Adversarial Defense via Learning to Generate Diverse Attacks

Yunseok Jang[1,*], Tianchen Zhao[1,*], Seunghoon Hong[1,2], Honglak Lee[1]
[1]University of Michigan, [2]Korea Advanced Institute of Science and Technology
[1]{yunseokj, ericolon, honglak}@umich.edu, [2]{seunghoon.hong}@kaist.ac.kr

Abstract

With the remarkable success of deep learning, Deep Neural Networks (DNNs) have been applied as dominant tools to various machine learning domains. Despite this success, however, it has been found that DNNs are surprisingly vulnerable to malicious attacks; adding a small, perceptually indistinguishable perturbations to the data can easily degrade classification performance. Adversarial training is an effective defense strategy to train a robust classifier. In this work, we propose to utilize the generator to learn how to create adversarial examples. Unlike the existing approaches that create a one-shot perturbation by a deterministic generator, we propose a recursive and stochastic generator that produces much stronger and diverse perturbations that comprehensively reveal the vulnerability of the target classifier. Our experiment results on MNIST and CIFAR-10 datasets show that the classifier adversarially trained with our method yields more robust performance over various white-box and black-box attacks.

1. Introduction

In the past decade, deep learning has achieved great breakthroughs in various domains of machine learning such as computer vision [22, 26, 49], speech recognition [18], and reinforcement learning [37, 48]. In parallel to these successes, Szegedy *et al*. [52] discovered that deep learning models are vulnerable to maliciously designed perturbations added to the input data. It is shown that even a small, perceptually indistinguishable perturbations can easily cause disastrous misprediction from the model. Later study suggested that such perturbations exist over a wide range of applications [27], and are even transferable across different models [33, 41, 42].

To make deep learning models robust against such malicious perturbations, various defense mechanisms [14, 19, 20, 23, 28, 34, 43, 47, 54, 58] have been proposed recently. Adversarial training [28] is one of the most effective strategies to defend against various attacks. The idea is to train the target neural network (*e.g*. classifier) jointly with clean data samples and their adversaries. Such a framework can be interpreted as a game between adversarial attack and defense; the adversary constantly attacks the target model, and the model learns to be robust against all attacks from the adversary. In this work, we address the attack mechanism in adversarial training, since its effectiveness is closely related to the method of generating adversarial data.

To generate adversarial data, early approaches formulate the task as a constrained optimization problem and utilize first-order gradient to generate adversarial perturbations [6, 9, 17, 27, 43]. Despite its success, however, an optimization in first-order methods is based on an oversimplified loss landscape [31]. The performance of the adversarially trained classifier is not guaranteed when it is challenged by more complicated, higher-order attacks.

More recently, approaches based on *learning-to-learn* framework have been proposed to improve adversarial training [7, 38, 44, 54]. The main idea is to *learn* a neural network, or a *generator*, to seek adversarial examples that effectively fool the target neural network. With the expressive power of DNNs, it is possible to discover attacks that cannot be obtained with the first-order methods, which is also useful to enhance the robustness of the classifier against future attacks.

However, the existing generator-based approaches employ a simple generator, or a naive scheme, that generates the perturbation in a single shot, leading to sub-optimal solutions that are less valuable for adversarial training. Also, such generators are often designed to produce deterministic perturbations (*i.e*. one perturbation per data), which makes adversarial training easily overfit to specific types of adversarial examples.

In this work, we propose a novel generator-based adversary to improve adversarial training. Based on recent learning-to-learn (L2L) framework [7], we design a generator so that it can achieve *stronger* and *diverse* attacks that lead to improving the robustness of the classifier. To build a stronger attack, we design the generator to recursively construct adversarial perturbations, which allows it to discover

*Equal Contribution

better optimization paths that lead to higher classification loss than a one-shot generation. In addition, we propose to regularize the generator to promote diversity in perturbations, which is important for discovering the comprehensive vulnerability of the classifier in adversarial training. From an optimization perspective, we argue that diversity encourages the generator to efficiently explore the space of possible perturbations, leading to more stable training.

The contributions of this paper are summarized below:

- We propose a novel defense mechanism based on the learning-to-learn framework [7]. The proposed generator constructs adversarial examples in a recursive manner, which utilize more informative optimization signals and produce stronger attacks.
- We further regularize the generator to produce stochastic and diverse outputs, to prevent the adversarial training from overfitting to specific attacks and to stabilize the generator training.
- We show that both recursive and diverse perturbations lead to significant improvements in the adversarial training over the baseline L2L approach. We also show that it leads to a non-trivial improvement over the existing approaches in white-box attacks and achieves competitive performance in black-box attacks.

The rest of this paper is organized as follows. We briefly review related work in Section 2 and provide the background of this problem in Section 3. Our approach for defense is introduced in Section 4. We discuss the experiment results on MNIST and CIFAR-10 datasets in Section 5.

2. Related Work

Adversarial attack. Observed initially by Szegedy *et al*. [52], modern deep neural networks are vulnerable against maliciously designed imperceptible perturbations, and a series of attack methods have been presented. Goodfellow *et al*. [17] proposed Fast Gradient Sign Method (FGSM) which takes the sign of a gradient obtained from the classifier.

To generate an adversarial attack in a more conservative way, Kurakin *et al*. [27] and Madry *et al*. [34] present Basic Iteration Method (BIM) and Projected Gradient Descent (PGD), respectively, which extend FGSM by applying it multiple times with a smaller step size. Because BIM is essentially a special case of PGD on the negative cross-entropy, we refer it as PGD throughout this paper.

In parallel to these gradient-based approaches, Papernot *et al*. [43], Dezfooli *et al*. [9] and Carlini and Wagner [6] take a direct optimization approach to discover adversarial examples. C&W attack [6] employs different forms of loss functions and is shown to be extremely powerful against many defense mechanisms.

Recent works propose stronger attacks compared to PGD. Dong *et al*. [11] add the notion of momentum to the PGD updates; Wang *et al*. [55] follow the distributional approach to adversarial training [50] and derive a stronger update scheme from an energy functional based on PGD; Wang *et al*. [55] introduce interval attack, claiming the mix-trained model is on par with PGD trained model in terms of test accuracy yet more verifiably robust. Following the literature, we use attacks from Madry *et al*. [34] and Carlini and Wagner [6] as our benchmarks.

Adversarial defense. In the past three years, many techniques have been proposed to improve the robustness of DNNs [10, 14, 15, 19, 20, 23, 24, 32, 35, 36, 40, 47, 51, 57, 60], but they are later shown to be not robust against the recent attacks [2, 4, 5, 39]. This hypothesizes that each defense mechanism is only robust against certain classes of attacks while being vulnerable to other attacks.

One of the most effective methods for training an adversarially robust network is *adversarial training* [28], namely training the model on adversarial attacks. Intuitively, the model will become robust if it is trained on very powerful attacks. Madry *et al*. [34] claims that PGD attack is the strongest attack obtainable from first-order methods, but this claim is later challenged by Wang *et al*. [55].

Instead of focusing on a hand-designed adversary based on the first-order gradient, our work aims to train a generator to learn how to create attacks through the mini-max game between the classifier and the generator. Hamm and Mehra [21] investigate the mini-max game between the attacker and the defender from both generator-based and gradient-based perspective and discover that a mini-max classifier outperforms non-minimax optimal classifiers. Xiao *et al*. [56] propose to use the GAN framework to generate attacks with a generator and filter out low-quality attacks with a discriminator. Similarly, Baluja and Fischer [3] generate attacks with a generator that either adversarially transforms images or adds adversarial perturbations to the images; however, only static classifier is considered in their experiments. Lee *et al*. [30] improve the supervised learning by training the classifier with outputs from the generator. Wang and Yu [54] improve the robustness of their classifier by training on generator-based adversarial attacks; Chen *et al*. [7] improves the idea by feeding both inputs and their gradients into the generator and show their adversarially trained Wide-ResNet outperforms the PGD trained Wide-ResNet on CIFAR-10/100.

3. Background

3.1. Problem Definition

The objective of this paper is to build a classifier robust to adversarial perturbations. Formally, we consider a clas-

sifier parameterized by θ and the labeled data $(\mathbf{x}, \mathbf{y})$. The goal of the classifier is to produce the correct class label $\mathbf{y}$ over adversarial examples $\mathbf{x}_{\text{adv}}$, obtained by adding a small perturbation δ to $\mathbf{x}$ (*i.e.* $\mathbf{x}_{\text{adv}} = \mathbf{x} + \delta$).

This task can be formulated as an optimization of the following saddle point problem [34]:

$$\min_{\theta} \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\mathcal{D}} \left[\max_{\mathbf{x}_{\text{adv}} \in \mathcal{S}(\mathbf{x})} \mathcal{L}(\mathbf{x}_{\text{adv}}, \mathbf{y}; \theta) \right], \quad (1)$$

where $\mathcal{L}(\cdot)$ is a classification loss (*i.e.* cross-entropy) and $\mathcal{S}(\mathbf{x})$ is a set of admissible attacks which constrains $\mathbf{x}_{\text{adv}}$ to be perceptually similar to $\mathbf{x}$ (*i.e.* ℓ_∞-bounded ball).

Eq. (1) implies that learning a robust classifier can be achieved via an adversarial game between the attacker and the classifier; the inner maximization corresponds to generating adversarial examples that maximize the classification loss, and the outer minimization corresponds to training the classifier with the augmented adversarial examples. If there exists an optimal θ^* that minimizes Eq. (1), the classifier parameterized by θ^* should be robust to any adversarial example $\mathbf{x}_{\text{adv}} \in \mathcal{S}(\mathbf{x})$.

Unfortunately, direct optimization on Eq. (1) is practically intractable due to the challenges in optimizing (non-concave) *inner maximization* over all training data. In practice, we instead approximate the optimal adversary with a local maxima $\mathbf{x}_{\text{adv}}$ that approximates the optimal adversary $\mathbf{x}^*_{\text{adv}} = \arg\max_{\mathbf{x}_{\text{adv}} \in \mathcal{S}(\mathbf{x})} \mathcal{L}(\mathbf{x}_{\text{adv}}, \mathbf{y}; \theta)$. From this perspective, the success of learning a robust classifier depends on the quality of the local maxima $\mathbf{x}_{\text{adv}}$.

In this work, we focus on the inner maximization problem of Eq. (1) to improve adversarial training. Specifically, we consider *strength* and *diversity* as the important properties for informative adversarial examples. From an optimization perspective, generating *strong* perturbation is important since we want to compute a better local maxima $\mathbf{x}_{\text{adv}}$ that minimizes $\mathcal{L}(\mathbf{x}^*_{\text{adv}}, \mathbf{y}; \theta) - \mathcal{L}(\mathbf{x}_{\text{adv}}, \mathbf{y}; \theta)$. To ensure the robustness of the classifier, it is also important to train a classifier with *diverse* adversarial examples revealing various vulnerabilities of the model.

3.2. Previous Methods

In this section, we briefly review existing approaches on optimizing the inner maximization problem of Eq. (1). For notational simplicity, we drop the dependency of $\mathbf{y}$ and θ from the loss $\mathcal{L}(\mathbf{x}, \mathbf{y}; \theta)$ and refer to it as $\mathcal{L}(\mathbf{x})$ for the rest of the paper.

Gradient-based methods. One simple and popular way to approximate the inner maximization of Eq. (1) is to exploit the first-order loss gradient to create adversarial perturbations. For instance, approaches such as First-order Gradient Sign Method (FGSM) [17] creates an adversarial example based on one-step gradient descent:

$$\mathbf{x}_{\text{adv}} = \text{Proj}_{\mathcal{X}} \left(\mathbf{x} + \epsilon \text{sign}(\nabla_{\mathbf{x}} \mathcal{L}(\mathbf{x})) \right), \quad (2)$$

where $\text{Proj}_{\mathcal{X}}$ denotes the projection of its element to a valid pixel value range, and ϵ denotes the size of ℓ_∞-ball. Madry *et al*. [34] show that approaches such as Projective Gradient Descent (PGD) can build much stronger attacks with iterative gradient descent:

$$\mathbf{x}^{(t+1)}_{\text{adv}} = \text{Proj}_{\mathcal{S}\cap\mathcal{X}} \left(\mathbf{x}^{(t)}_{\text{adv}} + \epsilon_{\text{step}} \text{sign}(\nabla_{\mathbf{x}} \mathcal{L}(\mathbf{x}^{(t)}_{\text{adv}})) \right), \quad (3)$$

where $\text{Proj}_{\mathcal{S}\cap\mathcal{X}}(\cdot)$ denotes the projection of its element to ℓ_∞-ball $\mathcal{S}$ and a valid pixel value range, and ϵ_{step} denotes a step size smaller than ϵ. The final adversarial example is then obtained by $\mathbf{x}_{\text{adv}} = \mathbf{x}^{(T)}_{\text{adv}}$. Note that FGSM [17] is a special case of PGD with $T = 1$ and $\epsilon_{\text{step}} = \epsilon$.

Generator-based methods. Instead of directly exploiting the gradient, generator-based approaches utilize a parameterized generator g_ϕ to *learn* how to create attacks to the target classifier [7, 38, 44, 54]. Among various approaches in this category, we focus on the method proposed by Chen *et al*. [7], which generates adversarial perturbations by

$$\mathbf{x}_{\text{adv}} = \text{Proj}_{\mathcal{X}} \left(\mathbf{x} + \epsilon g_\phi(\mathbf{x}, \nabla_{\mathbf{x}} \mathcal{L}(\mathbf{x})) \right), \quad (4)$$

where g_ϕ denotes our adversary, a neural network parameterized by ϕ. With the expressive power of deep neural networks, it can possibly generate more informative perturbations that lead to higher classification loss than the gradient-based methods.

However, one-shot generation, which is based solely on a single gradient, may not be informative enough to compute the optimal perturbation. It provides only a pointwise estimate of the entire loss landscape, which is usually too noisy to produce a reasonable solution. Also, the generator in Eq. (4) lacks a stochastic component, which constrains it to produce a deterministic perturbation for each data sample. Considering that there exist various perturbations corresponding to the local maxima of Eq. (1), as suggested by [34], it is important for the generator to be able to explore such perturbations to improve adversarial training.

4. Our Approach

The discussion about PGD attack [34] in Section 3.2 suggests that generator-based methods have a potential to improve adversarial training by generating stronger and more diverse attacks, but such capability is limited due to the simple generator. To address this issue, we propose a novel generator that produces both strong and diverse adversarial examples that better optimizes the inner maximization term of Eq. (1).

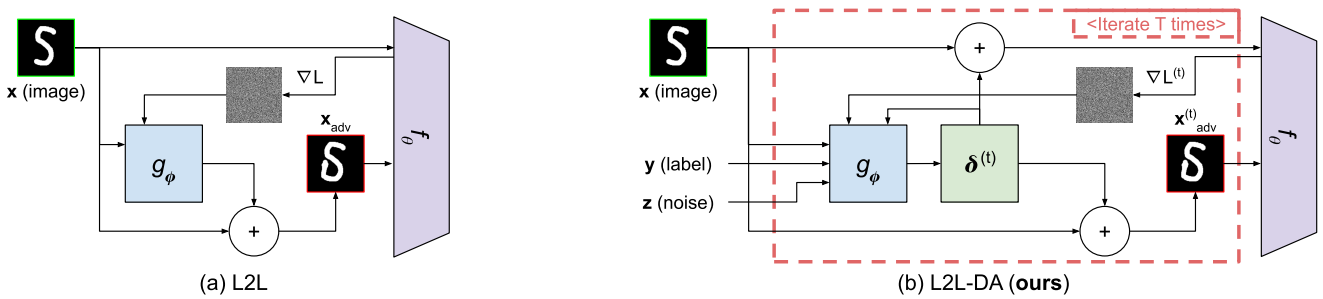


Figure 1: Comparison of L2L [7] and our attack network.

4.1. Recursive Generator

For better optimization of the inner maximization in Eq. (1), we define a recursive generator. Compared to the generator which exploits the fixed-point gradient (Figure 1.(a)), our generator utilizes much richer information, such as the trajectory of the perturbations and loss gradients, which is useful to estimate the loss landscape and the next perturbation that leads to a higher classification loss. Also, such a recursive update makes our generator more robust against noisy predictions, since the incorrect update at a certain step can be corrected by further updates.

We recursively generate an adversarial example by

$$\begin{aligned}\delta^{(t+1)} &= \text{Proj}_{\mathcal{S}}\left(\delta^{(t)} + \epsilon_{\text{step}} g_\phi(\mathbf{x}, \mathbf{y}, \mathbf{z}, \delta^{(t)}, \nabla_{\mathbf{x}}\mathcal{L}(\mathbf{x}_{\text{adv}}^{(t)}))\right), \\ \mathbf{x}_{\text{adv}}^{(t+1)} &= \text{Proj}_{\mathcal{X}}\left(\mathbf{x} + \delta^{(t+1)}\right), \end{aligned} \tag{5}$$

where $\delta^{(t)}$ denotes the adversarial perturbation accumulated up to the t-th step, which is initialized with zero ($\delta^{(0)} = 0$). To introduce stochasticity in the generation process, we additionally employ a random variable $\mathbf{z} \sim \mathcal{N}(0, \mathbf{I})$. It allows our generator to produce diverse perturbations through various optimization paths, helping the classifier become robust against diverse adversaries. The final adversarial example is then obtained by $\mathbf{x}_{\text{adv}} = \mathbf{x}_{\text{adv}}^{(T)}$. Figure 1 presents the overall framework of L2L-DA.

4.2. Loss Functions

To train our generator g_ϕ to produce stronger and more diverse perturbations, we employ two losses: the classification loss $\mathcal{L}_{\text{cls}}$ and the diversity loss $\mathcal{L}_{\text{div}}$. The classification loss $\mathcal{L}_{\text{cls}}$ is employed to ensure that each generator update leads to a stronger perturbation. This is simply achieved by making each perturbation increase the loss of the target classifier by

$$\mathcal{L}_{\text{cls}} = \frac{1}{T}\sum_{t=1}^{T} \mathcal{L}(\mathbf{x}_{\text{adv}}^{(t)}). \tag{6}$$

Eq. (6) encourages the perturbations created by our generator to learn stronger attacks at all steps, which roughly satisfy $\mathcal{L}(\mathbf{x}_{\text{adv}}^{(1)}) \leq \mathcal{L}(\mathbf{x}_{\text{adv}}^{(2)}) \leq \ldots \leq \mathcal{L}(\mathbf{x}_{\text{adv}}^{(T)})$. Considering the existing generator-based methods such as Chen *et al*. [7] as the special case of ours with $T = 1$, it is then evident that our method produces a stronger adversarial attack than these methods since $\mathcal{L}(\mathbf{x}_{\text{adv}}^{(1)}) \leq \mathcal{L}(\mathbf{x}_{\text{adv}}^{(T)})$. In Section 4.3, we show that attacks generated by our method indeed makes the classification loss worse than existing gradient- and generator-based methods.

On the other hand, the diversity loss $\mathcal{L}_{\text{div}}$ encourages the generator to produce diverse perturbations. Although the generator is supposed to produce stochastic outputs by sampling different latent variables $\mathbf{z}$, the generator trained with Eq. (6) tends to ignore $\mathbf{z}$ in the generation process; this phenomenon is known as mode-collapse [1, 16, 46]. To avoid the mode-collapse issue, we employ one of the most recent techniques from Yang *et al*. [59] that explicitly forces the generator outputs to be sensitive to the latent variables:

$$\mathcal{L}_{\text{div}} = \frac{\frac{1}{T}\sum_{t=1}^{T} \|\mathbf{x}_{\text{adv}}^{(t)}(\mathbf{z}_1) - \mathbf{x}_{\text{adv}}^{(t)}(\mathbf{z}_2)\|_1}{\|\mathbf{z}_1 - \mathbf{z}_2\|_1}, \tag{7}$$

where $\mathbf{x}_{\text{adv}}^{(t)}(\mathbf{z})$ denotes the adversarial examples generated by Eq. (5) using the latent variable $\mathbf{z}$, and $\mathbf{z}_1, \mathbf{z}_2$ are two i.i.d. samples of $\mathbf{z}$. Note that Eq. (7) encourages the diversity in the entire perturbation trajectory $\{\mathbf{x}_{\text{adv}}^{(t)}\}_{t=1}^{T}$, rather than the final adversarial output $\mathbf{x}_{\text{adv}}^{(T)}$.

Overall, we set our objective function as

$$\max_{\phi} \mathcal{L}_{g_\phi} = \mathcal{L}_{\text{cls}} + \lambda \mathcal{L}_{\text{div}}. \tag{8}$$

where λ is a hyperparameter to balance $\mathcal{L}_{\text{cls}}$ and $\mathcal{L}_{\text{div}}$. The overall training pipeline is described in Algorithm 1.

4.3. Analysis of Our Approach

As discussed in Section 3.1, training with powerful and diverse attacks, leads a model to be more robust to any adversarial sample $\mathbf{x}_{\text{adv}}$. In this section, we analyze the effectiveness of our recursive generation scheme and diversity loss that encourages stochastic perturbations.

Effectiveness of recursive generation. We employ PGD [34] and L2L-DA adversarially trained classifiers on

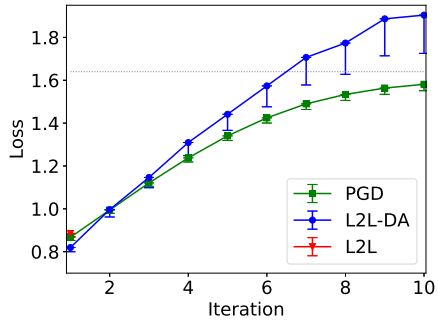
Algorithm 1 L2L-DA Training Algorithm

Input: Image $\mathbf{x}$, ground-truth label $\mathbf{y}$, $\nabla_{\mathbf{x}}\mathcal{L}(\mathbf{x})$, $\mathbf{z}_1$, $\mathbf{z}_2$
$\mathbf{x} \leftarrow [\mathbf{x}; \mathbf{x}]$, $\mathbf{y} \leftarrow [\mathbf{y}; \mathbf{y}]$, $\mathbf{z} \leftarrow [\mathbf{z}_1; \mathbf{z}_2]$ ($[\cdot;\cdot]$: concatenation)
for t in range [0, T) **do**
  $\delta^{(t+1)} \leftarrow \text{Proj}_{\mathcal{S}}\left(\delta^{(t)} + \epsilon_{\text{step}} g_\phi(\mathbf{x}, \mathbf{y}, \mathbf{z}, \delta^{(t)}, \nabla_{\mathbf{x}}\mathcal{L}(\mathbf{x}_{\text{adv}}^{(t)}))\right)$
  $\mathbf{x}_{\text{adv}}^{(t+1)} \leftarrow \text{Proj}_{\mathcal{X}}\left(\mathbf{x} + \delta^{(t+1)}\right)$
end for
$\mathcal{L}_{\text{cls}} \leftarrow \frac{1}{T}\sum_{t=1}^{T}\mathcal{L}(\mathbf{x}_{\text{adv}}^{(t)})$
$\mathcal{L}_{\text{div}} \leftarrow \frac{\frac{1}{T}\sum_{t=1}^{T}\|\mathbf{x}_{\text{adv}}^{(t)}(\mathbf{z}_1) - \mathbf{x}_{\text{adv}}^{(t)}(\mathbf{z}_2)\|_1}{\|\mathbf{z}_1 - \mathbf{z}_2\|_1}$
$\phi \leftarrow \phi + \nabla_{g_\phi}(\mathcal{L}_{\text{cls}} + \lambda\mathcal{L}_{\text{div}})$
$\theta \leftarrow \theta - \nabla_{f_\theta}(\mathcal{L}(\mathbf{x}) + \mathcal{L}(\mathbf{x}_{\text{adv}}^{(T)}))$

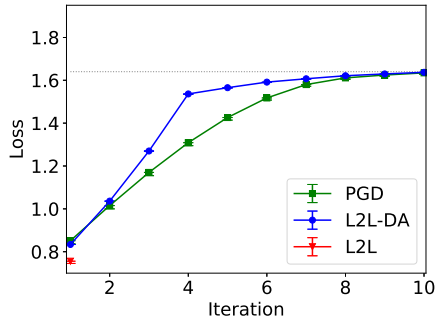
CIFAR-10 dataset (Please refer to Section 5.2 for details on PGD10 and L2L-DA, as well as the generator used for training the L2L-DA classifier (`full`). Note that our generator was not finetuned in Figure 2.(a)). Then, we perform a white-box attack with L2L [7], PGD [34], and L2L-DA method. We collect 10 attacks per data sample from each method, by using random initializations (restarts) for L2L and PGD, and randomly sampling $\mathbf{z}$ for L2L-DA. We compute the maximum and standard deviation of the losses from 10 attacks per image and calculate the mean over test data. We plot the values of the average maximum loss over 10 iterations in Figure 2, along with the corresponding average standard deviation as the lower error bar.

Both plots in Figure 2.(a) and (b) show that our method generates a more effective attack compared to L2L. Moreover, the loss increases as the time step increases, which empirically supports our argument about Eq. (6) that $\mathcal{L}(\mathbf{x}_{\text{adv}}^{(1)}) \leq \mathcal{L}(\mathbf{x}_{\text{adv}}^{(2)}) \leq \ldots \leq \mathcal{L}(\mathbf{x}_{\text{adv}}^{(T)})$: the strength of our attack is boosted over iterations. Our method also provide superior updates compared to the naive PGD scheme: PGD updates by the sign of the loss gradient to guarantee that the magnitude of the update is not too small, but it may deviate from the true gradient, resulting in an inferior performance; we train an adversary that knows how to update the perturbation optimally.

Effectiveness of stochastic perturbation. When the perturbation given by Eq. (5) reaches to the edge of ℓ_∞-ball at $t < T$, the deterministic generator could eventually get stuck in a local optima and produces the same perturbation over the remaining iterations (*i.e.* $\mathbf{x}_{\text{adv}}^{(t)} = \mathbf{x}_{\text{adv}}^{(t+1)}$). Eq. (7) encourages diversity in the entire perturbation trajectory $\{\mathbf{x}_{\text{adv}}^{(t)}\}_{t=1}^{T}$, to explore different optimization paths and discover rich and diverse perturbations that achieves a high classification loss. In Figure 2.(b), the standard deviations of the loss over the adversarial attacks generated on L2L-DA classifier are near zero, meaning the performance of our classifier trained with diverse attacks is not influenced by the stochasticity of the attack methods. Madry *et al.* [34] also observe that for CIFAR-10, the optimization trajectories of PGD and final loss values are fairly concentrated, especially for adversarially trained models that are more robust.



(a) Attacks on the classifier adversarially trained with PGD.



(b) Attacks on the classifier adversarially trained with L2L-DA.

Figure 2: Classification loss on (a) PGD (b) L2L-DA adversarially trained classifier. For each of the classifier, we perform three attack methods (PGD, L2L-DA, L2L) and plot its loss value. We additionally plot a gray dotted line in Loss=1.64 for a better comparison. L2L-DA attacks are generated by g_ϕ of L2L-DA (Eq.(5)).

The scales in Figure 2 also imply that our model is more robust than PGD trained model. Recall that our objective is to minimize Eq.(1) over θ; we can quantify the robustness of a model θ from the expected loss $\mathbb{E}\left[\mathcal{L}(\mathbf{x}_{\text{adv}}^{*}, \mathbf{y}; \theta)\right]$ over data distribution (empirically the average loss). Assuming our attack at the 10th iteration $\mathbf{x}_{\text{Ours}}^{(10)}$ is a good approximation to $\mathbf{x}_{\text{adv}}^{*}$, we can compare the robustness of each classifier by comparing its loss value. From Figure 2, we can see $\mathbb{E}\left[\mathcal{L}(\mathbf{x}_{\text{Ours}}^{(10)}, \mathbf{y}; \theta_{\text{Ours}})\right] \approx 1.64 < \mathbb{E}\left[\mathcal{L}(\mathbf{x}_{\text{Ours}}^{(10)}, \mathbf{y}; \theta_{\text{PGD}})\right] \approx 1.90$, which demonstrates that L2L-DA classifier is more robust than PGD classifier.

5. Experiments

In a high level, an adversarial is categorized as a white-box or black-box attack. White-box attack refers to the case that an adversary has full knowledge of the classifier

and training data, while black-box attack does not have knowledge about the classifier. [33, 42]. In this section, we first perform a white-box attack to show the effectiveness of our approach in Section 4. We then compare the robustness of our method with other baseline models by performing both white-box attack and black-box attack. The source codes used in the experiments are available at http://github.com/YunseokJANG/l2l-da.

5.1. Settings

Datasets. We evaluate our method for both white-box [17] and black-box settings [42] on MNIST [29] and CIFAR-10 [25] datasets. The MNIST dataset contains 70,000 28×28 gray-scale images of a single digit. The CIFAR-10 dataset contains a subset of the 80 million tiny images dataset [53] with 60,000 32×32 images from 10 object classes. For both datasets, we use the original train/test splits for training and evaluation.

Implementation. For the generator, we employ a convolutional encoder-decoder network based on the U-Net architecture [44, 45]. We first concatenate the accumulated perturbation and the loss gradient with the input image. Then, we feed the latent variable $\mathbf{z}$ to the input and the first two convolutional layers through spatial padding and concatenation. The class label $\mathbf{y}$ is injected by class-conditional batch normalization [8, 12]. For the target classifier, we employ LeNet-5 [29] for MNIST dataset and ResNet-20 [22] for CIFAR-10 dataset. We initialize the classifier with the weights pre-trained with the original data, and train both the generator and classifier jointly (Algorithm 1) by Stochastic Gradient Descent (SGD) with a momentum of 0.9 and weight decay of 0.00001 for 100K iterations with a batch size of 100. Following the literature [34], we set the size of ℓ_∞ ball as $\epsilon = 0.3$ for MNIST, and $\epsilon = 8/255$ for CIFAR-10 datasets. We use the step size of perturbation $\epsilon_{\text{step}} = \epsilon/4$.

Baselines. We compare our method against the classifiers that are adversarially trained upon existing gradient- and generator-based attack methods. We employ adversarially trained classifiers using FGSM [17], PGD [34], and C&W [6] as the gradient-based methods, and the one trained using L2L* [7] as the generator-based method [1].

Metric. To evaluate the robustness of the adversarially trained classifier, we measure its classification performance over the adversarial examples generated by existing gradient-based attack methods, such as FGSM [17], PGD [34], and C&W [6]. Since the iterative methods such as PGD and C&W tend to generate stronger attacks by running more iterations, we evaluate its performance over multiple versions with a different number of iterations (*e.g.* PGD10 and PGD100 for 10- and 100-step updates, respectively). For PGD, we employ random initialization without restart. In addition to the gradient-based attacks, we employ three generator-based attacks, AdvGAN [56], GAP [44], and L2L-DA, to measure the robustness of the classifier. As these methods are considered as a *proxy* for the real-world attack, we additionally report the lowest classifier accuracy among all attacks to gauge the robustness of the classifier in the worst case [13].

[1] We reproduced L2L [7] since there is no publicly available source code, but with additional skip-connection adhering to the advice from the authors. Our implementation is based on the draft submission on Nov. 3, 2018.

5.2. White Box Experiments

In this section, we present the results on a white-box attack. In this setting, the attacker has direct access to the target classifier parameters to create adversarial examples. We first conduct an ablation study to validate the effectiveness of the proposed regularization and architecture. Then we take the classifier adversarially trained with our best method to compare with the benchmarks.

5.2.1 Ablation Study

To understand how each component of the proposed method contributes to the robustness of adversarial training, we first conduct an ablation study on the MNIST dataset. In this experiment, we adversarially train the classifier by adding each component to our method and evaluate its performance based on the classification accuracy over various attacks. Table 1 summarizes the results.

Recursive generation. To understand the impact of the regularization terms, we first compare the second and fourth rows in Table 1.(a). By comparing them, we observe that our model achieves 7.85%p higher classification accuracy by recursively constructing adversarial examples. As discussed in Section 4.1, this is because the recursive generation utilizes richer optimization signals such as step-wise loss gradients to construct a stronger attack. This result is also consistent with our claim in Section 4.3 that the stronger attack improves the robustness of the adversarially trained classifier.

Diversity loss. We observe that the diversity loss improves the robustness of adversarial training. Comparing the second and third row in Table 1.(a), the 'Min' score is improved by 4.33%p. This improvement is observed across the recursive generation (the fourth and fifth rows by 1.18%p), showing that the contribution of diversity loss is orthogonal to the recursive generation. Interestingly, the improvement by the diversity loss seems to be more critical in a single-step generation. We believe this is because

Defense \ Attack	Natural	FGSM	PGD100	C&W1000	Min
L2L*	**98.95**	36.59	0.02	2.46	0.02
L2L-DA (R X, D X)	98.81	97.02	87.86	86.12	86.12
L2L-DA (R X, D O)	98.94	97.46	91.40	90.45	90.45
L2L-DA (R O, D X)	98.51	96.91	93.97	94.68	93.97
L2L-DA (R O, D O)	98.47	**97.87**	**95.15**	**95.26**	**95.15**

(a) Impact of the regularization terms.

Defense \ Attack	Natural	FGSM	PGD100	C&W1000	Min
L2L-DA (δ O, $\mathbf{y}$ O)	98.47	**97.87**	**95.15**	**95.26**	**95.15**
L2L-DA (δ O, $\mathbf{y}$ X)	**98.51**	96.75	93.44	94.02	93.44
L2L-DA (δ X, $\mathbf{y}$ O)	98.50	95.73	92.46	93.59	92.46
L2L-DA (δ X, $\mathbf{y}$ X)	98.40	95.58	92.36	93.24	92.36

(b) Impact of the generator architecture.

Table 1: Ablation study on MNIST dataset [29]. We measure the classification accuracy of adversarially trained classifiers (rows) against various attack methods (columns). R and D refer to the usage of recursive generation (Eq. (5)) and diversity loss (Eq. (7)) in our method. δ and $\mathbf{y}$ refer to the inputs in Eq. (5). Please check the footnote[1] for the details of L2L*.

Defense \ Attack	Natural	FGSM	PGD10	PGD100	C&W100	C&W1000	AdvGAN	GAP	L2L-DA	Min
MNIST [29]										
Plain	**99.14**	0.90	0.00	0.00	63.14	0.85	0.47	3.94	0.00	0.00
PGD10	98.52	96.14	92.21	90.22	96.66	88.92	**98.30**	96.56	89.49	88.92
PGD100	98.29	94.07	90.15	88.93	96.48	89.76	95.00	96.79	89.95	88.93
C&W100	99.09	17.54	0.25	0.25	69.99	0.05	4.90	3.74	34.51	0.05
L2L*	98.95	36.59	0.18	0.02	81.63	2.46	10.87	2.01	3.89	0.02
L2L-DA (full)	98.47	**97.87**	**95.70**	**95.15**	**97.26**	**95.26**	95.72	**96.92**	**94.61**	**94.61**
CIFAR-10 [25]										
Plain	**91.73**	17.63	0.00	0.00	0.00	0.00	24.13	17.42	0.04	0.00
PGD10	87.43	**72.00**	**41.77**	37.37	35.12	34.81	69.96	67.38	0.32	0.32
PGD100	81.11	44.96	37.60	35.80	36.25	35.85	77.79	**77.97**	36.95	35.80
C&W100	88.36	25.76	9.44	6.82	8.27	6.97	**83.11**	72.94	8.88	6.82
L2L*	88.96	59.48	0.00	0.00	0.00	0.00	30.13	19.36	0.00	0.00
L2L-DA (full)	78.91	45.77	39.69	**38.39**	**37.75**	**37.40**	75.31	76.35	**39.20**	**37.40**

Table 2: The result of White-box attacks on MNIST [29] and CIFAR-10 [25] datasets. We measure the classification accuracy of adversarially trained classifiers (rows) against various attack methods (columns).

optimization in a single-step is much more difficult than a recursive generation, and our diversity-sensitive term helps the optimization by exploring various optimization paths as discussed in Section 4.3.

Generator architecture. In addition to the regularization, we also analyze the impact of network architecture for generating adversarial attacks. The main difference between our base model (second row of Table 1.(a)) and L2L* (first row of Table 1.(a)) is that our method utilizes the additional class label $\mathbf{y}$ and the noise δ, and normalizes the input gradient (by dividing it by its Frobenius norm). Surprisingly, we observe that such architectural difference leads to substantially more robust model against strong attacks such as PGD100 and C&W1000.

To verify the effect of additional information, we train our model (*i.e.*, R and D) without $\mathbf{y}$, δ and gradient normalization operation. We first note that the performance drops significantly without gradient normalization to the gradient input ($\approx 1\%$ accuracy against PGD10 attack on CIFAR-10), which shows its importance in guiding the generator to produce meaningful attacks.

Comparing the first and third row in Table 1.(b), we observe that the performance drops by 2.69%p without δ. This is because being aware of the current state within ℓ_∞-ball via δ can help our model to explore the adversarial space more efficiently. Similarly, we observe 1.71%p gap when we compare the first and second row in Table 1.(b). This result implies that class-conditional batch normalization [8, 12] can also help our model to focus more on class-specific statistics.

The above ablation studies show that the regularization terms and the architecture change in our method complementary benefits the robustness in adversarial training. For the rest of the paper, we denote our full method (R O, D O) as L2L-DA (full).

5.2.2 Comparison to other methods

In this section, we compare our model to existing approaches. We use three popular attack methods in the literature, PGD [34], C&W [6], and L2L [7], as the baselines for adversarial training. The results on MNIST and CIFAR-10 datasets are presented in Table 2.

Defense \ Attack	A	B	C	A	B	C	A	B	C	Min
	L2L-DA			PGD10			PGD100			
Plain	0.00	42.70	90.36	0.01	44.46	86.04	0.00	42.36	84.54	0.00
PGD10	94.13	94.61	96.52	94.90	95.45	95.85	94.89	95.52	95.49	94.13
PGD100	94.23	94.91	96.32	94.91	95.52	95.29	95.05	95.59	94.91	94.23
CW100	15.15	41.76	93.49	23.55	41.72	89.30	21.47	39.10	88.63	15.15
L2L*	59.29	70.62	94.79	69.36	75.01	93.43	66.83	72.91	93.13	59.29
L2L-DA (`full`)	**95.08**	**95.40**	**96.65**	**95.72**	**96.00**	**96.31**	**95.63**	**96.09**	**96.03**	**95.08**

Table 3: The result of Black-box attack on MNIST dataset [29]. We first employ three surrogate models to perform a black-box attack, (A): PlainLeNet-5 (B): FGSMLeNet-5 (C): PGD10LeNet-5, which stands for the classifier that is adversarially trained over clean test data, FGSM, PGD10 respectively. We perform a set of White-box attack with L2L-DA, PGD10, PGD100 to those surrogates, and we report the accuracy by feeding the result of the white-box attack to the adversarially trained classifiers (rows).

First, we observe that the classifier trained with our method outperforms the others by a significant margin in both MNIST and CIFAR-10 datasets (5.68%p gap in MNIST and 1.60%p gap in CIFAR-10). Although the network trained with PGD10 achieves the best classification accuracy against similar attacks it is trained on, it experiences a noticeable performance drop against the unseen and stronger attacks, such as PGD100 and C&W1000, indicating an overfit to the attack used for training. On the other hand, our method avoids such problems by generating both stronger and diverse adversarial examples, leading to more efficient adversarial training that significantly improves the robustness of the classifier.

Also, we report the attack performance of AdvGAN [56], GAP [44] and L2L-DA, each of which are fine-tuned to its corresponding classifier. We observe that L2L-DA outperforms other generator-based attack methods [44, 56] in most cases. Therefore, the ability to generate strong adversarial samples helps improve classifier robustness.

5.3. Black Box Experiments

We call a phenomenon that some adversarial examples generated for one model is also misclassified by another model as transferability [33, 42]. We evaluate the robustness against transferability of the attacks by designing black-box attacks following Liu *et al*. [33]: to attack the targeted model f trained on the dataset $\mathcal{D}$, we first take a surrogate model f' pretrained on the same training data and perform white-box attacks to the surrogate model f' with test data to get adversarial examples. Then we measure the accuracy of the target classifier f on these adversarial test data. The results of the black-box experiment on MNIST dataset is shown in Table 3.

We use three surrogate models on MNIST dataset and name them as (A) PlainLeNet-5, (B) FGSMLeNet-5 and (C) PGD10LeNet-5. Each model is trained on LeNet-5 network, with (B: FSGM, C: PGD) or without (A: Plain test data) adversarial examples. Then, we generate adversarial attacks on the surrogate models with L2L-DA generator, PGD10, and PGD100, and feed them to the networks that are adversarially trained with the baseline methods.

Interestingly, black-box attacks generated from the surrogate model C (trained with PGD10) results in the best classification accuracy of the plain network, which is not intended for adversarial examples. One possible explanation is that black-box attacks are most effective when the same type of target models are used as the surrogate [33, 54].

Table 3 shows that our model outperforms the baselines for all tested cases. In particular, our model achieves the best 'Min' score (95.08%) across all tested adversarial attacks and networks. This implies that our model is more robust against various black-box attacks, *i.e*., transferability fails more often in L2L-DA.

6. Conclusion

In this work, we proposed a novel defense mechanism to learn a robust classifier against adversarial perturbation. Our defense mechanism utilizes the generator to create adversarial examples and trains the classifier with the adversarial generator to optimize the minimax objective. To make the classifier more robust against various and unexplored attacks, we design our generator to produce *strong* and *diverse* perturbations through the recursive generation and diversity-promoting regularization. Our experiment on two popular benchmark datasets demonstrates that both the strength and diversity of the perturbation play an important role in improving the robustness of the classifier. We also show that the classifier adversarially trained with our method is more robust against various white-box and black-box attacks.

Acknowledgements. We thank Wonkwang Lee, Kimin Lee and Juyong Kim for helpful discussions. We also appreciate Chris Dongjoo Kim, Jongwook Choi, Lajanugen Logeswaran and Mina Lee for constructive feedback of the manuscript. This work was supported in part by Kwanjeong Educational Foundation Scholarship, NSF CAREER IIS-1453651, and Sloan Research Fellowship.

References

[1] Martín Arjovsky and Léon Bottou. Towards Principled Methods for Training Generative Adversarial Networks. In *ICLR*, 2017.

[2] Anish Athalye, Nicholas Carlini, and David A. Wagner. Obfuscated Gradients Give a False Sense of Security: Circumventing Defenses to Adversarial Examples. In *ICML*, 2018.

[3] Shumeet Baluja and Ian Fischer. Adversarial Transformation Networks: Learning to Generate Adversarial Examples. In *AAAI*, 2018.

[4] Nicholas Carlini and David A. Wagner. Adversarial Examples Are Not Easily Detected: Bypassing Ten Detection Methods. In *AISec*, 2017.

[5] Nicholas Carlini and David A. Wagner. MagNet and "Efficient Defenses Against Adversarial Attacks" are Not Robust to Adversarial Examples. *arXiv:1711.08478*, 2017.

[6] Nicholas Carlini and David A. Wagner. Towards Evaluating the Robustness of Neural Networks. In *S&P*, 2017.

[7] Zhehui Chen, Haoming Jiang, Bo Dai, and Tuo Zhao. Learning to Defense by Learning to Attack. *arXiv:1811.01213*, 2018.

[8] Harm De Vries, Florian Strub, Jérémie Mary, Hugo Larochelle, Olivier Pietquin, and Aaron C Courville. Modulating Early Visual Processing by Language. In *NeurIPS*, 2017.

[9] Seyed Mohsen Moosavi Dezfooli, Alhussein Fawzi, and Pascal Frossard. DeepFool: A Simple and Accurate Method to Fool Deep Neural Networks. In *CVPR*, 2016.

[10] Guneet S. Dhillon, Kamyar Azizzadenesheli, Zachary C. Lipton, Jeremy Bernstein, Jean Kossaifi, Aran Khanna, and Anima Anandkumar. Stochastic Activation Pruning for Robust Adversarial Defense. In *ICLR*, 2018.

[11] Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Hang Su, Jun Zhu, Xiaolin Hu, and Jianguo Li. Boosting Adversarial Attacks with Momentum. In *CVPR*, 2018.

[12] Vincent Dumoulin, Jonathon Shlens, and Manjunath Kudlur. A Learned Representation For Artistic Style. In *ICLR*, 2017.

[13] Logan Engstrom, Andrew Ilyas, and Anish Athalye. Evaluating and Understanding the Robustness of Adversarial Logit Pairing. In *NeurIPS SECML*, 2018.

[14] Reuben Feinman, Ryan R. Curtin, Saurabh Shintre, and Andrew B. Gardner. Detecting Adversarial Samples from Artifacts. *arXiv:1703.00410*, 2017.

[15] Zhitao Gong, Wenlu Wang, and Wei-Shinn Ku. Adversarial and Clean Data Are Not Twins. *arXiv:1704.04960*, 2017.

[16] Ian Goodfellow. NIPS 2016 Tutorial: Generative Adversarial Networks. *arXiv:1701.00160*, 2016.

[17] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and Harnessing Adversarial Examples. In *ICLR*, 2015.

[18] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech Recognition with Deep Recurrent Neural Networks. In *ICASSP*, 2013.

[19] Kathrin Grosse, Praveen Manoharan, Nicolas Papernot, Michael Backes, and Patrick D. McDaniel. On the (Statistical) Detection of Adversarial Examples. *arXiv:1702.06280*, 2017.

[20] Chuan Guo, Mayank Rana, Moustapha Cissé, and Laurens van der Maaten. Countering Adversarial Images using Input Transformations. In *ICLR*, 2018.

[21] Jihun Hamm and Akshay Mehra. Machine vs Machine: Defending Classifiers Against Learning-based Adversarial Attacks. *arXiv:1711.04368*, 2017.

[22] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *CVPR*, 2016.

[23] Dan Hendrycks and Kevin Gimpel. Early Methods for Detecting Adversarial Images. In *ICLR Workshop*, 2017.

[24] Harini Kannan, Alexey Kurakin, and Ian J. Goodfellow. Adversarial Logit Pairing. *arXiv:1803.06373*, 2018.

[25] Alex Krizhevsky. Learning Multiple Layers of Features from Tiny Images. Technical report, Citeseer, 2009.

[26] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In *NeurIPS*, 2012.

[27] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. Adversarial Examples in the Physical World. In *ICLR Workshop*, 2017.

[28] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. Adversarial Machine Learning at Scale. In *ICLR*, 2017.

[29] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based Learning Applied to Document Recognition. *Proceedings of the IEEE*, 1998.

[30] Hyeungill Lee, Sungyeob Han, and Jungwoo Lee. Generative Adversarial Trainer: Defense to Adversarial Perturbations with GAN. *arXiv:1705.03387*, 2017.

[31] Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. Visualizing the Loss Landscape of Neural Nets. In *NeurIPS*, 2018.

[32] Xin Li and Fuxin Li. Adversarial Examples Detection in Deep Networks with Convolutional Filter Statistics. In *ICCV*, 2017.

[33] Yanpei Liu, Xinyun Chen, Chang Liu, and Dawn Song. Delving into Transferable Adversarial Examples and Black-box Attacks. In *ICLR*, 2017.

[34] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards Deep Learning Models Resistant to Adversarial Attacks. In *ICLR*, 2018.

[35] Dongyu Meng and Hao Chen. MagNet: A Two-Pronged Defense against Adversarial Examples. In *CCS*, 2017.

[36] Jan Hendrik Metzen, Tim Genewein, Volker Fischer, and Bastian Bischoff. On Detecting Adversarial Perturbations. In *ICLR*, 2017.

[37] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing Atari with Deep Reinforcement Learning. In *NeurIPS*, 2013.

[38] Konda Reddy Mopuri, Utkarsh Ojha, Utsav Garg, and R. Venkatesh Babu. NAG: Network for Adversary Generation. In *CVPR*, 2018.

[39] Marius Mosbach, Maksym Andriushchenko, Thomas Alexander Trost, Matthias Hein, and Dietrich Klakow. Logit Pairing Methods Can Fool Gradient-Based Attacks. In *NeurIPS SECML*, 2018.

[40] Taesik Na, Jong Hwan Ko, and Saibal Mukhopadhyay. Cascade Adversarial Machine Learning Regularized with a Unified Embedding. In *ICLR*, 2018.

[41] Nicolas Papernot, Patrick D. McDaniel, and Ian J. Goodfellow. Transferability in Machine Learning: from Phe-

nomena to Black-Box Attacks using Adversarial Samples. *arXiv:1605.07277*, 2016.

[42] Nicolas Papernot, Patrick D. McDaniel, Ian J. Goodfellow, Somesh Jha, Z. Berkay Celik, and Ananthram Swami. Practical Black-Box Attacks Against Machine Learning. In *ASIACCS*, 2017.

[43] Nicolas Papernot, Patrick D. McDaniel, Somesh Jha, Matt Fredrikson, Z. Berkay Celik, and Ananthram Swami. The Limitations of Deep Learning in Adversarial Settings. In *EuroS&P*, 2016.

[44] Omid Poursaeed, Isay Katsman, Bicheng Gao, and Serge Belongie. Generative Adversarial Perturbations. In *CVPR*, 2018.

[45] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation. In *MICCAI*, 2015.

[46] Tim Salimans, Ian J. Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved Techniques for Training GANs. In *NeurIPS*, 2016.

[47] Pouya Samangouei, Maya Kabkab, and Rama Chellappa. Defense-GAN: Protecting Classifiers Against Adversarial Attacks Using Generative Models. In *ICLR*, 2018.

[48] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the Game of Go with Deep Neural Networks and Tree Search. *Nature*, 2016.

[49] Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *ICLR*, 2014.

[50] Aman Sinha, Hongseok Namkoong, and John Duchi. Certifying Some Distributional Robustness with Principled Adversarial Training. In *ICLR*, 2018.

[51] Yang Song, Taesup Kim, Sebastian Nowozin, Stefano Ermon, and Nate Kushman. PixelDefend: Leveraging Generative Models to Understand and Defend against Adversarial Examples. In *ICLR*, 2018.

[52] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing Properties of Neural Networks. In *ICLR*, 2014.

[53] Antonio Torralba, Rob Fergus, and William T Freeman. 80 Million Tiny Images: A Large Data Set for Nonparametric Object and Scene Recognition. *TPAMI*, 2008.

[54] Huaxia Wang and Chun-Nam Yu. A Direct Approach to Robust Deep Learning Using Adversarial Networks. In *ICLR*, 2019.

[55] Shiqi Wang, Yizheng Chen, Ahmed Abdou, and Suman Jana. MixTrain: Scalable Training of Formally Robust Neural Networks. *arXiv:1811.02625*, 2018.

[56] Chaowei Xiao, Bo Li, Jun-Yan Zhu, Warren He, Mingyan Liu, and Dawn Song. Generating Adversarial Examples with Adversarial Networks. In *IJCAI*, 2018.

[57] Cihang Xie, Jianyu Wang, Zhishuai Zhang, Zhou Ren, and Alan L. Yuille. Mitigating Adversarial Effects Through Randomization. In *ICLR*, 2018.

[58] Cihang Xie, Yuxin Wu, Laurens van der Maaten, Alan Yuille, and Kaiming He. Feature Denoising for Improving Adversarial Robustness. In *CVPR*, 2019.

[59] Dingdong Yang, Seunghoon Hong, Yunseok Jang, Tiangchen Zhao, and Honglak Lee. Diversity-Sensitive Conditional Generative Adversarial Networks. In *ICLR*, 2019.

[60] Valentina Zantedeschi, Maria-Irina Nicolae, and Ambrish Rawat. Efficient Defenses Against Adversarial Attacks. In *AISec*, 2017.