Learning to Defend by Learning to Attack

Haoming Jiang* Georgia Tech
Zhehui Chen* Georgia Tech
Yuyang Shi Georgia Tech
Bo Dai Google Brain
Tuo Zhao Georgia Tech

Abstract

Adversarial training provides a principled approach for training robust neural networks. From an optimization perspective, adversarial training is essentially solving a bilevel optimization problem. The leader problem is trying to learn a robust classifier, while the follower maximization is trying to generate adversarial samples. Unfortunately, such a bilevel problem is difficult to solve due to its highly complicated structure. This work proposes a new adversarial training method based on a generic learning-to-learn (L2L) framework. Specifically, instead of applying existing hand-designed algorithms for the inner problem, we learn an optimizer, which is parametrized as a convolutional neural network. At the same time, a robust classifier is learned to defense the adversarial attack generated by the learned optimizer. Experiments over CIFAR-10 and CIFAR-100 datasets demonstrate that L2L outperforms existing adversarial training methods in both classification accuracy and computational efficiency. Moreover, our L2L framework can be extended to generative adversarial imitation learning and stabilize the training.

1 Introduction

This decade has witnessed great breakthroughs in deep learning in a variety of applications, such as computer vision (Taigman et al., 2014; Girshick et al., 2014; He et al., 2016; Liu et al., 2017). Recent studies (Szegedy et al., 2013), however, show that most of these deep learning models are very vulnerable to adversarial attacks. Specifically, by injecting a small perturbation to a normal sample, one can obtain an adversarial sample. Although the adversarial sample is semantically indistinguishable from the normal one, it can fool deep learning models and undermine the security of deep learning, causing reliability problems in autonomous driving, biometric authentication, etc.

Proceedings of the 24[th] International Conference on Artificial Intelligence and Statistics (AISTATS) 2021, San Diego, California, USA. PMLR: Volume 130.

Researchers have devoted many efforts to study efficient adversarial attack and defense (Szegedy et al., 2013; Goodfellow et al., 2014b; Nguyen et al., 2015; Zheng et al., 2016; Madry et al., 2017; Carlini and Wagner, 2017). There is a growing body of work on generating adversarial samples, e.g., fast gradient sign method (FGSM, Goodfellow et al. (2014b)), projected gradient method (PGM, Kurakin et al. (2016)), Carlini-Wagner (CW, Paszke et al. (2017)), etc. As for defense, existing methods can be unified as a bilevel optimization problem as follows:

$$\begin{aligned} &\text{(Leader)} \qquad \min_{\boldsymbol{\theta}} \quad \mathbb{E}_{P^*}\big[\ell(f_{\boldsymbol{\theta}}(\widetilde{\boldsymbol{x}}),\widetilde{\boldsymbol{y}})\big], \qquad (1)\\ &\text{(Follower) s.t. } P^* \in \underset{\widetilde{P}\in\mathcal{P}}{\operatorname{argmax}}\, \mathbb{E}_{\widetilde{P}}\big[q_{f_{\boldsymbol{\theta}}}\big((\boldsymbol{x},\boldsymbol{y}),(\widetilde{\boldsymbol{x}},\widetilde{\boldsymbol{y}})\big)\big], \end{aligned}$$

where ℓ denotes the loss function, $f_{\boldsymbol{\theta}}$ denotes the neural network classifier with parameter $\boldsymbol{\theta}$, $(\boldsymbol{x},\boldsymbol{y})$ denotes the clean sample from distribution D, $q_{f_{\boldsymbol{\theta}}}(\cdot,\cdot)$ denotes a measure depending on network $f_{\boldsymbol{\theta}}$, and $\mathcal{P}$ denotes a set of joint distributions of perturbed sample $(\widetilde{\boldsymbol{x}},\widetilde{\boldsymbol{y}})$ and clean sample $(\boldsymbol{x},\boldsymbol{y})$. Here $\widetilde{P}\in\mathcal{P}$ satisfies that in each sample $(\widetilde{\boldsymbol{x}},\widetilde{\boldsymbol{y}})$ is close to $(\boldsymbol{x},\boldsymbol{y})$ and the marginal distribution of $\widetilde{P}$ over $(\boldsymbol{x},\boldsymbol{y})$ is D. By solving the follower problem in (1), P^* essentially represents an effective adversarial distribution. Existing adversarial training methods use different approaches to find P^* under different $q_{f_{\boldsymbol{\theta}}}$ and $\mathcal{P}$. For example, Goodfellow et al. (2014b) consider a special case of this problem, distributionally robust optimization (DRO, Gao and Kleywegt (2016); Rahimian and Mehrotra (2019)). In DRO, $q_{f_{\boldsymbol{\theta}}}$ in (1) is the same as ℓ in (1) and $\widetilde{P}\in\mathcal{P}$ satisfies that in each sample $\widetilde{\boldsymbol{y}}=\boldsymbol{y}$, i.e., train the network $f_{\boldsymbol{\theta}}$ over adversarial samples and still require $f_{\boldsymbol{\theta}}$ to yield the correct labels. Another example is adversarial interpolation training (AIT, Haichao Zhang (2019)), where $q_{f_{\boldsymbol{\theta}}}$ is the cosine similarity between the features of adversarial sample and clean sample, and

$\mathcal{P}$ is a set of adversarial distribution yielded by mixup Zhang et al. (2017). More details are in Section 2.

(1) contains two optimization problems, referred to as leader and follower problems, respectively in the optimization literature. Such a bilevel formulation naturally provides us a unified perspective on prior works of robustifying the neural network: The leader aims to find a robust network so that the loss given by the training distribution from the follower problem is minimized; The follower targets on finding an optimal distribution that maximizes a certain measure, which yields a distribution of adversarial samples.

Though the bilevel problem is straightforward and well formulated, it is hard to solve. Even the simplest version of bilevel problem, linear-linear bilevel optimization, is shown to be NP-hard (Colson et al., 2007). In our case, the problem becomes more challenging, since loss function ℓ in the leader is highly nonconvex in $\boldsymbol{\theta}$ and the follower targets on finding an optimal distribution under a nonconcave measure $q_{f_{\boldsymbol{\theta}}}$. Besides, the feasible domain of the follower problem is a space of continuous distributions; while, in practice, we have finite samples to approximate the original problem. Such a gap makes the problem more challenging.

There are several approaches to solve the original problem (1). Under the DRO setting, Goodfellow et al. (2014b) propose to use FGSM to solve the DRO. However, Kurakin et al. (2016) then find that FGSM with true label suffers from a "label leaking" issue, which ruins adversarial training. Madry et al. (2017) further suggest to find adversarial samples by PGM and outperforms FGSM, since FGSM essentially is one iteration PGM; Alternatively, Haichao Zhang (2019) propose to combine FGSM and mixup to yield an adversarial samples for both feature and label. All these methods need to find an adversarial $(\widetilde{\boldsymbol{x}}_i, \widetilde{\boldsymbol{y}}_i)$ for each clean sample $(\boldsymbol{x}_i, \boldsymbol{y}_i)$, thus the dimension of the overall search space for all samples is substantial, which makes the computation expensive. More recently, Li et al. (2019) propose to use the natural evolution strategy to learn an adversarial distribution under the black-box setting, which is beyond the scope of this paper.

To address the above challenges, we propose a new learning-to-learn (L2L) framework that provides a more principled and efficient way for solving adversarial training. Specifically, we parameterize the optimizer of the follower problem by a neural network denoted by $g_{\boldsymbol{\phi}}(\mathcal{A}_{f_{\boldsymbol{\theta}}}(\boldsymbol{x}, \boldsymbol{y}))$, where $\mathcal{A}_{f_{\boldsymbol{\theta}}}(\boldsymbol{x}, \boldsymbol{y})$ denotes the input of the optimizer $g_{\boldsymbol{\phi}}$ with parameter $\boldsymbol{\phi}$. We also call the optimizer as the attacker. Since the neural network is very powerful in function approximation, our parameterization ensures that $g_{\boldsymbol{\phi}}$ is able to yield strong adversarial samples. Under our framework, instead of directly solving the follower problem in (1), we update the parameter $\boldsymbol{\phi}$ of the optimizer $g_{\boldsymbol{\phi}}$. Our training procedure becomes updating the parameters of two neural networks, which is quite similar to generative adversarial network (GAN, Goodfellow et al. (2014a)). The proposed L2L is a generic framework and can be extended to other bilevel optimization problems, e.g., generative adversarial imitation learning, which is studied in Section D.

Different from the hand-designed methods that compute adversarial perturbation $\boldsymbol{\delta}_i = \widetilde{\boldsymbol{x}}_i - \boldsymbol{x}_i$ for each individual sample $(\boldsymbol{x}_i, \boldsymbol{y}_i)$ using gradients from backpropagation, our methods generate perturbations for all samples through the shared optimizer $g_{\boldsymbol{\phi}}$. This enables optimizer $g_{\boldsymbol{\phi}}$ to learn potential common structures of the perturbations. Therefore, our method is capable of yielding strong perturbations and accelerating the training process. Furthermore, the L2L framework is very flexible: we can either choose different input $\mathcal{A}_{f_{\boldsymbol{\theta}}}(\boldsymbol{x}, \boldsymbol{y})$, or use different architecture. For example, we can include gradient information in $\mathcal{A}_{f_{\boldsymbol{\theta}}}(\boldsymbol{x}, \boldsymbol{y})$ and use a recurrent neural network (RNN) to mimic multi-step gradient-type methods. Instead of computing the high order information with finite difference approximation or multiple gradients, by parameterizing the algorithm as a neural network, our proposed method can capture this information in a much adaptive way (Finn et al., 2017). Our experiments demonstrate that L2L not only outperforms existing adversarial training methods, e.g., PGM training, but also enjoys computational efficiency over CIFAR-10 and CIFAR-100 datasets (Krizhevsky and Hinton, 2009).

The research on L2L has a long history (Schmidhuber, 1987, 1992, 1993; Younger et al., 2001; Hochreiter et al., 2001; Andrychowicz et al., 2016). The basic idea is that the updating formula of complicated optimization algorithms is first modeled in a parametric form, and then parameters are learned by some simple algorithms, e.g., stochastic gradient algorithm. Among existing works, Hochreiter et al. (2001) propose a system allowing the output of backpropagation from one network to feed into an additional learning network, with both networks trained jointly; Andrychowicz et al. (2016) then show that the design of an optimization algorithm can be cast as a learning problem. Specifically, they use long short-term memory RNNs to model the algorithm and allow the RNNs to exploit structure in the problems of interest in an adaptive way, which is one of the most popular methods for L2L.

However, there are two major drawbacks of the existing L2L methods: **(1)** It requires a large amount of datasets (or a large number of tasks in multi-task learning) to guarantee the learned optimizer to generalize, which limits their applicability (most of the related works only consider the image encoding as the motivating application); **(2)** The number of layers/iterations in RNNs for modeling algorithms cannot be large to avoid computational burden.

Our contribution is that we fill the blank of L2L framework in solving bilevel optimization problems, and our proposed methods do not suffer from the aforementioned drawbacks: **(1)** Different $f_{\boldsymbol{\theta}}$ and $(\boldsymbol{x}, \boldsymbol{y})$ yield different follower problems. Therefore, for adversarial training, we have sufficiently many tasks for L2L; **(2)** The follower problem does not need a large scale RNN, and we use a convolutional neural network (CNN) or a length-two RNN (sequence of length equals 2) as our attacker network, which eases computation. Our code is available at `https://github.com/YuyangShi/Learning-to-Defend-by-Learning-to-Attack`.

Notations. Given a scalar $a \in \mathbb{R}$, denote $(a)_+$ as $\max(a, 0)$. Given two vectors $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d$, denote x_i as the i-th element of $\boldsymbol{x}$, $||\boldsymbol{x}||_\infty = \max_i |x_i|$ as the ℓ_∞-norm of $\boldsymbol{x}$, $\boldsymbol{x} \circ \boldsymbol{y} = [x_1 y_1, \cdots, x_d y_d]^\top$ as element-wise product, and $\boldsymbol{e}_i$ is the vector with i-th element as 1 and others as 0. Denote the simplex in $\mathbb{R}^d$ by $\Delta(d) := \{\boldsymbol{x} : ||\boldsymbol{x}||_1 = 1\}$, the ℓ_∞-ball centered at $\boldsymbol{x}$ with radius ϵ by $\mathcal{B}(\boldsymbol{x}, \epsilon) = \{\boldsymbol{y} \in \mathbb{R}^d : ||\boldsymbol{y} - \boldsymbol{x}||_\infty \leq \epsilon\}$ and the projection to $\mathcal{B}(\boldsymbol{0}, \epsilon)$ as $\Pi_\epsilon(\boldsymbol{\delta}) = \mathrm{sign}(\boldsymbol{\delta}) \circ \max(|\boldsymbol{\delta}|, \epsilon)$, where sign and max are element-wise operators.

2 Preliminary

We focus on the defense against ℓ_∞-norm attack. In this section, we first introduce two popular cases of the original problem: distributionally robust optimization (DRO) and adversarial interpolation training (AIT). Then we discuss the fundamental hardness of solving these problems and the drawbacks of existing methods.

2.1 Adversarial Training

Instead of using population loss in (1), we use empirical loss in the following context, since in practice we only have finite samples. Given n samples $\{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^n$, where $\boldsymbol{x}_i$ is the i-th image and $\boldsymbol{y}_i$ is the corresponding label, DRO aims to solve:

$$\min_{\boldsymbol{\theta}} \quad \tfrac{1}{n} \sum_{i=1}^n \big[\ell(f_{\boldsymbol{\theta}}(\boldsymbol{x}_i + \boldsymbol{\delta}_i), \boldsymbol{y}_i)\big], \tag{2}$$

$$\text{s.t.} \quad \boldsymbol{\delta}_i \in \mathrm{argmax}_{\boldsymbol{\delta} \in \mathcal{B}(\boldsymbol{0}, \epsilon)} \ell(f_{\boldsymbol{\theta}}(\boldsymbol{x}_i + \boldsymbol{\delta}), \boldsymbol{y}_i). \tag{3}$$

The standard pipeline of DRO version is shown in Algorithm 1. Since the step of generating adversarial perturbation $\boldsymbol{\delta}_i$ in Algorithm 1 is intractable, most adversarial training methods adopt hand-designed algorithms. For example, Kurakin et al. (2016) propose to solve follower problem (3) approximately by first order methods like PGM. Specifically, PGM iteratively updates the adversarial perturbation by the projected sign gradient ascent method for each sample: Given sample $(\boldsymbol{x}_i, \boldsymbol{y}_i)$, at the t-th iteration, PGM takes

$$\boldsymbol{\delta}_i^t \leftarrow \Pi_\epsilon\big(\boldsymbol{\delta}_i^{t-1} + \eta \cdot \mathrm{sign}\left(\nabla_{\boldsymbol{x}} \ell(f_{\boldsymbol{\theta}}(\widetilde{\boldsymbol{x}}_i^t), \boldsymbol{y}_i)\right)\big), \tag{4}$$

where $\widetilde{\boldsymbol{x}}_i^t = \boldsymbol{x}_i + \boldsymbol{\delta}_i^{t-1}$, η is the perturbation step size, T is a pre-defined total number of iterations, and $\boldsymbol{\delta}_i^0 = \boldsymbol{0}$, $t = 1, \cdots, T$. Finally PGM takes $\boldsymbol{\delta}_i = \boldsymbol{\delta}_i^T$. Note that FGSM essentially is one-iteration PGM. Besides, some works adopt other optimization methods, e.g., momentum gradient method (Dong et al., 2018), and L-BFGS (Tabacof and Valle, 2016).

Algorithm 1 *Distributionally Robust Optimization.*

Input: $\{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^n$: data, α: learning rate, N: number of iterations, ϵ: perturbation magnitude.
for $t \leftarrow 1$ *to* N **do**
  Sample a minibatch $\mathcal{M}_t$
  for i *in* $\mathcal{M}_t$ **do**
    $\boldsymbol{\delta}_i \leftarrow \mathrm{argmax}_{\boldsymbol{\delta} \in \mathcal{B}(\boldsymbol{0}, \epsilon)} \ell(f_{\boldsymbol{\theta}}(\boldsymbol{x}_i + \boldsymbol{\delta}), \boldsymbol{y}_i)$
    // Generate adversarial data.
  $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha \frac{1}{|\mathcal{M}_t|} \sum_{i \in \mathcal{M}_t} \nabla_{\boldsymbol{\theta}} \ell(f_{\boldsymbol{\theta}}(\boldsymbol{x}_i + \boldsymbol{\delta}_i), \widetilde{\boldsymbol{y}}_i)$
  //Update $\boldsymbol{\theta}$ over adversarial data.

Alternatively, AIT adopts the mixup method to generate an adversarial distribution for a given sample $(\boldsymbol{x}_i, \boldsymbol{y}_i)$ and then randomly select a sample $(\widetilde{\boldsymbol{x}}_i, \widetilde{\boldsymbol{y}}_i)$ from this adversarial distribution. Specifically, AIT solves the following problem:

$$\min_{\boldsymbol{\theta}} \quad \tfrac{1}{n} \sum_{i=1}^n \mathbb{E}_{(\widetilde{\boldsymbol{x}}_i, \widetilde{\boldsymbol{y}}_i) \sim D_i} \big[\ell(f_{\boldsymbol{\theta}}(\widetilde{\boldsymbol{x}}_i), \widetilde{\boldsymbol{y}}_i)\big], \tag{5}$$

where $D_i = \{(\widetilde{\boldsymbol{x}}_i^j, \widetilde{\boldsymbol{y}}_i^j)\}_{j=1}^n$ is generated as follows:

$$\begin{aligned} \widetilde{\boldsymbol{x}}_i^j &= \mathrm{argmin}_{\widetilde{\boldsymbol{x}} \in \mathcal{B}(\boldsymbol{x}_i, \epsilon)} \tfrac{f_{\boldsymbol{\theta}}^s(\boldsymbol{x}_j) \cdot f_{\boldsymbol{\theta}}^s(\widetilde{\boldsymbol{x}})}{||f_{\boldsymbol{\theta}}^s(\boldsymbol{x}_j)||_2 ||f_{\boldsymbol{\theta}}^s(\widetilde{\boldsymbol{x}})||_2}, \\ \widetilde{\boldsymbol{y}}_i^j &= \mathrm{argmin}_{\widetilde{\boldsymbol{y}} \in \Delta(C) \cap \mathcal{B}(\boldsymbol{y}_i, \epsilon_{\boldsymbol{y}})} ||\widetilde{\boldsymbol{y}} - \tfrac{\boldsymbol{1} - \boldsymbol{y}_j}{C - 1}||_2^2, \end{aligned} \tag{6}$$

where $f_{\boldsymbol{\theta}}^s(\cdot)$ denotes the output of the s-th layer of network $f_{\boldsymbol{\theta}}$, C denotes the number of classes, and $\boldsymbol{1}$ denotes the vector with all elements 1. The standard pipeline is shown in Algorithm 2. To ease the computation, Haichao Zhang (2019) use one-step gradient update as the solution of (6).

2.2 Hardness

Now we present the hardness for solving these problems. Ideally, we want to obtain the optima for the follower problem, i.e.,

$$P^* := \mathrm{argmax}_{\widetilde{P} \in \mathcal{P}} \mathbb{E}_{\widetilde{P}} \left[q_{f_{\boldsymbol{\theta}}}\big((\boldsymbol{x}, \boldsymbol{y}), (\widetilde{\boldsymbol{x}}, \widetilde{\boldsymbol{y}})\big)\right].$$

However, the measure $q_{f_{\boldsymbol{\theta}}}$ depends on network $f_{\boldsymbol{\theta}}$, which makes solving P^* intractable. Therefore, in reality the sample $(\widetilde{\boldsymbol{x}}_i, \widetilde{\boldsymbol{y}}_i)$ from the obtained solution $\widetilde{P}$ is very unlikely to be the sample $(\boldsymbol{x}_i^*, \boldsymbol{y}_i^*)$ from P^*.

This then often leads to a highly unreliable or even completely wrong search direction, i.e.,

$$\langle \nabla_{\boldsymbol{\theta}} \ell(f_{\boldsymbol{\theta}}(\widetilde{\boldsymbol{x}}_i), \widetilde{\boldsymbol{y}}_i), \nabla_{\boldsymbol{\theta}} \ell(f_{\boldsymbol{\theta}}(\boldsymbol{x}_i^*), \boldsymbol{y}_i^*)\rangle < 0,$$

which may further result in a limiting cycle (See Appendix A). This becomes even worse when sample noises exist. Moreover, among the methods mentioned earlier, except FGSM, all require numerous queries for gradients, which is computationally expensive.

Algorithm 2 *Adversarial Interpolation Training.*

Input: $\{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^n$: data, α: learning rate, N: number of iterations, $\epsilon, \epsilon_{\boldsymbol{y}}$: perturbation magnitudes, s: the output layer of network, C: number of classes.

for $t \leftarrow 1$ *to* N **do**
  Sample a minibatch $\mathcal{M}_t$
  for i *in* $\mathcal{M}_t$ **do**
    Sample another index j
    $\widetilde{\boldsymbol{y}}_i \leftarrow (1-\epsilon_y)\boldsymbol{y}_i + \epsilon_y(\mathbf{1}-\boldsymbol{y}_j)/(C-1)$
    $\widetilde{\boldsymbol{x}}_i \leftarrow \underset{\widetilde{\boldsymbol{x}} \in \mathcal{B}(\boldsymbol{x}_i, \epsilon)}{\operatorname{argmin}} \frac{f_{\boldsymbol{\theta}}^s(\boldsymbol{x}_j) \cdot f_{\boldsymbol{\theta}}^s(\widetilde{\boldsymbol{x}})}{||f_{\boldsymbol{\theta}}^s(\boldsymbol{x}_j)||_2 ||f_{\boldsymbol{\theta}}^s(\widetilde{\boldsymbol{x}})||_2}$
    // Generate adversarial data.
  $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha \frac{1}{|\mathcal{M}_t|} \sum_{i \in \mathcal{M}_t} \nabla_{\boldsymbol{\theta}} \ell(f_{\boldsymbol{\theta}}(\widetilde{\boldsymbol{x}}_i), \widetilde{\boldsymbol{y}}_i)$
  // Update $\boldsymbol{\theta}$ over adversarial data.

3 Learning-to-Learn (L2L) Framework

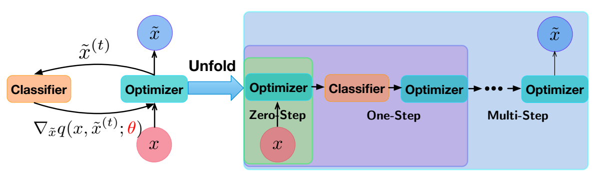


Figure 1: *An illustration of L2L: A neural network models optimizer for generating attack network.*

Since the hand-designed methods for bilevel problem (1) do not perform well, we propose to learn an optimizer for the follower problem. Specifically, we parameterize $\boldsymbol{\delta} = \widetilde{\boldsymbol{x}} - \boldsymbol{x}$, the perturbation[1], by a neural network $g_{\boldsymbol{\phi}}(\mathcal{A}_{f_{\boldsymbol{\theta}}}(\boldsymbol{x}, \boldsymbol{y}))$ with input $\mathcal{A}_{f_{\boldsymbol{\theta}}}(\boldsymbol{x}, \boldsymbol{y})$ summarizing the information of data and classifier $f_{\boldsymbol{\theta}}(\cdot)$. We first show how our method works on the DRO: We convert DRO problem (2) and (3) to

$$\min_{\boldsymbol{\theta}} \tfrac{1}{n} \textstyle\sum_{i=1}^n \ell(f_{\boldsymbol{\theta}}(\boldsymbol{x}_i + g_{\boldsymbol{\phi}}(\mathcal{A}_{f_{\boldsymbol{\theta}}}(\boldsymbol{x}_i, \boldsymbol{y}_i))), \boldsymbol{y}_i), \qquad (7)$$

where $\boldsymbol{\phi}^*$ is defined as the solution to the problem:

$$\boldsymbol{\phi}^* \in \operatorname{argmax}_{\boldsymbol{\phi}} \tfrac{1}{n} \textstyle\sum_{i=1}^n \ell(f_{\boldsymbol{\theta}}(\boldsymbol{x}_i + g_{\boldsymbol{\phi}}(\mathcal{A}_{f_{\boldsymbol{\theta}}}(\boldsymbol{x}_i, \boldsymbol{y}_i))), \boldsymbol{y}_i),$$
$$\text{s.t.} \quad g_{\boldsymbol{\phi}}(\mathcal{A}_{f_{\boldsymbol{\theta}}}(\boldsymbol{x}_i, \boldsymbol{y}_i)) \in \mathcal{B}(\mathbf{0}, \epsilon), i \in [1, ..., n].$$

The optimizer $g_{\boldsymbol{\phi}}$ targets on generating optimal perturbations under constraints $g_{\boldsymbol{\phi}}(\mathcal{A}_{f_{\boldsymbol{\theta}}}(\boldsymbol{x}_i, \boldsymbol{y}_i)) \in \mathcal{B}(\mathbf{0}, \epsilon)$.

[1]This helps to handle the constraints $\boldsymbol{\delta} \in \mathcal{B}(\mathbf{0}, \epsilon)$.

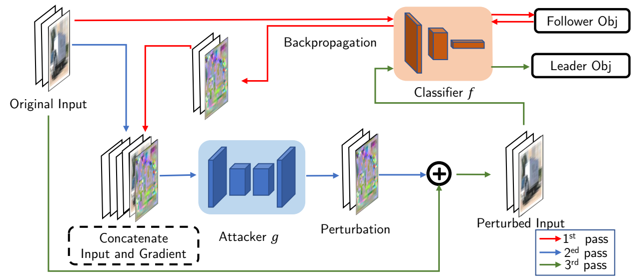


Figure 2: *The architecture of adversarial training with gradient attacker model.*

These constraints can be handled by a tanh function and an ϵ scaler in the last layer of $g_{\boldsymbol{\phi}}$. L2L framework is very flexible: We can choose different $\mathcal{A}_{f_{\boldsymbol{\theta}}}(\boldsymbol{x}, \boldsymbol{y})$ as the input and mimic multi-step algorithms shown in Figure 1. We provide three examples for DRO:

Naive Attacker. This is the simplest example among our methods, taking original image $\boldsymbol{x}_i$ as input, i.e.,

$$\mathcal{A}_{f_{\boldsymbol{\theta}}}(\boldsymbol{x}_i, \boldsymbol{y}_i) = \boldsymbol{x}_i \quad \text{and} \quad \boldsymbol{\delta}_i = g_{\boldsymbol{\phi}}(\boldsymbol{x}_i).$$

With this, L2L training is similar to GAN training. The major difference is that the generator in GAN yields synthetic data from random noises, while the naive attacker generates perturbations via samples.

Gradient Attacker. Motivated by FGSM, we design an attacker which takes the gradient information into consideration. Specifically, we concatenate image $\boldsymbol{x}_i$ and gradient $\Delta_i = \nabla_{\boldsymbol{x}} \ell(f_{\boldsymbol{\theta}}(\boldsymbol{x}_i), \boldsymbol{y}_i)$ as the input of g:

$$\mathcal{A}_{f_{\boldsymbol{\theta}}}(\boldsymbol{x}_i, \boldsymbol{y}_i) = \left[\boldsymbol{x}_i, \Delta_i\right] \quad \text{and} \quad \boldsymbol{\delta}_i = g_{\boldsymbol{\phi}}\left([\boldsymbol{x}_i, \Delta_i]\right).$$

With more information, the attacker is more effective to learn and yields more powerful perturbations.

Multi-Step Gradient Attacker. Motivated by PGM, we adapt the RNN to mimic a multi-step gradient update. Specifically, we use the gradient optimizer network as the cell of RNN sharing the same parameter $\boldsymbol{\phi}$. As we mentioned earlier, the number of layers/iterations in the RNN for modeling algorithms cannot be very large so as to avoid significant computational burden in backpropagation. In this paper, we focus on a length-two RNN to mimic a two-step gradient update. The corresponding perturbation becomes:

$$\widetilde{\boldsymbol{x}}_i = \boldsymbol{x}_i + \Pi_\epsilon\left(\boldsymbol{\delta}_i^{(0)} + g_{\boldsymbol{\phi}}\left([\widetilde{\boldsymbol{x}}_i^{(0)}, \nabla_{\boldsymbol{x}} \ell(f_{\boldsymbol{\theta}}(\widetilde{\boldsymbol{x}}_i^{(0)}), \boldsymbol{y}_i)]\right)\right).$$

Here $\boldsymbol{\delta}_i^{(0)} = g_{\boldsymbol{\phi}}\left([\boldsymbol{x}_i, \nabla_{\boldsymbol{x}} \ell(f_{\boldsymbol{\theta}}(\boldsymbol{x}_i), \boldsymbol{y}_i)]\right)$, $\widetilde{\boldsymbol{x}}_i^{(0)} = \boldsymbol{x}_i + \boldsymbol{\delta}_i^{(0)}$. Taking gradient attackers as an example, Figure 2 illustrates how L2L works and jointly trains two networks: The first forward pass is used to obtain gradient of the classification loss over the clean data; The second forward pass is used to generate perturbation $\boldsymbol{\delta}_i$ by the attacker g; The third forward pass is used to calculate the adversarial loss ℓ in (7). Since our gradient

Algorithm 3 *L2L-based DRO with gradient attacker.*

Input: $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$: clean data, α_1, α_2: learning rates, N: number of epochs.
for $t \leftarrow 1$ *to* N **do**
  Sample a minibatch $\mathcal{M}_t$
  for i *in* $\mathcal{M}_t$ **do**
    $\boldsymbol{u}_i \leftarrow \nabla_{\boldsymbol{x}} \ell(f_{\boldsymbol{\theta}}(\boldsymbol{x}_i), y_i)$, $\boldsymbol{\delta}_i \leftarrow g_{\boldsymbol{\phi}}([\boldsymbol{x}_i, \boldsymbol{u}_i])$
    //Generate perturbation by $g_{\boldsymbol{\phi}}$.
  $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \frac{\alpha_1}{|\mathcal{M}_t|} \sum_{i \in \mathcal{M}_t} \nabla_{\boldsymbol{\theta}} \ell(f_{\boldsymbol{\theta}}(\boldsymbol{x}_i + \boldsymbol{\delta}_i), \boldsymbol{y}_i)$
  // Update $\boldsymbol{\theta}$ over adversarial data.
  $\boldsymbol{\phi} \leftarrow \boldsymbol{\phi} + \frac{\alpha_2}{|\mathcal{M}_t|} \sum_{i \in \mathcal{M}_t} \nabla_{\boldsymbol{\phi}} \ell(f_{\boldsymbol{\theta}}(\boldsymbol{x}_i + \boldsymbol{\delta}_i), \boldsymbol{y}_i)$
  // Update $\boldsymbol{\phi}$ over adversarial data.

attacker only needs one backpropagation, it amortizes the adversarial training cost, which leads to better computational efficiency. Moreover, L2L may adapt to the underlying optimization problem and yield better solution for the follower problem. The corresponding procedure of L2L is shown in Algorithm 3.

Algorithm 4 *L2L-based AIT with gradient attacker.*

Input: $\{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^n$: data, α_1, α_2: learning rates, N: number of iterations, $\epsilon_{\boldsymbol{y}}$: perturbation magnitudes.
for $t \leftarrow 1$ *to* N **do**
  Sample a minibatch $\mathcal{M}_t$
  for i *in* $\mathcal{M}_t$ **do**
    Sample another index j
    $\widetilde{\boldsymbol{y}}_i \leftarrow (1 - \epsilon_{\boldsymbol{y}})\boldsymbol{y}_i + \epsilon_{\boldsymbol{y}}(\boldsymbol{1} - \boldsymbol{y}_j)/(C-1)$
    $\boldsymbol{u}_i = \nabla_{\boldsymbol{x}_i} q_{f_{\boldsymbol{\theta}}}(\boldsymbol{x}_i, \boldsymbol{x}_j)$, $\boldsymbol{\delta}_i \leftarrow g_{\boldsymbol{\phi}}(\boldsymbol{x}_i, \boldsymbol{u}_i)$
    //Generate perturbation by $g_{\boldsymbol{\phi}}$.
  $\boldsymbol{\phi} \leftarrow \boldsymbol{\phi} - \frac{\alpha_2}{|\mathcal{M}_t|} \sum_{i \in \mathcal{M}_t} \nabla_{\boldsymbol{\phi}} q_{f_{\boldsymbol{\theta}}}(\boldsymbol{x}_i + \boldsymbol{\delta}_i, \boldsymbol{x}_j)$
  //Update $\boldsymbol{\phi}$ over adversarial data.
  $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \frac{\alpha_1}{|\mathcal{M}_t|} \sum_{i \in \mathcal{M}_t} \nabla_{\boldsymbol{\theta}} \ell(f_{\boldsymbol{\theta}}(\boldsymbol{x}_i + \boldsymbol{\delta}_i), \widetilde{\boldsymbol{y}}_i)$
  //Update $\boldsymbol{\theta}$ over adversarial data.

It is straightforward to extend L2L to AIT as shown in Algorithm 4. We simply replace the gradient of ℓ, $\nabla_{\boldsymbol{x}} \ell(f_{\boldsymbol{\theta}}(\boldsymbol{x}_i), \boldsymbol{y}_i)$, by the gradient of $q_{f_{\boldsymbol{\theta}}}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \frac{f_{\boldsymbol{\theta}}^s(\boldsymbol{x}_i) \cdot f_{\boldsymbol{\theta}}^s(\boldsymbol{x}_j)}{||f_{\boldsymbol{\theta}}^s(\boldsymbol{x}_i)||_2 ||f_{\boldsymbol{\theta}}^s(\boldsymbol{x}_j)||_2}$, $\nabla_{\boldsymbol{x}_i} q_{f_{\boldsymbol{\theta}}}(\boldsymbol{x}_i, \boldsymbol{x}_j)$ in the attacker input. Taking gradient network as an example, given a sample $(\boldsymbol{x}_i, \boldsymbol{y}_i)$, we randomly select another sample $(\boldsymbol{x}_j, \boldsymbol{y}_j)$, and yield the adversarial sample as follows:

$$\widetilde{\boldsymbol{x}}_i = \boldsymbol{x}_i + g_{\boldsymbol{\phi}}\Big([\boldsymbol{x}_i, \nabla_{\boldsymbol{x}_i} q_{f_{\boldsymbol{\theta}}}(\boldsymbol{x}_i, \boldsymbol{x}_j)]\Big), \tag{8}$$

and adopt the corresponding label vector $\widetilde{\boldsymbol{y}}_i$ from (6).

4 Experiments

To demonstrate the effectiveness and computational efficiency of L2L, we conduct experiments over both CIFAR-10 and CIFAR-100 datasets. We compare our methods with original PGM training and adversarial interpolation training. All implementations are done in PyTorch with one single NVIDIA 2080 Ti GPU. Here we discuss the white-box setting, which is the most direct way to evaluate the robustness.

Classifier Network. All experiments adopt a 34-layer wide residual network (WRN-34-10, Zagoruyko and Komodakis (2016)) implemented by Zhang et al. (2019) as the classifier network. For each method, we train the classifier network from scratch.

Table 1: *Attacker Architecture: k, c, s, p Denote the Kernel Size, Output Channels, Stride and Padding Parameters of Convolutional Layers, Respectively.*

Conv:	$[k = 3 \times 3, c = 64, s = 1, p = 1]$, BN+ReLU
ResBlock:	$[k = 3 \times 3, c = 128, s = 1, p = 1]$
ResBlock:	$[k = 3 \times 3, c = 256, s = 1, p = 1]$
ResBlock:	$[k = 3 \times 3, c = 128, s = 1, p = 1]$
Conv:	$[k = 3 \times 3, c = 3, s = 1, p = 1]$, tanh

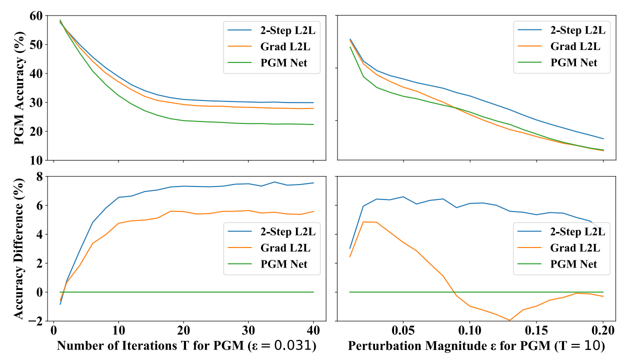


Figure 3: *Robust accuracy against perturbation magnitude and number of iteration of PGM over CIFAR-100;. (Top) Accuracy; (Bottom) Performance gain over PGM Net. See more results in Appendix E.*

Attacker. Table 1 presents the architecture of our attacker network[2]. We adopt the ResBlock proposed in Miyato et al. (2018). The detailed structure of ResBlock is provided in Appendix B. Batch normalization (BN) and activations, e.g., ReLU and tanh, are applied when specified. The tanh function can easily make the output of attacker satisfy the constraints.

White-box and Black-box[3]. We compare different methods under both white-box and black-box settings. Under the white-box setting, attackers can access all parameters of target models and generate adversarial examples based on the models; whereas under the black-box setting, we adopt the standard transfer at-

[2] We provide another attacker architecture with downsampling modules in the Section B. With such an attacker, L2L adversarial training is less stable, but faster.

[3] Due to space limit, the full results under the black-box setting are provided in Appendix C.

Table 2: *Results of Different Defense Methods under the White-box Setting.*

Defense Method	Attack	Data Set	Accuracy	
			Clean	Robust
Stability Train Zheng et al. (2016)	PGM-20	CIFAR-10	94.64%	0.15%
PGM Net Madry et al. (2017)	PGM-20		87.30%	47.04%
Naive L2L	PGM-20	CIFAR-10	94.53%	0.01%
Grad-only L2L	PGM-20		86.28%	49.94%
2-Step Grad-only L2L	PGM-20		85.8%	53.85%
Grad L2L	PGM-20		85.84%	51.17%
2-Step L2L	PGM-20		85.35%	54.32%
Grad L2L	PGM-100		85.84%	47.72%
2-Step L2L	PGM-100		85.35%	52.12%
Grad L2L	CW		85.84%	53.5%
2-Step L2L	CW		85.35%	57.07%
Grad L2L	Random		85.84%	82.67%
2-Step L2L	Random		85.35%	83.10%
Grad L2L	Grad L2L		85.84%	49.68%
2-Step L2L	2-Step L2L		85.35%	52.71%
PGM Net	PGM-20	CIFAR-100	62.68%	23.75%
Grad-only L2L	PGM-20		62.4%	27.64%
2-Step Grad-only L2L	PGM-20		60.25%	31.24%
Grad L2L	PGM-20		62.18%	28.67%
2-Step L2L	PGM-20		60.95%	31.03%
PGM Net	PGM-100		62.68%	22.06%
Grad L2L	PGM-100		62.18%	26.69%
2-Step L2L	PGM-100		60.95%	29.75%
PGM Net	CW		62.68%	25.95%
Grad L2L	CW		62.18%	29.65%
2-Step L2L	CW		60.95%	32.28%

tack method from Liu et al. (2016) as accessing parameters is prohibited.

Robust Evaluation[4]**.** We evaluate the robustness of the networks by PGM and CW attacks with the maximum perturbation magnitude $\epsilon = 0.031$ (after rescaling the pixels to $[0, 1]$) over CIFAR 10 and 100. For PGM attack, we use 20 and 100-iteration PGM with a perturbation step size $\eta = 0.003$, and for each sample we initialize the perturbation randomly in $\mathcal{B}(0, 10^{-4})$. For CW attack, we adopt the implementation in Paszke et al. (2017), and set the maximum number of iterations as 100. For each method, we repeat 5 runs with different random initial seed and report the *worst result*. For CIFAR-10, we also evaluate the robustness of Grad L2L and 2-Step L2L networks using random attacks, for which we uniformly sample 10^5 perturbations in $\mathcal{B}(0, 0.031)$ adding to each test sample. We also evaluate the robustness of Grad L2L and 2-Step L2L networks under their own attackers.

[4]More detailed robustness checklist is in Appendix E.

4.1 PGM Training

For simplicity, we denote PGM Net as the classifier with PGM training, and Naive L2L, Grad L2L, and 2-Step L2L as the classifiers using L2L training with corresponding attackers. For reference, we also include some results of Grad-only L2L and 2-Step Grad-only L2L, whose attackers take the gradient information only without the raw images.

Original PGM. For CIFAR-10, we directly report the result from Madry et al. (2017) as the baseline; For CIFAR-100, we train a PGM Net as the baseline: For optimizer, we use stochastic gradient descent (SGD) algorithm with Polyak's momentum (parameter 0.9, Liu et al. (2018)) and weight decay (parameter 2×10^{-4}, Krogh and Hertz (1992)). In addition, we adapt the setting from Madry et al. (2017) but train the network for 100 epochs with initial learning rate 0.1, decay schedule [30,60,90], and decay rate 0.1. For adversarial samples, we use a 10-iteration PGM with the perturbation step size 0.007 in (4).

Table 3: *One epoch running time. (Unit: s)*

Dataset	Plain Net	PGM Net	Naive L2L	Grad L2L	2-Step L2L
CIFAR-10	106.5 ± 1.5	1310.8 ± 14.2	293.7 ± 3.1	617.5 ± 6.1	805.1 ± 8.1
CIFAR-100	106.9 ± 1.4	1354.8 ± 14.1	310.0 ± 2.9	623.1 ± 6.3	824.7 ± 8.4

PGM+L2L. We train two networks for 100 epochs. For classifier's optimizer, we use the same configuration as original PGM training; For attacker's optimizer, we use Adam optimizer (parameter $[0.9, 0.999]$, Kingma and Ba (2014)) with initial learning rate 10^{-3} (no learning rate decay) and weight decay (parameter 2×10^{-4}) so that it adaptively balances the updates in both leader and follower optimization problems.

Experiment Results. Table 2 shows the results of all PGM training methods over CIFAR-10 and 100 under the white-box setting. As can be seen, without gradient information, Naive L2L is vulnerable to the PGM attack. However, when the attacker utilizes the gradient information, Grad L2L and 2-Step L2L *significantly outperform* the PGM Net over CIFAR-10 and 100, with a slight loss for the clean accuracy. From the experiments on CIFAR-10, our Grad L2L and 2-Step L2L are robust to random attacks, where the accuracy is only slightly lower than the clean accuracy. Furthermore, the accuracy of our Grad/2-Step L2L model under the Grad/2-Step L2L attacker is comparable to the accuracy under PGM attacks, which shows that L2L attackers are able to generate strong attacks. As can be seen, PGM-100 is stronger than Grad L2L attacker (47.72% vs. 49.68%), but similar to the 2-Step L2L attacker (52.07% vs. 52.71%), which means 2-Step L2L attacker is much stronger than Grad L2L attacker and explains why 2-Step L2L is stronger than Grad L2L and PGM net. In addition, comparing Grad-only L2L with Grad L2L, we see that without the raw images fed into the attackers, Grad-only L2L is less robust to the PGM attack, though 2-Step Grad-only L2L and 2-Step L2L achieves comparable performance.

In addition, Table 3 shows one epoch running time of all methods over CIFAR-10 and 100. As can be seen, Grad L2L and 2-Step L2L is much faster than PGM Net. By further comparing the accuracy of Grad/2-Step L2L and PGM Net in Table 2, we find that L2L methods enjoy computational efficiency. In addition, Figure 3 presents the robust accuracy against number of iterations with $\epsilon = 0.031$ and perturbation magnitude (number of iterations $T = 10$). As can be seen, 2-Step L2L is much more robust than PGM Net.

4.2 Adversarial Interpolation Training

We conduct the experiments of AIT over CIFAR-10 using the code from Haichao Zhang (2019). [5]

Original AIT. We follow the experimental setting in Haichao Zhang (2019), but use a WRN-34-10. For classifier's optimizer, we use the same configuration in original PGM training. We choose the perturbation magnitude over label $\epsilon_{\boldsymbol{y}}$ as 0.5. In addition, we train the whole network for 200 epochs with initial learning rate 0.1, decay schedule [60,90], and decay rate 0.1. Moreover, in each epoch, we first use FGSM to yield training samples via (6), and then train the AIT Net over these adversarial samples.

AIT+L2L. We train for 200 epochs. For classifier's optimizer, we adopt the configuration of SGD from the original AIT; For attacker's optimizer, we use Adam (parameter $[0.9, 0.999]$) with initial step size as 10^{-3} (no decay) and weight decay (parameter 2×10^{-4}).

Table 4: *Results of AIT based defense methods under the white-box setting (CIFAR-10).*

Defense Method	Attack	Accuracy	
		Clean	Robust
AIT	PGM-20	90.43%	75.33%
Grad L2L	PGM-20	91.65%	80.87%
AIT	PGM-100	90.43%	67.84%
Grad L2L	PGM-100	91.65%	79.20%
AIT	CW-20	90.43%	64.79%
Grad L2L	CW-20	91.65%	74.88%
AIT	CW-100	90.43%	61.69%
Grad L2L	CW-100	91.65%	73.46%

Experiment Results. Table 4 shows the results of AIT methods over CIFAR-10 under the white-box setting. As can be seen, Grad L2L *significantly improves upon* the AIT Net over CIFAR-10 on both clean accuracy and robust accuracy.

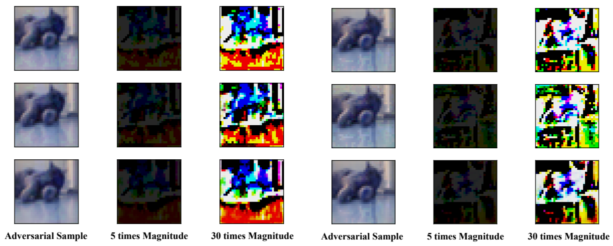
4.3 Visualization of Adversarial Examples

Figure 4 provides an illustrative example of adversarial perturbations generated by FGSM, PGM-20 and 2-Step L2L attacker for a *cat* in CIFAR-10. As can be seen, attacks for these two networks are different. Moreover, the perturbation generated by the 2-Step L2L attacker is much smoother than FGSM and PGM. In this example, 2-Step L2L labels all adversarial samples correctly; whereas the PGM Net is fooled by PGM-20 attack and misclassifies it as a *dog*.

Figure 5 provides an illustrative example of adversarial perturbations generated by PGM, AIT and Grad L2L

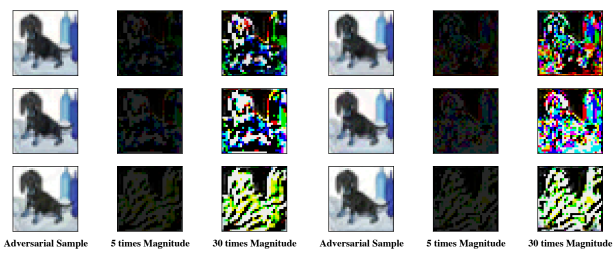
[5] `https://github.com/Adv-Interp/adv_interp`

for a *dog* in CIFAR-10. As can be seen, attacks for these two networks are very different: the attacks for the Grad L2L is more abundant in three channels. In this example, Grad L2L labels all adversarial samples correctly; whereas the AIT is fooled by all attacks and misclassifies it as a *horse*.



(a) PGM Net adv. samples (b) 2-Step L2L adv samples

Figure 4: *Adv. examples of FGSM (Top), PGM-20 (Mid), 2-Step L2L (Bottom) perturbations for a cat under PGM Net and 2-Step L2L with* $\epsilon = 0.031$.



(a) AIT adv. samples (b) Grad L2L adv. samples

Figure 5: *Illustrative adv. examples of PGM-20 (Top), AIT (Mid), and Grad L2L (Bottom) perturbations for a dog under AIT Net and Grad L2L with* $\epsilon = 0.031$.

5 Discussions

We discuss several closely related works:

• By leveraging the Fenchel duality and feature embedding technique, Dai et al. (2016) convert a learning conditional distribution problem to a minimax problem, which is similar to our naive attacker. Both approaches, however, lack the primal information. In contrast, gradient attacker network considers the gradient information of primal variables, and achieves good results with this key information.

• Goodfellow et al. (2014a) propose the GAN, which is very similar to our L2L framework. Both GAN and L2L contain one generator network and one classifier network, and jointly train these two networks. There are two major difference between GAN and our framework: (1) GAN aims to transform the random noises to the synthetic data which is similar to the training examples, while ours targets on transforming the training examples to the adversarial examples for robustifying the classifier; (2) Our attacker does not only take the training examples (analogous to the random noise in GAN) as the input, but also exploits the gradient information of the objective function, since it essentially represents an optimization algorithm. The training procedure of these two, however, are quite similar. We adopt some tricks from GAN training to our framework to stabilize training process, e.g., in Grad L2L, we use the two-time scale trick (Heusel et al., 2017).

• There are some other works simply combining the GAN framework and adversarial training together. For example, Baluja and Fischer (2017) and Xiao et al. (2018) propose some ad hoc GAN-based methods to robustify neural networks. Specifically, for generating adversarial examples, they only take training examples as the input of the generator, which lacks the information of the outer mimnimization problem. Instead, our proposed L2L methods (e.g., Grad L2L, 2-step L2L) connect outer and inner problems by delivering the gradient information of the objective function to the generator. This is a very important reason for our performance gain on the benchmark datasets. As a result, the aforementioned GAN-based methods are only robust to simple attacks, e.g., FGSM, on simple data sets, e.g., MNIST, but fail for strong attacks, e.g., PGM and CW, on complicated data sets, e.g. CIFAR, where our L2L methods achieve significantly better performance.

Training Stability: For improving the training stability, we use both clean image and the corresponding gradient as the input of the attacker. Without such gradient information, the attacker severely suffers from training instability, e.g., the Naive Attacker Network. Furthermore, we try another architecture with the widely used downsampling modules, called "slim attacker" in Section B. We observed that the slim attacker also suffers from training instability. We suspect that the downsampling causes the loss of information. Thus, we tried to enhance the slim attacker by skip layer connections. In this way, the training is stabilized. However, the robust performance is still worse than the proposed architecture.

Benefits of our L2L in adversarial training:
(1) Since neural networks have been known to be powerful in function approximation, our attacker g can yield strong adversarial perturbations. Since they are generated by the same attacker, attacker g learns some common structures across all samples;
(2) *Overparametrization* is conjectured to ease the training of deep neural networks. We believe that similar phenomena happen to our attacker network, and ease the adversarial training.

References

Andrychowicz, M., Denil, M., Gomez, S., Hoffman, M. W., Pfau, D., Schaul, T., Shillingford, B. and De Freitas, N. (2016). Learning to learn by gradient descent by gradient descent. In *Advances in Neural Information Processing Systems*.

Athalye, A., Carlini, N. and Wagner, D. (2018). Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. *arXiv preprint arXiv:1802.00420* .

Baluja, S. and Fischer, I. (2017). Adversarial transformation networks: Learning to generate adversarial examples. *arXiv preprint arXiv:1703.09387* .

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J. and Zaremba, W. (2016). Openai gym. *arXiv preprint arXiv:1606.01540* .

Carlini, N., Athalye, A., Papernot, N., Brendel, W., Rauber, J., Tsipras, D., Goodfellow, I. and Madry, A. (2019). On evaluating adversarial robustness. *arXiv preprint arXiv:1902.06705* .

Carlini, N. and Wagner, D. (2017). Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE.

Colson, B., Marcotte, P. and Savard, G. (2007). An overview of bilevel optimization. *Annals of operations research* **153** 235–256.

Dai, B., He, N., Pan, Y., Boots, B. and Song, L. (2016). Learning from conditional distributions via dual embeddings. *arXiv preprint arXiv:1607.04579* .

Dong, Y., Liao, F., Pang, T., Su, H., Zhu, J., Hu, X. and Li, J. (2018). Boosting adversarial attacks with momentum. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Finn, C., Abbeel, P. and Levine, S. (2017). Model-agnostic meta-learning for fast adaptation of deep networks. *arXiv preprint arXiv:1703.03400* .

Gao, R. and Kleywegt, A. J. (2016). Distributionally robust stochastic optimization with wasserstein distance. *arXiv preprint arXiv:1604.02199* .

Girshick, R., Donahue, J., Darrell, T. and Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. and Bengio, Y. (2014a). Generative adversarial nets. In *Advances in Neural Information Procesing Systems*.

Goodfellow, I. J., Shlens, J. and Szegedy, C. (2014b). Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572* .

Haichao Zhang, W. X. (2019). Adversarial interpolation training: A simple approach for improving model robustness.
URL `https://openreview.net/pdf?id=SyejjONYvr`

He, K., Zhang, X., Ren, S. and Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*.

He, K., Zhang, X., Ren, S. and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B. and Hochreiter, S. (2017). GANs trained by a two time-scale update ReLU converge to a local nash equilibrium. In *Advances in Neural Information Processing Systems*.

Ho, J. and Ermon, S. (2016). Generative adversarial imitation learning. *CoRR* **abs/1606.03476**.
URL `http://arxiv.org/abs/1606.03476`

Hochreiter, S., Younger, A. S. and Conwell, P. R. (2001). Learning to learn using gradient descent. In *International Conference on Artificial Neural Networks*. Springer.

Kingma, D. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* .

Krizhevsky, A. and Hinton, G. (2009). Learning multiple layers of features from tiny images. Tech. rep., Citeseer.

Krogh, A. and Hertz, J. A. (1992). A simple weight decay can improve generalization. In *Advances in neural information processing systems*.

Kurakin, A., Goodfellow, I. and Bengio, S. (2016). Adversarial machine learning at scale. *arXiv preprint arXiv:1611.01236* .

Li, Y., Li, L., Wang, L., Zhang, T. and Gong, B. (2019). Nattack: Learning the distributions of adversarial examples for an improved black-box attack on deep neural networks. *arXiv preprint arXiv:1905.00441* .

Liu, T., Chen, Z., Zhou, E. and Zhao, T. (2018). Toward deeper understanding of nonconvex stochastic optimization with momentum using diffusion approximations. *arXiv preprint arXiv:1802.05155* .

Liu, W., Zhang, Y.-M., Li, X., Yu, Z., Dai, B., Zhao, T. and Song, L. (2017). Deep hyperspherical learning. In *Advances in Neural Information Processing Systems*.

Liu, Y., Chen, X., Liu, C. and Song, D. (2016). Delving into transferable adversarial examples and black-box attacks. *arXiv preprint arXiv:1611.02770* .

Madry, A., Makelov, A., Schmidt, L., Tsipras, D. and Vladu, A. (2017). Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083* .

Miyato, T., Kataoka, T., Koyama, M. and Yoshida, Y. (2018). Spectral normalization for generative adversarial networks. In *International Conference on Learning Representations*. URL `https://openreview.net/forum?id=B1QRgziT-`

Nguyen, A., Yosinski, J. and Clune, J. (2015). Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L. and Lerer, A. (2017). Automatic differentiation in pytorch .

Rahimian, H. and Mehrotra, S. (2019). Distributionally robust optimization: A review. *arXiv preprint arXiv:1908.05659* .

Samangouei, P., Kabkab, M. and Chellappa, R. (2018). Defense-gan: Protecting classifiers against adversarial attacks using generative models. *arXiv preprint arXiv:1805.06605* .

Schmidhuber, J. (1987). *Evolutionary principles in self-referential learning, or on learning how to learn: the meta-meta-... hook.* Ph.D. thesis, Technische Universität München.

Schmidhuber, J. (1992). Learning to control fast-weight memories: An alternative to dynamic recurrent networks. *Neural Computation* **4** 131–139.

Schmidhuber, J. (1993). A neural network that embeds its own meta-levels. In *Neural Networks, 1993., IEEE International Conference on*. IEEE.

Schulman, J., Levine, S., Abbeel, P., Jordan, M. and Moritz, P. (2015). Trust region policy optimization. In *International conference on machine learning*.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I. and Fergus, R. (2013). Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199* .

Tabacof, P. and Valle, E. (2016). Exploring the space of adversarial images. In *2016 International Joint Conference on Neural Networks (IJCNN)*. IEEE.

Taigman, Y., Yang, M., Ranzato, M. and Wolf, L. (2014). Deepface: Closing the gap to human-level performance in face verification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Xiao, C., Li, B., Zhu, J.-Y., He, W., Liu, M. and Song, D. (2018). Generating adversarial examples with adversarial networks. *arXiv preprint arXiv:1801.02610* .

Younger, A. S., Hochreiter, S. and Conwell, P. R. (2001). Meta-learning with backpropagation. In *Neural Networks, 2001. Proceedings. IJCNN'01. International Joint Conference on*, vol. 3. IEEE.

Zagoruyko, S. and Komodakis, N. (2016). Wide residual networks. *arXiv preprint arXiv:1605.07146* .

Zhang, H., Cisse, M., Dauphin, Y. N. and Lopez-Paz, D. (2017). mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412* .

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. *arXiv preprint arXiv:1901.08573* .

Zheng, S., Song, Y., Leung, T. and Goodfellow, I. (2016). Improving the robustness of deep neural networks via stability training. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.