MITSUBISHI ELECTRIC RESEARCH LABORATORIES
http://www.merl.com

White-Box Adversarial Defense via Self-Supervised Data Estimation

Zhang, Z.; Lin, Z.; Pfister, H.

TR2019-104 September 19, 2019

Abstract

In this paper, we study the problem of how to defend classifiers against adversarial attacks that fool the classifiers using subtly modified input data. In contrast to previous works, here we focus on the white-box adversarial defense where the attackers are granted full access to not only the classifiers but also defenders to produce as strong attack as possible. In such a context we propose viewing a defender as a functional, a high-order function that takes functions as its argument to represent a function space, rather than fixed functions conventionally. From this perspective, a defender should be realized and optimized individually for each adversarial input. To this end, we propose RIDE, an efficient and provably convergent self-supervised learning algorithm for individual data estimation to rescue the predictions from adversarial attacks. We demonstrate the significant improvement of adversarial defense performance on image recognition, e.g. 98%, 76%, 43% test accuracy on MNIST, CIFAR-10, and ImageNet datasets respectively under the state-of-the-art BPDA attacker.

arXiv

White-Box Adversarial Defense via Self-Supervised Data Estimation

Zudi Lin[1*], **Hanspeter Pfister**[1] **and Ziming Zhang**[2†]
[1]Harvard University, [2]Mitsubishi Electric Research Laboratories (MERL)
{linzudi,pfister}@g.harvard.edu,zzhang@merl.com

Abstract

In this paper, we study the problem of how to defend classifiers against adversarial attacks that fool the classifiers using subtly modified input data. In contrast to previous works, here we focus on the *white-box* adversarial defense where the attackers are granted full access to not only the classifiers but also defenders to produce as strong attacks as possible. In such a context we propose viewing a defender as a *functional*, a higher-order function that takes functions as its argument to represent a *function space*, rather than fixed functions conventionally. From this perspective, a defender should be realized and optimized *individually* for each adversarial input. To this end, we propose RIDE, an efficient and provably convergent self-supervised learning algorithm for individual data estimation to protect the predictions from adversarial attacks. We demonstrate the significant improvement of adversarial defense performance on image recognition, *e.g.* , 98%, 76%, 43% test accuracy on MNIST, CIFAR-10, and ImageNet datasets respectively under the state-of-the-art BPDA attacker.

Introduction

Recent studies have demonstrated that as classifiers, deep neural networks (*e.g.* , CNNs) are quite vulnerable to adversarial attacks that only add quasi-imperceptible perturbations to the input data but completely change the predictions of the classifiers (*e.g.* , [Szegedy et al., 2013; Goodfellow, Shlens, and Szegedy, 2014; Madry et al., 2017; Carlini and Wagner, 2017; Athalye, Carlini, and Wagner, 2018]).

In general adversarial attackers can be categorized into two groups based on the accessibility to the classifiers. *White-box* attackers (*e.g.* , [Szegedy et al., 2013; Goodfellow, Shlens, and Szegedy, 2014; Carlini and Wagner, 2017]) have full knowledge of the classifiers such as network architectures as well as the trained weights. In this case, those attackers can directly calculate the perturbations that change the prediction using gradient ascent. *Black-box* attackers (*e.g.* , [Ilyas et al., 2018; Su, Vargas, and Sakurai, 2019]) have no right to access the classifiers, but they can observe the classification predictions given different inputs to estimate the perturbations. Black-box attackers are easier to apply on a classifier under privacy constraints but generally lead to lower success rate.

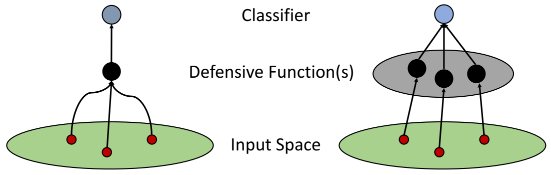
Adversarial training (*e.g.* , [Goodfellow, Shlens, and Szegedy, 2014; Madry et al., 2017]) was introduced to defend against such attacks by augmenting the training data with adversarial samples. However, defense using adversarial training is resource-intensive since it requires large labeled datasets and retraining of the classifier.

To counter such attacks without changing existing classifiers, in this paper, we focus on adversarial *defenders*, which work as pre-processing modules to estimate the unattacked clean data to recover the prediction. The defenders can be categorized into either *gray-box* (or *oblivious* [Carlini and Wagner, 2017]) defenders, where only the classifier but not the defender is accessible by the attackers, or *white-box* defenders, where the defender is also exposed to the attackers. State-of-the-art gray-box defenders use transformation models trained with labeled adversarial-clean image pairs (*e.g.* ,[Liao et al., 2018; Akhtar, Liu, and Mian, 2018]) to remove the perturbations. However, since such modules are differentiable, the attackers can directly regard the defender as part of the classifier and conduct gradient ascent under the white-box setting. Therefore previous attempts on white-box defenders mainly focus on the using of functions with *obfuscated gradients*, where either the functions are non-differentiable [Guo et al., 2018], or the gradients are hard to compute due to very deep computation [Samangouei, Kabkab, and Chellappa, 2018]. However, those defenders that claim white-box robustness are recently broken by the Backward Pass Differentiable Approximation (BPDA) technique [Athalye, Carlini, and Wagner, 2018], which approximates the derivatives by running the defender at the classifier forward pass and backpropagate the gradients using a differentiable approximation.

Since the invisibility assumption in gray-box defenders significantly increases the failure risk in practice, we therefore mainly focus on the *white-box* setting. That is, even the defender is exposed to the attackers, it can still prevent the attacker from not only direct derivative calculation but also gradient approximation. Besides, we hope such a defender does not require training and directly work at inference time. To this end, we propose viewing the defense from the perspective of a *functional*, a higher-order function that takes an individual adversarial sample as input and returns a new defender function exclusive for this sample (Fig.1). Such a design makes the defender *input-dependent*, whose gradients are hard to be estimated without knowing the data prior (*e.g.*

[*] Work was done during an internship at MERL.

[†] Corresponding author. Code is publicly available at www.merl.com/research/license#RIDE.



(a) Conventional Defenders (b) Proposed Defender

Figure 1: Illustration of the proposed *functional* defender. (**a**) A conventional defender usually performs the defense with a fixed function. (**b**) In a *functional* defender, every defensive function itself is a function of the input. Such a design significantly increases the difficulty of an adversarial attacker from gradient calculation/estimation.

, the prior distribution of natural images). Specifically, the functional is implemented to yield a parameterized neural network defender for each input, which is optimized by minimizing a *self-supervised* reconstruction loss using deep image prior [Ulyanov, Vedaldi, and Lempitsky, 2018] or denoising autoencoder [Vincent et al., 2008]. To further improve the robustness, we propose a novel Robust Iterative Data Estimation (RIDE) algorithm that accelerates the convergence of the network output to the underlying clean data, which shares similar high-level concepts with momentum [Qian, 1999].

To summarize, our main contributions includes: (1) we are the first, to the best of our knowledge, to propose viewing a defender as a *functional* that returns an input-dependent function for every adversarial observation; (2) we propose a novel iterative self-supervised optimization algorithm, called RIDE, as an effective realization of such a defender; (3) we demonstrate the significantly increased robustness of RIDE against state-of-the-art BPDA white-box attackers, which achieves 98%, 76%, 43% Top-1 test accuracy on MNIST, CIFAR-10, and ImageNet datasets respectively.

Related Work

Adversarial Attack. The vulnerability of deep learning models to small perturbations was first demonstrated by Szegedy et al. [2013]. Later, Goodfellow, Shlens, and Szegedy [2014] proposed an attacker of Fast Gradient Sign Method (FGSM) that utilizes the gradient sign to generate adversarial perturbations with only one-step update on the original image to fool classifiers. To improve this method, several iterative algorithms such as Basic Iterative Method (BIM) [Kurakin, Goodfellow, and Bengio, 2016] and the Projected Gradient Descent (PGD) attacker [Madry et al., 2017] were proposed as well. Moosavi-Dezfooli, Fawzi, and Frossard [2016] proposed DeepFool to find the minimal perturbation that changes the prediction. Athalye et al. [2017] proposed Expectation over Transformation (EOT) to attack against a random distribution of transformations. Moosavi-Dezfooli et al. [2017] showed the existence of universal adversarial perturbations that can fool a classifier with a single perturbation image. Such works above follow white-box attack, in contrast to black-box attackers (*e.g.* , [Ilyas et al., 2018; Su, Vargas, and Sakurai, 2019]). We refer readers to a nice survey by Akhtar and Mian [2018] for more details in this direction.

In particular, Athalye, Carlini, and Wagner [2018] proposed Backward Pass Differentiable Approximation (BPDA) based on the fact that a defender g always tries to recover the clean sample $\mathbf{x}$ from an adversarial sample $\tilde{\mathbf{x}}$. Therefore, one can approximate the gradient of a defender *w.r.t.* the input using an approximation with an identity matrix.

Adversarial Defense. Besides adversarial training (*e.g.* , [Goodfellow, Shlens, and Szegedy, 2014; Madry et al., 2017; Kannan, Kurakin, and Goodfellow, 2018; Sinha et al., 2019]), Xie et al. [2019] recently introduced non-local means into the classifiers as feature denoising blocks. More evaluations of the robustness of classifiers are conducted by Hendrycks and Dietterich [2019]. For defenders that do not require changing of the classifier, Guo et al. [2018] showed the defense using input transformations such as bit-depth reduction, total variance minimization (TVM), and image quilting while Prakash et al. [2018] introduced pixel deflection. CNN-based denoisers (*e.g.* , [Liao et al., 2018]) or rectifiers (*e.g.* , [Akhtar, Liu, and Mian, 2018]) can be trained with clean-adversarial image pairs to remove adversarial noise. Samangouei, Kabkab, and Chellappa [2018] proposed Defense-GAN that uses a generator trained to model the distribution of clean images, and projects the adversarial sample into the space of the generator before classifying them. However, successful training of a GAN on datasets like ImageNet is still extremely challenging.

Different from previous literature, we propose a *functional*-based defender that returns an *unique* defender function for every individual input sample, which is the key feature that makes our method much more robust to white-box attacks.

Self-Supervised Learning. Self-supervised learning is a form of unsupervised learning where the data itself provides the supervision [Zisserman, 2018]. Representative works in this direction include numerous variants of autoencoders [Tschannen, Bachem, and Lucic, 2018]. Recently, Ulyanov, Vedaldi, and Lempitsky [2018] proposed *deep image prior* (DIP) demonstrating that a neural network can be used as a handcrafted prior with excellent results in inverse problems such as denoising, super-resolution, and inpainting. Shocher, Cohen, and Irani [2018] proposed *deep internal learning* (DIL) with similar observations. In this paper, we explore the applicability of those self-supervision algorithms as the implementation of the functional-based defenders.

Preliminaries

Generally adversarial attacks can be expressed as the following optimization problem (*e.g.* ,[Simon-Gabriel et al., 2019]):

$$\max_{\|\Delta\mathbf{x}\|\leq\epsilon} \ell\left(f(\mathbf{x}+\Delta\mathbf{x}), f(\mathbf{x})\right), \tag{1}$$

where $\Delta\mathbf{x}$ denotes the perturbation for input $\mathbf{x}$, f denotes the classifier, $\epsilon \geq 0$ denotes a predefined (small) constant, ℓ denotes a proper loss function, and $\|\cdot\|$ denotes a norm operator. As we see, the attacker is essentially seeking for the adversarial sample $\mathbf{x}+\Delta\mathbf{x}$ that is *locally* around $\mathbf{x}$ but can change the prediction of the classifier as much as possible.

As discussed before, white-box attackers have full knowledge of the classifiers and can utilize the derivatives of the classifiers *w.r.t.* the inputs, *i.e.*, $\frac{\partial f}{\partial \mathbf{x}}$ in Eq. 1, to determine the perturbation $\Delta\mathbf{x}$ using back propagation. On the contrary, learning a defender can be formulated as a minimization

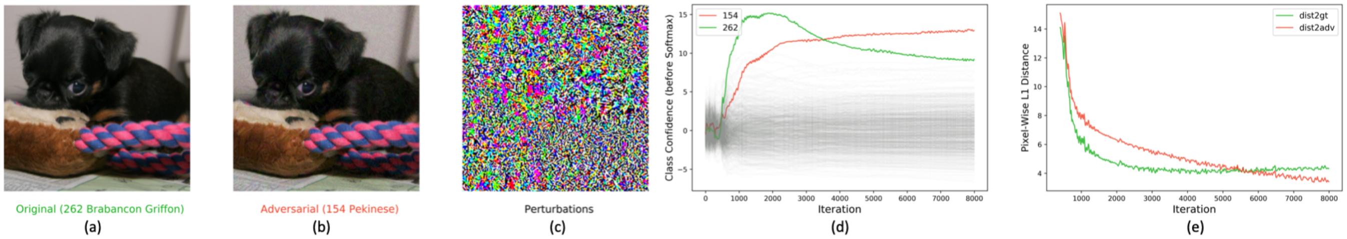


Figure 2: Characteristics of the reconstructed image using DIP when minimizing the reconstruction loss to the *adversarial image*. From left to right: **(a)** clean image, **(b)** adversarial image, **(c)** (rescaled) adversarial perturbations, **(d)** class confidence scores (before softmax) over iterations, and **(e)** Manhattan distances of the reconstructed image to either clean (dist2gt) or adversarial image (dist2adv) over iterations. In (d), the green and red curves are for ground-truth and adversarial class labels, respectively, and the light gray curves are for the other classes.

optimization problem as follows:

$$\min_{g \in \mathcal{G}} \ell\left(f(g(\mathbf{x}+\Delta\mathbf{x})), f(\mathbf{x})\right), \tag{2}$$

where $g \in \mathcal{G}$ denotes a defender function that tries to recover the original sample to minimize the prediction loss. In the literature such defenders are often assumed to be *invisible* to the attackers, which leads to *gray-box* defenders. clearly, if a fixed differentiable g was known, an attacker could take $\tilde{f} = f \circ g$ as a new classifier to generate adversarial samples using $\frac{\partial \tilde{f}}{\partial \mathbf{x}}$. For g with *obfuscated gradients*, $\frac{\partial g}{\partial \mathbf{x}} \approx \mathbf{I}$ is shown to be an effective approximation by the BPDA attacker.

Adversarial Defenders as Functionals. As we discussed above, any fixed-function defender can hardly survive white-box attacks. This motivates us to rethink the solution from the perspective of *funtional*, a higher-order function that takes functions as its argument to represent a function space (*i.e.* , a space made of functions where each function can be thought of as a point). Conceptually, a *functional* defender is equivalent to an ensemble of different conventional function-based defenders. Now consider a functional $g(\mathbf{x}; \omega(\mathbf{x}))$ as the defender, whose input $\omega : \mathcal{X} \to \mathcal{W}$ is a (hidden) function over the a single input $\mathbf{x} \in \mathcal{X}$. In order to attack defender g, the derivative $\frac{\partial g}{\partial \mathbf{x}}$ can be written as follows:

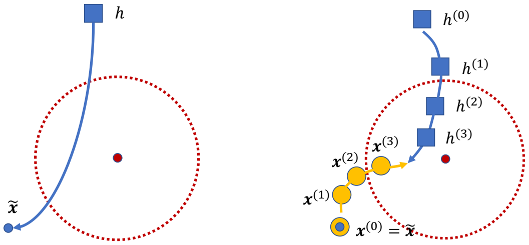
$$\frac{\partial g}{\partial \mathbf{x}} = \left[\mathbf{I}, \nabla\omega(\mathbf{x})\right] \nabla g(\mathbf{x}; \omega(\mathbf{x})), \tag{3}$$

where ∇ denotes the gradient operator, $[\cdot, \cdot]$ denotes the matrix concatenation operator, and $\mathbf{I}$ denotes the identity matrix. With full access to f and g, one may obtain ∇g. However, the calculation of $\nabla\omega$ for hidden function ω without any prior knowledge at all will be challenging for the attack, in return leading to a significant increase in the success rate of defense.

Self-Supervision. To realize the *functional* defenders, we refer to recent works in self-supervised learning that works on each individual sample without seeing external data. Specifically, DIP manages to estimate the true image by fitting a corrupted image $\tilde{\mathbf{x}}$, using a convolutional neural network (CNN) h, with randomly initialized weights $\mathbf{w}$, and an input consisting of random noise $\mathbf{z}$. Mathematically DIP tries to solve the following optimization problem:

$$\min_{\mathbf{w} \in \mathcal{W}} \ell_h\left(\tilde{\mathbf{x}}, h(\mathbf{z}; \mathbf{w}, \theta)\right), \tag{4}$$

where ℓ_h denotes the loss function for optimizing the network, and θ denotes the network hyper-parameters during learning.



(a) DIP (b) Proposed RIDE

Figure 3: Hypothesis of learning trajectories (*i.e.* the arrow curves) in the input space for **(a)** DIP and **(b)** our proposed RIDE algorithm. Here the red dot denotes the unknown clean input, while the red dotted circle denotes the region where all the samples share the same prediction. RIDE iteratively changes the estimation by convex interpolation to prevent overfitting and improve prediction recovery.

In the end, the functional h returns a defender parameterized by $\mathbf{w} = \omega(\tilde{\mathbf{x}})$, which becomes an *input-dependent hidden* function. To the best of our knowledge, however, they have never been explored for adversarial defense. To verify their potential usage, given an adversarial image, we feed the intermediate reconstructed outputs from DIP into a pretrained ResNet50 model [He et al., 2016] to see whether the prediction can be correctly recovered (Fig. 2). As we see in Fig. 2(d), the output indeed manages to recover the original prediction of the classifier before starting to overfit the adversarial image. Meanwhile, we observe in Fig. 2(e) that when the prediction is recovered, the distances of the output to the clean image (*dist2gt*) are consistently smaller than the distances to the adversarial image (*dist2adv*). This indicates that during optimization, the reconstructed image approaches the original image first, and then gradually diverges towards the adversarial image, as illustrated in Fig. 3(a). Therefore, how to effectively avoid such an overfitting problem in self-supervised learning becomes the key challenge of the defense. Later on we show two regularization techniques that significantly improve the performance of DIP as a defender.

White-Box Defense with Self-Supervision

Problem Setup. Let $(\mathbf{x}, y) \in \mathcal{X} \times \mathcal{Y}$ be a clean data pair with a label y, and $\tilde{\mathbf{x}} = \mathbf{x} + \Delta\mathbf{x} \in \tilde{\mathcal{X}}$ be an adversarial input. In the adversarial defense, we observe $(\tilde{\mathbf{x}}, y)$ only, but not $\mathbf{x}$.

Now given a pretrained classifier $f : \mathcal{X} \to \mathcal{Y}$ and a proper loss function $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$, we aim to learn our defender

$g : \tilde{\mathcal{X}} \times \mathcal{W} \rightarrow \mathcal{X}$ by optimizing the following problem:

$$\min_{\omega\in\Omega} \mathbb{E}_{(\tilde{\mathbf{x}},y)\in\tilde{\mathcal{X}}\times\mathcal{Y}}\Big[\ell\left(f(g(\tilde{\mathbf{x}},\omega(\tilde{\mathbf{x}}))),y\right)\Big], \tag{5}$$

where $\mathbb{E}$ denotes the expectation operator, and $\omega : \tilde{\mathcal{X}} \rightarrow \mathcal{W}$ denotes another (hidden) function. As discussed before, two fundamental differences between our method and the literature such as denoisers [Liao et al., 2018] are: (1) no clean data is needed to learn our defender, and (2) our defender adaptively changes given each adversarial input. Such discrimination is derived from the nature of a functional.

Hyper-Parameter Learning as Realization. From our analysis for Eq. 3, in order to design an effective white-box defender, function ω has to be a hidden function, namely, its explicit formula is unknown (otherwise, in the white-box defense the attacker can compute the gradient in Eq. 3 as well). To this end, inspired by DIP we propose a novel self-supervised data estimation algorithm to embed the realization of ω and g into the optimization of a neural network $h\left(\mathbf{z};\mathbf{w},\theta\right)$ by defining $\omega(\tilde{\mathbf{x}}) = \mathbf{w}$ that outputs different network parameters for every individual data point.

With the help of $h\left(\mathbf{z};\mathbf{w},\theta\right)$ we can further rewrite Eq. 5 as a network hyper-parameter learning problem as follows:

$$\min_{\theta\in\Theta} \mathbb{E}_{(\tilde{\mathbf{x}},y)\in\tilde{\mathcal{X}}\times\mathcal{Y}}\Big[\ell\left(f(\mathbf{x}_{est}),y\right)\Big], \tag{6}$$

$$\text{s.t.}\, \mathbf{u},\mathbf{w}\in\arg\min_{\mathbf{u}',\mathbf{w}'}\mathcal{L}(\tilde{\mathbf{x}},\mathbf{u}',\mathbf{w}',\theta), \mathbf{x}_{est}=\mathbb{E}_{\mathbf{z}\sim\mathcal{Z}}[h(\mathbf{z};\mathbf{w},\theta)],$$

where $\mathcal{L}$ denotes a reconstruction loss, $\mathcal{Z}$ denotes a data distribution, and $\mathbf{u}$ denotes an auxiliary variable that will be discussed in RIDE. Here we consider the clean data estimator $\mathbf{x}_{est}$ as the mean of the network outputs over the entire distribution given the learned parameters, and define the output of our defender as $g(\tilde{\mathbf{x}},\omega(\tilde{\mathbf{x}})) \equiv \mathbf{x}_{est}$.

Efficient Solver by Converting Learning to Tuning. Eq. 6 essentially defines a bilevel optimization problem [Colson, Marcotte, and Savard, 2007], and how to solve it effectively and efficiently, in general, is nontrivial. However, since the network hyper-parameters θ (*e.g.* , number of iterations) often do not require continuity, to minimize ℓ we can

1. Predefine Θ by *discretizing* the hyper-parameter space;
2. Evaluate each $\theta \in \Theta$ using the objective with learned $\mathbf{w}$'s;
3. Choose the hyper-parameters with the minimum objective.

This methodology simplifies the hyper-parameter learning problem in Eq. 6 to a tuning problem. For instance, empirically, we tune the number of iterations as our stop criterion. Later on, we will discuss how to use self-supervision to design the lower-level objective $\mathcal{L}$ in Eq. 6.

Self-Supervised Data Estimation

Denosing Autoencoder for Robustness. In contrast to DIP that takes a fixed random noise image as input and the adversarial image as the target, we also propose employing the denoising autoencoder (DAE) [Vincent et al., 2008] to improve the stability in data recovery using DIP. Different from the original goal of DAE which is to learn a low-dimensional encoding of the statistics of a dataset, here we apply DAE on every single input to model its *internal statistics* similar to DIL [Shocher, Cohen, and Irani, 2018]. Considering further improvement on robustness over conventional DAE (see supplementary material for comparison), in our implementation we also add random Gaussian noise to the target, leading to the following lower-level objective:

$$\begin{aligned}&\mathcal{L}_{DAE}\left(\tilde{\mathbf{x}},\emptyset,\mathbf{w},\theta\right)\\&\quad=\mathbb{E}_{(\mathbf{s},\mathbf{z})\sim\mathcal{N}(\tilde{\mathbf{x}},\boldsymbol{\sigma})\times\mathcal{N}(\tilde{\mathbf{x}},\boldsymbol{\sigma})}\Big[\ell_h\left(\mathbf{s},h(\mathbf{z};\mathbf{w},\theta)\right)\Big],\end{aligned} \tag{7}$$

where $\mathbf{s}$ and $\mathbf{z}$ denote the reconstruction target and the network input, respectively, and each $\mathcal{N}$ denotes a Gaussian distribution with mean $\tilde{\mathbf{x}}$ and std $\boldsymbol{\sigma}$. To solve Eq. 7, we use mini-batch stochastic gradient descent (SGD) by randomly sampling the input-output pairs. We list our single-input DAE algorithm in Alg. 1 for reference.

Algorithm 1 Denosing Autoencoder (DAE)

```
Input  : adversarial sample $\tilde{\mathbf{x}}$, initial weights $\mathbf{w}$, network $h$, loss
         $\ell_h$, Gaussian std $\boldsymbol{\sigma}$, number of iterations $K$, mini-batch
         size $n$, learning rate $\eta$
Output : estimated true data $\mathbf{x}_{est}$
for $k = 1, 2, \cdots, K$ do
    Randomly draw $n$ samples $(\mathbf{s}, \mathbf{z}) \sim \mathcal{N}(\tilde{\mathbf{x}}, \boldsymbol{\sigma}) \times \mathcal{N}(\tilde{\mathbf{x}}, \boldsymbol{\sigma})$;
    $\mathbf{w} \leftarrow \mathbf{w} - \eta \cdot \frac{\partial \mathcal{L}_{DAE}}{\partial \mathbf{w}}$ ;                        // Eq. 7
end
$\mathbf{x}_{est} = \mathbb{E}_{\mathbf{z}\sim\mathcal{N}(\tilde{\mathbf{x}},\boldsymbol{\sigma})}\left[h(\mathbf{z};\mathbf{w},\theta)\right]$ ;              // mini-batch
return $\mathbf{x}_{est}$;
```

Algorithm 2 Robust Iterative Data Estimation (RIDE)

```
Input  : adversarial sample $\tilde{\mathbf{x}}$, network $h$, least-square loss $\ell_h$,
         Gaussian std $\boldsymbol{\sigma} = 0.5$, numbers of iterations and check-
         points $K = 2000, T = 5$, mini-batch size $n = 1$, learning
         rate $\eta = 10^{-4}$, interpolation parameter $\alpha = 0.9$
Output : estimated true data $\mathbf{x}_{est}$
Initialize the network weights $\mathbf{w}^{(0)}$, and $\mathbf{x}^{(0)} \leftarrow \tilde{\mathbf{x}}$;
for $t = 1, 2, \cdots, T$ do
    $\mathbf{x}_{est}^{(t)} \leftarrow$DAE$\left(\mathbf{x}^{(t-1)}, \mathbf{w}^{(t-1)}, h, \ell_h, \boldsymbol{\sigma}, K, n, \eta\right)$;
    $\mathbf{x}^{(t)} \leftarrow \alpha\mathbf{x}^{(t-1)} + (1-\alpha)\mathbf{x}_{est}^{(t)}$;
end
return $\mathbf{x}_{est}^{(T)}$;
```

Robust Iterative Data Estimation (RIDE). DAE manages to improve the robustness in data estimation, but it converges slowly, as illustrated in Fig. 4(b), which may have a negative impact on the defense within a limited number of iterations. To remedy this, we propose RIDE, a novel integration of DAE with iterative optimization that gradually estimates the true data (the auxiliary variable $\mathbf{u} = \mathbf{x}$ in Eq. 6) as the target for fitting the network. As illustrated in Fig. 3(b), the lower-level objective, $\mathcal{L}_{RIDE}$, can be formulated as follows:

$$\begin{aligned}&\mathcal{L}_{RIDE}\left(\mathbf{x}^{(t-1)},\mathbf{x}^{(t)},\mathbf{w}^{(t)},\theta\right)\\&\quad=\mathcal{L}_{DAE}\left(\mathbf{x}^{(t)},\emptyset,\mathbf{w}^{(t)},\theta\right)+\lambda d\left(\mathbf{x}^{(t)},\mathbf{x}^{(t-1)}\right),\forall t,\end{aligned} \tag{8}$$

where $\{\mathbf{x}^{(t)}\}$ denotes a sequence of the true data estimations with $\mathbf{x}^{(0)} = \tilde{\mathbf{x}}$, d denotes a distance measure as our regularizer, and $\lambda \geq 0$ is a predefined constant. Comparing to Eq. 7,

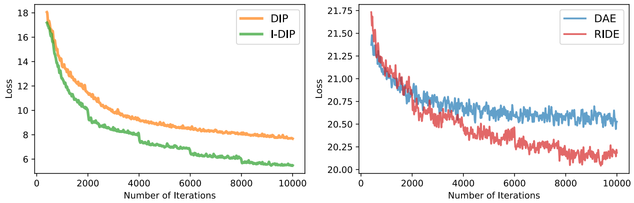


Figure 4: Illustration of convergence improvement of the proposed iterative optimization mechanism on (**left**) DIP and (**right**) DAE.

DAE is a special case of RIDE with $\mathbf{x}^{(t)} = \tilde{\mathbf{x}}, \forall t$. In particular, if both ℓ_h and d in Eq. 8 are mean squared error, we then have a close-form solution for updating $\mathbf{x}^{(t)}$ as follows:

$$\mathbf{x}^{(t)} = \alpha \mathbf{x}^{(t-1)} + (1-\alpha)\mathbf{x}_{est}^{(t)}, \forall t, \tag{9}$$

where $\alpha = \frac{\lambda}{1+\lambda}$ and $\mathbf{x}_{est}^{(t)} = \mathbb{E}_{\mathbf{z}\sim\mathcal{N}(\mathbf{x}^{(t-1)},\boldsymbol{\sigma})}[h(\mathbf{z};\mathbf{w}^{(t)},\theta)]$. Such update rule appears to be widely used in deep learning optimizers as momentum (*e.g.* , Adam [Kingma and Ba, 2014]) that helps accelerate gradients vectors in the right directions, thus leading to faster converging. Along with DAE, we list our RIDE algorithm in Alg.2, where all the numbers denote the default hyper-parameter values in the defender[1].

Note that such an iterative mechanism can also be integrated into DIP, leading to the *Iterative DIP* (I-DIP) algorithm. We illustrate the effects of our iterative mechanism in Fig. 4 that clearly shows the convergence acceleration for both DIP and DAE in reconstruction.

Theorem 1 (Convergence of RIDE)**.** *Suppose that* $\forall t$,

$$\mathcal{L}_{DAE}\left(\mathbf{x}^{(t-1)}, \emptyset, \mathbf{w}^{(t)}, \theta\right) \le \mathcal{L}_{DAE}\left(\mathbf{x}^{(t-1)}, \emptyset, \mathbf{w}^{(t-1)}, \theta\right)$$

holds with sufficiently large number of iterations. Then RIDE *in Alg. 2 is guaranteed to converge locally with* $t \to \infty$.

Please refer to the supplementary material for the proof.

Ablation Study on RIDE

Implementation of RIDE. By default we use an 8-layer fully convolutional network (FCN) similar to the one used in DIL [Shocher, Cohen, and Irani, 2018]. Except for the input and output layers, all the other layers use 32 filters with size 3×3. If using dropout, dropout layers with a ratio of 0.5 are added at the 4, 5 and 6-th conv layers (check details in the supplementary material). No batch normalization layers are used. We initialize the model weights with random uniform distribution. The loss function is the mean squared error for DIP, DAE and RIDE. The model is optimized by the Adam optimizer [Kingma and Ba, 2014] with a learning rate 10^{-4} and weight decay 10^{-4}. We fix the number of iterations $K = 2,000$ and check-points $T = 5$ for RIDE. These parameters are fixed for all the following experiments in this paper without further fine-tuning.

Dataset. We conduct experiments on the ImageNet [Deng et al., 2009] classification dataset. The dataset comprises 1.2 million training and 50,000 validation images, where each image has a label corresponding to one of the 1,000 classes. Following the setting of Prakash et al. [2018], we utilize a subset of the validation data by randomly sampling two images from each class, which consists of totally 2,000 images. Before being fed into the classifier and defenders, the images are center-cropped into 224×224 pixels.

Classifier. By default, we use the pretrained ResNet50 model provided by PyTorch [Paszke et al., 2017] to evaluate our defender[2]. The pretrained model achieves a top-1 classification accuracy of 76.15% and 77.5% on the validation set and our subset, respectively. For all the following experiments, the classifier stays in evaluation mode, and no gradient information is needed by our defenders.

Attacker. We employ the one-step FGSM attacker [Goodfellow, Shlens, and Szegedy, 2014] with step size 0.015.

Results

Dropout *vs.* DAE *vs.* Iterative Optimization. We compare the accuracy of the proposed defenders (DIP, I-DIP, DAE and RIDE) against the number of iterations using the default hyper-parameters. As shown in Fig. 5(a), we observe that:

- Either dropout, or DAE, or iterative optimization can significantly help alleviate the overfitting in DIP, leading to better robustness in defense and better prediction recovery.
- Among the three, DAE seems to be the most important for defense, and iterative optimization is the second.
- Our RIDE algorithm, which involves all the three techniques, achieves the highest prediction accuracy.

We also show qualitative results in Fig. 6 to illustrate the reconstruction process of a single image using our proposed method. Note that the probability, p, of the ground-truth class increases with the increase of the number of iterations during the learning. Besides, the distance between the reconstructed image and the clean image (*dist2gt*) is consistently smaller than that between the reconstruction and the adversarial image (*dist2adv*), which further supports the motivation of the proposed algorithm as shown in Fig. 2(e).

Reconstruction Model h. We compare our FCN model with U-Net [Ronneberger, Fischer, and Brox, 2015], which features a symmetric encoder-decoder architecture with skip connections, by conducting the same experiments in Fig. 5(a). We observed a similar tendency of the curves, but the performance is $\sim 8\%$ lower for the U-Net model. Note that finding the optimal network architecture for reconstruction is out of the scope of this paper, and we will consider it in future work.

Mini-Batch Size n. We verify the performance of $n = \{1, 2, 4, 8\}$ due to the limit of the GPU memory. Note that we still take a single image as input, while augmenting the input and target images by adding different Gaussian noise. As illustrated in Fig. 5(b), the performance using $n = 1$ is

[1] Theoretically $\mathbf{x}^{(T)}$ and $\mathbf{x}_{est}^{(T)}$ should merge when $T \to \infty$. Our empirical studies on ImageNet show that $\mathbf{x}^{(T)}$ is closer to $\tilde{\mathbf{x}}$ and performs worse than $\mathbf{x}_{est}^{(T)}$ by $\sim 15\%$ in terms of test accuracy.

[2] We also tested RIDE on VGG19 [Simonyan and Zisserman, 2015] and Inception V3 [Szegedy et al., 2016] using the default hyper-parameter setting for ResNet50, which show 4.3% and 8.2% performance drop compared to the defense result of ResNet50. Please refer to the supplementary material for more details.

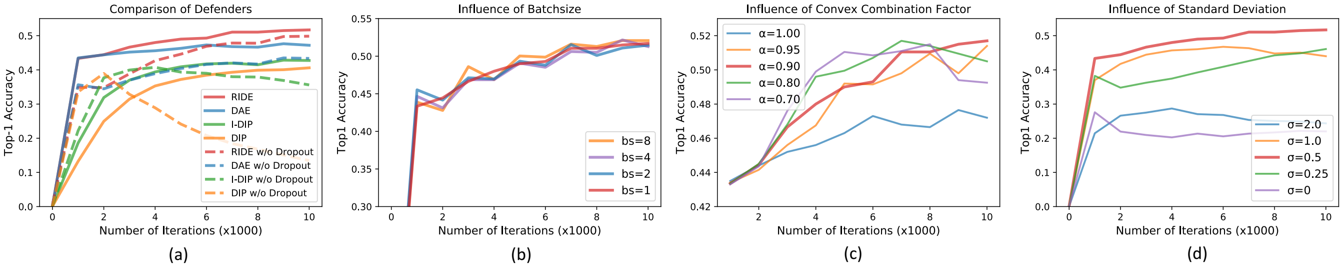
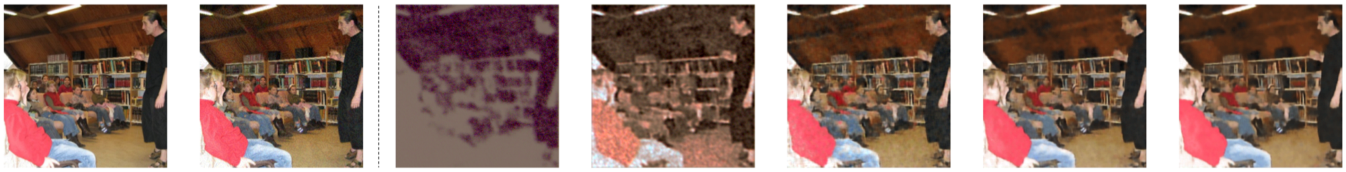


Figure 5: Ablation study on RIDE in Alg. 2 by comparing the effects of: **(a)** dropout, DAE, and iterative optimization; **(b)** batch size n; **(c)** interpolation parameter α; and **(d)** Gaussian std $\boldsymbol{\sigma}$ of DAE.



Clean (*p*=0.988)

Adv (*p*=0.003)
(dist2gt, dist2adv) :

Iter 10 (*p*=1e-5)
(33.11 < 33.46)

Iter 100 (*p*=1e-3)
(13.24 < 14.07)

Iter 1,000 (*p*=0.106)
(7.89 < 9.24)

Iter 5,000 (*p*=0.255)
(7.03 < 8.72)

Iter 10,000 (*p*=0.491)
(7.01 < 8.77)

Figure 6: Visualization of the reconstruction process. From left to right: clean image, adversarial image, and reconstructed images via self-supervision at different iterations. The class probability of the correct class (624 *library*) are shown under each image, as well as the averaged Manhattan distances of the reconstructed image to the clean and adversarial images (dist2gt and dist2adv in $[0, 255]$ space).

comparable with the others. Considering that both time and memory consumption increase linearly with batch size, we fix the mini-batch size as 1 for all the following experiments.

Interpolation Parameter α**.** As illustrated in Fig. 5(c) we verify the performance of $\alpha = \{0.7, 0.8, 0.9, 0.95, 1\}$. Note that when $\alpha = 1$, RIDE becomes the DAE defender. As we see in Fig. 5(c), $\alpha = 0.9$ achieves the highest accuracy, while $\alpha = 0.95$ performs marginally worse. $\alpha = 0.8$ and 0.7 reach peak performance at earlier stages, and we hypothesize that the estimator $\mathbf{x}^{(t)}$ becomes too far away from the unobserved sample, which may be regarded as an underfit problem.

Gaussian Standard Deviation $\boldsymbol{\sigma} = \sigma \times \mathbf{I}, \sigma \geq 0$**.** For simplicity, we represent $\boldsymbol{\sigma}$ as a scalar times an identity matrix. Fig. 5(d) shows the influence of standard deviation in RIDE. Recall that we add random Gaussian noise to both input and target images to further improve the robustness. When $\sigma = 0$, our model becomes a vanilla autoencoder-decoder. When σ is too small, the model tends to learn an identity mapping, while when σ is too large (*e.g.* , 2), the model is hard to converge. By default, we set $\sigma = 0.5$ as this setting achieves the best performance among what we tested.

Running Time and Memory Footprint. For each image in the ImageNet dataset with size of 224×244 pixels, our RIDE takes $\sim 140s$ to finish the optimization on an NVIDIA Titan X GPU with a memory footprint of ~ 1 gigabyte.

State-of-the-art Performance Comparison

Defense Against Gray-Box Attack

Following the experimental setup in the ablation study, we first compare the defensive performance on ImageNet against gray-box attacks as listed in Table 1, where the attack has full access to the classifier, but not the defender. We use the default parameters in RIDE, and the default classifier ResNet50 with 77.5% top-1 accuracy using clean images.

Table 1: Performance comparison of different defenders on ImageNet against gray-box attacks. We show the top-1 accuracy of median filtering [Guo et al., 2018], total variance minimization (TVM) [Guo et al., 2018], and our RIDE against both single-step FGSM [Szegedy et al., 2013] and iterative PGD [Madry et al., 2017].

	No Attack	No Defender	Median	TVM	RIDE (**Ours**)
FGSM	0.775	0.075	0.306	0.359	**0.517**
PGD		0.000	0.251	0.301	**0.476**

For the attacks, we use the public code released by Guo et al. [2018], and for TVM defense, we use the Python scikit-image package [van der Walt et al., 2014].

Median Filtering. We test different kernel sizes for the median filter, *i.e.* $k \in \{3, 5, 7, 9, 11\}$. For a given kernel size k, the input image is zero-padded by $\left\lfloor \frac{k}{2} \right\rfloor$ along the boundaries to keep the input size unchanged (*i.e.* , 224×224). We report the best result under each attacker in Table 1. Please refer to our supplementary material for more results.

TV Minimization. Total variance minimization (TVM) tries to find an image similar to the input image but with less total variance. The trade-off is controlled by the weight factor where a larger one results in an image with less total variance at the cost of less similarity. The application of TVM on adversarial defense was recently introduced by Guo et al. [2018] and tested on ImageNet. There are two commonly used TVM algorithms called the split-Bregman and Chambolle. In Table 1 we show the best results among the two. Please refer to our supplementary material for more results.

Results & Discussion. As we see in Table 1, our RIDE defender significantly outperforms the rest by at least 15.8% and 17.5% under the attack of FGSM and PGD, respectively.

Recall that our RIDE is developed for the stronger white-box attack, it should also work for the gay-box attacks. As a demonstration, we compare with median filtering and TVM, which have achieved (near) state-of-the-art performance on

Table 2: Performance comparison with the state-of-the-art defenders against the BPDA attacker. Our RIDE defender achieves the best performance on all the three datasets under stronger attacks (*i.e.* , larger maximum distances). Except RIDE, all the other numbers are cited from the results reported by Athalye, Carlini, and Wagner [2018] under similar experimental settings.

Defense	Dataset	Distance	Accuracy
Samangouei *et al.* (2018)	MNIST	0.005 (ℓ_2)	55%
RIDE (**Ours**)	MNIST	0.1 (ℓ_∞)	**98%**
Buckman *et al.* (2018)	CIFAR	0.031 (ℓ_∞)	0%
Dhillon *et al.* (2018)	CIFAR	0.031 (ℓ_∞)	0%
Ma *et al.* (2018)	CIFAR	0.031 (ℓ_∞)	5%
Song *et al.* (2018)	CIFAR	0.031 (ℓ_∞)	9%
Na *et al.* (2018)	CIFAR	0.015 (ℓ_∞)	15%
Madry *et al.* (2018)	CIFAR	0.031 (ℓ_∞)	47%
RIDE (**Ours**)	CIFAR	0.050 (ℓ_∞)	**76%**
Xie *et al.* (2018)	ImageNet	0.031 (ℓ_∞)	0%
Guo *et al.* (2018)	ImageNet	0.005 (ℓ_2)	0%
RIDE (**Ours**)	ImageNet	0.050 (ℓ_∞)	**43%**

several benchmark datasets without classifier re-training or external data for defense. Those methods share the same setting as RIDE. We are aware that there exist some other methods that may perform better. For instance, the image quilting defense also proposed by Guo et al. [2018]. However, it requires a collection of one million clean images as an “image bank”, whose patches are used to replace the regions in the adversarial images at inference time. Since it does not follow the setting that no external data should be used, for a fair comparison, we do not directly compare with such defenders. However, the performance of RIDE is still highly comparable to the accuracy reported by Guo et al. [2018].

Defense Against White-Box Attack

Datasets. Besides our randomly selected dataset from ImageNet, here we also test our RIDE on MNIST and CIFAR-10 to demonstrate the broad applicability of our algorithm. Both datasets contain 50k training and 10k test images, divided into 10 classes. Images in MNIST are grayscale of size 28×28, while CIFAR contains color images of size 32×32.

Classifiers. For ImageNet, we continue to use ResNet50 as before. For MNIST, we train a simple CNN model with two convolutional and two fully connected layers, which reports a test accuracy of 98.98%. For CIFAR-10, we train a ResNet18 model with a test accuracy of 94.78%.

Attackers. We employ the BPDA attacker [Athalye, Carlini, and Wagner, 2018] to verify our white-box defense performance because as far as we know, BPDA is currently the only attacker that can utilize the information (*i.e.* , the outputs) of the defenders to attack the input images iteratively as a wrapper where in the inner loop another attacker is utilized.

Specifically, for the inner loop, we apply the iterative ℓ_∞ PGD attack [Madry et al., 2017] for 10 iterations. For ImageNet and CIFAR-10, we perform the iterative attack with a step size of 0.01 and maximum perturbation of 0.05 on images normalized to $(0, 1)$. For MNIST, we perform a stronger attack with maximum ℓ_∞ of 0.1 and step size of 0.02.

Networks for RIDE. For ImageNet, we use the same network as the one in the ablation study. Since the images in both MNIST and CIFAR-10 are much smaller, here we propose another two 4-layer (*i.e.* , $1 \to 16 \to 16 \to 16 \to 1$ with ReLU activations) and 5-layer (*i.e.* , $3 \to 32 \to 32 \to 32 \to 32 \to 3$ with ReLU activations) FCNs for them, respectively.

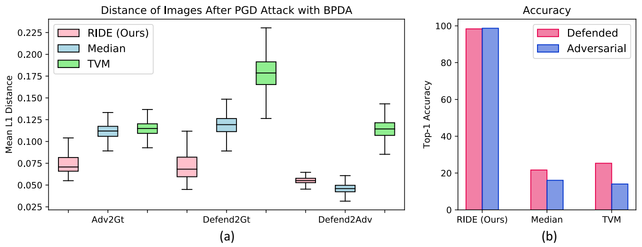


Figure 7: Analysis of defense against BPDA on MNIST: **(a)** Distances among adversarial, defended, and ground-truth images; **(b)** Top-1 test accuracy of both the defended and adversarial images.

Results & Discussion. We list all the comparison results in Table 2. The accuracy indicates that even under stronger attacks, our RIDE still can significantly outperform the state-of-the-art defenders by large margins on all three datasets.

Recall that the BPDA attacker assumes that $g(\mathbf{x}) \approx \mathbf{x}$ holds for a defender g and estimate the gradient through $\nabla_{\mathbf{x}} f(g(\mathbf{x}))|_{\mathbf{x}=\tilde{\mathbf{x}}} \approx \nabla f(\mathbf{x})|_{\mathbf{x}=g(\tilde{\mathbf{x}})}$. To analyze why RIDE can survive the attack, we compute the image distances on MNIST (Fig. 7(a)). After 10-iteration attacks, the distances of the defended images to the adversarial images (defend2adv) for RIDE have a similar distribution as median filtering, both of which are much lower than TVM. This indicates that the assumption of $g(\mathbf{x}) \approx \mathbf{x}$ in BPDA is very likely to hold for RIDE. However, the distances of the clean images to the adversarial (adv2gt) and defended images (defense2gt) using RIDE are significantly lower than the other methods. This indicates that the gradients *w.r.t.* the input are hindered by RIDE, making the adversarial images much “cleaner”.

Besides, we feed both the adversarial and defended images after the attacks into the classifier. Fig. 7(b) indicates that for both kinds of images RIDE achieves $\sim 98\%$ accuracy while the other two perform much worse. Refer to Eq. 3, we have:

$$\begin{aligned} \nabla_{\mathbf{x}} f(g(\mathbf{x}; \omega(\mathbf{x}))) &= \frac{\partial g}{\partial \mathbf{x}} \cdot \nabla f(g(\mathbf{x}; \omega(\mathbf{x}))) \\ &= [\mathbf{I}, \nabla \omega(\mathbf{x})] \, \nabla g(\mathbf{x}; \omega(\mathbf{x})) \nabla f(g(\mathbf{x}; \omega(\mathbf{x}))). \end{aligned} \quad (10)$$

With full access to both defender and classifier, the attacker can still compute $\nabla f(g(\mathbf{x}; \omega(\mathbf{x})))$, $\nabla g(\mathbf{x}; \omega(\mathbf{x}))$. However, calculating $\nabla \omega(\mathbf{x})$ can be extremely difficult because ω differs for each individual sample. So the explicit form of ω can not be estimated without knowing the prior distribution of the inputs. Such an input-dependent function breaks the assumption $\nabla_{\mathbf{x}} f(g(\mathbf{x}))|_{\mathbf{x}=\tilde{\mathbf{x}}} \approx \nabla f(\mathbf{x})|_{\mathbf{x}=g(\tilde{\mathbf{x}})}$ in BPDA, which is fundamental for the success of a *functional* defender.

Conclusion

In this paper, we propose a novel functional-based adversarial defender, RIDE, against white-box adversarial attacks without any modification of well-trained classifiers. Our defender utilizes an iterative self-supervised optimization algorithm to estimate the clean data for each individual adversarial input. For future work, we will focus on improving the optimizing efficiency of RIDE and target for real-time applications.

References

Akhtar, N., and Mian, A. 2018. Threat of adversarial attacks on deep learning in computer vision: A survey. *IEEE Access* 6:14410–14430.

Akhtar, N.; Liu, J.; and Mian, A. 2018. Defense against universal adversarial perturbations. In *CVPR*, 3389–3398.

Athalye, A.; Engstrom, L.; Ilyas, A.; and Kwok, K. 2017. Synthesizing robust adversarial examples. *arXiv preprint arXiv:1707.07397*.

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

Carlini, N., and Wagner, D. 2017. Adversarial examples are not easily detected: Bypassing ten detection methods. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, 3–14. ACM.

Colson, B.; Marcotte, P.; and Savard, G. 2007. An overview of bilevel optimization. *Annals of operations research* 153(1):235–256.

Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; and Fei-Fei, L. 2009. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR*.

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

Guo, C.; Rana, M.; Cisse, M.; and van der Maaten, L. 2018. Countering adversarial images using input transformations. In *ICLR*.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *CVPR*, 770–778.

Hendrycks, D., and Dietterich, T. 2019. Benchmarking neural network robustness to common corruptions and perturbations. In *ICLR*.

Ilyas, A.; Engstrom, L.; Athalye, A.; and Lin, J. 2018. Black-box adversarial attacks with limited queries and information. In *ICML*, 2142–2151.

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

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

Kurakin, A.; Goodfellow, I.; and Bengio, S. 2016. Adversarial examples in the physical world. *arXiv preprint arXiv:1607.02533*.

Liao, F.; Liang, M.; Dong, Y.; Pang, T.; Hu, X.; and Zhu, J. 2018. Defense against adversarial attacks using high-level representation guided denoiser. In *CVPR*, 1778–1787.

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*.

Moosavi-Dezfooli, S.-M.; Fawzi, A.; Fawzi, O.; and Frossard, P. 2017. Universal adversarial perturbations. In *CVPR*, 1765–1773.

Moosavi-Dezfooli, S.-M.; Fawzi, A.; and Frossard, P. 2016. Deepfool: a simple and accurate method to fool deep neural networks. In *CVPR*, 2574–2582.

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.

Prakash, A.; Moran, N.; Garber, S.; DiLillo, A.; and Storer, J. 2018. Deflecting adversarial attacks with pixel deflection. In *CVPR*, 8571–8580.

Qian, N. 1999. On the momentum term in gradient descent learning algorithms. *Neural networks* 12(1):145–151.

Ronneberger, O.; Fischer, P.; and Brox, T. 2015. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, 234–241.

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

Shocher, A.; Cohen, N.; and Irani, M. 2018. "zero-shot" super-resolution using deep internal learning. In *CVPR*, 3118–3126.

Simon-Gabriel, C.-J.; Ollivier, Y.; Bottou, L.; Schölkopf, B.; and Lopez-Paz, D. 2019. First-order adversarial vulnerability of neural networks and input dimension. In *ICML*, 5809–5817.

Simonyan, K., and Zisserman, A. 2015. Very deep convolutional networks for large-scale image recognition. *ICLR*.

Sinha, A.; Singh, M.; Kumari, N.; Krishnamurthy, B.; Machiraju, H.; and Balasubramanian, V. 2019. Harnessing the vulnerability of latent layers in adversarially trained models. *arXiv preprint arXiv:1905.05186*.

Su, J.; Vargas, D. V.; and Sakurai, K. 2019. One pixel attack for fooling deep neural networks. *IEEE Transactions on Evolutionary Computation*.

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*.

Szegedy, C.; Vanhoucke, V.; Ioffe, S.; Shlens, J.; and Wojna, Z. 2016. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2818–2826.

Tschannen, M.; Bachem, O.; and Lucic, M. 2018. Recent advances in autoencoder-based representation learning. *arXiv preprint arXiv:1812.05069*.

Ulyanov, D.; Vedaldi, A.; and Lempitsky, V. 2018. Deep image prior. In *CVPR*, 9446–9454.

van der Walt, S.; Schönberger, J. L.; Nunez-Iglesias, J.; Boulogne, F.; Warner, J. D.; Yager, N.; Gouillart, E.; Yu, T.; and the scikit-image contributors. 2014. scikit-image: image processing in Python. *PeerJ* 2:e453.

Vincent, P.; Larochelle, H.; Bengio, Y.; and Manzagol, P.-A. 2008. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, 1096–1103. ACM.

Xie, C.; Wu, Y.; van der Maaten, L.; Yuille, A. L.; and He, K. 2019. Feature denoising for improving adversarial robustness. In *CVPR*, 501–509.

Zisserman, A. 2018. Self-supervised learning. https://project.inria.fr/paiss/files/2018/07/zisserman-self-supervised.pdf.

Appendix

Proof of Theorem 1

Proof. To prove this theorem, we consider the first three arguments in $\mathcal{L}_{RIDE}$ in Eq. 8 as variables, and try to minimize the loss by optimizing these variables one-by-one. To do so, based on the assumption in the theorem, Eq. 8 and Alg. 2, we have that $\forall t$,

$$\begin{aligned} &\mathcal{L}_{RIDE}\left(\mathbf{x}^{(t-1)}, \mathbf{x}^{(t-1)}, \mathbf{w}^{(t)}, \theta\right) \\ &\geq \mathcal{L}_{RIDE}\left(\mathbf{x}^{(t-1)}, \mathbf{x}^{(t)}, \mathbf{w}^{(t)}, \theta\right) \Longleftarrow Eq.9 \quad (11) \\ &\geq \mathcal{L}_{RIDE}\left(\mathbf{x}^{(t)}, \mathbf{x}^{(t)}, \mathbf{w}^{(t)}, \theta\right) \Longleftarrow d(\mathbf{x}^{(t)}, \mathbf{x}^{(t-1)}) = 0 \quad (12) \\ &\geq \mathcal{L}_{RIDE}\left(\mathbf{x}^{(t)}, \mathbf{x}^{(t)}, \mathbf{w}^{(t+1)}, \theta\right). \Longleftarrow Eq.7, 8, assmp. \quad (13) \end{aligned}$$

Eq. 11 optimizes the second variable based on Eq. 9 while fixing the rest variables. Eq. 12 optimizes the first variable by making the distance term equal to 0 while fixing the rest. Eq. 13 optimizes the third variable by training the network while fixing the rest. Moreover, since $\mathcal{L}_{RIDE}$ is lower-bounded by 0, we then can complete our proof. □

Implementation Details

Non-Maximum Suppression. To further prevent the reconstruction model from overfitting, we apply the non-maximum suppression at the convex combination step. That is, for current reconstructed and target images $\mathbf{x}_{est}^{(t)}$ and $\mathbf{x}^{(t)} \in \mathbb{R}^{3\times H\times W}$, we calculate the pixel-wise ℓ_1 norm of the difference and get a set of distances at time step t.

$$S_d^{(t)} = \{d_i | d_i = \|x_i - x_i'\|_1, x_i \in \mathbf{x}^{(t)}, x_i' \in \mathbf{x}_{est}^{(t)}\} \quad (14)$$

where i is the index of the corresponding pixel. Because the optimization goal is to recover the image without overfitting to the adversarial target, the non-maximum suppression operation here is that for a given threshold τ, the pixel pairs with $d_i < \tau$ are excluded in the convex combination steps and the pixels in the target image $\mathbf{x}^{(t)}$ are directly substituted by the corresponding pixels in the reconstructed image $\mathbf{x}_{est}^{(t)}$. The reason is that when the distance between the reconstructed image and target image are already close enough, substitute pixels in the adversarial image with pixels in the reconstructed image can decrease the tendency of overfitting but cause little influence on the learning of image content and texture. For all the experiments with iterative updating described in this paper, we just set the threshold $\tau = \text{median}\{S_d\}$, which is calculated adaptively on each individual image without substantial parameter searching.

Random Loss Masking. Another technique we apply to the self-supervised image reconstruction process is to randomly mask out part of the pixels in the loss calculation. Because the reconstruction network works on a single image, therefore no matter what are the noise patterns in the adversarial image, the model can easily overfit to it. Therefore we can randomly exclude p portion of the pixels in loss calculation at each iteration. Although after updating the model for thousands of iterations by expectation every pixel in the target image should be directly supervised for several times, the random loss masking can still alleviate the risk of overfitting because every iteration the pixel value for some pixels are learned not based on its noisy version but the context in the corresponding field of view. We show experiments on RIDE with fixed convex combination factor $\alpha = 0.9$ using different random masking ratio p in Fig. 9. Accordingly, for all the experiments in the paper, we fix the random masking ratio of $p = 0.9$.

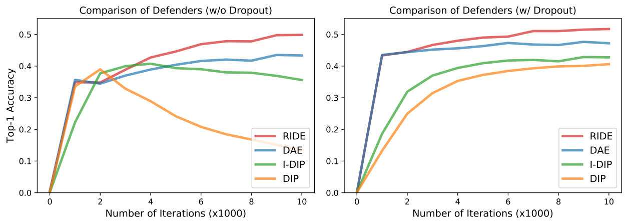


Figure 8: Comparison of different proposed defenders on classification performance **(left)** without, or **(right)** with dropout layers in the reconstruction model. This figure just separates the curves in Fig. 5(a) to clearly indicate the incremental performance improvement of different proposed techniques.

Model Architecture. Here we show the architectures of the reconstruction models used in our experiments (Table 3). The random dropout ratio for Dropout layers is 0.5.

Image Normalization. Pre-trained PyTorch classifiers (*e.g.* , ResNet50, VGG19, and InceptionV3) suppose that an input image is normalized to have zero mean. Specifically, for our ablation study and gray-box defense experiments on ImageNet dataset, the adversarial images are pre-computed, and before feeding the images into the RIDE defender, the pixel values of the images are normalized by means $\mu = (0.485, 0.456, 0.406)$ and standard deviation $\sigma = (0.229, 0.224, 0.225)$. Therefore there is no *sigmoid* activation for the reconstruction model in RIDE for ablation study and gray-box defense experiments. However, for the white-box setting (*i.e.*, BPDA) the perturbations need to be calculated on-the-fly, therefore we add a *sigmoid* activation to the reconstruction model to make the values of the output images restricted to $(0, 1)$, and make the normalization a differentiable layer at the beginning of the classifier.

We apply the same operation for CIFAR-10 and MNIST under the white-box defense setting. The mean and standard deviation for CIFAR-10 are the same as those used for ImageNet. For MNIST, which consists of gray-scale images, the mean is 0.1307 while the standard deviation is 0.3081.

Additional Results

Other Classifier & Clean Images. To show the transferability of the proposed defense algorithm, we also tested the RIDE defender on VGG19 [Simonyan and Zisserman, 2015] and Inception V3 [Szegedy et al., 2016] models. Using the optimal hyper-parameter settings for ResNet50, the RIDE defender show only 4.3% and 8.2% performance drop on VGG19 and Inception V3 models compared to the FGSM de-

Table 3: Architecture of reconstruction models. All layers except the output layer (`convn`) are followed by a ReLU activation. There is a sigmoid activation following the output layer if the image range is $(0, 1)$, and no activation for image normalized with $mean = 0$.

Dataset	Layer Name	Input Planes	Output Planes	Dropout
ImageNet	`conv1`	3	32	
	`conv2`	32	32	
	`conv3`	32	32	
	`conv4`	32	32	✓
	`conv5`	32	32	✓
	`conv6`	32	32	✓
	`conv7`	32	32	
	`convn`	32	3	
MNIST	`conv1`	1	16	
	`conv2`	16	16	✓
	`conv3`	16	16	✓
	`convn`	16	1	
CIFAR-10	`conv1`	3	32	
	`conv2`	32	32	✓
	`conv3`	32	32	✓
	`conv4`	32	32	
	`convn`	32	3	

Table 4: Performance of median filtering against *gray-box* attacks. We show the top-1 accuracy of different $kernel\_size$ on ImageNet validation set. The highest value is reported in Table 1.

Attack	3	5	7	9	11
FGSM	0.180	0.281	**0.306**	0.277	0.241
PGD	0.005	0.128	0.238	**0.251**	0.228

Table 5: Performance of total-variation denoising using split-Bregman optimization against *gray-box* attacks. We show the top-1 accuracy of different $denoising\_weight$ on ImageNet validation set. The highest value is reported in Table 1.

Attack	0.075	0.125	0.25	0.5	0.75
FGSM	0.312	0.339	**0.359**	0.326	0.288
PGD	0.281	**0.301**	0.289	0.218	0.149

Table 6: Performance of Chambolle total-variation denoising against *gray-box* attacks. We show the top-1 accuracy of different $denoising\_weight$ on ImageNet validation set. The highest values under two attacks are lower than the Bregman method.

Attack	0.25	0.5	0.75	1.0	1.25	1.5	1.75
FGSM	0.174	0.277	0.321	**0.341**	0.332	0.323	0.315
PGD	0.017	0.138	0.234	0.267	0.279	0.283	**0.285**

fense result reported on ResNet50 (*i.e.*, 51.7%). Not that the adversarial samples are calculated using the target classifiers instead of using the ones generated for ResNet50. Besides, when taking the clean images as inputs, the test accuracy is 61.75% using the ResNet50 model.

Median Filtering. We show the results of using median filtering as defenders under gray-box setting on the ImageNet validation dataset in Table 4. For a given kernel size k (an odd number), the input image is zero-padded by $\lfloor \frac{k}{2} \rfloor$ along the boundaries to keep the input size unchanged (224×224).

TV Minimization. Total variance minimization (TVM) tries to find an image similar to the input image but with less total variance. The trade-off is controlled by the weight factor w where a larger w results in an image with less total variance at the expense of less similarity. The application of TVM on adversarial defense was recently introduced by Guo et al. [2018] and are tested on the ImageNet dataset. There are two commonly used TVM algorithms called the split-Bregman and Chambolle. Here we show the defense performance of the two algorithms in Table 5 and 6. The highest scores are reported in Table 1.

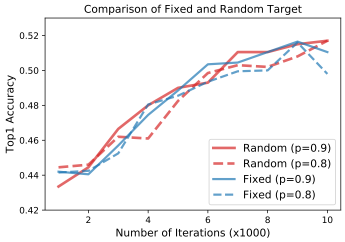
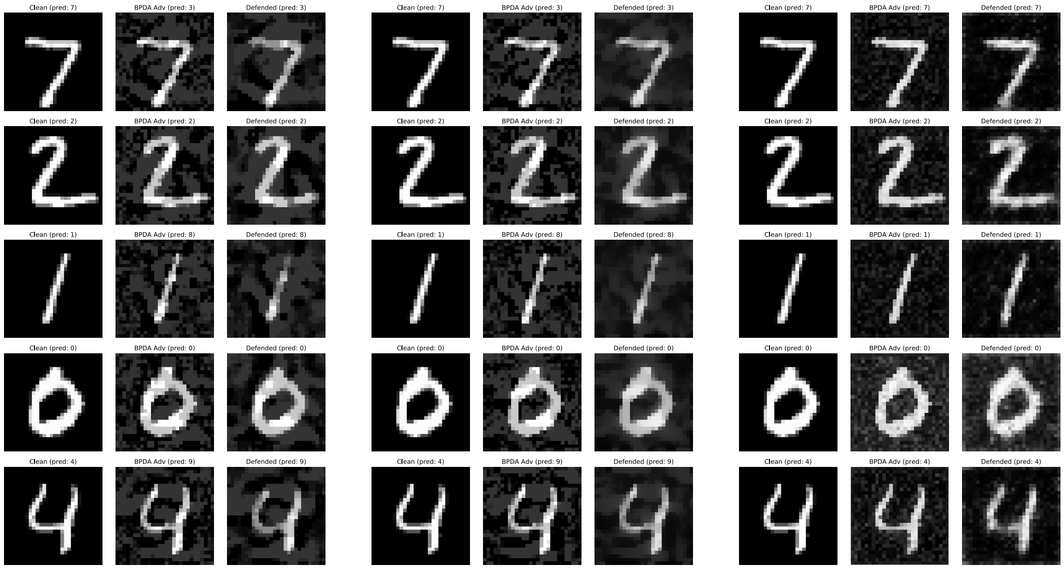


Figure 9: Comparison of the proposed RIDE algorithm using fixed target image (blue curves) or target image with random Gaussian noise (red curves). We show the comparison with random loss masking ratio of $p = 0.9$ and $p = 0.8$.

Qualitative Results

We show the qualitative comparisons on MNIST, CIFAR, and ImageNet datasets in the following figures.

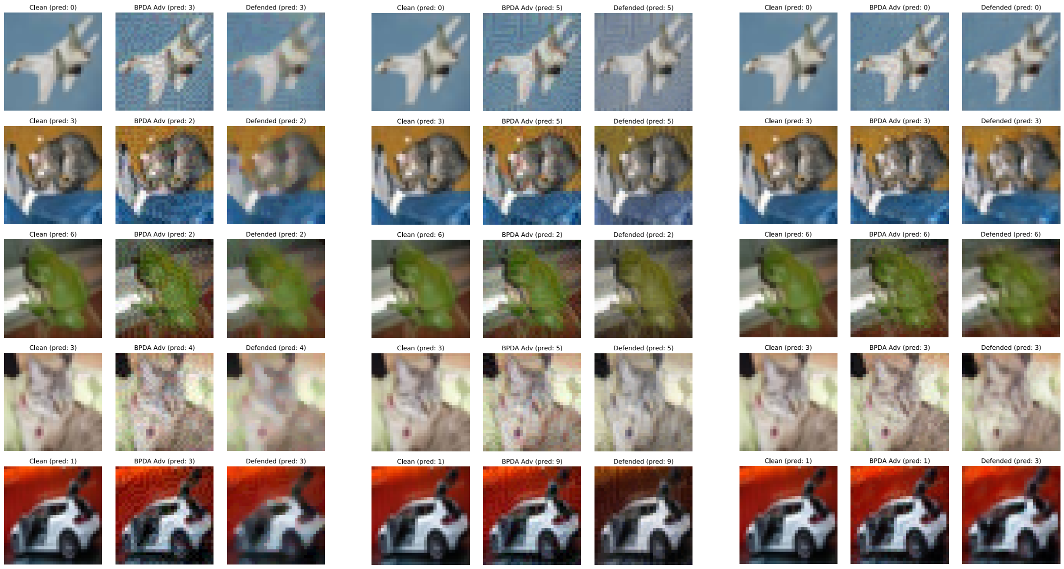


(a) Median Filtering

(b) TV Minimization

(c) RIDE (**Ours**)

Figure 10: Qualitative results of defense against 10-iteration PGD attack (with BPDA) on MNIST dataset using median filtering **(a)**, total-variance minimization **(b)** and the proposed RIDE algorithm **(c)**. The predicted label of each image is shown on top of it.

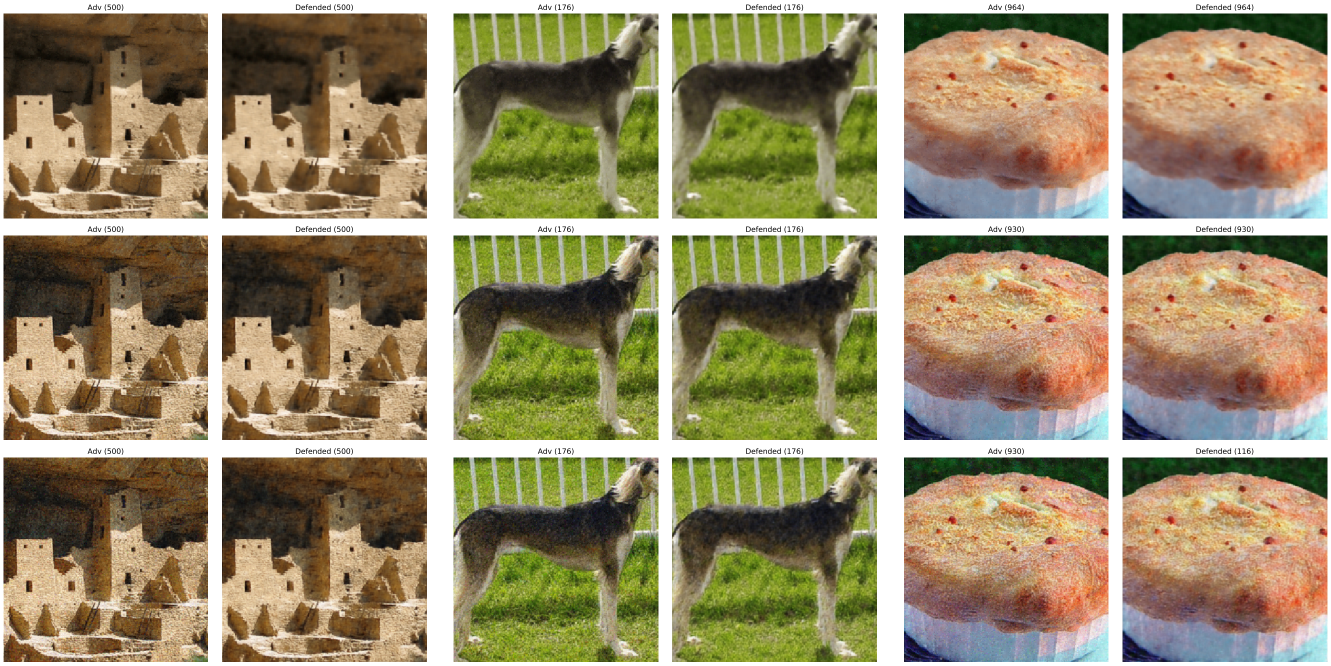


(a) Median Filtering

(b) TV Minimization

(c) RIDE (**Ours**)

Figure 11: Qualitative results of defense against 10-iteration PGD attack (with BPDA) on CIFAR-10 dataset using median filtering **(a)**, total-variance minimization **(b)** and the proposed RIDE algorithm **(c)**. The predicted label of each image is shown on top of it. The last row of RIDE denote a failure case.



(**a**) cliff dwelling (500)

(**b**) Saluki, gazelle hound (176)

(**c**) potpie (964)

Figure 12: Qualitative results of defense against 10-iteration PGD attack (with BPDA) on ImageNet dataset, using the proposed RIDE defender. Every two-column shows both the adversarial and defended images after 1,4 and 10 iterations of attack. The predicted label of each image is shown on top of it, where (**c**) denotes a failure case.