Adversarial Feature Desensitization

Anonymous Author(s)
Affiliation
Address
email

Abstract

Neural networks are known to be vulnerable to adversarial attacks – slight but carefully constructed perturbations of the inputs which can drastically impair the network's performance. Many defense methods have been proposed for improving robustness of deep networks by training them on adversarially perturbed inputs. However, these models often remain vulnerable to new types of attacks not seen during training, and even to slightly stronger versions of previously seen attacks. In this work, we propose a novel approach to adversarial robustness, which builds upon the insights from the domain adaptation field. Our method, called *Adversarial Feature Desensitization (AFD)*, aims at learning features that are invariant towards adversarial perturbations of the inputs. This is achieved through a game where we learn features that are both predictive and robust (insensitive to adversarial attacks), i.e. cannot be used to discriminate between natural and adversarial data. Empirical results on several benchmarks demonstrate the effectiveness of the proposed approach against a wide range of attack types and attack strengths.

1 Introduction

When training a classifier, it is common to assume that the training and test samples are drawn from the same underlying distribution. In adversarial machine learning, however, this assumption is intentionally violated by using the classifier itself to perturb the samples from the original (natural) data distribution towards a new distribution over which the classifier's error rate is increased [52]. As expected, when tested on such adversarially generated input distribution, the classifier severely underperforms. To date, various methods have been proposed to defend the neural networks against adversarial attacks [34, 2], additive noise patterns and corruptions [24, 25, 45], and transformations [17]. Among these methods, two of the most successful adversarial defense methods to date are adversarial training [34], which trains the neural network with examples that are perturbed to maximize the loss on the target model, and TRADES [57], which regularizes the classifier to push the decision boundary away from the data. While past adversarial defence methods have successfully improved the neural network robustness against adversarial examples, it has also been shown that these robust networks remain susceptible to even slightly larger adversarial perturbations or other forms of attacks [19, 46, 48].

In this paper, we propose to view the problem of adversarial robustness through the lens of domain adaptation, and to consider distributions of natural and adversarial images as distinct input domains that a classifier is expected to perform well on. We then focus our attention on learning features that are invariant under such domain shifts. Building upon domain adaptation literature [4], we use the classification-based $\mathcal{H}\Delta\mathcal{H}$-divergence to quantify the distance between the natural and adversarial domains. The theory of domain adaptation allows us to formulate a bound on the adversarial classification error (i.e. the error under the distribution of adversarial examples) in terms of the classification error on natural images and the divergence between the natural and adversarial features.

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

We further propose an algorithm for minimizing the adversarial error using this bound. For this, we train a classifier and a domain discriminator to respectively minimize their losses on the label classification and domain discrimination tasks. The feature extractor is trained to minimize the label classifier's loss and maximise the discriminator's loss. In this way, the feature extractor network is encouraged to learn features that are both predictive for the classification task and insensitive to the adversarial attacks. The proposed setup is conceptually similar to prior work in adversarial domain adaptation [18, 53], where domain-invariant features are learned through an adversarial game between the domain discriminator and a feature extractor network.

This setup is similar to the adversarial learning paradigm widely used in image generation and transformation [20, 28, 60], unsupervised and semi-supervised learning [39], video prediction [35, 31], active learning [47], and continual learning [16]. Some prior work have also considered adversarial learning to tackle the problem of adversarial examples [54, 36, 9, 8]. These methods used generative models to learn the distribution of the adversarial images[54, 36], or to learn the distribution of input gradients[9, 8]. Unlike our method which learns a discriminator function between distributions of adversarial and natural features and updates the feature extractor to reduce the discriminability of those distributions.

The main contributions of this work are as follows:

- We apply domain-adaptation theory to adversarial robustness problem; this allows to bound the adversarial error in terms of the error on the natural inputs and the divergence between the feature (representation) distributions of adversarial and natural domains.

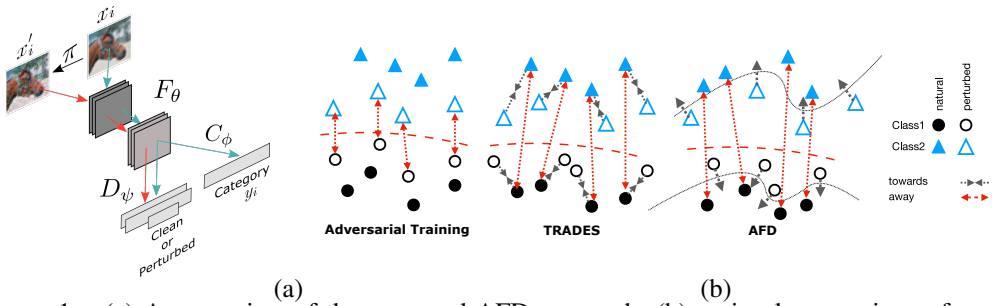
- Aiming to minimize this bound, we propose a method which learns adversarially robust features that are both predictive and insensitive to adversarial attacks, i.e. cannot be used to discriminate between natural and adversarial data.

- We empirically demonstrate the effectiveness of the proposed method in learning robust models against a wide range of attack types and attack strengths, and show that our proposed approach often significantly outperforms most previous defense methods.

2 Related Work

There is an extensive literature on mitigating susceptibility to adversarial perturbations [34, 57, 13, 59, 3, 22, 7]. Adversarial training [34] is one of the earliest successful attempts to improve robustness of the learned representations to potential perturbations to the input pattern by solving a "saddle point" problem composed of an inner and outer adversarial optimization. TRADES [57] adds a regularization term to the cross-entropy loss which penalizes the network for assigning different labels to natural images and their corresponding perturbed images. [41] proposed an additional regularization term (local linearity regularizer) that encourages the classification loss to behave linearly around the training examples. [55, 51] proposed to regularize the flatness of the loss to improve adversarial robustness.

Our work is closely related to the domain adaptation literature in which adversarial optimization has recently gained much attention [18, 32, 53]. From this viewpoint one could consider the clean and perturbed inputs as two distinct domains for which a network aims to learn an invariant feature set. Although in our setting, i) the perturbed domain continuously evolves while the parameters of the feature network are tuned; ii) unlike the usual setting in domain-adaptation problems, here we have access to the labels associated with samples from the perturbed (target) domain. Recent work[49] regularized the network to have similar logit values in response to clean and perturbed inputs and showed that this additional term leads to better robust generalization to unseen perturbations. Related to this, Adversarial Logit Pairing [27] increases robustness by directly matching the logits for clean and adversarial inputs. JARN [9] Another line of work is on developing certified defenses which consist of methods with provable bounds over which the network is *certified* to operate robustly [58, 56, 10]. While these approaches provide a sense of guarantee about the proposed defenses, they are usually prohibitively expensive to train, drastically reduce the performance of the network on natural images, and the empirical robustness gained against standard attacks is low.



(a) (b)

Figure 1: (a) An overview of the proposed AFD approach; (b) a visual comparison of several adversarial robustness methods (Adversarial training [34], TRADES [57], and AFD). The dotted black line corresponds to the decision boundary of the adversarial discriminator.

3 Our approach

We will now make a connection between the domain adaptation and adversarial robustness, and build upon this connection to develop an approach for improving the network's robustness against adversarial attacks.

3.1 Preliminaries

Let $F_\theta(x) : \mathcal{X} \to \mathcal{Z}$, where $\mathcal{X} \subseteq \mathbb{R}^n$, $\mathcal{Z} \subseteq \mathbb{R}^m$, be a *feature extractor* (e.g. a neural network with parameters θ) mapping the input $x \in \mathcal{X}$ into the feature vector (representation) $z \in \mathcal{Z}$, and let $C_\phi : \mathcal{Z} \to \mathcal{Y}$, where $\mathcal{Y} = \{1, \dots, K\}$ are the class labels, be a *classifier*, with parameters ϕ (e.g., the last linear layer of a neural network plus the softmax function, on top of the extracted features).

Adversarial attack: Let $\pi(x, \epsilon)$ denote a perturbation function (an adversarial attack) which, for a given $(x, y) \in \mathcal{X} \times \mathcal{Y}$, generates a perturbed sample $x' \in \mathcal{B}(x, \epsilon)$ within the ϵ-neighborhood of x, $\mathcal{B}(x, \epsilon) = \{x' \in \mathcal{X} : \|x' - x\| < \epsilon\}$, by solving the following maximization problem

$$\max_{t \in \mathcal{B}(x,\epsilon)} \mathcal{L}(C_\phi(F_\theta(t)), y), \tag{1}$$

where $\mathcal{L}$ is the task classification loss function. In practice, however, the perturbed sample x' found by an attacker is typically an approximate rather than the exact solution to this maximization problem.

In order to characterize the distance between the natural and adversarial data distributions, the following notion of distance between two probability distributions, defined in [4, 18], will be used later to make a connection with domain adaptation theory.

$\mathcal{H}\Delta\mathcal{H}$-distance: Let $\mathcal{H}$ be a set of binary classifiers (hypotheses), called a hypothesis space; then the symmetric difference hypothesis space $\mathcal{H}\Delta\mathcal{H}$ defines the set of hypotheses that capture the disagreements between two hypotheses in $\mathcal{H}$, as in [4]:

$$g \in \mathcal{H}\Delta\mathcal{H} \iff g(x) = h(x) \oplus h'(x) \quad \text{for some } h, h' \in \mathcal{H}, \tag{2}$$

where $\oplus$ denotes the XOR function. Then the $\mathcal{H}\Delta\mathcal{H}$-distance [4, 18] between two data distributions (domains) $\mathcal{S}$ and $\mathcal{T}$, with respect to the hypothesis space $\mathcal{H}$, is defined as:

$$d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{S}, \mathcal{T}) = 2 \sup_{h \in \mathcal{H}\Delta\mathcal{H}} \left| P_{x\sim\mathcal{S}}\big[h(x) = 1\big] - P_{x\sim\mathcal{T}}\big[h(x) = 1\big] \right|. \tag{3}$$

This equation turns into an inequation when the supremum is taken over the hypothesis space $\mathcal{H}$ instead of $\mathcal{H}\Delta\mathcal{H}$ [18].

3.2 A Domain Adaptation View of Adversarial Robustness

A *domain* is defined as a data distribution $\mathcal{D}$ on the set of inputs $\mathcal{X}$ [5]. In the adversarial robustness setting, we consider two domains – the natural and the adversarial domains, corresponding respectively to the target and source domains in domain adaptation. We denote by $\mathcal{D}_\mathcal{X}$ and $\mathcal{D}'_\mathcal{X}$ the natural and adversarial distributions of instances in , respectively, and by $\mathcal{D}_\mathcal{Z}$ and $\mathcal{D}'_\mathcal{Z}$ their corresponding induced distributions over the feature space $\mathcal{Z}$. As in domain adaptation, we assume that $f : \mathcal{X} \to \mathcal{Y}$ is a

labeling function common to both domains. The expected classification error $\epsilon_{\mathcal{Z}}$ of the classifier C_ϕ over $\mathcal{D}_{\mathcal{Z}}$ is defined as the probability that the classifier C_ϕ disagrees with the function $\tilde{f}$:

$$\epsilon_{\mathcal{Z}}(C_\phi) = E_{z\sim\mathcal{D}_{\mathcal{Z}}}\big[y \neq C_\phi(z)\big], \tag{4}$$

where $\tilde{f} : \mathcal{Z} \to \mathcal{Y}$ is a mapping from the features to the class label such that $f(x) = \tilde{f}(F_\theta(x))$. We similarly define $\epsilon'_{\mathcal{Z}}$ as the expected error of C_ϕ over $\mathcal{D}_{\mathcal{Z}'}$. Using theorem 2 from [4] that relates the source and the target domain errors, we get an upper bound on the expected adversarial error $\epsilon'_{\mathcal{Z}}$ as:

$$\epsilon'_{\mathcal{Z}}(h) \leq \epsilon_{\mathcal{Z}}(h) + \frac{1}{2} d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_{\mathcal{Z}}, \mathcal{D}'_{\mathcal{Z}}) + c, \tag{5}$$

where c is a constant term w.r.t. h. Eq. 5 essentially gives a bound on the adversarial error $\epsilon'_{\mathcal{Z}}$ in terms of the natural error $\epsilon_{\mathcal{Z}}$ and a divergence $d_{\mathcal{H}\Delta\mathcal{H}}$ between the natural and adversarial domains with respect to their induced representation distributions $\mathcal{D}_{\mathcal{Z}}$ and $\mathcal{D}'_{\mathcal{Z}}$. In the next section, we will describe an algorithm for improving adversarial robustness of a model by iteratively estimating and minimizing these two components of the error bound.

3.3 Adversarial Feature Desensitization

Based on Eq. 5, the expected adversarial error could be reduced by jointly minimizing the natural error and the divergence between the distributions of natural and adversarial representations $d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_{\mathcal{Z}}, \mathcal{D}'_{\mathcal{Z}})$. While minimizing the natural error ϵ_X is straightforward, minimizing the cross-domain divergence requires us to estimate $d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_{\mathcal{Z}}, \mathcal{D}'_{\mathcal{Z}})$. As was shown before [18], training a *domain discriminator* D_ψ is closely related to estimating the $d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_{\mathcal{Z}}, \mathcal{D}'_{\mathcal{Z}})$. The domain discriminator is a classifier trained to assign a label of 1 to samples from $\mathcal{D}_{\mathcal{Z}}$, and -1 to samples from $\mathcal{D}'_{\mathcal{Z}}$. Namely, it is shown [18] that

$$d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_{\mathcal{Z}}, \mathcal{D}'_{\mathcal{Z}}) \leq 2 \sup_{h\in\mathcal{H}} |\alpha_{\mathcal{D}_{\mathcal{Z}}, \mathcal{D}'_{\mathcal{Z}}}(h) - 1|, \tag{6}$$

where $\alpha_{\mathcal{D}_{\mathcal{Z}}, \mathcal{D}'_{\mathcal{Z}}}(h) = P_{z\sim\mathcal{D}_{\mathcal{Z}}}\big[h(z) = 1\big] + P_{z\sim\mathcal{D}'_{\mathcal{Z}}}\big[h(z) = -1\big]$ combines the true positives and true negatives, and is thus maximized by the optimal domain discriminator $h = D_\psi$. Note that, if the domain distributions $\mathcal{D}_{\mathcal{Z}}$ and $\mathcal{D}'_{\mathcal{Z}}$ are the same, then even the best choice of domain discriminator D_ψ will achieve chance-level accuracy, corresponding to $\alpha_{\mathcal{D}_{\mathcal{Z}}, \mathcal{D}'_{\mathcal{Z}}}(D_\psi) = 1$. *Our approach will aim at minimizing this estimated distance $d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_{\mathcal{Z}}, \mathcal{D}'_{\mathcal{Z}})$ by tuning the feature extractor network parameters θ in the direction that pushes the distributions $\mathcal{D}_{\mathcal{Z}}$ and $\mathcal{D}'_{\mathcal{Z}}$ closer together.* In parallel, we train the domain discriminator to estimate and guide the progress of the feature extractor's tuning.

We now describe the proposed approach (see Algorithm 1) which essentially involves simultaneous training of the feature extractor F_θ, the task classifier C_ϕ and the domain discriminator D_ψ (see Figure 1a)[1]. One iteration of the training procedure consists of the following three steps.

First, parameters of the feature extractor F_θ and classifier C_ϕ are updated aiming to minimize the natural error $\epsilon_{\mathcal{X}}$ using the cross-entropy loss on natural inputs:

$$\mathcal{L}_C = -\frac{1}{m}\sum_{i=1}^{m} \tilde{y}_i \cdot \log\Big(\text{softmax}(C_\phi(F_\theta(x_i)))\Big), \tag{7}$$

where $\tilde{y}_i$ is a one-hot encoding of the true label of the i-th sample x_i.

Next, steps two and three essentially implement a two-player minimax game similar to that in Generative Adversarial Networks (GAN) [20], carried out between the feature extractor network F_θ and the domain discriminator D_ψ, with a value function

$$V(F_\theta,\, D_\psi) = \mathbb{E}_{p(y)}\big[\mathbb{E}_{p(x|y)}[\mathcal{S}(-D_\psi(F_\theta(x),\, y))]\big] + \mathbb{E}_{q(y)}\big[\mathbb{E}_{q(x|y)}[\mathcal{S}(D_\psi(F_\theta(x),\, y))]\big], \tag{8}$$

where $\mathcal{S}$ is the softplus function. In particular, parameters of the domain discriminator D_ψ are

[1]Note that we will somewhat abuse the notation, assuming that C_ϕ and D_ψ below correspond to the logits (last-layer output) of the corresponding networks. Also, we will use class-conditional discriminators, $D_\psi(F_\theta(x, y))$, i.e. train different domain discriminator for different label values y.

Algorithm 1: AFD training procedure

Input: Adversarial perturbation function (attack) π, feature extractor F_θ, task classifier C_ϕ, domain discriminator D_ψ, learning rates α, β, and γ.

repeat

  input next mini-batch $\{(x_i,\, y_i), ..., (x_m,\, y_m)\}$
  for i=1 to m: $x'_i \leftarrow \pi(x_i,\, \epsilon)$
  Compute $\mathcal{L}_C$ according to Eq. 7
  Compute $\mathcal{L}_D$ according to Eq. 9
  Compute $\mathcal{L}_F$ according to Eq. 10
  $(\theta,\, \phi) \leftarrow (\theta,\, \phi) - \alpha \nabla_{\theta,\, \phi} \mathcal{L}_C$ % update feature extractor and task classifier
  $\psi \leftarrow \psi - \beta \nabla_\psi \mathcal{L}_D$ % update domain discriminator
  $\theta \leftarrow \theta - \gamma \nabla_\theta \mathcal{L}_F$ % update feature extractor

until *convergence*;

updated to minimize the cross-entropy loss associated with discriminating natural and adversarial inputs, maximizing $\alpha(h)$ in Eq. 6.

$$\mathcal{L}_D = \frac{1}{m} \sum_{i=1}^{m} \Big[\mathcal{S}(-D_\psi(F_\theta(x_i),\, y_i)) + \mathcal{S}(D_\psi(F_\theta(x'_i),\, y_i)) \Big], \tag{9}$$

while the parameters of the feature extractor function F_θ are adversarially updated to maximize the domain discriminator's loss from Eq. 9

$$\mathcal{L}_F = \frac{1}{m} \sum_{i=1}^{m} \mathcal{S}(-D_\psi(F_\theta(x'_i),\, y_i)). \tag{10}$$

In Figure 1b, we visually compare the learning dynamics in adversarial training, TRADES and AFD. Essentially, the adversarial training solves the classification problem by pushing the representation of adversarial examples from different classes away. TRADES regularizes the normal classification loss on the natural inputs with an additional term that encourages the representation of adversarial and natural images to match. Similar to TRADES, in AFD, the regular classification loss on natural inputs is augmented but with an adversarial game which consists of training the domain discriminator that distinguishes between the adversarial and natural inputs for each class followed by updates to the feature extractor to make the representations for natural and adversarial examples to become indistinguishable from each other. Notably, because the parameter update for the feature extractor network is done to maximize the domain discriminator loss and not to decrease the loss for particular adversarial examples (as is done in adversarial training or TRADES), it potentially increases the network robustness against any perturbation that could be correctly classified using the same domain discriminator. This could potentially lead to a broader form of generalization learned by the network.

Discussion: Relation to Adversarial Training. Adversarial training minimizes the expected error on adversarial examples (the perturbed versions of the natural samples), generated by an attacker in order to maximize the classification loss. The adversarial training procedure involves a minimax optimization problem consisting of an inner maximization to find adversarial examples that maximize the classification loss and an outer minimization to find model parameters that minimize the adversarial loss. From the domain adaptation point of view, the inner optimization of adversarial training is equal to a sampling procedure that generates samples from the target domain. Intuitively, direct training of the classifier on samples from the target domain would be the best way to improve the accuracy in that domain (i.e. adversarial classification accuracy). However, it's important to note that the adversarial examples found through the inner optimization only approximately maximize the classification loss, and therefore the adversarial error associated with these samples only act as a lower bound on the adversarial error and therefore the outer loop of the adversarial training method essentially minimizes a lower bound on the adversarial classification error. In contrast to this setup, our proposed method minimizes a conservative upper bound on the adversarial error and therefore is more likely to generalize to a larger set of unseen attacks, and to stronger versions of previously seen attacks (i.e. ones that generate higher-loss samples in the inner optimization loop).

4 Experiments

4.1 Experimental setup

Datasets. We validated our proposed method on several common datasets including MNIST [30], CIFAR10, CIFAR100 [29], and tiny-Imagenet [26]. The inputs for all datasets were used in their original resolution except for tiny-Imagenet where the inputs were resized to 32×32 to allow the experiments to finish within reasonable time on two GPUs.

Adversarial attacks. To fairly assess the generalization ability of each defense method across attack types, we tested each network on 8 well-known adversarial attacks from the literature, using existing implementations from the Foolbox [42] and Advertorch [12] Python packages. Namely, we tested the models against different variations of the Projected Gradient Descent (PGD) [34] (L_∞, L_2, L_1), Fast Gradient Sign Method (FGSM) [21], Momentum Iterative Method (MIM) [14], Decoupled Direction and Norm (DDN) [43], Deepfool [40], and C&W [6] attacks. Also to assess the generalization in robustness across stronger adversarial attacks, for each attack we also varied the ϵ value across a wide range and validated different models on each. Specific hyperparameters used for each attack are listed in Table-A2.

Feature extractor network F_θ and classifier C_ϕ. We used the same network architecture, ResNet18 [23] for the feature extractor and classifier networks in experiments on all datasets and only increased the number of features for more challenging datasets. The number of base filters in the ResNet architecture was set to 16 for MNIST and 64 for other datasets. We used the activations before the last linear layer as the the output of the feature extractor network ($\mathcal{Z}$) and the last linear layer as the classifier network C_ϕ. We added an activation normalization layer to the output of feature extractor network to provide normalized features to both C_θ and D_ψ networks.

Domain discriminator network D_ψ. We compared several variations of the domain discriminator architecture and evaluated its effect on robust classification on MNIST dataset (Table A5). Overall, we found that using deeper networks for domain discriminator and adding projection discriminator layer improves the robust classification accuracy. The number of hidden units in all layers of D_ψ were equal (64 for MNIST and 512 for other datasets). Following the common design principles in Generative Adversarial Networks literature, we used the spectral normalization [37] on all layers of D_ψ. In all experiments, the domain discriminator (D_ψ) consisted of three fully connected layers with Leaky ReLU nonlinearity followed by a projection discriminator layer that incorporated the labels into the adversarial discriminator through a dot product operation [38]. Further details of training for each experiment are listed in Table-A1.

Training parameters and baselines. All networks including baselines were trained on an adaptive version of PGD attack [11] that adaptively tunes the step size during the attack with virtually no computational overhead compared to standard PGD attack. We used $\epsilon = 0.3$, 0.031, and 0.016 for MNIST, CIFAR, and Tiny-Imagenet datasets respectively. To find the best learning rates, we randomly split the CIFAR10 train set into a train and validation sets (45000 and 5000 images in train and validation sets respectively). We then carried out a grid-search using the train-validation sets and picked the learning rates with highest validation performance. Based on this analysis, we selected the learning rate $\gamma = 0.5$ for tuning the feature extractor F_θ, and $\alpha = \beta = 0.1$ for tuning the parameters in domain discriminator D_ψ, and the task classifier C_ϕ.

In all experiments we trained two versions of the AFD model, one with losses L_D and L_F according to Eq. 9 and 10 which we call AFD-DCGAN and another version where we substitute the losses with those from the Wasserstein GAN [1] dubbed AFD-WGAN (see Eq. 11 and 12 in the Appendix). We mainly compared the performance of our proposed method with two prominent defense methods, adversarial training and TRADES. We used a re-implementation of adversarial training (AT) method [34] and the official code for TRADES[2] [57] and denoted these results with † in the tables. All experiments were run on NVIDIA V100 GPUs. We used one GPU for experiments on MNIST and 2 GPUs for other datasets.

4.2 Robust classification against nominal attacks

We first evaluated our method against adversarial attacks under similar settings to those used during training ($\epsilon = 0.3$, 0.031, and 0.015 for MNIST, CIFAR, and Tiny-Imagenet datasets respectively).

[2] https://github.com/yaodongyu/TRADES.git

Table 1: Comparison of adversarial accuracy against various attacks on different datasets. For PGD_∞ attack we used $\epsilon = 0.3$, 0.031, and 0.015 for MNIST, CIFAR10/CIFAR100, and Tiny-Imagenet datasets respectively and for C&W attack we used $\epsilon = 1$ for all datasets. † indicates replicated results. NT: natural training; AT: adversarial training; AFD: adversarial feature desensitization; WB: white-box attack; BB: black-box attack where the adversarial examples were produced by running the attack on the NT ResNet18 model. Numbers reported with $\mu \pm \sigma$ denote mean and std values over three independent runs with different random initialization. * RST[7] additionally uses 500K unlabeled images during training.

Method	**Dataset**	**Network**	**Clean**	**PGD$_\infty$** (WB)	**C&W$_2$** (WB)	**PGD$_\infty$** (BB)	**C&W$_2$** (BB)
NT†	MNIST	RN18	98.84	0.	62.43	50.82	96.48
AT[34] †		RN18	**99.35**	95.66	92.78	**98.92**	**98.95**
TRADES[57] †		RN18	99.14	94.81	90.08	98.5	98.57
AFD-DCGAN		RN18	99.24	95.72	93.78	98.65	98.49
AFD-WGAN		RN18	99.14	**97.68**	**97.68**	98.59	98.71
AT[34]	CIFAR10	RN18	87.3	45.8	-	86.0	-
TRADES[57]		RN18	84.92	56.61	-	87.60	-
RLFAT[50]		WRN-32-10	82.72	58.75	-	-	-
RST+[55, 7]*		WRN-34-10	89.82	64.86	-	-	-
LLR[41]		WRN-28-8	**86.83**	52.99	-	-	-
JARN[9]		WRN-34-10	84.8	46.7	-	-	-
NT†		RN18	94.89	0.55	0.31	17.93	-
AT[34] †		RN18	85.92	40.07	40.27	85.14	85.84
TRADES[57] †		RN18	81.94	53.3	40.24	80.82	81.74
AFD-DCGAN		RN18	**86.82**	44.35	50.93	**85.73**	**86.68**
AFD-WGAN		RN18	85.95	**59.38**	**62.43**	84.74	85.79
NT†	CIFAR100	RN18	76.76	0.01	0.52	-	-
AT[34] †		RN18	56.49	18.54	17.71	56.07	56.42
TRADES[57] †		RN18	60.32	**25.11**	20.52	59.62	60.29
AFD-DCGAN		RN18	**60.95**	18.06	21.47	**60.31**	**60.86**
AFD-WGAN		RN18	58.87	22.35	**25.33**	58.15	58.75
NT†	Tiny-IN	RN18	58.30	0.3	0.0	-	-
AT[34] †		RN18	43.80	12.62	4.90	41.87	42.86
TRADES[57] †		RN18	37.70	**13.26**	4.11	36.26	36.72
AFD-WGAN		RN18	**47.70**	11.49	**5.90**	**43.5**	**44.69**

Table 1 compares the robust classification performance of AFD and several other defense methods against PGD-L_∞ and C&W-L_2 white-box and black-box attacks. The black-box attacks were carried out by constructing the adversarial examples using a ResNet18 architecture trained on the natural inputs $x \sim D_\mathcal{X}$. Overall both versions of AFD (AFD-DCGAN and AFD-WGAN) were highly robust against all four tested attacks with AFD-WGAN being consistently at the top. In addition to the high adversarial accuracy, AFD-trained models also achieved higher "Clean" accuracy (on natural data) compared to strong baseline models like TRADES and Adversarial Training. The only exceptions among these was the $PGD_{\infty+}$ white-box attack on CIFAR100 and Tiny-Imagenet datasets in which AFD performed lower than some of the baselines. However, in both cases AFD models consistently outperformed the baselines against the other three attacks and, as we discuss in the next section, to most of the unseen attacks of various strengths. In addition, we also evaluated the AFD model against transfer black-box attacks from Adversarial Training and TRADES models which further showed higher robustness to those attacks too (Table-A3).

4.3 Robust classification against stronger and unseen attacks

To evaluate how each network generalizes to unseen domains of adversarial inputs (i.e. adversarial attacks generated with unseen forms of adversarial attacks), we additionally validated the classification robustness against a range of possible ϵ values for several widely used attacks that were not used during training. To fairly compare different models while considering both attack types and ϵ values, we computed the area-under-the-curve (AUC) for accuracy vs. epsilon for each attack (similar to Figure-2). Table-2 summarizes the AUC values for all 8 attack methods on four tested datasets. Compared with the baselines, we found that, AFD-trained networks consistently performed better on various datasets and on almost all the tested attacks even for substantially larger ϵ values (Figure 2, also see Figures A1,A3 in the appendix). These results show that compared to other baselines, AFD-trained networks are robust against a wider range of attacks and attack strengths (ϵ). This further

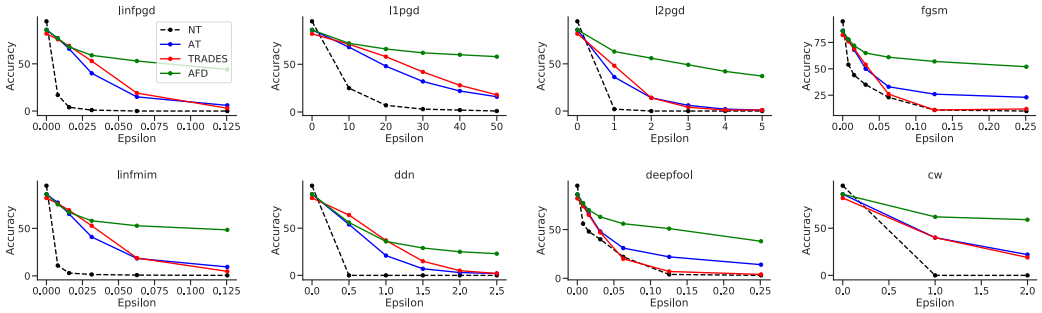


Figure 2: Adversarial accuracy for different methods against white-box attacks on CIFAR10 dataset with ResNet18 architecture.

Table 2: AUC measures for different perturbations and methods on MNIST, CIFAR10, CIFAR100, and tiny-imagenet datasets. AUC values are normalized to have a maximum allowable value of 1. Evaluations on AT and TRADES were made on networks trained using reimplemented or official code.

Dataset	Model	PGD_{L_∞}	PGD_{L2}	PGD_{L1}	FGSM	MIM	DDN	DeepFool	C&W
MNIST	NT	0.16	0.06	0.07	0.3	0.19	0.09	0.21	0.57
	AT	0.74	0.29	0.19	0.83	0.95	0.49	0.55	0.87
	TRADES	0.71	0.26	0.15	0.79	0.88	0.42	0.47	0.86
	AFD-DCGAN	0.77	0.33	0.3	0.78	0.91	0.51	0.49	0.9
	AFD-WGAN	**0.92**	**0.54**	**0.55**	**0.9**	**0.98**	**0.68**	**0.63**	**0.94**
CIFAR10	NT	0.05	0.1	0.17	0.19	0.05	0.1	0.16	0.1
	AT	0.28	0.2	0.44	0.33	0.31	0.26	0.29	0.31
	TRADES	0.32	0.22	0.5	0.24	0.32	0.33	0.18	0.28
	AFD-DCGAN	0.34	**0.54**	0.43	0.4	0.31	**0.4**	0.43	0.47
	AFD-WGAN	**0.56**	**0.54**	**0.66**	**0.59**	**0.56**	**0.4**	**0.52**	**0.62**
CIFAR100	NT	0.03	0.08	0.1	0.07	0.03	0.08	0.06	0.08
	AT	0.13	0.1	0.24	0.13	0.14	0.14	0.12	0.15
	TRADES	0.16	0.13	0.31	0.12	0.17	**0.18**	0.1	0.16
	AFD-DCGAN	0.14	0.12	0.27	0.17	0.16	0.15	0.16	0.18
	AFD-WGAN	**0.18**	**0.16**	**0.31**	**0.22**	**0.19**	0.16	**0.19**	**0.23**
Tiny-IN	NT	0.04	0.03	0.08	0.05	0.04	0.06	0.07	0.07
	AT	**0.10**	0.03	0.16	**0.15**	**0.09**	0.14	0.13	0.11
	TRADES	**0.10**	0.03	0.16	0.07	**0.09**	**0.15**	0.11	0.09
	AFD-WGAN	**0.10**	**0.04**	**0.19**	0.12	**0.09**	**0.15**	**0.16**	**0.12**

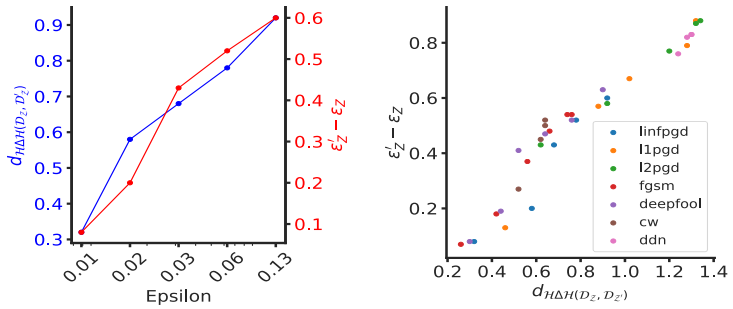
suggests that the features learned through AFD generalize better across various forms of attacks and can sustain larger perturbations.

4.4 Estimated $\mathcal{H}\Delta\mathcal{H}$-distance and adversarial-vs-natural error gap

As stated in Eq. 5, the upper bound on the adversarial error can be stated in terms of the natural error, the divergence between the two domains, and a constant term. In practice, this means that the smaller the divergence term $d_{\mathcal{H}\Delta\mathcal{H}}$ is, the smaller the gap between the adversarial and natural errors ($\epsilon'_Z - \epsilon_Z$) can be. We empirically tested this prediction using the domain discriminator trained on CIFAR10 dataset using the PGD-L_∞ attack. Figure-3a shows that the estimated $d_{\mathcal{H}\Delta\mathcal{H}}$ using the domain discriminator (i.e., using the corresponding empirical value of α in Eq. 6) trained on $PGD - L_\infty$ with $\epsilon = 0.031$ is closely related to the adversarial-vs-natural error gap over different ϵ values as predicted by Eq. 5. Moreover, estimations from the same domain discriminator also predicts the gap in generalization error attained for other forms of attack and ϵ values with high accuracy (Figure-3b). This further supports the proposal that minimizing the estimated distance between the natural and adversarial representations can be an efficient way to improve the model robustness against various adversarial attacks.

4.5 Learning a sparse representation

Because the AFD method aims to learn a representation that is insensitive to adversarial attacks, we expected the learned representational space to potentially be of lower dimensionality (i.e. less number of orthogonal features). To test this, we compared the dimensionality of the learned representation using two measures. i) number of non-zero features over the test set within each dataset and ii) number of Principal Compoenent Analysis (PCA) dimensions that explains more than 99% of the



(a) (b)

Figure 3: (a) Estimated $d_{\mathcal{H}\Delta\mathcal{H}}$ distance (based on empirical value of α in Eq. 6) and gap in adversarial and natural error $\epsilon'_Z - \epsilon_Z$ as a function of epsilon for $PGD - L_\infty$ attack; (b) scatter plot of the estimated $d_{\mathcal{H}\Delta\mathcal{H}}$ distance using the domain discriminator and the gap in adversarial and natural error across different attack types and magnitudes (i.e. ϵ). Colors correspond to different attack types. Each dot corresponds to one attack evaluated at a particular epsilon value. Estimations of the $d_{\mathcal{H}\Delta\mathcal{H}}$ distance for all attacks and epsilons are made with the domain discriminator trained on PGD-L_∞ with $\epsilon = 0.031$.

variance in the representation computed over the test-set of each dataset. We found that the same network architecture (i.e. ResNet18), when trained with AFD method learns a much sparser and lower dimensional representational space (Table A4) compared to the naturally trained, adversarial training and TRADES models. The representational spaces learned with AFD on MNIST, CIFAR10, and CIFAR100 datasets had only 6, 9, and 76 principal components respectively.

4.6 Adversarial and norm-based desensitization

To investigate whether the same level of robustness could be achieved by encouraging the network to produce similar representations in response to natural and adversarial inputs, we ran an additional experiment on the MNIST dataset in which we added a regularization term to the classification loss to directly minimize the representation sensitivity $S_e = \frac{1}{n}\sum_x \|F(x) - F(x')\|$, during training. We observed that although this augmented loss led to learning robustness representations, it achieved modest levels of robustness ($\sim 80\%$) and showed only weak generalization to stronger and other unseen attacks (Figure-A4). This result suggests that more direct forms of enforcing representational similarity may not lead to the same form of robustness with generalization properties similar to that achieved using an adversarial training with domain discriminator (e.g. as in AFD).

5 Conclusion and limitations

Decreasing the input-sensitivity of features has long been desired in training neural networks [15] and has been suggested as a way to improve adversarial robustness [44, 61]. In this work we proposed an algorithm to decrease the sensitivity of neural network representations using an adversarial learning paradigm that involves joint training of a domain discriminator, a feature extractor, and a task classifier. Essentially, our proposed algorithm iteratively estimates a bound on the adversarial error in terms of the natural error and a classification-based measure of distance between the distributions of natural and adversarial features and then minimizes the adversarial error by concurrently reducing the natural error as well as the distance between the two feature distributions.

Limitations. The empirical results presented here suggest that AFD-trained models are robust against a wide range of adversarial attacks (distributions) even compared to strong baselines like Adversarial Training and TRADES. However, it is not guaranteed that the model would remain robust any unseen attacks that we have not tested or may be invented in the future - as is the case in lack of theoretical guarantees in the domain adaptation literature. With regards to the computational cost, while AFD requires three SGD updates per batch, the additional computational cost is not significantly higher than many prior methods when considering that most of the computational cost is associated with generating the adversarial examples during training.

References

[1] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. *ICML*, 1:298–321, 2017.

[2] Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. *35th International Conference on Machine Learning, ICML 2018*, 1:436–448, 2018.

[3] Yang Bai, Yuyuan Zeng, Yong Jiang, Shu-Tao Xia, Xingjun Ma, and Yisen Wang. Improving Adversarial Robustness via Channel-wise Activation Suppressing. In *ICLR*, pages 1–19, 2021.

[4] Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. A theory of learning from different domains. *Machine Learning*, 79(1-2):151–175, 2010.

[5] Shai Ben-David, John Blitzer, Koby Crammer, and Fernando Pereira. Analysis of representations for domain adaptation. In *Advances in Neural Information Processing Systems*, pages 137–144, 2007.

[6] Nicholas Carlini and David Wagner. Towards Evaluating the Robustness of Neural Networks. *Proceedings - IEEE Symposium on Security and Privacy*, pages 39–57, 2017.

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

[8] Alvin Chan, Yi Tay, and Yew-Soon Ong. What it thinks is important is important: Robustness transfers through input gradients. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 332–341, 2020.

[9] Alvin Chan, Yi Tay, Yew Soon Ong, and Jie Fu. Jacobian Adversarially Regularized Networks for Robustness. *ICLR*, 2020.

[10] Jeremy Cohen, Elan Rosenfeld, and J. Zico Kolter. Certified adversarial robustness via randomized smoothing. *36th International Conference on Machine Learning, ICML 2019*, 2019-June:2323–2356, 2019.

[11] Francesco Croce and Matthias Hein. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. *arXiv preprint arXiv:2003.01690*, 2020.

[12] Gavin Weiguang Ding, Luyu Wang, and Xiaomeng Jin. AdverTorch v0.1: An adversarial robustness toolbox based on pytorch. *arXiv preprint arXiv:1902.07623*, 2019.

[13] Yinpeng Dong, Zhijie Deng, Tianyu Pang, Hang Su, and Jun Zhu. Adversarial Distributional Training for Robust Deep Learning. In *Neural Information Processing Systems (NIPS)*, number NeurIPS, pages 1–14, 2020.

[14] Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Hang Su, Jun Zhu, Xiaolin Hu, and Jianguo Li. Boosting Adversarial Attacks with Momentum. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 9185–9193, 2018.

[15] Harris Drucker and Yann Le Cun. Improving generalization performance using double back-propagation. *IEEE Transactions on Neural Networks*, 3(6):991–997, 1992.

[16] Sayna Ebrahimi, Franziska Meier, Roberto Calandra, Trevor Darrell, and Marcus Rohrbach. Adversarial continual learning. *arXiv preprint arXiv:2003.09553*, 2020.

[17] Logan Engstrom, Brandon Tran, Dimitris Tsipras, Ludwig Schmidt, and Aleksander Madry. Exploring the landscape of spatial robustness. *36th International Conference on Machine Learning, ICML 2019*, 2019-June:3218–3238, 2019.

[18] Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. *32nd International Conference on Machine Learning, ICML 2015*, 2(i):1180–1189, 2015.

[19] Ian Goodfellow, Dan Boneh, and Patrick Mcdaniel. Ensemble Adversarial Training: Attacks and Defenses. In *ICLR*, pages 1–20, 2018.

[20] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

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

[22] Sven Gowal, Chongli Qin, Jonathan Uesato, Timothy Mann, and Pushmeet Kohli. Uncovering the limits of adversarial training against norm-bounded adversarial examples. *arXiv*, (2017):1–30, 2020.

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

[24] Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. In *ICLR*, pages 1–16, 2019.

[25] Dan Hendrycks, Mantas Mazeika, Saurav Kadavath, and Dawn Song. Using pre-training can improve model robustness and uncertainty. *NeurIPS*, pages 4815–4826, 2019.

[26] Qiuyuan Huang, Paul Smolensky, Xiaodong He, Li Deng, and Dapeng Wu. Tensor product generation networks for deep NLP modeling. *arXiv*, pages 1263–1273, 2017.

[27] Harini Kannan, Alexey Kurakin, and Ian Goodfellow. Adversarial logit pairing. *arXiv preprint arXiv:1803.06373*, 2018.

[28] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4401–4410, 2019.

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

[30] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[31] Alex X Lee, Richard Zhang, Frederik Ebert, Pieter Abbeel, Chelsea Finn, and Sergey Levine. Stochastic adversarial video prediction. *arXiv preprint arXiv:1804.01523*, 2018.

[32] Hong Liu, Mingsheng Long, Jianmin Wang, and Michael Jordan. Transferable Adversarial Training: A General Approach to Adapting Deep Classifiers. *International Conference on Machine Learning*, pages 4013–4022, 2019.

[33] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[34] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

[35] Michael Mathieu, Camille Couprie, and Yann LeCun. Deep multi-scale video prediction beyond mean square error. *arXiv preprint arXiv:1511.05440*, 2015.

[36] Alexander Matyasko and Lap-Pui Chau. Improved network robustness with adversary critic. In *Advances in Neural Information Processing Systems*, pages 10578–10587, 2018.

[37] Takeru Miyato, Toshiki Kataoka, Koyama Masanori, and Yoshida Yuichi. Spectral normalization for generative adversarial networks. *ICLR*, 2018.

[38] Takeru Miyato and Masanori Koyama. cgans with projection discriminator. *arXiv preprint arXiv:1802.05637*, 2018.

[39] Takeru Miyato, Shin Ichi Maeda, Shin Ishii, and Masanori Koyama. Virtual Adversarial Training: A Regularization Method for Supervised and Semi-Supervised Learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–16, 2018.

[40] Seyed Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. DeepFool: A Simple and Accurate Method to Fool Deep Neural Networks. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016-Decem:2574–2582, 2016.

[41] Chongli Qin, James Martens, Sven Gowal, Dilip Krishnan, Krishnamurthy Dvijotham, Alhussein Fawzi, Soham De, Robert Stanforth, and Pushmeet Kohli. Adversarial Robustness Through Local Linearization. (NeurIPS):1–10, 2020.

[42] Jonas Rauber, Wieland Brendel, and Matthias Bethge. Foolbox: A python toolbox to benchmark the robustness of machine learning models. In *Reliable Machine Learning in the Wild Workshop, 34th International Conference on Machine Learning*, 2017.

[43] Jerome Rony, Luiz G. Hafemann, Luiz S. Oliveira, Ismail Ben Ayed, Robert Sabourin, and Eric Granger. Decoupling direction and norm for efficient gradient-based l2 adversarial attacks and defenses. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2019-June:4317–4325, 2019.

[44] Andrew Slavin Ros and Finale Doshi-Velez. Improving the adversarial robustness and interpretability of deep neural networks by regularizing their input gradients. *32nd AAAI Conference on Artificial Intelligence, AAAI 2018*, pages 1660–1669, 2018.

[45] Evgenia Rusak, Lukas Schott, Roland S. Zimmermann, Julian Bitterwolf, Oliver Bringmann, Matthias Bethge, and Wieland Brendel. Increasing the robustness of DNNs against image corruptions by playing the Game of Noise. 2020.

[46] Lukas Schott, Jonas Rauber, Matthias Bethge, and Wieland Brendel. Towards the first adversarially robust neural network model on mnist. *arXiv preprint arXiv:1805.09190*, 2018.

[47] Samrath Sinha, Sayna Ebrahimi, and Trevor Darrell. Variational adversarial active learning. *Proceedings of the IEEE International Conference on Computer Vision*, 2019-Octob:5971–5980, 2019.

[48] Chawin Sitawarin, Supriyo Chakraborty, and David Wagner. Improving adversarial robustness through progressive hardening. *arXiv preprint arXiv:2003.09347*, 2020.

[49] Chuanbiao Song, Kun He, Liwei Wang, and John E Hopcroft. Improving the generalization of adversarial training with domain adaptation. In *ICLR*, pages 1–14, 2019.

[50] Chuanbiao Song, He Kun, Lin Jiadong, John E Hopcroft, and Liwei Wang. Robust local features for improving the generalization of adversarial training. In *ICLR*, pages 1–12, 2020.

[51] Gaurang Sriramanan, Sravanti Addepalli, Arya Baburaj, and R. Venkatesh Babu. Guided Adversarial Attack for Evaluating and Enhancing Adversarial Defenses. In *Neural Information Processing Systems (NIPS)*, number NeurIPS, 2020.

[52] Christian Szegedy, Joan Bruna, Dumitru Erhan, Ian Goodfellow, Joan Bruna, Rob Fergus, and Dumitru Erhan. Intriguing properties of neural networks. pages 1–10, 2013.

[53] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. Adversarial discriminative domain adaptation. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, 2017-January:2962–2971, 2017.

[54] Huaxia Wang and Chun-Nam Yu. A direct approach to robust deep learning using adversarial networks. *arXiv preprint arXiv:1905.09591*, 2019.

[55] Dongxian Wu, Yisen Wang, and Xia Shu-Tao. Revisiting Loss Landscape for Adversarial Robustness. *ICML*, 2019.

[56] Runtian Zhai, Chen Dan, Di He, Huan Zhang, Boqing Gong, Pradeep Ravikumar, Cho-Jui Hsieh, and Liwei Wang. Macer: Attack-free and scalable robust training via maximizing certified radius. *arXiv preprint arXiv:2001.02378*, 2020.

[57] Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric P Xing, Laurent El Ghaoui, and Michael I Jordan. Theoretically principled trade-off between robustness and accuracy. *arXiv preprint arXiv:1901.08573*, 2019.

[58] Huan Zhang, Hongge Chen, Chaowei Xiao, Sven Gowal, Robert Stanforth, Bo Li, Duane Boning, and Cho-Jui Hsieh. Towards Stable and Efficient Training of Verifiably Robust Neural Networks. pages 1–25, 2019.

[59] Jingfeng Zhang, Jianing Zhu, Gang Niu, Bo Han, Masashi Sugiyama, and Mohan Kankanhalli. Geometry-aware instance-reweighted adversarial training. In *ICLR*, pages 1–28, 2021.

[60] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232, 2017.

[61] Sicheng Zhu, Xiao Zhang, and David Evans. Learning adversarially robust representations via worst-case mutual information maximization. *ICML*, 2020.

Checklist

The checklist follows the references. Please read the checklist guidelines carefully for information on how to answer these questions. For each question, change the default **[TODO]** to [Yes] , [No] , or [N/A] . You are strongly encouraged to include a **justification to your answer**, either by referencing the appropriate section of your paper or providing a brief inline description. For example:

- Did you include the license to the code and datasets? [Yes] See Section **??**.
- Did you include the license to the code and datasets? [No] The code and the data are proprietary.
- Did you include the license to the code and datasets? [N/A]

Please do not modify the questions and only use the provided macros for your answers. Note that the Checklist section does not count towards the page limit. In your paper, please delete this instructions block and only keep the Checklist section heading above along with the questions/answers below.

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] We provide theoretical motivation for our method in section 3. We then implement and test our approach in section 4.
   (b) Did you describe the limitations of your work? [Yes] We describe the limitations of our work in section 5.
   (c) Did you discuss any potential negative societal impacts of your work? [No] Our work is concerned with improving robustness against adversarial attacks. As such, we actually may mitigate some vulnerabilities of machine learning models that could negatively affect society. Arguably, the most likely negative outcome is that providing new defenses may encourage researching stronger attacks.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] The full set of assumptions are stated in the beginning of Section 3.
   (b) Did you include complete proofs of all theoretical results? [No] We do not present any new theories in this work.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] The details of the method are fully described in the paper. The code along with model checkpoints are also included in the supplementary material.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] This information is provided in the Experiments section.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No] We have not reported error bars for experiments on different datasets. The current results include many individual experiments on multiple datasets that were not repeated. We are going to include error bars in all reported results in the next revision.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] The type and number of GPUs used to conduct the experiments were reported in section 4.1.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] This information is provided in the Experiments section.
   (b) Did you mention the license of the assets? [Yes] In the supplmental material.
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] Yes, code is provided in the supplemental material.

(d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [No] We did not use any assets that require consent from the creators.

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [No] We only used public benchmarks in our work that do not contain any personally identifiable information or offensive content.

5. If you used crowdsourcing or conducted research with human subjects...

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]