Reducing Accuracy Gap in Adversarial Training by Discriminating Adversarial Samples

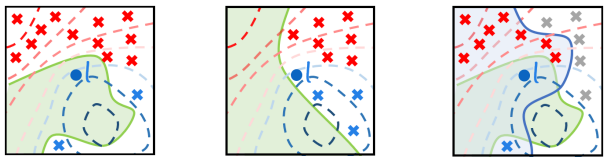
Abstract

Adversarial training is a time-tested effective defense method for norm-bounded adversarial attacks. However, it is known that adversarial training usually causes non-trivial accuracy degradation, leading to a large gap between robust accuracy and natural accuracy. This can be largely attributed to adversarial samples having a different distribution from natural samples. Therefore directly training with adversarial samples causes undesirable side-effects. We propose a novel method to mitigate such distribution disparity. We devise an unsupervised surrogate game to train a discriminator that classifies adversarial samples to in-distribution or out-of-distribution. The latter is precluded during inference. We compare our method with four state-of-the-art adversarial training methods and two discriminators under three threat models. Our method can bring robust accuracy for adversarially trained models closer to the natural accuracy whereas others suffer substantial degradation.

Introduction

Adversarial training is one of the most effective methods to counter adversarial attacks on deep learning models (Mądry et al. 2018). It leverages various attack methods to generate adversarial samples and adds such samples to the training set to improve model robustness. The state-of-the-art adversarial training techniques can greatly improve model robustness. It has prominent advantages over other methods such as random smoothing (Jia et al. 2020) which has substantial runtime overhead, model verification (Katz et al. 2017) which can verify a relatively smaller bound and often has scalability issues, and runtime adversarial detection (Feinman et al. 2017) that is vulnerable to white-box attack. Therefore, it is becoming a standard step in model training.

However, existing adversarial training techniques often induce non-trivial accuracy gap between clean and adversarial samples. The root cause is that adversarial samples follow a distribution that is different from the clean sample distribution, which is also called *natural distribution*. As such, directly using adversarial samples during inference undermines the model's ability to predict natural distribution. Meanwhile, using such samples in training compromises the model's ability to learn the natural distribution, despite of the ability of improving robustness. These are called the *distribution disparity problem* (Xie and Yuille 2020). Figure (1a) and (1b) illustrate the problem. Figure (1a) shows the case of a naturally trained model. The solid blue circle in the center denotes a natural test sample of some label ℓ. The dashed contour lines delineate the natural distribution density. The darker blue/red color indicates a higher/lower density of ℓ. Samples in a low density area are out-of-distribution and most of the malicious adversarial samples are in this area. The green area depicts the classification boundary of the model and the irregular shape indicates its lack of robustness. The crosses represent the adversarial samples being classified to other labels than ℓ by the model. The square box denotes the l_p ball of the central sample. Figure (1b) shows the case of an adversarially trained model. The adversarial training works by including adversarial samples (crosses) in the training. Observe that the robustness is improved as the green area is enlarged compared to Figure (1a). However, there still exist a significant number of out-of-distribution samples misclassified by the model, due to the hardness of this task. Moreover, observe that the decision boundary shifts away from blue area (dense area), which leads to the degradation of the clean accuracy. This is because most of the adversarial samples are concentrated on the low-density area, unlike natural samples.



(a) Normal Classifier (b) Adv. Trained Classifier (c) Adversarial Detector

Figure 1: Distribution disparity problem in adversarial training and adversarial sample detection

A straightforward idea is using a discriminator to determine if an adversarial sample falls in the natural distribution. Out-of-distribution samples are precluded from training and inference. Runtime adversarial sample detection techniques can be considered an approximation of discriminator (Feinman et al. 2017; Ma et al. 2019). These techniques often model the natural distribution in the input space or feature space. For example, *kernel density estimation* (KDE) (Feinman et al. 2017) approximates the density of internal activations with kernel functions and uses the density information to identify adversarial samples. These methods can be used to filter out unnatural samples so that they are not used in adversarial training and/or inference. However, these discriminators/detectors are closely coupled with the subject model, making the training of a high quality discriminator very challenging. First, the irregular/complex classification boundary of the subject model makes the acquisition of good classification boundary between natural and adversarial samples difficult. Second, adversarial training of the discriminator/detector is often less effective and hence the resulted discrim-

inator/detector is often vulnerable to adaptive attack (Yin, Kolouri, and Rohde 2019). The reason again is the close coupling with the subject model that may not have a good classification boundary to begin with. Intuitively, *the discriminator is not able to distinguish natural and unnatural samples in general, but rather samples that are in the distribution denoted by the model and the rest*. Figure (1c) illustrates the concept. The solid blue line denotes the decision boundary of the discriminator/detector. It depends on the subject model's decision boundary. The irregularity of the model's boundary makes it difficult to achieve a good boundary for the discriminator. Furthermore, using the boundary to preclude adversarial samples in training or testing leads to misclassification due to its misalignment with the natural distribution (i.e., dashed blue contour lines). In Figure (1c), the grey crosses in the dense area are undesirably precluded. Please see our results in Section Comparison with Discriminators.

To address the distribution disparity problem, we argue that adversarial samples ought to be *representative* and *in-distribution*. The first goal is to ensure that these samples can effectively change the decision boundary such that the model becomes more robust. As such, a simple idea of precluding out of distribution adversarial samples may not work as that may filter out representative adversarial samples. The second goal is to ensure that the adversarial samples do not substantially compromise the model's ability of learning the natural distribution. However, finding representative and in-distribution adversarial samples is hard. In addition, training a general discriminator is highly challenging. Note that discriminators in pre-trained GANs cannot serve the purpose because (1) they themselves are vulnerable to adversarial perturbations and (2) they are closely coupled with the corresponding generators in the pre-trained GANs and hence not general purpose.

In this paper, through rigorous problem reduction, we decouple the training into two parts: one is the standard adversarial training that considers only the representativeness goal and the other is a general-purpose and *model agnostic* discriminator that is trained by a sophisticated surrogate game. At the inference time, the discriminator is used to filter out unnatural samples. The remaining natural samples are then fed to the adversarial trained model to derive the final classification results. Our evaluation shows that the method substantially reduces the accuracy gap, compared to the state-of-the-art adversarial training methods.

In summary, our main contributions include:

- We reduce the hard problem of adversarial training on dense area into two independent parts: adversarial training and a surrogate game. We theoretically prove the decomposed problem upper-bounds the original one.
- We propose a novel δ-Hardness-Adjustable Surrogate Game for effective optimization.
- We build a prototype in TensorFlow. Our evaluation on two datasets and four state-of-the-art adversarial training methods and two discriminators under oblivious/unseen/adaptive attacks shows our method is effective and achieves the robust accuracy closer to the natural accuracy.

Related Work

Adversarial Training. A lot of standard adversarial training methods have been devised based on different attacks: L-BFGS (Szegedy et al. 2014), FGSM (Goodfellow, Shlens, and Szegedy 2015), one-step methods (Kurakin, Goodfellow, and Bengio 2017), PGD (Mądry et al. 2018; Shafahi et al. 2019; Kannan, Kurakin, and Goodfellow 2018; Xie et al. 2020; Zhang et al. 2020b). Regularization (Zhang et al. 2019; Wang et al. 2019) and early stop (Rice, Wong, and Kolter 2020) have been proposed to improve adversarial training. TRADES (Zhang et al. 2019) utilized Kullback-Leibler regularization to bound the adversarial risk and use an coefficient to control the trade-off between natural accuracy and robust accuracy. Rice *et al.* (Rice, Wong, and Kolter 2020) proposed early-stop adversarial training to address the robust overfitting. Tramèr *et al.* (Tramèr et al. 2017) introduced ensemble adversarial training to improve robustness against black-box attacks. Unsupervised adversarial training (UAT) (Alayrac et al. 2019) and robust self-training (RST) (Carmon et al. 2019) use additional unlabeled samples to improve robustness. While existing methods treat all adversarial samples as the same, our method only utilizes adversarial samples within the high-density area to address the distribution disparity issue. Most of the existing adversarial training methods can be employed to solve the standard adversarial training part in our relaxed problem.

Related Detection Methods. Kernel density estimation (KDE) (Feinman et al. 2017) models the outputs of the final hidden layer with a Gaussian Mixture Model. An input is considered adversarial sample if its estimated likelihood is smaller than a threshold. However, KDE is unable to defend adaptive attacks (Carlini and Wagner 2017a). Pang *et al.* (Pang et al. 2018) proposed the reverse cross-entropy (RCE) loss to facilitate KDE detection. These density based methods are usually easy to breach with adaptive attacks, since their metric is manually designed. Generative adversarial training (GAT) (Yin, Kolouri, and Rohde 2019) adversarially trains N binary discriminators $\{f_k\}_{k=1}^N$ for N-class classifier g to detect adversarial input. If g predicts an input x as label k, f_k is then selected for detection. And if $f_k(x)$ is smaller than a selected threshold, x is considered as adversarial sample. As mentioned previously, GAT will draw a decision boundary closely coupled with the classifier, and this coupling is problematic for good performance. Instead, our method in theory estimates the density through a surrogate game, which is independent from the underlying classifier.

Verification/Certification-based Defenses. There are other defense mechanism parallel to adversarial training. AI^2 (Gehr et al. 2018) applies abstract interpretation to prove over-approximated properties of convolutional networks. Wong and Kolter (Wong and Kolter 2018) used linear relaxation. Some work (Cohen, Rosenfeld, and Kolter 2019; Lecuyer et al. 2019) added random noises to classifiers. COLT (Balunovic and Vechev 2019) combines adversarial training and provable defenses. Zhang *et al.* (Zhang et al. 2020a) utilized linear programming and interval bound propagation. Different from verified/certified defenses, our method falls into the empirical defense category.

Design

Our overarching goal is to use *representative* and *in-distribution* adversarial samples in training. The former requirement is to ensure that the adversarial samples are effective in improving robustness. The latter requires that the adversarial samples are natural and hence avoids accuracy degradation. As such, a simple method that first generates adversarial samples and then precludes those that are not in natural distribution is not effective in practice because many representative adversarial samples are filtered out due to the lack of a good general purpose discriminator for natural samples (see results in Section Experiment). On the other hand, existing adversarial training methods ensure representativeness but neglect the in-distribution requirement. As a result, they suffer non-trivial accuracy degradation. In this section, we show how we gradually (and rigorously) reduce the objective to a realizable process that consists of a standard adversarial training step and a stand-alone discriminator training step. The discriminator is used at the inference time to screen out unnatural (adversarial) samples to prevent accuracy degradation. It is trained by a surrogate game that increases difficulty over time to achieve the best discrimination capabilities. In the following, we first define the notations used in our description. Then we discuss the detailed problem reduction and the final design.

Notations. We use a capital symbol (e.g. X) to denote a random variable and a calligraphic font (e.g. $\mathcal{X}$) for the corresponding distribution. We denote a data sample as $X \in \mathbb{R}^d$ with a label $Y \in \mathbb{N}$. They jointly follow distribution $(\mathcal{X}, \mathcal{Y})$. Our technique aims to harden a classifier $g(\cdot;\theta)$ from domain $\mathbb{R}^d$ to classes $\mathbb{N}$. Our technique leverages a discriminator denoted as $f(\cdot;\sigma)$ from $\mathbb{R}^d$ to real interval $[0,1]$. θ and σ are parameters. We represent the loss functions for f and g as $\mathcal{L}_f$ and $\mathcal{L}_g$, respectively. $B(x,\epsilon)$ represents a closed set of feasible adversarial samples which is defined through some norm (i.e., ℓ_∞-norm in our experiment) and budget ϵ. Given a distribution $\mathcal{A}$ and a set $\mathbb{S}$, $\mathcal{A}|\mathbb{S}$ represents $\mathcal{A}$ truncated on $\mathbb{S}$. $\mathbb{1}(\cdot)$ represents a characteristic function; $\circ$ represents the Hadamard product between vectors. $\mathcal{U}(\cdot)$ denotes a uniform distribution over a range. □

Adversarial Training with a Discriminator

With the aforementioned notations, the standard adversarial training can be described by a minmax formula as follows.

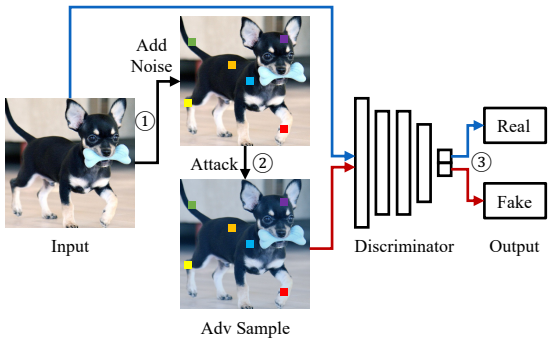
$$\min_{\theta} \mathbb{E}_{(X,Y)\sim(\mathcal{X},\mathcal{Y})} \left[\max_{x' \in B(X,\epsilon)} \mathcal{L}_g(g(x';\theta), Y) \right] . \quad (1)$$

To support our later formal discussion, we consider that an adversarial sample A_θ, dependent on (X,Y), is drawn from a distribution $\mathcal{A}_\theta$. We use $\mathcal{U}$ to randomly select among multiple maximal values. The conditional definition is as follows,

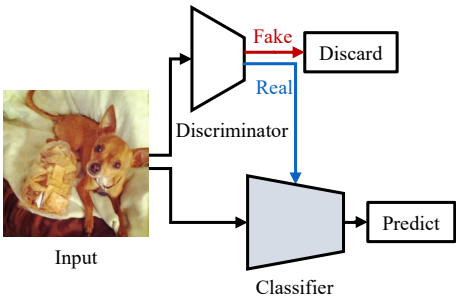
$$\begin{aligned} A_\theta^{(X,Y)} &\sim \mathcal{A}_\theta, \\ A_\theta|(X,Y) &\sim \mathcal{U}\left[\arg\max_{x' \in B(X,\epsilon)} \mathcal{L}_g(g(x';\theta), Y) \right] . \end{aligned} \quad (2)$$

Assuming the argmax function of X and Y is measurable, the loss function Eq. (1) can hence be rewritten as follows.

$$\min_{\theta} \mathbb{E}_{(A_\theta,Y)\sim(\mathcal{A}_\theta,\mathcal{Y})} \mathcal{L}_g(g(A_\theta;\theta), Y) . \quad (3)$$



(a) Training Phase



(b) Inference Phase

Figure 2: Illustration of δ-hardness-adjustable game. This game stresses a balance between hardness and effective optimization (Eq. 8). (a) During training, in step ①, we randomly sample some positions for masking (colorful dots) and add the uniform noise on the masked area. The amount of masking area is controlled by δ, a.k.a the hardness of the game. In step ②, the attacker generates malicious samples by modifying the unmasked area. In step ③, the discriminator learns to tell the difference between the adversarial samples and benign samples. (b) During inference, the system simply rejects low-density samples and only predicts for high-density ones.

As mentioned earlier, distribution $\mathcal{A}_\theta$ and $\mathcal{X}$ can be very different, making it difficult to achieve both robust accuracy and natural accuracy. A plausible solution is to harden the model with only adversarial samples in the distribution of natural samples and preclude the out-of-distribution ones. As such, we might retain good accuracy but fail to gain robustness as many adversarial samples are precluded. On the other hand, if we allow arbitrary adversarial samples, these samples may be so out of the natural distribution that the natural accuracy degrades substantially, in spite of the better robustness. *Our idea is to acquire adversarial samples in a common high-density areas of both the natural and the adversarial distributions and use such samples in training. We also regulate the differences between the two distributions to make sure they are not far off.* Specifically, given the density functions of $\mathcal{X}$ and $\mathcal{A}_\theta$, denoted as $\rho_X(\cdot)$ and $\rho_{\mathcal{A}_\theta}(\cdot)$ respectively, we define the *high density area* as $\mathbb{S}_\theta = \{\frac{\rho_X(x)}{\rho_{\mathcal{A}_\theta}(x)} \geq c\}$ where c is a constant to control the balance. For example, $c = 1$ means we only train with the adversarial samples when they are as likely in the adversarial distribution as in the natural distribution. By avoiding the low-density adversarial samples,

we can redefine the goal as Eq. (4).

$$\min_{\theta} \mu(\mathbb{S}_\theta)\mathbb{E}_{(A_\theta,Y)\sim(\mathcal{A}_\theta|\mathbb{S}_\theta,\mathcal{Y})}\mathcal{L}_g(g(A_\theta;\theta),Y) + 2\mathbb{D}_{JS}(\mathcal{A}_\theta,\mathcal{X}) \tag{4}$$

The Jensen–Shannon divergence $\mathbb{D}_{JS}$ is a penalty term to make sure the adversarial distribution $\mathcal{A}_\theta$ is close to $\mathcal{X}$. Intuitively, by this term we want to draw adversarial samples from a distribution that is sufficiently natural, just like how humans harden their decision makings with realistic samples. Since $\mathcal{A}_\theta$ may have a disjoint support with $\mathbb{S}_\theta$ which makes the expectation ill-defined, we scale the first term with $\mu(\mathbb{S}_\theta)$ to rule out this case, where μ is the measure of $\mathcal{A}_\theta$.

The term $\mathbb{S}_\theta$ requires the knowledge of $\rho_X(\cdot)$ and $\rho_{\mathcal{A}_\theta}(\cdot)$, which is not practically attainable. Hence, we leverage a discriminator f to take its place. In particular, $f(x;\sigma)$ can be considered as the probability of a sample from $\mathcal{A}_\theta$. Inversely, $1-f(x;\sigma)$ represents the probability of that from $\mathcal{X}$. We use $\mathbb{T}_\sigma = \{x|f(x;\sigma) \le \alpha\}$ to approximate $\mathbb{S}_\theta$. We can rewrite the goal using the discriminator f as follows,

$$\min_{\theta,\sigma} \overbrace{\mu(\mathbb{T}_\sigma)\mathbb{E}_{(A_\theta,Y)\sim(\mathcal{A}_\theta|\mathbb{T}_\sigma,\mathcal{Y})}\mathcal{L}_g(g(A_\theta;\theta),Y)}^{\text{Classification Loss}} + \overbrace{\mathbb{E}_{\mathcal{A}_\theta}\mathcal{L}_f(f(A_\theta;\sigma),1) + \mathbb{E}_{\mathcal{X}}\mathcal{L}_f(f(X;\sigma),0)}^{\text{Discrimination Loss}} . \tag{5}$$

Here $\mathcal{L}_f$ is a cross-entropy loss function. In the following, we prove that using the discriminator f provides a good approximation of the goal in Eq. (4).

Lemma 1. *Given the cross-entropy loss $\mathcal{L}_f(x,0) = -\log(1-x)$, $\mathcal{L}_f(x,1) = -\log(x)$, and assuming $\mathbb{T}_\sigma = \mathbb{S}_\theta$, we have* Eq.(4) $\le$ Eq.(5)*, and this upper bound is tight when*

$$f(\cdot;\sigma) = \frac{\rho_{\mathcal{A}_\theta}(\cdot)}{\rho_{\mathcal{X}}(\cdot) + \rho_{\mathcal{A}_\theta}(\cdot)}, \text{and } c = \frac{1-\alpha}{\alpha}.$$

Note that c is the constant used to define $\mathbb{S}_\theta$ and α is the threshold in the definition of $\mathbb{T}_\sigma$. The proof of the lemma is in Appendix A. Lemma 1 and Eq. (5) provide a way to measure the density ratio without explicitly expressing the distributions. However, the two terms in Eq. (5) are interdependent. $\mathbb{T}_\sigma$ in the classification loss depends on the discriminator and $\mathcal{A}_\theta$ in the discrimination loss depends on the classifier. This interdependence is problematic for practical optimization. Specifically, the dependency causes competition and oscillation during adversarial training, and makes convergence difficult if not impossible. To this end, we further relax the problem and decouple the loss into two independent parts.

Theorem 1. *Eq. (5) is upper-bounded by two independent terms,*

$$\text{Eq.}(5) \le \overbrace{\min_{\sigma} \mathbb{E}_{\mathcal{A}_\sigma}\mathcal{L}_f(f(A_\sigma;\sigma),1) + \mathbb{E}_{\mathcal{X}}\mathcal{L}_f(f(X;\sigma),0)}^{\textit{Surrogate Game}} + \overbrace{\min_{\theta} \mathbb{E}_{(\mathcal{A}_\theta,\mathcal{Y})}\mathcal{L}_g(g(A_\theta;\theta),Y)}^{\textit{Standard Adv Training}} \tag{6}$$

where adversarial sample A_σ (dependent on X) is drawn from distribution $\mathcal{A}_\sigma$, as defined in the following.

$$A_\sigma^{(X)} \sim \mathcal{A}_\sigma, A_\sigma|X \sim \mathcal{U}\left[\underset{x'\in B(X,\epsilon)}{\arg\max}\ \mathcal{L}_f(f(x';\sigma),1)\right] \tag{7}$$

Its proof is in Appendix A. Intuitively, compared to Eq. (5), the first term is relaxed by removing the truncation of $\mathcal{A}_\sigma$ on $\mathbb{T}_\sigma$, which makes it the same loss term as in standard adversarial training. In the second term, according to Eq. (7), the adversarial sample A_σ is to cause the discriminator to misclassify, whereas A_θ in Eq. (5) is to cause the subject model to misclassify. *Since the discriminator f is derived from natural and adversarial samples, the second loss term suggests that we can devise a surrogate game to train f as part of the process*. More importantly, the two terms are independent such that the hardening process can be decoupled to a standard adversarial training and the surrogate game. At inference time, the (trained) discriminator is first used to filter out the unnatural samples and the hardened model is then used to produce the classification results.

δ-Hardness-Adjustable Surrogate Game. Practically, the surrogate game mentioned above is hard for the discriminator. Intuitively, the adversary can introduce arbitrary perturbations, e.g., simply making $A_\sigma^{(X)} = X$, such that the discriminator has substantial difficulties learning how to distinguish adversarial and natural samples. Thus we propose an innovative *δ-hardness-adjustable Surrogate Game* (δ-SG) which strikes a balance in the hardship and effectiveness of training. The basic idea is to use an argument δ to control the level of hardship. Specifically, allowing the adversary to mutate a given sample in an arbitrary fashion (in order to defeat the discriminator) represents the hardest extreme, whereas disallowing the adversary to mutate at all represents the easiest extreme. The argument $\delta \in [0,1]$ denotes some middle point in between the two extremes. Formally, we construct adversarial samples $H_{\sigma,\delta}$ as follows.

$$\begin{aligned}&\text{Let } h(m,x_1,x_2) = m\circ x_1 + (1-m)\circ x_2, X\sim\mathcal{X},\\ &N\sim\mathcal{U}(B(X,\epsilon)), R\sim\mathcal{U}([0,1]^d), m(R) = \mathbb{1}[R<\delta],\\ &H_{\sigma,\delta}^{(X,N,R)} \sim \mathcal{H}_{\sigma,\delta}, H_{\sigma,\delta}|(X,N,R) \sim\\ &\mathcal{U}\left[\underset{h(m(R),N,x')|x'\in B(X,\epsilon)}{\arg\max} \mathcal{L}_f(f(h(m(R),N,x');\sigma),1)\right]\end{aligned} \tag{8}$$

Function h uses a binary vector mask m to combine two samples into one. For each dimension, it either takes the value from the corresponding dimension in x_1, or that from x_2. We use it to restrict the dimensions that can be perturbed for an adversarial sample. The mask m is randomly generated and controlled by the hardness parameter δ and a uniform random seed R. For the dimensions that are not mutable, their values are filled in from a noise vector N, which is a random sample from $\mathcal{U}(B(x,\epsilon))$. The δ-SG game is then formally defined as follows.

$$\mathcal{G}(\delta) = \min_{\sigma} \mathbb{E}_{H\sim\mathcal{H}_{\sigma,\delta}}\mathcal{L}_f(f(H;\sigma),1) + \mathbb{E}_{X\sim\mathcal{X}}\mathcal{L}_f(f(X;\sigma),0). \tag{9}$$

Here, we use $\mathcal{G}(\delta)$ to denote the minimum value that can be achieved by the δ-SG game. Intuitively, a smaller $\mathcal{G}(\delta)$ value indicates a harder game. The following theme shows that $\delta = 0$ and $\delta = 1$ denote the hardest and easiest games

Algorithm 1: Discriminator Training by Hardness-Adjustable Surrogate Game

Input : Training Data $(\mathcal{X}, \mathcal{Y})$, training epochs T, curriculum learning epochs T_c, number of batches each epoch L, surrogate game hardness limit δ, adversary iteration I, set of adversary learning rates $\mathcal{E}$, set of feasible adversarial samples $B(\cdot, \epsilon)$.

Output : δ-SG robust discriminator σ

1 $\sigma \leftarrow$ `RandomInit()`, $\hat{\delta} \leftarrow 1.0$
2 **for** $t \leftarrow 1$ **to** T **do**
3 **for** $b \leftarrow 1$ **to** L **do**
4 $x_b \sim \mathcal{X}$ `// Sample a batch of data`
5 Sample a learning rate $\eta \sim \mathcal{E}$ `// Ensemble of attackers`
6 Sample seed R and noise N through $\hat{\delta}$
7 $x'_b \leftarrow h(m(R), N, x_b)$ `// see Eq.(8)`
8 **for** $i \leftarrow 1$ **to** I **do**
9 $x'_b \leftarrow x'_b + \eta\,\text{sign}[\nabla_\sigma \mathcal{L}_f(f(x'_b;\sigma), 1)]$ `// Adversarial optimization`
10 $x'_b \leftarrow$ `Clip(`x'_b, $B(x_b, \epsilon)$`)` `// Clip within the boundary`
11 $x'_b \leftarrow h(m(R), N, x'_b)$ `// Keep the game rule`
12 Update σ with $\nabla_\sigma [\mathcal{L}_f(f(x'_b;\sigma), 1) + \mathcal{L}_f(f(x_b;\sigma), 0)]$
13 **if** $t \leq T_c$ **then** $\hat{\delta} \leftarrow \hat{\delta} - (1-\delta)/(T_c L)$ `// Increase hardship level till reaching` δ

respectively, and the hardship monotonically decreases when δ changes from 0 to 1.

Theorem 2. *(1) When $\delta = 0$, the δ-SG is equivalent to that in the Surrogate Game in Eq. (6) and when $\delta = 1$, the δ-SG is equivalent to the following.*

$$\mathcal{G}(1) = \min_{\sigma} \mathbb{E}_{U \sim \mathcal{U}(B(X,\epsilon))} \mathcal{L}_f(f(U;\sigma), 1) + \mathbb{E}_{X \sim \mathcal{X}} \mathcal{L}_f(f(X;\sigma), 0)\,. \tag{10}$$

2) We have $\delta_1 \geq \delta_2 \implies \mathcal{G}(\delta_1) \leq \mathcal{G}(\delta_2)$ assuming B is defined on l_∞-norm.

The proof is in Appendix A. The game of $\delta = 1$ aims to differentiate the input distribution and a uniformly perturbed distribution. Intuitively, it's to distinguish inputs with random noises from clean inputs and hence much easier than differentiating inputs with crafted perturbations (to fool the discriminator) from the clean ones. The proof of (2) is to show that a larger δ indicates a smaller domain of the min function in Eq. (9), suggesting it's easier to find the minimum.

Training Algorithm. The high-level steps of the discriminator training process are illustrated in Figure 2a. Algorithm 1 describes the detailed process. Note that besides the discriminator, we also apply standard adversarial training to the subject model, as indicated by the first term of Eq.(7). Since δ-SG by its nature allows different levels of hardness, we leverage curriculum learning where difficulty gradually increases. To increase generalization, we also uses ensemble of attackers by using different learning rates in during training. Specifically, initially, the algorithm set the starting $\hat{\delta}$ to 1.0 (i.e., the easiest) at line 1. It will gradually decreases to δ, the target hardness level. Line 5 samples a random learning rate, denoting a random adversary. Line 6 samples a mask m and N, the random perturbations of x_b. The former is used to combine N and x_b to acquire x'_b (i.e., Eq.(8)). In lines 9-11, N stays the same whereas x'_b is mutated through I steps of adversarial attack. Line 12 updates the discriminator weight values. Line 13 updates $\hat{\delta}$. Observe that during the first T_c epochs, δ gradually decreases from 1. It reaches 0 and doesn't change after T_c epochs.

Inference with Discriminator

Figure 2b shows the inference phase of our defense. Given well-trained $f(\cdot;\sigma)$,$g(\cdot;\theta)$, and $(X, Y) \sim (\mathcal{X}, \mathcal{Y})$ as we described earlier, we consider the following defense. The discriminator judges whether the sample falls into the low density area. When the sample is from low density area, the system simply discard it. In another word, we negligibly shrink the domain of feasible input in exchange of better robustness. In this manner, the model correctly classifies the sample iff. $g(X;\theta) = Y \wedge f(X;\sigma) < \alpha$. Inversely an attacker succeeds iff. $g(X;\theta) \neq Y \wedge f(X;\sigma) < \alpha$. Note that this setting is somewhat similar to defense, with the difference that we reject samples with a small probability (1%).

Experiment

Experimental Setup

Datasets and Models. Two datasets are used: CIFAR-10 (Krizhevsky, Hinton et al. 2009) and Tiny ImageNet (Deng et al. 2017). CIFAR10 includes 60000 images of size 32x32 for 10 classes. Tiny ImageNet includes 100000 images of size 64x64 for 200 classes. We use the original classification models as the baselines.

Threat Models. We consider three types of threat models. 1) Oblivious attack: attackers have the access to the classifier but know nothing about the new defense. 2) Unseen attack: attackers use a generation mechanism which is quite different from attack used in the adversarial training. 3) Adaptive attack: attackers know everything of the system and the defenses.We adopt the common setting of ℓ_∞-norm and set $\epsilon = 8.0/255$. We set the α in a way where the false-positive ratio on training data equals 1%.

Baselines. We choose three popular adversarial training methods Mądry (Mądry et al. 2018), TRADE(Zhang et al. 2019), UAT (Alayrac et al. 2019), as well as two detection methods including KDE (Feinman et al. 2017) and GAT (Yin, Kolouri, and Rohde 2019) as our baselines. The introduction to these baselines can be found in Section Related Work.

Metrics. For each defense method, we report three metrics. The accuracy (Acc.) means the accuracy for either the normal samples (without attacks) or adversarial ones (with different attacks) using the classifier alone. The low density rate (LD Rate) means how many samples are in the low-density area and thus rejected by the discriminator. We also report the accuracy of adversarial samples in high-density area (HD Acc.). This metric mimics the behavior of adaptive attackers who simultaneously evade the discrimination and mislead the classifier. It calculates the accuracy of the best adversarial samples within the high-density area. Specifically,

Table 1: Compare accuracy with high density accuracy on CIFAR10.

Threat	Attack	Mądry			Trade			UAT		
		Acc.	HD Acc.	LD Rate	Acc.	HD Acc.	LD Rate	Acc.	HD Acc.	LD Rate
None	Nature	87.3 %	85.8 %	1.7 %	84.9 %	83.5 %	1.7 %	86.4 %	85.0 %	1.7 %
Oblivious	Auto	44.1 %	85.1 %	99.2 %	53.4 %	86.8 %	99.6 %	57.0 %	84.8 %	99.8 %
	PGD	45.1 %	84.3 %	99.4 %	54.5 %	82.1 %	99.7 %	62.7 %	84.6 %	99.8 %
	CW	46.0 %	85.4 %	99.7 %	53.5 %	82.3 %	99.7 %	62.4 %	84.7 %	99.9 %
	BIM	44.9 %	83.6 %	99.5 %	54.7 %	81.7 %	99.5 %	62.5 %	84.4 %	99.9 %
	MIFGSM	50.4 %	81.7 %	99.0 %	58.7 %	80.8 %	99.7 %	66.4 %	83.8 %	99.8 %
	DeepFool	68.3 %	83.6 %	89.0 %	82.7 %	83.1 %	82.8 %	84.4 %	84.4 %	83.4 %
	JSMA	84.0 %	85.3 %	94.5 %	82.4 %	82.3 %	94.8 %	85.3 %	84.7 %	96.2 %
Unseen	Square	71.2 %	80.9 %	82.3 %	72.5 %	81.1 %	82.6 %	77.6 %	82.4 %	83.7 %
Adpative	g prior	82.0 %	80.6 %	81.6 %	81.3 %	80.2 %	81.8 %	84.4 %	83.0 %	86.9 %
	f prior	47.1 %	77.6 %	79.7 %	54.0 %	76.4 %	78.3 %	62.6 %	79.0 %	81.1 %
	Margin	82.8 %	81.2 %	82.7 %	81.5 %	80.6 %	82.7 %	84.4 %	83.3 %	87.7 %
	Penalty	56.5 %	80.1 %	80.9 %	66.0 %	80.1 %	81.2 %	75.3 %	83.0 %	86.2 %
Worst Case		44.9 %	77.6 %	-	53.5 %	76.4 %	-	62.4 %	79.0 %	-
Acc. Gap		42.4 %	**8.2** %	-	31.4 %	**7.1** %	-	24.0 %	**6.0** %	-

Table 2: Comparison of different discriminators.

Threat	Attack	HD Acc.			LD Rate		
		Ours	KDE	GAT	Ours	KDE	GAT
None	Nature	85.8 %	85.8 %	85.6 %	1.7 %	2.0 %	6.0 %
Oblivious	Auto	**85.1** %	45.2 %	57.3 %	**99.2** %	3.7 %	29.0 %
	PGD	**84.3** %	45.6 %	58.4 %	**99.4** %	3.3 %	17.4 %
	CW	**85.4** %	47.6 %	65.5 %	**99.7** %	4.7 %	24.8 %
	BIM	**83.6** %	45.7 %	57.8 %	**99.5** %	3.5 %	16.8 %
	MIFGSM	**81.7** %	51.3 %	62.4 %	**99.0** %	3.5 %	15.8 %
	DeepFool	**83.6** %	67.6 %	75.9 %	**89.0** %	3.8 %	16.4 %
	JSMA	**85.3** %	82.8 %	84.0 %	**94.5** %	2.3 %	9.1 %
Unseen	Square	**80.9** %	71.5 %	77.1 %	**82.3** %	2.6 %	13.8 %
Adpative	g prior	**80.6** %	46.2 %	51.0 %	**81.6** %	0.8 %	2.8 %
	f prior	**77.6** %	46.1 %	50.3 %	**79.7** %	1.4 %	3.0 %
	Margin	**81.2** %	48.1 %	51.2 %	**82.7** %	5.0 %	3.3 %
	Penalty	**80.1** %	46.1 %	49.6 %	**80.9** %	1.4 %	2.4 %
Worst Case		**77.6** %	45.6 %	49.6 %	-	-	-

each attacker iteratively generates adversarial samples for sample (X, Y) in k steps. Among the adversarial samples it generate at different steps $\{x_i\}_{i=1}^k$, we select one with the largest loss value in the high-density area, defined as $\arg\max_{x\in\{x_i\}_{i=1}^k} \mathcal{L}_g(g(x;\theta), Y)\mathbb{1}(f(x;\sigma) \leq \alpha)$. Note that HD Acc. is a slightly different concept compared with Acc., for it negligibly shrinks the defense area of ℓ_p-ball. For more details on the experimental setup, please refer to Appendix B.

Adversarial attacks

Oblivious Attack. We evaluate our method against the following representative attacks. 1) Auto (Croce and Hein 2020) combines several latest attackers and uses an adaptive learning rate to attack. 2) PGD (Mądry et al. 2018) is claimed as the most representative first-order attack. It uses random start and clips the perturbation with a pre-defined ℓ_∞ bound. 3) CW (Mądry et al. 2018) denotes PGD with Carlini-Wagner (CW) loss function (Carlini and Wagner 2017b). 4) BIM (Kurakin, Goodfellow, and Bengio 2016) is an iterative version of FGSM (Goodfellow, Shlens, and Szegedy 2015). 5) MIFGSM (Dong et al. 2018) extends BIM with momentum. 6) DeepFool (Moosavi-Dezfooli, Fawzi, and Frossard 2016) is an iterative attack with a linear approximation. 7) JSMA (Papernot et al. 2016) perturbs the most important features denoted by the Jacobian-based saliency map.

Unseen Attack. To better evaluate our defense against potential unseen attack(Athalye, Carlini, and Wagner 2018), we include an attacker of different mechanism called square attack (Andriushchenko et al. 2020). It generates adversarial samples by iteratively covering images with shadow squares. This is quite different from gradient-based attacks which we adopts in the surrogate game training. Thus results on the square attack demonstrate the generalization ability of our surrogate game.

Adaptive Attack. In this threat model, the adversaries are fully aware of the defense and thus can design adaptive attacks to breach the model and the defense. To mislead both the classifier and the discriminator, they need to incorporate both models into the loss function. We consider four different strategies. Let us denote the loss function of f and g by $\mathcal{L}_f$ and $\mathcal{L}_g$ and the indicator function of successful attack on f and g by $\mathbb{1}_f$ and $\mathbb{1}_g$. $\mathbb{1}_g(x) = 1$ if the classifier misclassifies x. $\mathbb{1}_f = 1$ if x falls in the low-density area. We practically use CW loss for $\mathcal{L}_f$ and $\mathcal{L}_g$ since it can avoid potential numerical issues(Carlini and Wagner 2017b).

$$f\text{-priority: } \mathcal{L}_{f\text{-prior}}(x) = \mathbb{1}_g(x)\mathcal{L}_f(x) + (1 - \mathbb{1}_g(x))\mathcal{L}_g(x)$$
$$g\text{-priority: } \mathcal{L}_{g\text{-prior}}(x) = \mathbb{1}_f(x)\mathcal{L}_g(x) + (1 - \mathbb{1}_f(x))\mathcal{L}_f(x)$$
$$\text{Penalty: } \mathcal{L}_{\text{Penalty}}(x) = \mathcal{L}_g(x) + \lambda\mathcal{L}_f(x)$$

Intuitively, f-prior strategy minimizes $\mathcal{L}_f$ once the attacker misleads the classifier g. It actually prioritizes the best adversarial sample against f. Oppositely, g-prior strategy minimizes $\mathcal{L}_g$ once the attacker misleads the discriminator f, which prioritizes attacking g. Penalty strategy combines two goals through a penalty weight λ. Margin strategy has the same loss function with g-priority strategy except that it extends $\mathbb{1}_f$ to be a continuous function. In this way, it creates a smooth transition margin between target f and target g. In our experiments, we grid search the hyper-parameters (iterations, step size and penalty weight) for the attacker (details in Appendix D) and uses the best one to report the result.

Comparison with Adversarial Training Baselines

Table 1 shows the whole results of ours with the three adversarial training baselines on CIFAR10. Further results on Tiny ImageNet show our approach is also effective on larger images. Please refer to Table 3 in Appendix F for details.

Elaboration of Metrics. Take the result of Mądry's model in Table 1 as an example. The first row denoted by "None" Threat shows the results on natural samples. Its test accuracy on natural samples is 87.3% (Acc.). LD Rate is model-agnostic. It means our approach filters out 1.7% natural samples. After rejecting the 1.7% natural samples, Mądry model's test accuracy is 85.8% (HD Acc.). We use another row PGD to explain metrics in adversarial settings. Mądry's model have a 45.1% test accuracy on PDG attack. 99.4% PGD-generated

samples fall in the low-density area and thus can be distinguished by our approach. If we select the best PGD-generated samples in high-density areas from the optimization iterations and feed them to Mądry's model, it has 84.3% test accuracy.

Accuracy Gap. From Table 1, we can find large accuracy gaps between the normal accuracy and the robust accuracy on adversarially trained models. Let us use Mądry's model as a example. This model has 44% robust accuracy and 87% normal accuracy, which leads to a 43% accuracy gap. Meanwhile, more than 80% adversarial examples fall into the low-density area, which are discriminated by our approach. By restricting the adversarial samples into high-density area, these adversarial samples are less malicious. HD Acc. reveals the accuracy after using these dense adversarial samples. We can find 32% robust accuracy improvement through this restriction. More importantly, the accuracy gap is now vastly shrinked to 8%.

Adversarial Samples vs Out-of-distribution Samples. One general question is that whether adversarial samples are out-of-distribution samples. And we empirically find that the majority of malicious adversarial samples fall into the low-density area (relative to natural distribution). From table 1, each attacker has different level of attack success rate (ASR), which is defined as $100\% - \text{Acc}$. A higher ASR indicates this attack generates more malicious adversarial samples. And detection rate indicates whether the samples are out-of-distribution. We can find that the detection rate are roughly positively correlated with ASR [1]. For example on Madry's model, DeepFool has 31.7% ASR compared with Auto 55.9% ASR, and DeepFool has 89.0% detection rate while Auto attack has 99.2% detection rate.

Unseen and Adaptive Attacks. We also evaluate on unseen and adaptive attacks. The result shows that our discriminator can generalize to unseen attack well. Although there is a moderate drop (6%) on detection rate compared to DeepFool (which has a similar Acc.), the HD Acc. remains high. Our results on several adaptive attacks further support the robustness of our model. We find the adaptive attacker is not able to simultaneously reduce accuracy on classifier and detection rate on the discriminator. This result suggests it is difficult to optimize in-distribution malicious adversarial samples.

Comparison with Discriminators

In the experiments, we compare our discriminator with other related detectors including density detection (Feinman et al. 2017), and generative trained detector(Yin, Kolouri, and Rohde 2019). We align the thresholds for different detectors to make the HD Acc. comparable for benign samples. Original papers select a larger threshold for better LD Rate, which instead significantly reduce the HD Accuracy. We conduct the experiment on CIFAR10 and Mądry model. Note that the other two discriminators require the knowledge of specific classifier for detection while ours don't, which further enlightens our results. As shown by Table 2, our approach outperforms KDE and GAT by a large margin. On adversarial examples, our approach has the highest HD accuracy and LD Rate. We also report the performance of our standalone discriminator on different δ-SG games and adversarial attacks. Results of Area Under Curve (AUC) score during training and Receiver Operating Characteristic (ROC) during inference can be found in Appendix C.

[1] It doesn't apply to adaptive attacks since the LD rate is adaptively manipulated during attack.

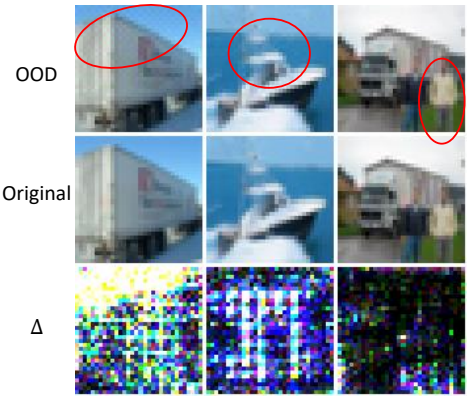


Figure 3: Visualization of adversarial samples for discriminator

Effect on Normal Models

Note that in Theorem 1, the underlying classifier in our derivation are required to be adversarially trained. However, we interestingly find that our discriminator is also effective to some extent on the naturally trained models. Please refer to Appendix F for more details.

How the Discriminator Works

To understand how the discriminator works, we visualize the typical out-of-the-distribution (OOD) sample recognized by the discriminator in Figure 3. These samples exaggerate the feature learned by the discriminator and help us uncover how it works. To generate these samples, we maximize the discrimination loss within the ℓ_p bound through sign gradient optimizer. The first row represents OOD samples, the second row represents benign sample and the third one indicates highlighted perturbance. We find these OOD samples generally present artifacts or unnatural pattern, e.g. the checkerboard pattern in the first column, or the sudden value change in the third column). This suggests that these unnatural patterns are the key features used to recognize the OOD samples.

Conclusion

In this paper, we identify Distribution Disparity as one important reason of the gap between nature accuracy and robust accuracy. We starts from a improved adversarial training and theoretically show that can be reduced to independent goals. We propose a novel δ-Hardness-Adjustable Surrogate Game to train the discriminator in a feasible way. Experimental results show our defense is robust against different threat model and greatly narrow down the gap.

Limitations. Our defense and proof only considers ℓ_∞ norm. As a trade-off for our mathematical reduction, we only consider the problem of reducing of the accuracy gap in this paper. It's among our future work to study the effect on other norms and improving accuracy degradation on the clean data.

References

Abadi, M.; Agarwal, A.; Barham, P.; Brevdo, E.; Chen, Z.; Citro, C.; Corrado, G. S.; Davis, A.; Dean, J.; Devin, M.; Ghemawat, S.; Goodfellow, I.; Harp, A.; Irving, G.; Isard, M.; Jia, Y.; Jozefowicz, R.; Kaiser, L.; Kudlur, M.; Levenberg, J.; Mané, D.; Monga, R.; Moore, S.; Murray, D.; Olah, C.; Schuster, M.; Shlens, J.; Steiner, B.; Sutskever, I.; Talwar, K.; Tucker, P.; Vanhoucke, V.; Vasudevan, V.; Viégas, F.; Vinyals, O.; Warden, P.; Wattenberg, M.; Wicke, M.; Yu, Y.; and Zheng, X. 2015. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. Software available from tensorflow.org.

Alayrac, J.-B.; Uesato, J.; Huang, P.-S.; Fawzi, A.; Stanforth, R.; and Kohli, P. 2019. Are Labels Required for Improving Adversarial Robustness? In Wallach, H.; Larochelle, H.; Beygelzimer, A.; d'Alché-Buc, F.; Fox, E.; and Garnett, R., eds., *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Andriushchenko, M.; Croce, F.; Flammarion, N.; and Hein, M. 2020. Square attack: a query-efficient black-box adversarial attack via random search. In *European Conference on Computer Vision*, 484–501. Springer.

Athalye, A.; Carlini, N.; and Wagner, D. 2018. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *International Conference on Machine Learning*, 274–283. PMLR.

Balunovic, M.; and Vechev, M. 2019. Adversarial training and provable defenses: Bridging the gap. In *International Conference on Learning Representations*.

Carlini, N.; and Wagner, D. 2017a. Adversarial Examples Are Not Easily Detected: Bypassing Ten Detection Methods. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, AISec '17, 3–14. New York, NY, USA: Association for Computing Machinery. ISBN 9781450352024.

Carlini, N.; and Wagner, D. 2017b. Towards Evaluating the Robustness of Neural Networks. In *IEEE Symposium on Security and Privacy (S&P)*, 39–57.

Carmon, Y.; Raghunathan, A.; Schmidt, L.; Duchi, J. C.; and Liang, P. S. 2019. Unlabeled Data Improves Adversarial Robustness. In Wallach, H.; Larochelle, H.; Beygelzimer, A.; d'Alché-Buc, F.; Fox, E.; and Garnett, R., eds., *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Cohen, J.; Rosenfeld, E.; and Kolter, Z. 2019. Certified adversarial robustness via randomized smoothing. In *International Conference on Machine Learning*, 1310–1320. PMLR.

Croce, F.; and Hein, M. 2020. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *ICML*.

Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; and Fei-Fei, L. 2017. Imagenet: A large-scale hierarchical image database.

Dong, Y.; Liao, F.; Pang, T.; Su, H.; Zhu, J.; Hu, X.; and Li, J. 2018. Boosting adversarial attacks with momentum. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 9185–9193.

Feinman, R.; Curtin, R. R.; Shintre, S.; and Gardner, A. B. 2017. Detecting adversarial samples from artifacts. *arXiv preprint arXiv:1703.00410*.

Gehr, T.; Mirman, M.; Drachsler-Cohen, D.; Tsankov, P.; Chaudhuri, S.; and Vechev, M. 2018. AI2: Safety and Robustness Certification of Neural Networks with Abstract Interpretation. In *2018 IEEE Symposium on Security and Privacy (SP)*, 3–18.

Goodfellow, I.; Shlens, J.; and Szegedy, C. 2015. Explaining and Harnessing Adversarial Examples. In *International Conference on Learning Representations*.

Jia, J.; Cao, X.; Wang, B.; and Gong, N. Z. 2020. Certified Robustness for Top-k Predictions against Adversarial Perturbations via Randomized Smoothing. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Kannan, H.; Kurakin, A.; and Goodfellow, I. 2018. Adversarial logit pairing. *arXiv preprint arXiv:1803.06373*.

Katz, G.; Barrett, C.; Dill, D. L.; Julian, K.; and Kochenderfer, M. J. 2017. Reluplex: An efficient SMT solver for verifying deep neural networks. In *International Conference on Computer Aided Verification*, 97–117. Springer.

Krizhevsky, A.; Hinton, G.; et al. 2009. Learning multiple layers of features from tiny images.

Kurakin, A.; Goodfellow, I.; and Bengio, S. 2016. Adversarial Examples in the Physical World. *arXiv preprint arXiv:1607.02533*.

Kurakin, A.; Goodfellow, I. J.; and Bengio, S. 2017. Adversarial Machine Learning at Scale. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Lecuyer, M.; Atlidakis, V.; Geambasu, R.; Hsu, D.; and Jana, S. 2019. Certified robustness to adversarial examples with differential privacy. In *2019 IEEE Symposium on Security and Privacy (SP)*, 656–672. IEEE.

Ma, S.; Liu, Y.; Tao, G.; Lee, W.; and Zhang, X. 2019. NIC: Detecting Adversarial Samples with Neural Network Invariant Checking. In *NDSS*.

Moosavi-Dezfooli, S.-M.; Fawzi, A.; and Frossard, P. 2016. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2574–2582.

Mądry, A.; Makelov, A.; Schmidt, L.; Tsipras, D.; and Vladu, A. 2018. Towards Deep Learning Models Resistant to Adversarial Attacks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

Pang, T.; Du, C.; Dong, Y.; and Zhu, J. 2018. Towards Robust Detection of Adversarial Examples. In Bengio, S.; Wallach, H.; Larochelle, H.; Grauman, K.; Cesa-Bianchi, N.; and Garnett, R., eds., *Advances in Neural Information Processing Systems (NeurIPS)*, volume 31. Curran Associates, Inc.

Papernot, N.; McDaniel, P.; Jha, S.; Fredrikson, M.; Celik, Z. B.; and Swami, A. 2016. The limitations of deep learning

in adversarial settings. In *2016 IEEE European symposium on security and privacy (EuroS&P)*, 372–387. IEEE.

Rice, L.; Wong, E.; and Kolter, J. Z. 2020. Overfitting in adversarially robust deep learning. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, 8093–8104. PMLR.

Rudin, W.; et al. 1976. *Principles of mathematical analysis*, volume 3. McGraw-hill New York.

Shafahi, A.; Najibi, M.; Ghiasi, M. A.; Xu, Z.; Dickerson, J.; Studer, C.; Davis, L. S.; Taylor, G.; and Goldstein, T. 2019. Adversarial training for free! In Wallach, H.; Larochelle, H.; Beygelzimer, A.; d'Alché-Buc, F.; Fox, E.; and Garnett, R., eds., *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Szegedy, C.; Zaremba, W.; Sutskever, I.; Bruna, J.; Erhan, D.; Goodfellow, I.; and Fergus, R. 2014. Intriguing properties of neural networks. In *International Conference on Learning Representations*.

Tramèr, F.; Kurakin, A.; Papernot, N.; Goodfellow, I.; Boneh, D.; and McDaniel, P. 2017. Ensemble adversarial training: Attacks and defenses. *arXiv preprint arXiv:1705.07204*.

Wang, Y.; Zou, D.; Yi, J.; Bailey, J.; Ma, X.; and Gu, Q. 2019. Improving adversarial robustness requires revisiting misclassified examples. In *International Conference on Learning Representations*.

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

Xie, C.; Tan, M.; Gong, B.; Yuille, A.; and Le, Q. V. 2020. Smooth Adversarial Training. *arXiv preprint arXiv:2006.14536*.

Xie, C.; and Yuille, A. L. 2020. Intriguing Properties of Adversarial Training at Scale. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Yin, X.; Kolouri, S.; and Rohde, G. K. 2019. Gat: Generative adversarial training for adversarial example detection and robust classification. In *International Conference on Learning Representations (ICLR)*.

Zhang, H.; Chen, H.; Xiao, C.; Gowal, S.; Stanforth, R.; Li, B.; Boning, D.; and Hsieh, C.-J. 2020a. Towards Stable and Efficient Training of Verifiably Robust Neural Networks. In *International Conference on Learning Representations*.

Zhang, H.; Yu, Y.; Jiao, J.; Xing, E. P.; Ghaoui, L. E.; and Jordan, M. I. 2019. Theoretically Principled Trade-off between Robustness and Accuracy. In Chaudhuri, K.; and Salakhutdinov, R., eds., *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, 7472–7482. PMLR.

Zhang, J.; Xu, X.; Han, B.; Niu, G.; Cui, L.; Sugiyama, M.; and Kankanhalli, M. 2020b. Attacks which do not kill training make adversarial learning stronger. In *International Conference on Machine Learning*, 11278–11287. PMLR.

Appendix

A. Proofs

Proof of Lemma 1

Lemma 1. *Given the cross-entropy loss $\mathcal{L}_f(x,0) = -\log(1-x)$, $\mathcal{L}_f(x,1) = -\log(x)$, and assuming $\mathbb{T}_\sigma = \mathbb{S}_\theta$, we have* $\text{Eq.}(4) \leq \text{Eq.}(5)$*, and this upper bound is tight when*

$$f(\cdot;\sigma) = \frac{\rho_{\mathcal{A}_\theta}(\cdot)}{\rho_{\mathcal{X}}(\cdot) + \rho_{\mathcal{A}_\theta}(\cdot)}\,, \text{and}\, c = \frac{1-\alpha}{\alpha}\,.$$

Proof.

$$\begin{aligned} &\mathbb{E}_{A_\theta \sim \mathcal{A}_\theta} \mathcal{L}_f(f(A_\theta;\sigma),1) + \mathbb{E}_{X\sim\mathcal{X}} \mathcal{L}_f(f(X;\sigma),0) \\ =&\int_{x\in\mathbb{R}^d} -\rho_{\mathcal{A}_\theta}(x)\log(f(x;\sigma)) - \rho_{\mathcal{X}}(x)\log(1-f(x;\sigma)) \end{aligned} \tag{11}$$

We then leverage calculus of variation to minimize the function f. Let $t(x) = f(x;\sigma)$, and define

$$\begin{aligned} l(x,t(x)) &= -\rho_{\mathcal{A}_\theta}(x)t(x) - \rho_{\mathcal{X}}(x)\log(1-t(x)) \\ J(t) &= \int_{x\in\mathbb{R}^d} l(x,t(x))\,. \end{aligned}$$

By Euler-Lagrange equation, the extrema of functional $J(t)$ is attained when $\frac{\partial l}{\partial t} = 0$. Solving this derivative will get the condition

$$t^*(\cdot) = f(\cdot;\sigma) = \frac{\rho_{\mathcal{A}_\theta}(\cdot)}{\rho_{\mathcal{X}}(\cdot) + \rho_{\mathcal{A}_\theta}(\cdot)}\,. \tag{12}$$

Note that $l(x,t(x))$ is a convex function of t given fixed x, and therefore this t^* is the minimum of $J(t)$. Put Equation 12 back into Equation 11, we get

$$\begin{aligned} &\text{eq.}(11) \\ \geq &\int_{x\in\mathbb{R}^d} -\rho_{\mathcal{A}_\theta}(x)\log\frac{\rho_{\mathcal{A}_\theta}(\cdot)}{\rho_{\mathcal{X}}(\cdot)+\rho_{\mathcal{A}_\theta}(\cdot)} - \rho_{\mathcal{X}}(x)\log\frac{\rho_{\mathcal{X}}(\cdot)}{\rho_{\mathcal{X}}(\cdot)+\rho_{\mathcal{A}_\theta}(\cdot)} \\ =&2\mathbb{D}_{JS}(\mathcal{A}_\theta,\mathcal{X})\,. \end{aligned} \tag{13}$$

And we can connect α and c, where

$$\begin{aligned} \mathbb{T}_\sigma &= \{x|f(x;\sigma) < \alpha\} \\ &= \{x|\frac{\rho_{\mathcal{A}_\theta}(x)}{\rho_{\mathcal{X}}(x)+\rho_{\mathcal{A}_\theta}(x)} < \alpha\} \\ &= \{x|\frac{\rho_{\mathcal{X}}(x)}{\rho_{\mathcal{A}_\theta}(x)} > \frac{1-\alpha}{\alpha}\} \\ &= \mathbb{S}_\theta\,. \end{aligned} \tag{14}$$

□

Proof of Theorem 1

Theorem 1. *Eq. (5) is upper-bounded by two independent terms,*

$$\begin{aligned} \text{Eq.}(5) \leq &\overbrace{\min_\sigma \mathbb{E}_{\mathcal{A}_\sigma}\mathcal{L}_f(f(A_\sigma;\sigma),1) + \mathbb{E}_{\mathcal{X}}\mathcal{L}_f(f(X;\sigma),0)}^{\textit{Surrogate Game}} + \\ &\overbrace{\min_\theta \mathbb{E}_{(\mathcal{A}_\theta,\mathcal{Y})}\mathcal{L}_g(g(A_\theta;\theta),Y)}^{\textit{Standard Adv Training}} \end{aligned} \tag{6}$$

where adversarial sample A_σ (dependent on X) is drawn from distribution $\mathcal{A}_\sigma$, as defined in the following.

$$A_\sigma^{(X)} \sim \mathcal{A}_\sigma, A_\sigma|X \sim \mathcal{U}\left[\underset{x'\in B(X,\epsilon)}{\arg\max}\ \mathcal{L}_f(f(x';\sigma),1)\right] \tag{7}$$

Proof. Let

$$g^*(X) = \underset{x' \in B(X,\epsilon)}{\arg\max}\, \mathcal{L}_g(g(x';\theta),1)\,.$$

By extreme value theorem (Rudin et al. 1976), within the closed set $B(x,\epsilon)$, there exist maximum and minimum for the continuous function g. Thus this function g^* is bijective and well-defined. Note that $g^*(X)$ produces a set instead of a single value.

$$\begin{aligned}\text{eq.(5)} = \min_{\theta,\sigma} &\overbrace{\mu(\mathbb{T}_\sigma)\mathbb{E}_{(A_\theta,Y)\sim(\mathcal{A}_\theta|\mathbb{T}_\sigma,\mathcal{Y})}\mathcal{L}(g(A_\theta;\theta),y)}^{\text{(a)}}\\ &+\overbrace{\mathbb{E}_{A_\theta\sim\mathcal{A}_\theta}\mathcal{L}(f(A_\theta;\sigma),1)+\mathbb{E}_{X\sim\mathcal{X}}\mathcal{L}(f(X;\sigma),0)}^{\text{(b)}}\,.\end{aligned}$$

Remember that μ is the measure of $\mathcal{A}_\theta$.

$$\begin{aligned}\text{(a)} &= \min_{\theta,\sigma}\mu(\mathbb{T}_\sigma)\mathbb{E}_{(X,Y)\sim(\mathcal{X},\mathcal{Y})}\frac{\mathcal{L}_g(g(g^*(X);\theta),Y)\mathbb{1}[\mathcal{U}(g^*(X))\in\mathbb{T}_\sigma]}{\Pr_{A_\theta\sim\mathcal{A}_\theta}(\mathcal{U}(g^*(A_\theta))\in\mathbb{T}_\sigma)}\\ &= \min_{\theta,\sigma}\mathbb{E}_{(x,y)\sim(\mathcal{X},\mathcal{Y})}\mathcal{L}_g(g(\mathcal{U}(g^*(X));\theta),Y)\mathbb{1}[\mathcal{U}(g^*(X))\in\mathbb{T}_\sigma]\\ &\le \min_{\theta,\sigma}\mathbb{E}_{(X,Y)\sim(\mathcal{X},\mathcal{Y})}\mathcal{L}_g(g(\mathcal{U}(g^*(X));\theta),Y)\\ &= \min_{\theta}\mathbb{E}_{(A_\theta,Y)\sim(\mathcal{A}_\theta,\mathcal{Y})}\mathcal{L}_g(g(A_\theta;\theta),Y)\,. \qquad \text{(By definition of } \mathcal{A}_\theta)\end{aligned} \tag{15}$$

Similarly, let

$$f^*(X) = \underset{x' \in B(X,\epsilon)}{\arg\max}\, \mathcal{L}_f(f(x';\sigma),1)\,,$$

and we have $\forall x'' \in B(x,\epsilon)$,

$$\begin{aligned}&\mathcal{L}_f(f(\mathcal{U}(f^*(x));\sigma),1)\\ =&\max_{x'\in B(X,\epsilon)}\mathcal{L}_f(f(x';\sigma),1)\\ \ge&\mathcal{L}_f(f(x'';\sigma),1)\,.\end{aligned} \tag{16}$$

Therefore

$$\begin{aligned}&\mathcal{L}_f(f(\mathcal{U}(g^*(x));\sigma),1)\\ =&\mathcal{L}_f(f(\underset{x'\in B(x,\epsilon)}{\arg\max}\,\mathcal{L}_f(g(x';\theta),1);\sigma),1)\\ \le&\mathcal{L}_f(f(\mathcal{U}(f^*(x));\sigma),1)\,. \qquad \text{(See Eq. 16)}\end{aligned}$$

$$\begin{aligned}(b) &= \min_{\theta,\sigma}\mathbb{E}_{X\sim\mathcal{X}}\mathcal{L}_f(f(\mathcal{U}(g^*(X));\sigma),1)+\mathbb{E}_{X\sim\mathcal{X}}\mathcal{L}_f(f(X;\sigma),0)\\ &\le \mathbb{E}_{X\sim\mathcal{X}}\mathcal{L}_f(f(\mathcal{U}(f^*(X));\sigma),1)+\mathbb{E}_{X\sim\mathcal{X}}\mathcal{L}_f(f(X;\sigma),0)\\ &= \mathbb{E}_{A_\sigma\sim\mathcal{A}_\sigma}\mathcal{L}_f(f(A_\sigma;\sigma),1)+\mathbb{E}_{X\sim\mathcal{X}}\mathcal{L}_f(f(X;\sigma),0)\,. \qquad \text{(By definition of } \mathcal{A}_\sigma)\end{aligned} \tag{17}$$

Combining Equation 15 and Equation 17 finishes the proof. □

Proof of Theorem 2

Theorem 2. *(1) When $\delta = 0$, the δ-SG is equivalent to that in the Surrogate Game in Eq. (6) and when $\delta = 1$, the δ-SG is equivalent to the following.*

$$\begin{aligned}\mathcal{G}(1) = \min_{\sigma}\,&\mathbb{E}_{U\sim\mathcal{U}(B(X,\epsilon))}\mathcal{L}_f(f(U;\sigma),1)\\ &+\mathbb{E}_{X\sim\mathcal{X}}\mathcal{L}_f(f(X;\sigma),0)\,.\end{aligned} \tag{10}$$

2) We have $\delta_1 \ge \delta_2 \implies \mathcal{G}(\delta_1) \le \mathcal{G}(\delta_2)$ assuming B is defined on l_∞-norm.

Proof. Let $\boldsymbol{M}_0 \in \mathbb{R}^d$ be an all-zero tensor, and $\boldsymbol{M}_1 \in \mathbb{R}^d$ be an all-one tensor. Recall that we have $\delta - SG$ game defined as

$$\begin{aligned}
&\text{Let } h(m, x_1, x_2) = m \circ x_1 + (1-m) \circ x_2, X \sim \mathcal{X}, \\
&N \sim \mathcal{U}(B(X, \epsilon)), R \sim \mathcal{U}([0,1]^d), m(R) = \mathbb{1}[R < \delta], \\
&H_{\sigma,\delta}^{(X,N,R)} \sim \mathcal{H}_{\sigma,\delta}, H_{\sigma,\delta}|(X,N,R) \sim \\
&\mathcal{U}\left[\underset{h(m(R),N,x')|x' \in B(X,\epsilon)}{\arg\max} \mathcal{L}_f(f(h(m(R), N, x'); \sigma), 1)\right]
\end{aligned} \tag{18}$$

When $\delta = 0$, for some adversarial sample $x \in B(X, \epsilon)$, we have

$$\begin{aligned}
&m(R) = \boldsymbol{M}_0. \\
&N \sim \mathcal{U}(B(X, \epsilon)), \\
&h(m(R), N, x) = x, \\
&H_{\sigma,\delta}^{(X)} \sim \mathcal{H}_{\sigma,\delta},\ \mathcal{H}_{\sigma,\delta}|X = \mathcal{U}[\underset{x \in B(X,\epsilon)}{\arg\max}\, \mathcal{L}((x;\sigma), 1)]\,.
\end{aligned}$$

This is nothing but the definition of the $H_{\sigma,\delta}$ in Surrogate Game.

Similarly, When $\delta = 1$, we have

$$\begin{aligned}
&m(R) = \boldsymbol{M}_1. \\
&N \sim \mathcal{U}(B(X, \epsilon)), \\
&h(m(R), N, x) = N, \\
&H_{\sigma,\delta}^{(N)} \sim \mathcal{H}_{\sigma,\delta} = N\,.
\end{aligned}$$

This is the definition of $H_{\sigma,\delta}$ in game $\mathcal{G}(1)$.

Now, we prove the loss is monotonically decreasing with respect to δ. Let us consider the range of random variable $\{h(m(R), N, x)|x \in B(X, \epsilon)\}$, which is the function of δ, R, N and X. Let us denote it as $\mathcal{C}(\delta, R, N, X)$. For simplicity, we overload some operators including $\circ$ and $+$ for set operation. For a vector x, and a set of vectors X, $x \circ X = \{x \circ x'|x' \in X\}$ and $x + X = \{x + x'|x' \in X\}$. Note that $m(R)$ is parameterized on δ, and we will use $m_\delta(R)$ for a clear notation when the underlying δ is unclear in the context.

Before going into the details, let us prove an important lemma.

Lemma 2. *Given $\delta_1 \geq \delta_2$, others fixed and B defined on ℓ_∞-norm, we have*

$$(1 - m_{\delta_1}(R)) \circ B(X - N, \epsilon) \subset (1 - m_{\delta_2}(R)) \circ B(X - N, \epsilon)\,.$$

Proof. Consider any x where

$$x \in (1 - m_{\delta_1}(R)) \circ B(X - N, \epsilon)\,,$$

and denote $c = X - N$. The inclusion of x implies there exists a y to compute x,

$$\exists y \in B(c, \epsilon), x = (1 - m_{\delta_1}(R)) \circ y\,.$$

In the following, we will construct another y' based on y, and will show that the same x is the result of Hadamard product between $(1 - m_{\delta_2}(R))$ and y' , and thus finish the proof.

We show all-zero tensor $\boldsymbol{M}_0$ and y are both in this set $B(c, \epsilon)$.

$$\begin{aligned}
&N \in B(X, \epsilon) \\
\implies &||X - N||_\infty = ||c||_\infty = ||c - \boldsymbol{M}_0||_\infty \leq \epsilon \\
\implies &\boldsymbol{M}_0 \in B(c, \epsilon)\,.
\end{aligned}$$

Let's define operation $\mathbb{F}(y)$. Intuitively it fills some or all dimensions of y to 0, enumerates all such possible fillings and collects them as a set. Let i represent a dimension index of y,

$$\mathbb{F}(y) = \{y''|y''_i = y_i \vee y''_i = 0\}\,.$$

For any element y'' in the set $\mathbb{F}(y)$, each dimension i of this element y'' takes value from either $\boldsymbol{M}_0$ or y. And note that y and $\boldsymbol{M}_0$ are both in this set $B(c,\epsilon)$. Therefore y'' also belongs to the set $B(c,\epsilon)$.

$$\begin{aligned}
&\forall y'' \in \mathbb{F}(y), ||y'' - c||_\infty \\
\leq &\max(||y-c||_\infty, ||\boldsymbol{M}_0 - c||_\infty) \\
\leq &\epsilon \\
\Longrightarrow &\mathbb{F}(y) \subset B(c,\epsilon)\,.
\end{aligned}$$

Given $\delta_1 \geq \delta_2$ and X,R, and N fixed, by definition, $m_{\delta_1}(R) \geq m_{\delta_2}(R)$ element-wisely. Thus $1 - m_{\delta_1}(R) \leq 1 - m_{\delta_2}(R)$ element-wisely. Let's consider the all the dimensions $j \in J$ where the vector $1 - m_{\delta_1}(R)$ and the vector $1 - m_{\delta_2}(R)$ differ. Since the two vectors are both binary, the only case of their difference will be

$$\begin{aligned}
&\forall j \in J\,, \\
&(1 - m_{\delta_1}(R))_j = 0\,, \text{and} \\
&(1 - m_{\delta_2}(R))_j = 1\,.
\end{aligned}$$

And we construct y' based on y such that,

$$y'_j = \begin{cases} 0 & j \in J \\ y_j & j \notin J \end{cases}\,.$$

Note that $y' \in \mathbb{F} \subset B(c,\epsilon)$, and thus

$$\begin{aligned}
&x \\
=&(1 - m_{\delta_1}(R)) \circ y \\
=&(1 - m_{\delta_2}(R)) \circ y' \\
\in&(1 - m_{\delta_2}(R)) \circ B(c,\epsilon)\,.
\end{aligned}$$

□

Given the lemma 2, let us analyze the property of function h. We have

$$\begin{aligned}
&h(m, x_1, x_2) \circ m \\
=&m \circ x_1 \circ m + (1-m) \circ x_2 \circ m \\
=&m \circ x_1\,.
\end{aligned} \tag{19}$$

And similarly,

$$\begin{aligned}
&h(m, x_1, x_2) \circ (1-m) \\
=&(1-m) \circ x_2\,.
\end{aligned} \tag{20}$$

Combine Eq. (19) and Eq. (20) through definition of $\mathcal{C}$, and we have

$$\begin{aligned}
&\mathcal{C}(\delta, R, N, X) \\
=&h(m(R), N, B(x,\epsilon)) \\
=&h(m(R), N, B(x,\epsilon)) \circ (m(R) + (1 - m(R))) \\
=&m(R) \circ N + (1 - m(R)) \circ B(X,\epsilon)\,.
\end{aligned} \tag{21}$$

From Eq. (21), we have,

$$\begin{aligned}
&\mathcal{C}(\delta_1, R, N, X) \\
=&m_{\delta_1}(R) \circ N + (1 - m_{\delta_1}(R)) \circ B(X,\epsilon) \\
=&m_{\delta_2}(R) \circ N + (m_{\delta_1}(R) - m_{\delta_2}(R)) \circ N + (1 - m_{\delta_1}(R)) \circ B(X,\epsilon) \\
=&m_{\delta_2}(R) \circ N + (m_{\delta_1}(R) - m_{\delta_2}(R)) \circ N + (1 - m_{\delta_1}(R)) \circ B(X,\epsilon) \\
=&m_{\delta_2}(R) \circ N + (m_{\delta_1}(R) - m_{\delta_2}(R)) \circ N + (1 - m_{\delta_1}(R)) \circ N + (1 - m_{\delta_1}(R)) \circ B(X - N,\epsilon) \\
=&m_{\delta_2}(R) \circ N + (1 - m_{\delta_2}(R)) \circ N + (1 - m_{\delta_1}(R)) \circ B(X - N,\epsilon) \\
\subset&m_{\delta_2}(R) \circ N + (1 - m_{\delta_2}(R)) \circ N + (1 - m_{\delta_2}(R)) \circ B(X - N,\epsilon) \qquad (Lemma\,2) \\
=&m_{\delta_2}(R) \circ u + (1 - m_{\delta_2}(R)) \circ B(x,\epsilon) \\
=&\mathcal{C}(\delta_2, R, N, X)\,.
\end{aligned}$$

Thus the domain $\mathcal{C}$ of the $\arg\max$ function in Eq.(18) is monotonically decreasing, which implies this loss function is monotonically decreasing. □

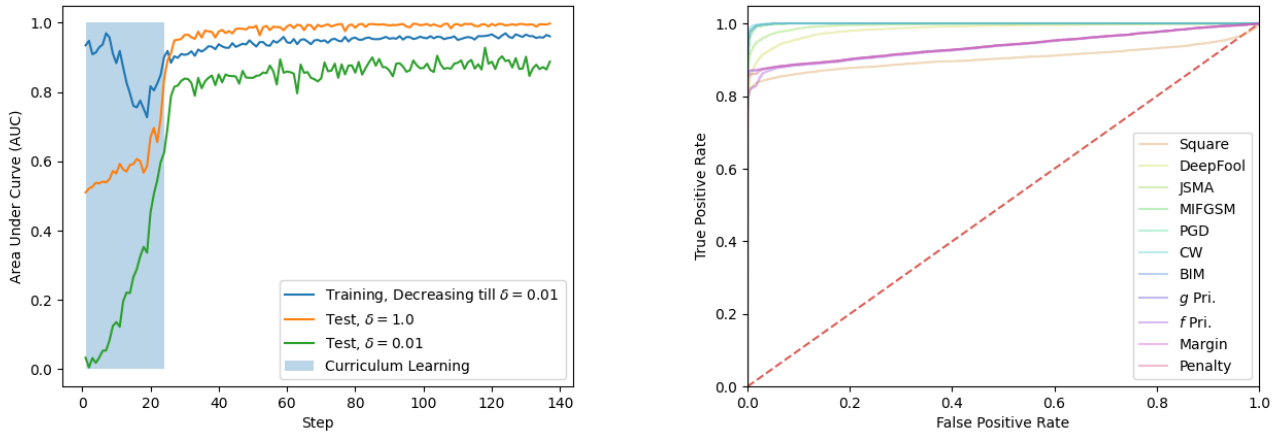
B. Experiment Details

We adopt the ResNet-18 network structure in our discriminator. Specifically, the ResNet-18 model contains an input block and three residual blocks, where each residual block has two residual layers. The numbers of channels for these blocks are [16, 32, 64, 128] with strides of [1, 1, 2, 2], respectively. During the training of the discriminator, we leave out the data augmentations that are commonly employed in the training of classification tasks, such as random cropping and padding. This is because those transformations applied on the training data will greatly shift the data distribution. For instance, the random cropping and padding will introduce black margins on most of the training samples. These black margins however do not exist in real samples, and hence create distribution disparity between the training set and the test set. Such a distribution disparity will hence make the discriminator generalize poorly on the test set. As such, we use the same data processing procedure as for the test set during training. We train the discriminator from scratch and use ℓ_2 model parameters regularization with the penalty weight of $1e$-4. We use Adam optimizer and set the learning rate as $1e$-4 for the training. During the training described in Algorithm 1, we set the total training epoch $T = 140$ and the curriculum learning epoch $T_c = 25$. We use an ensemble set $\mathcal{E} = [1e\text{-}2, 4]$ and sample the learning rate log-uniformly from the set. We conduct the adversarial attack for $I = 1000$ steps in order to cover different attack parameters in the ensemble set.

For the evaluation of model robustness, we use the common settings for Auto, PGD, MIFGSM, CW, BIM, DeepFool from the original papers (Croce and Hein 2020; Mądry et al. 2018; Kurakin, Goodfellow, and Bengio 2016; Dong et al. 2018; Moosavi-Dezfooli, Fawzi, and Frossard 2016). Specifically, we use standard setting introduced in Auto attack. For PGD, MIFGSM, CW and BIM attack, we use step size 2 and iteration numbers 40. For PGD, we randomly initialize the perturbation for 10 times. We increase the attack iterations for JSMA as it only manipulates one pixel a time, which requires more steps for larger input images (e.g., 64×64 for Tiny ImageNet). For the square attack, we run the attack for 1000 steps. For the adaptive attacks, we use the grid search for the four strategies discussed in Section Adversarial attacks and report the worst accuracy. We also stress test the adaptive attack for 1000 steps in Appendix D.

We implemented our framework in TensorFlow (Abadi et al. 2015). Experiments were conducted on a server equipped with 256 GB RAM, two Intel Xeon Silver 4214 2.20GHz 12-core CPUs and eight NVIDIA Quadro RTX 6000 GPUs. We used 4 RTX 6000 GPUs for distributed adversarial training. The training on CIFAR-10 took 1 day and 16 hours. Details of the training are discussed in section C.

C. Performance of the Discriminator



(a) AUC scores of the Surrogate Game during Training

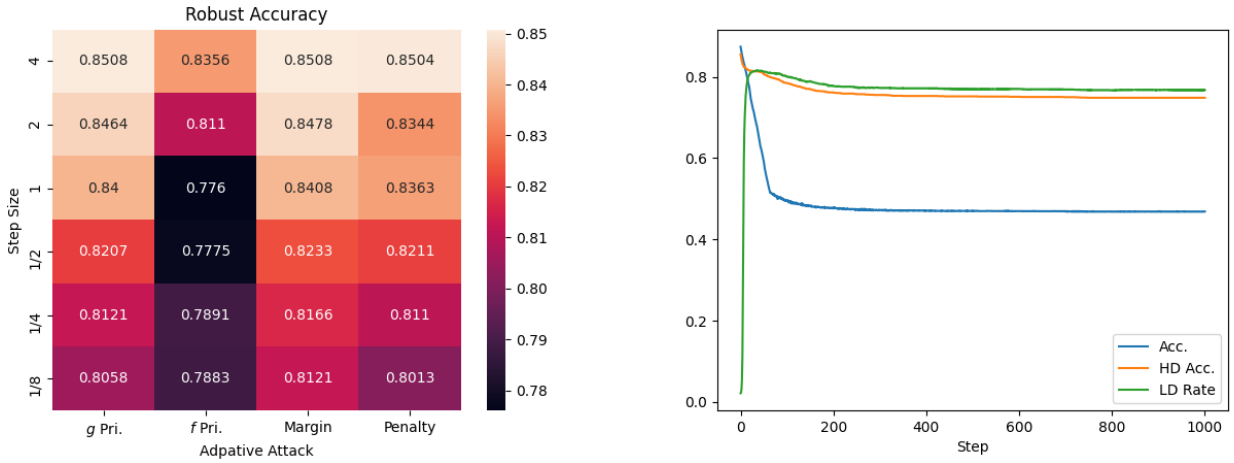
(b) ROC curves of different attacks

Figure 4: AUC scores and ROC curves of different δ-SGs and attacks. (a) AUC scores of δ-SGs during the training process. We calculate the AUC scores based on the output of our discriminator. (b) ROC curves of various attacks at the inference time.

In this section, we evaluate the performance of our discriminator against different δ-SGs and a few attacks described in Section 13. We denote the adversarial samples from each attack and δ-SG as positive (label of 1), and the natural samples (in the training set) as negative (label of 0). With this definition, we calculate the false positive rate and the true positive rate under different thresholds and draw the Receiver Operating Characteristic (ROC) curve. The Area Under Curve (AUC) score quantifies the area under the ROC curve. A larger AUC score indicates a better performance of the discriminator, where a random guess has 0.5 AUC score. We conduct the experiments on CIFAR-10 with our discriminator trained against δ-SG with $\delta = 0.01$. In Figure (4a), we show the AUC scores of different δ-SGs on the training set and the test set. The blue shaded area illustrates the curriculum learning stage where the hardship gradually increases. Observe that there is a "V" shape of the training curve within

this area. Initially, the blue line has a downward trend, meaning that the discriminator struggles to defend against the δ-SG. After the tipping point, the discriminator starts to learn the pattern of the δ-SG samples and gradually has a better performance (a larger AUC score). This delineates the importance of the curriculum learning, where the difficulty of δ-SG increases gradually. Without the curriculum learning, the discriminator may get trapped in a hard game (with no performance improvement). Note that the δ-SG with $\delta = 0.01$ is much more difficult than that with $\delta = 1$. Observe that in Figure (4a), the green curve ($\delta = 0.01$) is always lower than the orange curve ($\delta = 1$). Similar to standard adversarial training, the AUC score on the training set (blue curve) is larger than that on the test set (green curve). In Figure (4b), we present ROC curves for different attacks. As discussed in Section , adversarial samples concentrate on the low-density area. A very small threshold in the discriminator hence can easily filter out a large portion of adversarial samples. This is clearly demonstrated by the steep ROC curves, where our discriminator has a high true positive rate and a very low false positive rate.

D. Grid Search on Hyper-Parameters for Adaptive Attacks



(a) Grid search of step sizes and different strategies

(b) Stress test using f-priority strategy

Figure 5: Effect of step sizes and extensive attack steps

We study the effects of different hyper-parameters of adaptive attacks. We employ the Mądry model and the CIFAR-10 dataset as the study subject. We use the grid search on the step size ranging from $1/8$ to 4 for different adaptive attacks. We also conduct the grid search on the penalty weight from $1e$-1 to 100. The observations are similar and hence omitted. In Figure (5a), we report the effect of different step sizes on adaptive attacks. Observe that the robust accuracy of our technique against different adaptive attacks with various step sizes is stable (standard deviation of 2.2%). We then choose the f-priority strategy that has the best attack result from the grid search, and use it to stress test our technique with extensive attack iterations. Figure (5b) shows that with the increase of iterations, the attack quickly degrades the accuracy of the model in the first 50 iterations as shown by the blue line. In the meantime, the adversarial samples generated by the attack also speedily fall into the low-density area as shown by the green line of the LD Rate. As our discriminator discards low-density adversarial samples, we hence have a high HD Acc. (orange line).

E. Results on Tiny Imagenet

In this section, we reports the result on Tiny ImageNet in Table 3. This result illustrates the scalability of our method on higher-resolution data. We choose two popular adversarial training methods including Mądry (Mądry et al. 2018) and ALP (Kannan, Kurakin, and Goodfellow 2018). We report the accuracy, HD Acc. and LD Rate for different attacks (see Section Experimental Setup for the definition of these metrics). We can find the accuracy of classifier is much lower than CIFAR10. The results shows that our method can bring accuracy gap down from 28.2% to 4.5%. The findings are also consistent with the result on CIFAR10.

F. Results on Naturally Trained Models

Our technique requires the classifier to be adversarially trained according to our derivation of Theorem 1. It is interesting to study the performance of the discriminator on naturally trained models. The results on CIFAR-10 are shown in Table 4. We again report the accuracy, HD Acc. and LD Rate for different attacks (see Section Experimental Setup for the definition of these metrics). Observe that the HD Acc. on naturally trained models is much lower than that on adversarially trained ones under black-box and adaptive attacks. This result reflects our explanation in Figure 1a. That is, natural models have a irregular decision boundary, which will misclassify samples in the high-density area. Since these samples are still in-distribution samples, the

Table 3: Compare accuracy with high density accuracy on Tiny Imagenet. Some attacks are dropped due to timeout when scaling to this larger data set.

Threat	Attack	Mądry			ALP		
		Acc.	HD Acc.	LD Rate	Acc.	HD Acc.	LD Rate
None	Nature	45.4 %	44.7 %	1.1 %	42.2 %	41.7 %	1.1 %
Oblivious	PGD	18.7 %	44.0 %	97.3 %	19.4 %	41.1 %	97.7 %
	CW	17.2 %	44.5 %	98.8 %	16.5 %	41.5 %	98.9 %
	BIM	18.7 %	43.5 %	97.5 %	19.3 %	40.7 %	97.6 %
	MIFGSM	20.5 %	41.8 %	97.4 %	20.3 %	39.0 %	97.4 %
Unseen	Square	34.2 %	41.1 %	55.7 %	31.8 %	37.8 %	53.9 %
Adpative	g prior	40.2 %	40.3 %	42.2 %	36.9 %	37.4 %	39.5 %
	f prior	17.7 %	40.3 %	40.3 %	16.9 %	37.4 %	37.3 %
	Margin	40.9 %	40.9 %	43.3 %	37.7 %	37.8 %	40.2 %
	Penalty	25.8 %	40.2 %	42.0 %	25.2 %	37.4 %	39.3 %
Worst Case		17.2 %	40.2 %	-	16.5 %	37.4 %	-
Acc. Gap		28.2%	**4.5** %	-	25.7 %	**4.3** %	-

Table 4: Compare accuracy of naturally trained classifier on CIFAR-10

Threat	Attack	Nature Classifier		
		Acc.	HD Acc.	LD Rate
None	Nature	95.0 %	93.4 %	1.7 %
Oblivious	PGD	0.0 %	93.1 %	99.9 %
	CW	0.0 %	93.5 %	100.0 %
	BIM	0.0 %	89.7 %	100.0 %
	MIFGSM	0.0 %	54.2 %	99.0 %
	DeepFool	3.3 %	84.1 %	88.5 %
	JSMA	22.4 %	93.0 %	99.1 %
Unseen	Square	1.4 %	64.9 %	66.4 %
Adpative	g prior	44.5 %	48.7 %	50.9 %
	f prior	0.0 %	48.6 %	50.8 %
	Margin	48.6 %	50.3 %	52.9 %
	Penalty	0.0 %	42.9 %	45.5 %
Worst Case		0.0 %	42.9 %	-
Acc. Gap		95.0 %	**52.1** %	-

discriminator will not reject them, which results in a low HD Acc. Comparing to the standalone naturally trained classifier, our method greatly improves the robustness from 0% to 42.9% HD Acc in the worst case. And thus bring down the accuracy gap from 95.0 % to 52.1%.