymbol{w}_2}(\boldsymbol{x}))^2 - \frac{\gamma}{2}\end{aligned} \tag{45}$$

The first inequality in (45) is based on Hölder's inequality; the second inequality is based on $y \in \{-1, +1\}$ and the range of $\forall f_{\boldsymbol{w}} \in \mathcal{F}$ is $[-1, +1]$.

We combine (43) with (45), then:

$$\begin{aligned}\mathbb{P}(\exists f_{\boldsymbol{w}} \in \mathcal{F}_L : \frac{1}{n}\sum_{i=1}^{n}(y_i - f_{\boldsymbol{w}}(\boldsymbol{x}'_i))^2 \leq C) &\leq \mathbb{P}(\exists f_{\boldsymbol{w}} \in \mathcal{F}_{\gamma,L} : \frac{1}{n}\sum_{i=1}^{n}(y_i - f_{\boldsymbol{w}}(\boldsymbol{x}'_i))^2 \leq C + \frac{1}{2}\gamma) \\ &\leq 4e^{-\frac{n\gamma^2}{2^{11}}} + 2e^{b\log(\frac{4WJ}{\gamma}) - \frac{nm\gamma^2}{2^{14}cL^2}}\end{aligned} \tag{46}$$

Note that $\mathcal{F}_L$ is the set of functions in $\mathcal{F}$ whose Lipschitz constant is no larger than L. We set the right hand side of (46) to be δ and then get $L = \frac{\gamma}{2^7}\sqrt{\frac{nm}{c\left(b\log(4WJ\gamma^{-1}) - \log(\delta/2 - 2e^{-2^{-11}n\gamma^2})\right)}}$. This concludes the proof.

■

B.5 Proof of Corollary 8

Based on the definition of $\{\gamma_i\}_{i=1}^K$, we can apply Theorem 7 to each subset of the training set. Each of these subsets is sampled from one component of the data distribution. For instances sampled from the i-th components, we can derive the lower bound of the model's Lipschitz by the following formulation:

$$Lip^{(i)}(f_{\boldsymbol{w}}) \geq \begin{cases} 0, & \gamma_i < 0 \ , \\ \frac{\gamma}{2^7}\sqrt{\frac{nm}{c\left(b\log(4WJ\gamma^{-1}) - \log(\delta/2 - 2e^{-2^{-11}n\gamma^2})\right)}}, & \gamma_i \geq 0 \ , \end{cases} \tag{47}$$

Since $\{Lip^{(i)}(f_{\boldsymbol{w}})\}_{i=1}^K$ are all valid Lipschitz lower bounds for the same model, we can refine the Lipschitz lower bound by choosing the biggest number of them. We can then get the Lipschitz lower bound as in (9).

Appendix C. Experimental Settings

C.1 General Settings

The ResNet-18 (RN18) architecture is same as the one in Wong et al. (2020); the WideResNet-34 (WRN34) architecture is same as the one in Madry et al. (2018). Unless specified, the l_∞ adversarial budget used for CIFAR10 dataset Krizhevsky et al. (2009) [6] is 8/255 and for SVHN dataset Netzer et al. (2011) [7] is 0.02. In PGD adversarial training, the step size is 2/255 for CIFAR10 and 0.005 for SVHN; PGD is run for 10 iterations for both datasets. For adversarial attacks using a different adversarial budget, the step size is always 1/4 of the adversarial budget's size, and we always run it for 10 iterations. To comprehensively and reliably evaluate the robustness of the model, we use AutoAttack Croce and Hein (2020b), which is an ensemble of 4 different attacks: AutoPGD on cross entropy, AutoPGD on difference of logits ratio, fast adaptive boundary (FAB) attack Croce and Hein (2020a) and square attack Andriushchenko et al. (2020). Unless specified, we use stochastic gradient descent (SGD) with a momentum to optimize the model parameters, we also use weight decay whose factor is 0.0005. Unless specified, the momentum factor is 0.9, the learning rate starts with 0.1 and is divided by 10 in the 1/2 and 3/4 of the whole training duration. The size of the mini-batch is always 128.

We run the experiments on a machine with 4 NVIDIA TITAN XP GPUs. It takes about 6 hours to adversarially train a RN18 model for 200 epochs, and a whole day to adversarially train a WRN34 model for 200 epochs.

6. Data available for download on https://www.cs.toronto.edu/ kriz/cifar.html. MIT license. Free to use.
7. Data available for download on http://ufldl.stanford.edu/housenumbers/. Free for non-commercial use.

C.2 Settings in the Case Studies

Fast Adversarial Training Our experiments in this section is on CIFAR10 and use the l_∞ norm based adversarial budget with $\epsilon = 8/255$. The step size α in Algorithm 1 is 4/255. Unless explicitly stated, the coefficient ρ and β is 0.9 and 0.1. We train the model for 38 epochs, the learning rate is 0.1 on the first 30 epochs, it decays to 0.01 in the next 6 epochs and further decays to 0.001 in the last 2 epochs. When we use adaptive targets, the first 5 epochs are the warmup period in which we use fixed targets. Since the goal here is to accelerate adversarial training, we do not use a validation set to do model selection as in Rice et al. (2020). We use the standard data augmentation on CIFAR10: random crop and random horizontal flip.

Adversarial Fine-tuning with Additional Data For CIFAR10, we use 500000 images from 80 Million Tiny Images dataset Torralba et al. (2008) with pseudo labels in Carmon et al. (2019) [8]. For SVHN, we use the extra held-out set provided by SVHN itself, which contains 531131 somewhat less difficult samples. When we construct a mini-batch, half of its instances are sampled from the original training set and the other half are sampled from the additional data. The experimental settings are the same as Carmon et al. (2019) except the learning rate. We tune the learning rate and find that fixing it to 10^{-3} is the best choice.

Appendix D. Additional Experiments and Discussion

D.1 Properties of the Difficulty Metric

To study the factors affecting the difficulty function defined in (2), let us denote by d_1, d_2 the difficulty functions obtained under two different training settings, such as different network architectures and training methods. We then define the difficulty distance (*D-distance*) between two such functions d_1, d_2 under the same perturbation type $\mathcal{A}$ as $D_{\mathcal{A}}(d_1, d_2)$, which is calculated as follows:

$$D_{\mathcal{A}}(d_1, d_2) = \mathbb{E}_{\boldsymbol{x} \sim U(\mathcal{D})} |d_1(\boldsymbol{x}, \mathcal{A}) - d_2(\boldsymbol{x}, \mathcal{A})| \; . \tag{48}$$

Similarly, the D-distance between the same function d but under two different perturbation types $\mathcal{A}_1$, $\mathcal{A}_2$ is represented by $D_d(\mathcal{A}_1, \mathcal{A}_2)$:

$$D_d(\mathcal{A}_1, \mathcal{A}_2) = \mathbb{E}_{\boldsymbol{x} \sim U(\mathcal{D})} |d(\boldsymbol{x}, \mathcal{A}_1) - d(\boldsymbol{x}, \mathcal{A}_2)| \; . \tag{49}$$

For both $D_{\mathcal{A}}(d_1, d_2)$ and $D_d(\mathcal{A}_1, \mathcal{A}_2)$, the expected D-distance between two random difficulty functions with random perturbation types is 0.375, which is calculated based on the random shuffle of the average loss for each training instance.

We then study the properties of the difficulty functions in Equation (2) by performing experiments on the CIFAR10 and CIFAR10-C (Hendrycks and Dietterich, 2019) dataset, varying factors of interest and calculating the D-distances between different difficulty functions.

We first study the influence of the network architectures and training durations by using either a RN18 model, trained for either 100 or 200 epochs (RN18-100 or RN18-200), or a

8. Data available for download on https://github.com/yguooo/semisup-adv. MIT license. Free to use.

WRN34 model trained for 200 epochs (WRN34). To generate adversarial attacks, we always use of PGD perturbation $\mathcal{A}_{PGD}$ with an adversarial budget based on the l_∞ norm with $\epsilon = 8/255$. This corresponds to the settings used in other works Hendrycks and Dietterich (2019); Madry et al. (2018). The other hyper-parameters follow the general settings in Appendix C. In the left part of Table 5, we report the D-distance $D_{\mathcal{A}_{PGD}}(d_1, d_2)$ for all pairs of settings. Each result is averaged over 4 runs, the variances are all below 0.012 and thus negligible. The D-distances in all scenarios are very small and close to 0, indicating the architecture and the training duration have little influence on instance difficulty based on our definition.

$d_1 \backslash d_2$	RN18-100	RN18-200	WRN34
RN18-100	0.0189	0.0232	0.0355
RN18-200	0.0232	0.0159	0.0299
WRN34	0.0355	0.0299	0.0178

$\mathcal{A}_1 \backslash \mathcal{A}_2$	Clean	FGSM	PGD
Clean	0.0189	0.0607	0.1713
FGSM	0.0607	0.0843	0.1677
PGD	0.1713	0.1677	0.0857

Table 5: D-distances ($D_{\mathcal{A}}(d_1, d_2)$ for the left table and $D_d(\mathcal{A}_1, \mathcal{A}_2)$ for the right table) between difficulty functions in different settings, including different model architectures, training duration (left table), and different types of perturbations (right table).

We then perform experiments by varying the attack strategy using a RN18 network. As shown by the D-distances $D_d(\mathcal{A}_1, \mathcal{A}_2)$ reported in the right portion of Table 5, the discrepancy between values obtained with clean, FGSM-perturbed and PGD-perturbed inputs is much larger, thus indicating that our difficulty function correctly reflects the influence of an attack on an instance. In addition, Table 6 demonstrates the D-distance between the difficulty functions based on clean instances, FGSM-perturbed instance, PGD-perturbed instances and different common corruptions from CIFAR10-C Hendrycks and Dietterich (2019)[9]. Note that Hendrycks and Dietterich (2019) only provides corrupted instances on the test set, so we train models on the clean training set and test model on corrupted test set in these cases. We use RN18 architecture and train it for 100 epochs in all cases, results are reported on the test set. Compared with the results in the left half of Table 5, the D-distance is much larger here. This indicates the difficulty function depends on the perturbation type applied to the input, including the common corruptions.

The results in Table 5 and 6 demonstrate that our difficulty metric mainly depends on the data and on the perturbation type; not the model architecture or the training duration. This is why we include the data $\boldsymbol{x}$ and the perturbation type $\mathcal{A}$ explicitly in the parameter list in the definition of the difficulty function d in Equation (2).

In the definition of our difficulty metric in Equation (2), the difficulty of one instance is based on its average loss values during the training procedure. It is intuitive, because the values of the loss objective represents the cost that model needs to fit the corresponding data point. The bigger this cost is, the more difficulty this instance will be. To make the metric stable and prevent the metric from being sensitive to the stochasticity in the training dynamics, we use the average value of the loss objective for each instance to define its difficulty. In

9. Data available for download on https://github.com/hendrycks/robustness. Apache License 2.0. Free to use.

$\mathcal{A}_1 \backslash \mathcal{A}_2$	brightness	contrast	defocus	elastic	fog	gaussian blur
Clean	0.1279	0.3219	0.2646	0.2115	0.2324	0.3069
FGSM	0.1303	0.3128	0.2642	0.2098	0.2289	0.3064
PGD	0.1873	0.3082	0.2616	0.2319	0.2414	0.2959

$\mathcal{A}_1 \backslash \mathcal{A}_2$	glass blur	jpeg	motion blur	pixelate	gaussian noise	impulse noise
Clean	0.2809	0.1838	0.2520	0.2365	0.2999	0.2869
FGSM	0.2760	0.1853	0.2520	0.2417	0.2918	0.2807
PGD	0.2825	0.2026	0.2605	0.2551	0.2980	0.2866

$\mathcal{A}_1 \backslash \mathcal{A}_2$	saturate	shot noise	snow	spatter	zoom blur	speckle noise
Clean	0.1335	0.2832	0.2033	0.1930	0.2654	0.2829
FGSM	0.1329	0.2754	0.2003	0.1946	0.2657	0.2759
PGD	0.1932	0.2841	0.2148	0.2297	0.2711	0.2901

Table 6: D-distances between difficulty functions of vanilla / FGSM / PGD training and training based on 18 different corruptions on CIFAR10-C. We run each experiment for 4 times and report the average value.

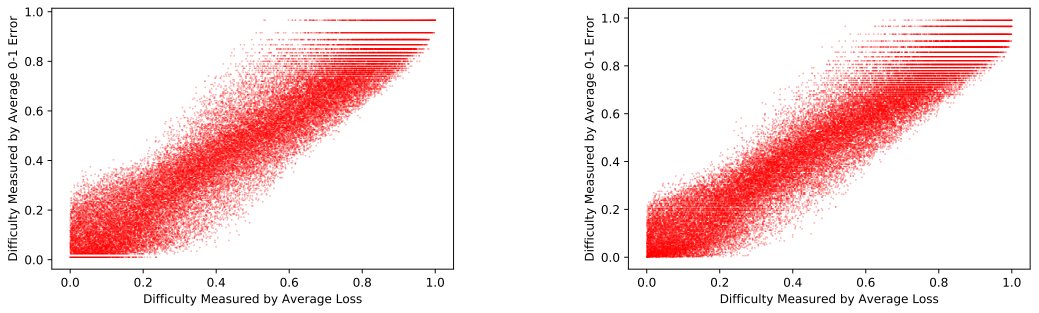


Figure 8: The relationship between the difficulty function based on the average loss values and the one based on the average 0-1 errors. The left figure is based on the RN18-200 model; the right figure is based on the WRN34 model. The correlation between these two metrics are 0.9466 (left) and 0.9545 (right), respectively.

addition to the average loss objectives, we can also use the average 0-1 error to define the difficulty function. In Figure 8, we plot the relationship between the difficulty metric based on the average loss values and the one based on the average 0-1 error for instances in the CIFAR10 training set when we train a RN18-100 model and a WRN34 model. We can see a strong correlation between them for both models. The correlation of the difficulty measured by two metrics for the same instance is 0.9466 in the RN18-100 case and 0.9545 in the WRN34 case. The high correlation indicates we can use either metric to measure the difficulty. Since the loss objective values are continuous and finer-grained, we choose it as the basis of the difficulty function we use in this paper.

D.2 Consistency of the Difficulty Definition

The difficulty definitions used in our theoretical analyses and empirical experiments are consistent with the definition of d function in Equation (2) in Section 3.

Theoretical Analyses in Section 5 In the analysis of the linear model, we assume the data distribution follows a K-component Gaussian mixture model. In our definition (4), the average distance between the positive instances and the negative instances of the k-th component is $2r_k$. Based on symmetry, the average distance between the decision boundary and the adversarial training instances is $r_k + \epsilon$. Since the loss of the linear model decreases with the increase of the distance between the input and the decision boundary, bigger the value of r_k is, smaller the average loss objective is. Therefore, in this case, the difficulty level of such training instances, which are defined on their loss objectives, is lower.

In the analysis of the general model, we use the conditional variance σ_k^2 to represent the difficulty of the k-th component of the data distribution. Based on Bubeck and Sellke (2021), the conditional variance σ_k^2 is the average error of a well-trained model. Since the difficulty is defined on the loss objective, it can be concluded that bigger the σ_k is, more difficulty the samples from the corresponding component will be.

Case Studies in Section 6

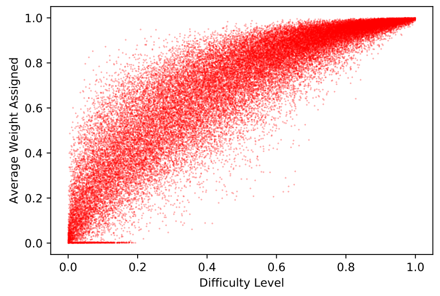


Figure 9: The relationship between the difficulty value and the weight assigned to each instances when using reweighting. We use the average weight across epochs. The correlation between them is 0.8900.

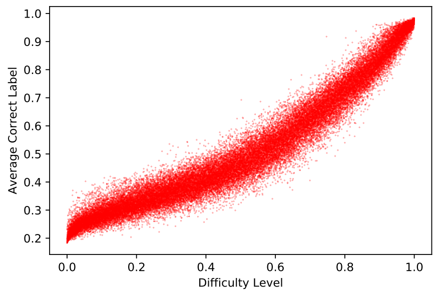


Figure 10: The relationship between the difficulty value and the average value of the true label's probability when using the adaptive targets. The correlation between them is 0.9604.

To confirm that the Algorithm 1 is consistent with our difficulty definition, we study the relationship between the instance difficulty and the weight assigned to them when using reweighting, as well as the soft target when using adaptive targets. Since the evaluation of model robustness is based on the PGD attack, the difficulty value here is also based on the PGD perturbation. In Figure 9, we demonstrate the relationship between the difficulty value and the average assigned weight for each instance when using reweighting. We calculate the correlation between these two values on the training set, it is 0.8900. This indicates we indeed assign smaller weights for hard training instances and assign bigger weights for easy training instances. In Figure 10, we show the relationship between the difficulty value and the average value of the true label's probability in the soft target when we use the adaptive targets. Similarly, we calculate the correlation between these two values on the training set, it is 0.9604. This indicates the adaptive target is similar to the ground-truth one-hot target for the easy training instances, while the adaptive target is very different from the ground-truth one-hot target for the hard training instances. This means, adaptive targets prevent the model from fitting hard training instances while encourage the model to fit the easy training instances.

D.3 Training on a Subset

Results on SVHN dataset

Figure 11 demonstrates the learning curves of PGD adversarial training based on a subset of the easiest, the random and the hardest instances of SVHN dataset. We let the size of each subset be 20000, because the training set of SVHN is larger than that of CIFAR10. The model architecture is RN18 in these cases. We have the same observations here: training on the hardest subset yields trivial performance, training on the random subset has significant generalization decay in the late phase of training while there is no such phenomenon when the model is trained on the easiest instances.

In Figure 12, we conduct PGD adversarial training using increasing more training instances in SVHN dataset, starting with the easiest ones. The observation here is consistent with Figure 3c: although fitting hard adversarial instances can cause overfitting, they can improve the model performance if we use easy stopping by a validation set. Therefore, we should not simply remove the hard training instances, but need to utilize them adaptively.

Different Values of ϵ and l_2-based Adversarial Budget Figure 13 and Figure 14 demonstrate the learning curves of RN18 models under different adversarial budgets on CIFAR10, in both l_∞ and l_2 cases. In l_∞ cases, the adversarial budgets are 2/255, 4/255 and 6/255; in l_2 cases, the adversarial budgets are 0.5, 0.75 and 1. With the increase in the size of the adversarial budget, we can see a clear transition from the vanilla training: more and more severe generalization decay when training on the random or the hardest subset.

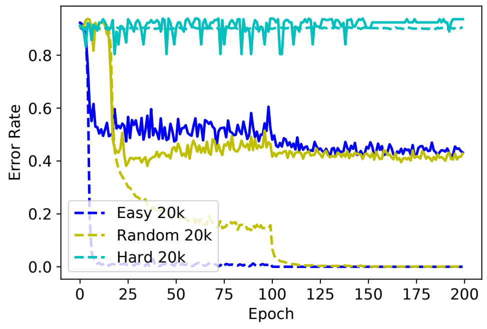


Figure 11: Learning curves of training using the easiest, the random and the hardest 20000 instances of the SVHN training set. The training error (dashed lines) is the robust error on the selected instances, and the robust test error (solid lines) is always the error on the entire test set.

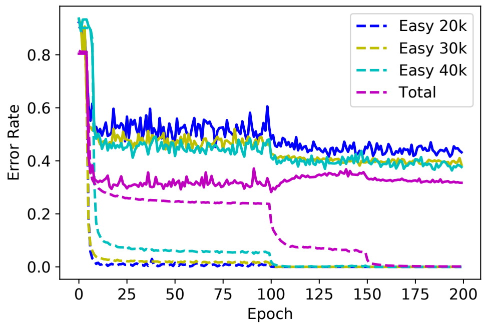
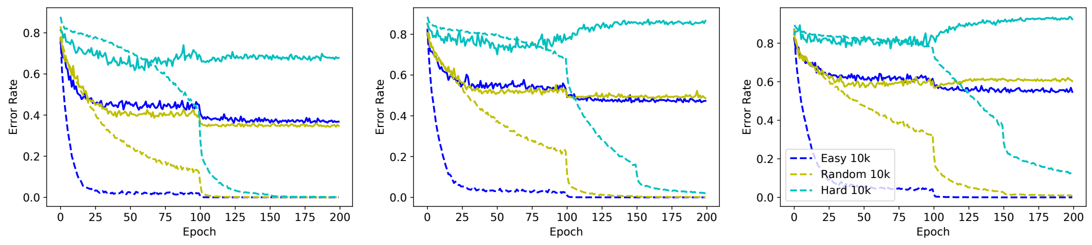


Figure 12: Learning curves of PGD adversarial training using increasing more training data of SVHN. The dashed lines represent the robust training error on the selected training instances; the solid lines represent the robust test error on the entire test set.

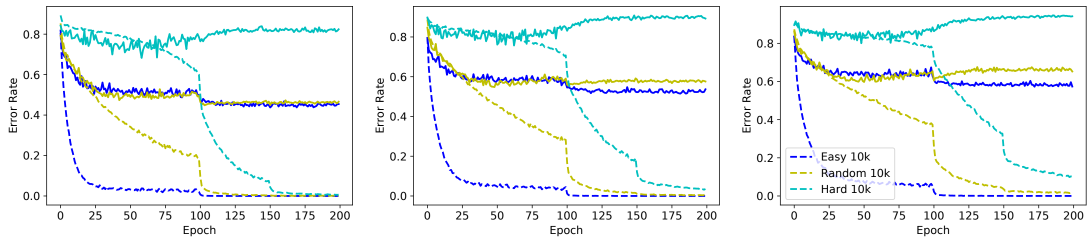


(a) $\epsilon = 2/255$ (b) $\epsilon = 4/255$ (c) $\epsilon = 6/255$

Figure 13: Learning curves of training on PGD-perturbed inputs against different sizes of l_∞ norm based adversarial budgets using the easiest, the random and the hardest 10000 training instances. The instance difficulty is determined by the corresponding adversarial budget and is thus different under different adversarial budgets. The dashed lines are robust training error on the selected training set, the solid lines are robust test error on the entire test set.

D.4 Revisiting Existing Methods Mitigating Adversarial Overfitting

Existing methods mitigating adversarial overfitting can be generally divided into two categories: one is to use adaptive inputs, such as Balaji et al. (2019); the other is to use adaptive targets, such as Chen et al. (2021b); Huang et al. (2020). Both categories aim to prevent the model from fitting hard input-target pairs. In this section, we pick one example from each category for investigation. We provide the learning curves of the methods we study in Figure 15. We use the same hyper-parameters as in these methods' original paper, except for the training duration and learning rate scheduler, which follow our settings. These methods



(a) $\epsilon = 0.50$ (b) $\epsilon = 0.75$ (c) $\epsilon = 1.00$

Figure 14: Learning curves of training on PGD-perturbed inputs against different size of l_2 norm based adversarial budgets using the easiest, the random and the hardest 10000 training instances. The instance difficulty is determined by the corresponding adversarial budget and is thus different under different adversarial budgets. The dashed lines are robust training error on the selected training set, the solid lines are robust test error on the entire test set.

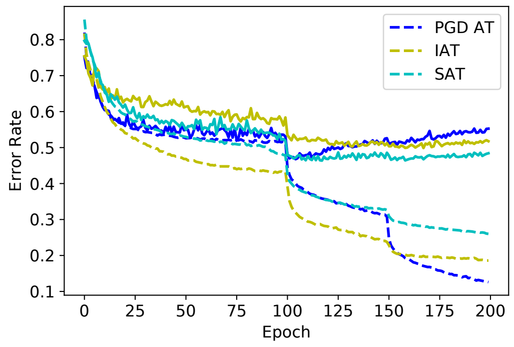


Figure 15: Learning curves of PGD adversarial training (PGD AT), instance-adaptive training (IAT) and self-adaptive training (SAT). Dashed lines and solid lines represent the robust training error and the robust test error, respectively.

clearly mitigate adversarial overfitting: The robust test error does not increase much in the late phase of training, and the generalization gap is much smaller that that of PGD adversarial training.

Instance-Adaptive Training Using an instance-adaptive adversarial budget has been shown to mitigate adversarial overfitting and yield a better trade-off between the clean and robust accuracy Balaji et al. (2019). In instance-adaptive adversarial training (IAT), each training instance $\boldsymbol{x}_i$ maintains its own adversarial budget's size ϵ_i during training. In each epoch, ϵ_i increases to $\epsilon_i + \epsilon_\Delta$ if the instance is robust under this enlarged adversarial budget. By contrast, ϵ_i decreases to $\epsilon_i - \epsilon_\Delta$ if the instance is not robust under the original adversarial budget. Here, ϵ_Δ is the step size of the adjustment.

We use the same settings as in Balaji et al. (2019) except that we use the same number of training epochs and learning rate scheduling as the one in other experiments for fair comparison. Specially, we set the value of ϵ and ϵ_Δ to be $8/255$ and $1.9/255$, respectively, same as in Balaji et al. (2019). The first 5 epochs are warmup, when we use vanilla adversarial training Madry et al. (2018).

Self-Adaptive Training Self-adaptive training (SAT) Huang et al. (2020) solves the adversarial overfitting issue by adapting the target. By contrast with common practice consisting of using a fixed target, usually the ground-truth, SAT adapts the target of each instance to the model's output. Specifically, after a warm-up period, the target $\boldsymbol{t}_i$ for an instance $\boldsymbol{x}_i$ is initialized as a one-hot vector by its ground-truth label y_i and updated in an iterative manner after each epoch as $\boldsymbol{t}_i \leftarrow \rho \boldsymbol{t}_i + (1-\rho)\boldsymbol{o}_i$. Here, ρ is a predefined momentum factor and $\boldsymbol{o}_i$ is the output probability of the current model on the corresponding clean instance. SAT uses the loss of TRADES Zhang et al. (2019b) but replaces the ground-truth label y with the adaptive target $\boldsymbol{t}_i$: $\mathcal{L}_{SAT}(\boldsymbol{x}_i) = \mathcal{L}(\boldsymbol{x}_i, \boldsymbol{t}_i) + \lambda \max_{\Delta_i \in \mathcal{S}(\epsilon)} KL(\boldsymbol{o}_i || \boldsymbol{o}'_i)$, where KL refers to the Kullback–Leibler divergence and λ is the weight for the regularizer. Furthermore, SAT uses a weighted average to calculate the loss of a mini-batch; the weight assigned to each instance $\boldsymbol{x}_i$ is proportional to the maximum element of its target $\boldsymbol{t}_i$ but normalized to ensure that all instances' weights sum up to 1. By weighted averaging, the instances with confident predictions are strengthened, whereas the ambiguous instances are downplayed.

Similarly, we use the same settings as in Huang et al. (2020) except we use the same number of training epochs and learning rate scheduling: we train the model for 200 epochs and the first 90 epochs are the warmup period.

References

Decoupled kullback-leibler divergence loss, 2024.

Jean-Baptiste Alayrac, Jonathan Uesato, Po-Sen Huang, Alhussein Fawzi, Robert Stanforth, and Pushmeet Kohli. Are labels required for improving adversarial robustness? *Advances in Neural Information Processing Systems*, 32, 2019.

Maksym Andriushchenko, Francesco Croce, Nicolas Flammarion, and Matthias Hein. Square attack: a query-efficient black-box adversarial attack via random search. In *European Conference on Computer Vision*, pages 484–501. Springer, 2020.

Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. *arXiv preprint arXiv:1802.00420*, 2018.

Yogesh Balaji, Tom Goldstein, and Judy Hoffman. Instance adaptive adversarial training: Improved accuracy tradeoffs in neural nets. *arXiv preprint arXiv:1910.08051*, 2019.

Robert Baldock, Hartmut Maennel, and Behnam Neyshabur. Deep learning through the lens of example difficulty. *Advances in Neural Information Processing Systems*, 34:10876–10889, 2021.

Peter L Bartlett, Philip M Long, Gábor Lugosi, and Alexander Tsigler. Benign overfitting in linear regression. *Proceedings of the National Academy of Sciences*, 117(48):30063–30070, 2020.

George Boole. *The mathematical analysis of logic.* Philosophical Library, 1847.

Sebastien Bubeck and Mark Sellke. A universal law of robustness via isoperimetry. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021. URL `https://openreview.net/forum?id=z71OSKqTFh7`.

Jacob Buckman, Aurko Roy, Colin Raffel, and Ian Goodfellow. Thermometer encoding: One hot way to resist adversarial examples. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=S18Su--CW`.

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

Jinghui Chen, Yu Cheng, Zhe Gan, Quanquan Gu, and Jingjing Liu. Efficient robust training via backward smoothing, 2021a. URL `https://openreview.net/forum?id=49V11oUejQ`.

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*, 2021b. URL `https://openreview.net/forum?id=qZzy5urZw9`.

Jeremy M Cohen, Elan Rosenfeld, and J Zico Kolter. Certified adversarial robustness via randomized smoothing. *arXiv preprint arXiv:1902.02918*, 2019.

Francesco Croce and Matthias Hein. Minimally distorted adversarial examples with a fast adaptive boundary attack. In *International Conference on Machine Learning*, pages 2196–2205. PMLR, 2020a.

Francesco Croce and Matthias Hein. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *International Conference on Machine Learning*, 2020b.

Francesco Croce and Matthias Hein. Mind the box: l_1-apgd for sparse adversarial attacks on image classifiers. In *International Conference on Machine Learning*, pages 2201–2211. PMLR, 2021.

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

Guneet S. Dhillon, Kamyar Azizzadenesheli, Jeremy D. Bernstein, Jean Kossaifi, Aran Khanna, Zachary C. Lipton, and Animashree Anandkumar. Stochastic activation pruning for robust adversarial defense. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=H1uR4GZRZ`.

Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Hang Su, Jun Zhu, Xiaolin Hu, and Jianguo Li. Boosting adversarial attacks with momentum. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9185–9193, 2018.

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

Sven Gowal, Krishnamurthy Dj Dvijotham, Robert Stanforth, Rudy Bunel, Chongli Qin, Jonathan Uesato, Relja Arandjelovic, Timothy Mann, and Pushmeet Kohli. Scalable verified training for provably robust image classification. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4842–4851, 2019.

Sven Gowal, Chongli Qin, Jonathan Uesato, Timothy Mann, and Pushmeet Kohli. Uncovering the limits of adversarial training against norm-bounded adversarial examples. *arXiv preprint arXiv:2010.03593*, 2020.

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*, 34:4218–4233, 2021.

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

Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=HJz6tiCqYm`.

Dan Hendrycks, Kimin Lee, and Mantas Mazeika. Using pre-training can improve model robustness and uncertainty. In *International Conference on Machine Learning*, pages 2712–2721, 2019.

Dorjan Hitaj, Giulio Pagnotta, Iacopo Masi, and Luigi V Mancini. Evaluating the robustness of geometry-aware instance-reweighted adversarial training. *arXiv preprint arXiv:2103.01914*, 2021.

Wassily Hoeffding. Probability inequalities for sums of bounded random variables. In *The collected works of Wassily Hoeffding*, pages 409–426. Springer, 1994.

Roger A Horn and Charles R Johnson. *Matrix analysis*. Cambridge university press, 2012.

Lang Huang, Chao Zhang, and Hongyang Zhang. Self-adaptive training: beyond empirical risk minimization. *Advances in Neural Information Processing Systems*, 33, 2020.

Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Logan Engstrom, Brandon Tran, and Aleksander Madry. Adversarial examples are not bugs, they are features. In *Advances in Neural Information Processing Systems*, pages 125–136, 2019.

Ziwei Ji and Matus Telgarsky. The implicit bias of gradient descent on nonseparable data. In *Conference on Learning Theory*, pages 1772–1798. PMLR, 2019.

Yulun Jiang, Chen Liu, Zhichao Huang, Mathieu Salzmann, and Sabine Süsstrunk. Towards stable and efficient adversarial training against l_1 bounded adversarial attacks. In *International Conference on Machine Learning*. PMLR, 2023.

Matt Jordan and Alexandros G Dimakis. Exactly computing the local lipschitz constant of relu networks. *arXiv preprint arXiv:2003.01219*, 2020.

Yiwen Kou, Zixiang Chen, Yuanzhou Chen, and Quanquan Gu. Benign overfitting in two-layer relu convolutional neural networks. In *International Conference on Machine Learning*, pages 17615–17659. PMLR, 2023.

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

Nupur Kumari, Mayank Singh, Abhishek Sinha, Harshitha Machiraju, Balaji Krishnamurthy, and Vineeth N Balasubramanian. Harnessing the vulnerability of latent layers in adversarially trained models. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 2779–2785. International Joint Conferences on Artificial Intelligence Organization, 7 2019. doi: 10.24963/ijcai.2019/385. URL `https://doi.org/10.24963/ijcai.2019/385`.

Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial examples in the physical world. *arXiv preprint arXiv:1607.02533*, 2016.

Zhu Li, Zhi-Hua Zhou, and Arthur Gretton. Towards an understanding of benign overfitting in neural networks. *arXiv preprint arXiv:2106.03212*, 2021.

Chen Liu, Mathieu Salzmann, Tao Lin, Ryota Tomioka, and Sabine Süsstrunk. On the loss landscape of adversarial training: Identifying challenges and how to overcome them. *Advances in Neural Information Processing Systems*, 33, 2020.

Xingjun Ma, Bo Li, Yisen Wang, Sarah M. Erfani, Sudanthi Wijewickrema, Grant Schoenebeck, Michael E. Houle, Dawn Song, and James Bailey. Characterizing adversarial subspaces using local intrinsic dimensionality. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=B1gJ1L2aW`.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=rJzIBfZAb`.

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

Behnam Neyshabur, Srinadh Bhojanapalli, David McAllester, and Nati Srebro. Exploring generalization in deep learning. *Advances in neural information processing systems*, 30, 2017.

Tianyu Pang, Kun Xu, Chao Du, Ning Chen, and Jun Zhu. Improving adversarial robustness via promoting ensemble diversity. In *International Conference on Machine Learning*, pages 4970–4979, 2019.

Tianyu Pang, Kun Xu, Yinpeng Dong, Chao Du, Ning Chen, and Jun Zhu. Rethinking softmax cross-entropy loss for adversarial robustness. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=Byg9A24tvB`.

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*, 34:20596–20607, 2021.

Aditi Raghunathan, Jacob Steinhardt, and Percy Liang. Certified defenses against adversarial examples. *arXiv preprint arXiv:1801.09344*, 2018.

Leslie Rice, Eric Wong, and Zico Kolter. Overfitting in adversarially robust deep learning. In *International Conference on Machine Learning*, pages 8093–8104. PMLR, 2020.

Wenjie Ruan, Xiaowei Huang, and Marta Kwiatkowska. Reachability analysis of deep neural networks with provable guarantees. In *IJCAI*, pages 2651–2659, 2018. URL `https://doi.org/10.24963/ijcai.2018/368`.

Hadi Salman, Greg Yang, Jerry Li, Pengchuan Zhang, Huan Zhang, Ilya Razenshteyn, and Sebastien Bubeck. Provably robust deep learning via adversarially trained smoothed classifiers. *arXiv preprint arXiv:1906.04584*, 2019.

Pouya Samangouei, Maya Kabkab, and Rama Chellappa. Defense-GAN: Protecting classifiers against adversarial attacks using generative models. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=BkJ3ibb0-`.

Amartya Sanyal, Puneet K Dokania, Varun Kanade, and Philip HS Torr. How benign is benign overfitting? *arXiv preprint arXiv:2007.04028*, 2020.

Kevin Scaman and Aladin Virmaux. Lipschitz regularity of deep neural networks: analysis and efficient estimation. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 3839–3848, 2018.

Ali Shafahi, Mahyar Najibi, Amin Ghiasi, Zheng Xu, John Dickerson, Christoph Studer, Larry S Davis, Gavin Taylor, and Tom Goldstein. Adversarial training for free! *arXiv preprint arXiv:1904.12843*, 2019.

Daniel Soudry, Elad Hoffer, Mor Shpigel Nacson, Suriya Gunasekar, and Nathan Srebro. The implicit bias of gradient descent on separable data. *The Journal of Machine Learning Research*, 19(1):2822–2878, 2018.

Swabha Swayamdipta, Roy Schwartz, Nicholas Lourie, Yizhong Wang, Hannaneh Hajishirzi, Noah A Smith, and Yejin Choi. Dataset cartography: Mapping and diagnosing datasets with training dynamics. In *EMNLP (1)*, 2020.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *International Conference on Learning Representations*, 2014. URL `http://arxiv.org/abs/1312.6199`.

Mariya Toneva, Alessandro Sordoni, Remi Tachet des Combes, Adam Trischler, Yoshua Bengio, and Geoffrey J Gordon. An empirical study of example forgetting during deep neural network learning. In *International Conference on Learning Representations*, 2018.

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.

Florian Tramer, Nicholas Carlini, Wieland Brendel, and Aleksander Madry. On adaptive attacks to adversarial example defenses. *Advances in Neural Information Processing Systems*, 33, 2020.

Ramon Van Handel. Probability in high dimension. Technical report, PRINCETON UNIV NJ, 2014.

Roman Vershynin. *High-dimensional probability: An introduction with applications in data science*, volume 47. Cambridge university press, 2018.

Ke Wang and Christos Thrampoulidis. Benign overfitting in binary classification of gaussian mixtures. *arXiv preprint arXiv:2011.09148*, 2020.

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*, 2020. URL `https://openreview.net/forum?id=rklOg6EFwS`.

Zekai Wang, Tianyu Pang, Chao Du, Min Lin, Weiwei Liu, and Shuicheng Yan. Better diffusion models further improve adversarial training. In *International Conference on Machine Learning*, pages 36246–36263. PMLR, 2023.

Lily Weng, Huan Zhang, Hongge Chen, Zhao Song, Cho-Jui Hsieh, Luca Daniel, Duane Boning, and Inderjit Dhillon. Towards fast computation of certified robustness for relu networks. In *International Conference on Machine Learning*, pages 5276–5285. PMLR, 2018a.

Tsui-Wei Weng, Huan Zhang, Pin-Yu Chen, Jinfeng Yi, Dong Su, Yupeng Gao, Cho-Jui Hsieh, and Luca Daniel. Evaluating the robustness of neural networks: An extreme value theory approach. In *International Conference on Learning Representations*, 2018b. URL `https://openreview.net/forum?id=BkUHlMZ0b`.

Eric Wong and Zico Kolter. Provable defenses against adversarial examples via the convex outer adversarial polytope. In *International Conference on Machine Learning*, pages 5286–5295. PMLR, 2018.

Eric Wong, Leslie Rice, and J. Zico Kolter. Fast is better than free: Revisiting adversarial training. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=BJx040EFvH`.

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

Chang Xiao, Peilin Zhong, and Changxi Zheng. Enhancing adversarial defense by k-winners-take-all. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=Skgvy64tvr.

Cihang Xie and Alan Yuille. Intriguing properties of adversarial training at scale. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=HyxJhCEFDS.

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

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=Sy8gdB9xx.

Dinghuai Zhang, Tianyuan Zhang, Yiping Lu, Zhanxing Zhu, and Bin Dong. You only propagate once: Accelerating adversarial training via maximal principle. In *Advances in Neural Information Processing Systems*, pages 227–238, 2019a.

Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric Xing, Laurent El Ghaoui, and Michael Jordan. Theoretically principled trade-off between robustness and accuracy. In *International Conference on Machine Learning*, pages 7472–7482, 2019b.

Jingfeng Zhang, Jianing Zhu, Gang Niu, Bo Han, Masashi Sugiyama, and Mohan Kankanhalli. Geometry-aware instance-reweighted adversarial training. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=iAX0l6Cz8ub.

Haizhong Zheng, Ziqi Zhang, Juncheng Gu, Honglak Lee, and Atul Prakash. Efficient adversarial training with transferable adversarial examples. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1181–1190, 2020.

Xuyang Zhong, Yixiao Huang, and Chen Liu. Towards efficient training and evaluation of robust models against l_0 bounded adversarial perturbations. 2024.