Robustifying Models Against Adversarial Attacks by Langevin Dynamics

Vignesh Srinivasan[1], Csaba Rohrer[1], Arturo Marban[1,2,3],
Klaus-Robert Müller[2,3,4,5,*], Wojciech Samek[1,3,*] and Shinichi Nakajima[2,3,6,*]
[1]Machine Learning Group, Fraunhofer Heinrich Hertz Institute, 10587 Berlin, Germany,
[2]Machine Learning Group, Technische Universität Berlin, 10587 Berlin, Germany, [3]BiFOLD,
[4]Dept. of Artificial Intelligence, Korea University, Seoul 136-713, South Korea,
[5]Max Planck Institute for Informatics, 66123 Saarbrücken, Germany,
[6]RIKEN AIP, 1-4-1 Nihonbashi, Chuo-ku, Tokyo 103-0027, Japan.
{wojciech.samek}@hhi.fraunhofer.de
{klaus-robert.mueller,nakajima}@tu-berlin.de

Abstract

Adversarial attacks on deep learning models have compromised their performance considerably. As remedies, a number of defense methods were proposed, which however, have been circumvented by newer and more sophisticated attacking strategies. In the midst of this ensuing arms race, the problem of robustness against adversarial attacks still remains a challenging task. This paper proposes a novel, simple yet effective defense strategy where off-manifold adversarial samples are driven towards high density regions of the data generating distribution of the (unknown) target class by the Metropolis-adjusted Langevin algorithm (MALA) with *perceptual boundary taken into account.* To achieve this task, we introduce a *generative* model of the conditional distribution of the inputs given labels that can be learned through a supervised Denoising Autoencoder (sDAE) in alignment with a *discriminative* classifier. Our algorithm, called MALA for DEfense (MALADE), is equipped with significant dispersion—projection is distributed broadly. This prevents white box attacks from accurately aligning the input to create an adversarial sample effectively. MALADE is applicable to any existing classifier, providing robust defense as well as off-manifold sample detection. In our experiments, MALADE exhibited state-of-the-art performance against various elaborate attacking strategies.

1. Introduction

Deep neural networks (DNNs) [1, 2, 3, 4, 5] have shown excellent performance in many applications, while they are known to be susceptible to adversarial attacks, i.e., examples crafted intentionally by adding slight noise to the input [6, 7, 8, 9, 10, 11]. These two aspects are considered to be two sides of the same coin: deep structure induces complex interactions between weights of different layers, which provides flexibility in expressing complex input-output relation with relatively small degrees of freedom, while it can make the output function unpredictable in *spots* where training samples exist sparsely. If adversarial attackers manage to find such spots in the input space close to a *real* sample, they can

*Corresponding author

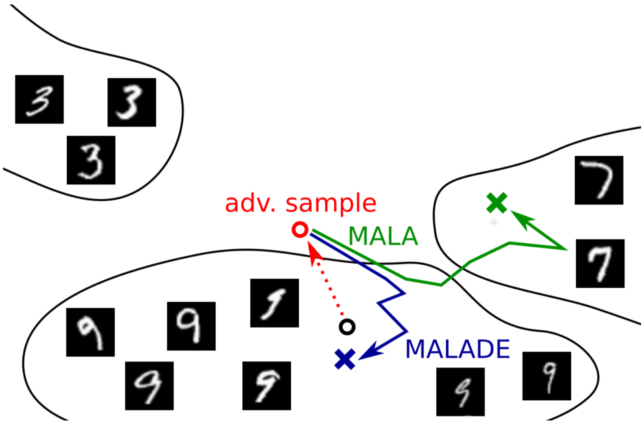
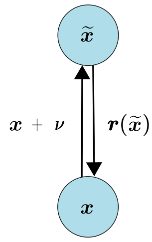
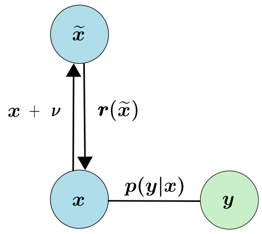


Figure 1: Adversarial attack and defense by MALA and MALADE. An adversarial sample (red circle) is created by moving the data point (black circle) away from the data manifold, here the manifold of images of digit "9". In this low density area, the DNN is not well trained and thus misclassifies the adversarial sample. Unsupervised sampling techniques such as MALA (green line) project the data point back to high density areas, however, not necessarily to the manifold of the original class. The proposed MALADE (blue line) takes into account of class information and thus projects the adversarial sample back to the manifold of images of digit "9".



$\boldsymbol{r}(\boldsymbol{x}) - \boldsymbol{x} = \sigma^2 \boldsymbol{\nabla}_{\boldsymbol{x}} \log p(\boldsymbol{x})$

(a) Denoising Autoencoder (DAE)



$\boldsymbol{r}(\boldsymbol{x}) - \boldsymbol{x} = \sigma^2 \boldsymbol{\nabla}_{\boldsymbol{x}} \log p(\boldsymbol{x}|\boldsymbol{y})$

(b) Supervised Denoising Autoencoder (sDAE)

Figure 2: DAE and sDAE. (a) DAE is trained so that corrupted images with Gaussian noise $\boldsymbol{\nu}$ are cleaned, and is known to provide a score function estimator of the marginal distribution $p(\boldsymbol{x})$. (b) sDAEs additionally learns from the classification loss, and provides a score function estimator of the conditional distribution $p(\boldsymbol{x}|\boldsymbol{y})$. Detailed discussion is given in Section 3.

manipulate the behavior of classification, which can lead to a critical risk of security in applications, e.g., self-driving cars, for which high reliability is required. Different types of defense strategies were proposed, including *adversarial training* [12, 13, 14, 15, 16, 17, 18, 19, 20] which incorporates adversarial samples in the training phase, *projection methods* [21, 22, 23, 24, 16, 25] which denoise adversarial samples by projecting them onto the data manifold, and *preprocessing methods* [26, 27, 28, 29] which try to destroy elaborate spatial coherence hidden in adversarial samples. Although those defense strategies were shown to be robust against the attacking strategies that had been proposed before, most of them have been circumvented by newer attacking strategies. Another type of approaches, called *certification-based methods* [30, 31, 32, 33, 20], minimize (bounds of) the worst case loss over a defined range of perturbations, and provide theoretical guarantees on robustness against any kind of attacks. However, the guarantee holds only for small perturbations, and the performance of those methods against existing attacks are typically inferior to the state-of-the-art. Thus, the problem of robustness against adversarial attacks still remains unsolved.

In this paper, we propose a novel defense strategy, which drives adversarial samples towards high density regions of the data distribution. Figure 1 explains the idea of our approach. Assume that an attacker created an adversarial sample (red circle) by moving an original sample (black circle) to an *untrained spot* where the target classifier gives a wrong prediction. We can assume that the spot is in a low density area of the training data, i.e., off the data manifold, where the classifier is not able to perform well, but still close to the original high density area so that the adversarial pattern is imperceptible to a human. Our approach is to *relax* the adversarial sample by the Metropolis-adjusted Langevin algorithm

(MALA) [34, 35], in order to project the adversarial sample back to the original high density area.

MALA requires the gradient of the energy function, which corresponds to the gradient $\nabla_{\boldsymbol{x}} \log p(\boldsymbol{x})$ of the log probability, a.k.a., the score function, of the input distribution. As discussed in [36], one can estimate this score function by a Denoising Autoencoder (DAE) [37] (see Figure 2a). However, naively applying MALA would have an apparent drawback: if there exist high density regions (clusters) close to each other but not sharing the same label, MALA could drive a sample into another cluster (see the green line in Figure 1), which degrades the classification accuracy. To overcome this drawback, we perform MALA driven by the score function of the conditional distribution $p(\boldsymbol{x}|\boldsymbol{y})$ given label $\boldsymbol{y}$. We will show that the score function can be estimated by a Supervised DAE (sDAE) [38, 39] (see Figure 2b) with the weights for the reconstruction loss and the classification loss appropriately set.

By using sDAE, our novel defense method, called MALA for DEfense (MALADE), relaxes the adversarial sample based on the conditional gradient $\nabla_{\boldsymbol{x}} \log p(\boldsymbol{x}|\boldsymbol{y})$ *without knowing the label* $\boldsymbol{y}$ *of the test sample*. Thus, MALADE drives the adversarial sample towards high density regions of the data generating distribution for the original class (see the blue line in Figure 1), where the classifier is well trained to predict the correct label.

Our proposed MALADE can be seen as one of the projection methods, most of which have been circumvented by recent attacking methods. However, MALADE has two essential differences from the previous projection methods:

Significant dispersion Most projection methods, including Magnet [28], Defense-GAN [23], PixelDefend [22], and others [24, 25], try to pull the adversarial sample back to the original point (so that the adversarial pattern is removed). On the other hand, MALADE drives the input sample to anywhere (randomly) in the closest cluster having the original label. In this sense, MALADE has much larger inherent randomness and thus resilience than the previous projection methods,

Perceptual boundary taken into account All previous projection methods pull the input sample into the closest point on the data manifold without the label information into account. On the other hand, MALADE is designed to drive the input sample into the data manifold of the original class.

The previous projection methods were broken down by aligning adversarial samples such that the classifier is fooled even after the projection or by finding adversarial samples that is not significantly moved by the projector [40, 41]. Significant dispersion of MALADE makes these attacking strategies harder: it prevents any whitebox attack from aligning the input so that MALADE *stably* moves it to a targeted untrained spot. Here, the second property is essential: when making dispersion of projection broad, it can happen that Langevin dynamics carries a sample from the original cluster to a neighboring cluster with different label, which results in a wrong prediction. sDAE, taking the perceptual boundary into account, allows us to safely perform Langevin Dynamics within the clusters of the correct label.

Concisely, our contributions in this paper are three fold:

- We prove that a sDAE can provide an estimator for the conditional gradient $\nabla_{\boldsymbol{x}} \log p(\boldsymbol{x}|\boldsymbol{y})$, without knowing the label at the test time.

- We propose to perform a modified version of MALA suited for defense, i.e., MALADE, which drives samples towards the high density area of the conditional, instead of the marginal.
- We empirically show that MALADE alone can protect the standard classifiers to get robust performance on MNIST. On ImageNet, the standard classifiers are completely broken down, and MALADE alone cannot make them robust. However, MALADE improves the performance of adversarially trained classifiers. A combined strategy of detection and defense enhances the performance, and achieves state-of-the-art results in countering adversarial samples on ImageNet.

This paper is organized as follows. We first summarize existing attacking and defense strategies in Section 2. Then, we propose our method with a novel conditional gradient estimator in Section 3. In Section 4, we evaluate our defense method against various attacking strategies, and show advantages over the state-of-the-art defense methods. Section 5 concludes.

2. Existing Methods

In this section, we introduce existing attacking and defense strategies.

2.1. Attacking Strategies

There are two scenarios considered in adversarial attacking. The whitebox scenario assumes that the attacker has the full knowledge on the target classification system, including the architecture and the weights of the DNN and the respective defense strategy,[4] while the blackbox scenario assumes that the attacker has access only to the classifier outputs.

2.1.1. Whitebox Attacks (General)

We first introduce representative whitebox attacks, which are effective against general classifiers with or without defense strategy.

Projected Gradient Descent (PGD) [17]. The PGD attack, a.k.a., the basic iterative method, solves the following problem iteratively by the projected gradient descent:

$$\begin{aligned} &\min_{\boldsymbol{x}'} -J(\boldsymbol{x}', \boldsymbol{y}) \\ &\text{s.t.} \quad \|\boldsymbol{x}' - \boldsymbol{x}\|_p \leq \varepsilon, \quad \boldsymbol{x}' \in [0,1]^L, \end{aligned} \tag{1}$$

where $\boldsymbol{x} \in [0,1]^L$ is the original image, $\boldsymbol{y} \in \{0,1\}^K$ is its (true) label, and ε is the upper-bound of the amplitude of adversarial patterns. $\|\cdot\|_p$ denotes the L_p-norm, and

$$J(\boldsymbol{x}, \boldsymbol{y}) = -\boldsymbol{y}^\top \log \widehat{\boldsymbol{y}}(\boldsymbol{x}) \tag{2}$$

is the cross entropy loss of the classifier output $\widehat{\boldsymbol{y}}(\boldsymbol{x}) \in [0,1]^K$. Here, $\top$ denotes the transpose of a vector, and $\log(\cdot)$ applies entry-wise.

[4] As in [40], we assume that the attacker cannot access to the random numbers (nor random seed) generated and used in stochastic defense processes.

While Fast Gradient Sign Method (FGSM) [6] corresponds to the first iteration of PGD, iteratively solving Eq.(1) makes it a strong first-order attack. Several variations can be incorporated into PGD, e.g., L_1 or L_2-norm for the perturbation bound, and use of momentum for better convergence [42].

Carlini-Wagner (CW) [43]. The CW attack optimizes the adversarial pattern $\boldsymbol{\tau} \in \mathbb{R}^L$ by solving

$$\begin{aligned} &\min_{\boldsymbol{\tau}} \|\boldsymbol{\tau}\|_p + c \cdot F(\boldsymbol{x} + \boldsymbol{\tau}, \boldsymbol{y}) \\ &\text{s.t.} \qquad \boldsymbol{x} + \boldsymbol{\tau} \in [0,1]^L, \end{aligned} \tag{3}$$

where

$$F(\boldsymbol{x} + \boldsymbol{\tau}, \boldsymbol{y}) = \max\{0, \log \widehat{y}_{k^*}(\boldsymbol{x} + \boldsymbol{\tau}) - \max_{k \neq k^*} \log \widehat{y}_k(\boldsymbol{x} + \boldsymbol{\tau}) + \iota\}.$$

Here, c is a trade-off parameter balancing the pattern intensity and the adversariality, k^* is the true label id, i.e., $y_{k^*} = 1$, and ι is a margin for the sample to be adversarial. Elastic-net Attack to Deep neural networks (EAD) [44, 45] is a modification of the CW attack where the L_p regularizer is replaced with the elastic-net regularizer, the sum of L_1 and L_2 norms.

2.1.2. Whitebox Attacks (Specialized)

Some attacking strategies target specific features of defense strategies, and enhance general whitebox attacks (introduced in the previous subsection). Reconstruction (R) Regularization [46] is suited for attacking defense strategies which are equipped with a denoising process, where adversarial pattern is removed by projection, e.g., by an autoencoder. The sum of the reconstruction loss by the denoising process and the (negative) cross entropy loss is minimized. Back Pass Differentiable Approximation (BPDA) [40] was strategized for defenses which prevent whitebox attackers from stably computing the gradient, e.g., by having non-differentiable layers or artificially inducing randomness. BPDA simply replaces such layers with identity maps, in order to stably estimate the gradient. This method is effective when the replaced layer is the denoising process that reconstructs the original input well, so that $\boldsymbol{\nabla}_{\boldsymbol{x}} \widehat{y}_k(r(\boldsymbol{x}')) \approx \boldsymbol{\nabla}_{\boldsymbol{x}} \widehat{y}_k(\boldsymbol{x})$ holds. The Expectation over Transformation (EOT) [40] method estimates the gradient by averaging over multiple trails, so that the randomness is averaged out. This method is effective against any stochastic defense methods.

2.1.3. Blackbox Attacks

In the blackbox scenario, attackers are assumed not to have the knowledge of the system but have access only to the output decision. Distillation Attack [47, 7] trains a student network and then use whitebox attacking strategies. Boundary Attack [48] performs random exploration on the input space to find the closest adversarial point to the original image. Assume that the attacker has full access to another classifier which was trained for the same purpose as the target classifier. Then, the attacker can create adversarial samples by any whitebox attack against the known model, and use them to attack the target classifier. Such an attack is called Transfer Attack [49]. Although, blackbox attacks are a more likely threat scenario, such attacks are, by definition, weaker than whitebox attacks.

2.2. Defense Strategies

As mentioned in Section 1, existing methods can be roughly classified into four categories.

Projection Methods. The methods in this category are generally seen as weaker than those in the other categories. Though many methods were proposed for adversarial defense using preprocessing for projection, (e.g., bit depth reduction, JPEG compression and decompression, random padding) [26, 29], [40] showed that they can be easily broken down by BPDA or EOT. Autoencoders can also be used as a preprocessor to remove adversarial patterns [27, 28], which however were broken down by CW [41].

Generative models have been shown to be useful to reconstruct the original image (or to remove the adversarial patterns) from an adversarial sample [23, 24, 25]. Since the generative model is trained to generate samples in the data manifold, the generative model effectively projects off-manifold samples onto the data manifold. The Analysis By Synthesis (ABS) method [21] uses a variational autoencoder to find the optimal latent vector maximizing the lower bound of the log likelihood of the given input to each of the class. Most existing strategies in this category have been rendered as ineffective against recent attacking strategies: Defense-GAN and PixelDefend have been broken down by BPDA [40].

Adversarial Training. In this strategy, adversarial samples are generated by known attacking strategies, and added to the training data, in order to make the classifier robust against those attacks [7, 15, 50, 16, 17, 18, 19, 20]. The method [17] proposed by Madry et al. (2017), which we refer to as "Madry" in this paper, withstood many adversarial attacks on MNIST and CIFAR10, and is considered to be the current state-of-the-art defense strategy that outperforms most of the other existing defense methods against most of the attacking methods. Adversarial Logit Pairing (ALP) [18] and Feature Denoising (FD) [20] utilize adversarial training on ImageNet. Methods in this category, typically show higher robustness than methods in the other categories. However, they have a risk of overfitting to the known attacking strategies [44, 45] and to intensity of perturbations [17].

In our experiment in Section 4, we choose Madry and FD as the state-of-the-art baselines, respectively, on MINST and on ImageNet. Madry is known to be the state-of-the-art on MNIST, as mentioned above. However, it did not show good performance on ImageNet, and was outperformed by ALP, another adversarial training method with logit pairing [18]. It was later found that the good performance of ALP in the original paper [18] was due to the small number of iterations in generating adversarial samples, and the same PGD attack with a larger number of iterations broke it down [51]. FD is equipped with feature denoising process on top of Madry, and showed excellent performance on ImageNet. However, it was found that the success of FD comes primarily from successful hyperparameter tuning of Madry (see the author's GitHub page[5]). Thus, Madry and FD are considered to be essentially the same method. Due to availability of pretrained networks, we use in our experiments a Madry implementation on MNIST, and a FD implementation on ImageNet.

Certification-based Methods. Certification-based methods employ robust optimization, and obtain *provably robust* networks. The idea is to train the classifier by minimizing (upper-bounds of) the worst-case loss over

[5]https://github.com/facebookresearch/ImageNet-Adversarial-Training/issues/1

Table 1: Existing adversarial defenses and attacks.

	Method	Related Work
Defenses	Projection	ABS [21], PixelDefend[22], Defense-GAN[23]
		Magnet [28],[24], [25], [26], [27], [29]
	Adversarial Training	Madry [17], ALP [18], [19],
		FD [20], [7], [15], [16], [50]
	Certification based	[30, 31, 32, 33, 52]
Attacks	Blackbox	Distillation Attack [47, 7], Boundary Attack [48], Transfer Attack [49]
	Whitebox	PGD [17], FGSM [6], CW [43], MIM [42], EAD [44, 45]
	Whitebox (Specialized)	Reconstruction (R) Regularization [46], BPDA, EOT [40]

a defined range of perturbations, so that creating adversarial samples in the range is impossible. To this end, one needs to solve a nested optimization problem, which consists of the inner optimization finding the worst case sample (or the strongest adversarial sample) and the outer optimization minimizing the worst-case loss. Since the inner optimization is typically non-convex, different relaxations [30, 52, 32, 33, 31] have been applied for scaling this approach.

The certification-based approach seems a promising direction towards the end of the arms race—it might protect classifiers against any (known or unknown) attacking strategy in the future. However, the existing methods still have limitations in many aspects, e.g., structure of networks, scalability, and the guaranteed range of the perturbation intensity. Typically, the robustness is guaranteed only for small perturbations, e.g., PGD-L_∞ Eq.(1) for $\varepsilon \leq 0.1$ in MNIST [31, 32, 33], and no method in this category has shown comparable performance to the state-of-the-art.

Table 1 summarizes the related work for adversarial defense and attack.

3. Proposed Method

In this section, we propose our novel defense strategy, which drives the input sample (if it lies in low density regions) towards high density regions. We achieve this by using Langevin dynamics.

3.1. Denoising Autoencoders

A denoising autoencoders (DAE) [37, 53] is trained such that data samples contaminated with artificial noise is cleaned. More specifically, it minimizes the reconstruction error:

$$\mathbb{E}_{p'(\boldsymbol{x})p'(\boldsymbol{\nu})}\left[\|\boldsymbol{r}(\boldsymbol{x}+\boldsymbol{\nu})-\boldsymbol{x}\|^2\right], \tag{4}$$

where $\mathbb{E}_p[\cdot]$ denotes the expectation over the distribution p, $\boldsymbol{x} \in \mathbb{R}^L$ is a training sample subject to a distribution $p(\boldsymbol{x})$, and $\boldsymbol{\nu} \sim \mathcal{N}_L(\boldsymbol{0}, \sigma^2\boldsymbol{I})$ is an L-dimensional artificial Gaussian noise with mean zero and variance σ^2. $p'(\cdot)$ denotes an empirical (training) distribution of the distribution $p(\cdot)$, namely, $\mathbb{E}_{p'(\boldsymbol{x})}[g(\boldsymbol{x})] = N^{-1}\sum_{n=1}^{N} g(\boldsymbol{x}^{(n)})$ where $\{\boldsymbol{x}^{(n)}\}_{n=1}^{N}$ are the training samples.

Proposition 1. *[36] Under the assumption that $\boldsymbol{r}(\boldsymbol{x}) = \boldsymbol{x} + o(1)$[6], the minimizer of the DAE objective Eq.(4) satisfies*

$$\boldsymbol{r}(\boldsymbol{x}) - \boldsymbol{x} = \sigma^2 \boldsymbol{\nabla}_{\boldsymbol{x}} \log p(\boldsymbol{x}) + o(\sigma^2), \tag{5}$$

as $\sigma^2 \to 0$.

Proposition 1 states that a DAE trained with a small σ^2 can be used to estimate the gradient of the log probability. In a blog [54], it was shown that the residual is proportional to the score function of the noisy input distribution for any σ^2, i.e.,

$$\boldsymbol{r}(\boldsymbol{x}) - \boldsymbol{x} = \sigma^2 \boldsymbol{\nabla}_{\boldsymbol{x}} \log \int \mathcal{N}_L(\boldsymbol{x}; \boldsymbol{x}', \sigma^2 \boldsymbol{I}_L) p(\boldsymbol{x}') d\boldsymbol{x}'. \tag{6}$$

3.2. Metropolis-adjusted Langevin Algorithm (MALA)

MALA is an efficient Markov chain Monte Carlo (MCMC) sampling method which uses the gradient of the energy (negative log-probability $E(\boldsymbol{x}) = -\log p(\boldsymbol{x})$). Sampling is performed sequentially by

$$\boldsymbol{x}_{t+1} = \boldsymbol{x}_t + \alpha \boldsymbol{\nabla}_{\boldsymbol{x}} \log p(\boldsymbol{x}_t) + \boldsymbol{\kappa}, \tag{7}$$

where α is the step size, and $\boldsymbol{\kappa}$ is random perturbation subject to $\mathcal{N}(\boldsymbol{0}, \delta^2 \boldsymbol{I}_L)$. By appropriately controlling the step size α and the noise variance δ^2, the sequence is known to converge to the distribution $p(\boldsymbol{x})$.[7]

3.3. Supervised Denoising Autoencoders (sDAE)

Let $\boldsymbol{y} \in \{0, 1\}^K$ be (the 1-of-K representation of) the label of a training image $\boldsymbol{x} \in [0, 1]^L$, and $\widehat{\boldsymbol{y}}(\boldsymbol{x}) \in [0, 1]^K$ be the classifier output (normalized by the final soft-max layer). We propose to train a *supervised denoising autoencoder* (sDAE) [38, 39] by minimizing the following functional with respect to the function $\boldsymbol{r} : \mathbb{R}^L \mapsto \mathbb{R}^L$:

$$\mathbb{E}_{p'(\boldsymbol{x},\boldsymbol{y})p'(\boldsymbol{\nu})} \left[\|\boldsymbol{r}(\boldsymbol{x} + \boldsymbol{\nu}) - \boldsymbol{x}\|^2 + 2\sigma^2 J\left(\boldsymbol{r}(\boldsymbol{x} + \boldsymbol{\nu}), \boldsymbol{y}\right) \right]. \tag{8}$$

The difference from the DAE objective Eq.(4) is in the second term, which is proportional to the cross entropy loss, defined in Eq.(2) . With this additional term, sDAE provides the gradient estimator of the log-*joint*-probability $\log p(\boldsymbol{x}, \boldsymbol{y})$ averaged over the training (conditional) distribution, as shown below. Note that, unlike the previous work [38, 39], we fix the balance (weights) between the reconstruction loss (first term) and the supervised loss (second term), which is essential in the following analysis.

One can see the classifier output as an estimator for the conditional distribution on the label given an image. We denote by $\widetilde{p}$ the estimated probability based on the classifier output, i.e., we use the following notation:

$$\widetilde{p}(\boldsymbol{y}|\boldsymbol{x}) \equiv \boldsymbol{y}^\top \widehat{\boldsymbol{y}}(\boldsymbol{x}), \qquad \widetilde{p}(\boldsymbol{x}, \boldsymbol{y}) \equiv \widetilde{p}(\boldsymbol{y}|\boldsymbol{x}) p(\boldsymbol{x}), \qquad \widetilde{p}(\boldsymbol{x}|\boldsymbol{y}) \equiv \frac{\widetilde{p}(\boldsymbol{y}|\boldsymbol{x})}{p(\boldsymbol{y})}. \tag{9}$$

[6]This assumption is not essential as we show in the proof in Appendix A.

[7]For convergence, a rejection step after Eq.(7) is required. However, it was observed that a variant, called MALA-approx [55], without the rejection step gives reasonable sequence for moderate step sizes. We use MALA-approx in our proposed method.

Theorem 1. *The minimizer of the sDAE objective Eq.*(8) *satisfies*

$$\boldsymbol{r}(\boldsymbol{x}) - \boldsymbol{x} = \sigma^2 \mathbb{E}_{p(\boldsymbol{y}|\boldsymbol{x})} \left[\boldsymbol{\nabla}_{\boldsymbol{x}} \log \widetilde{p}(\boldsymbol{x}, \boldsymbol{y}) \right] + O(\sigma^3). \tag{10}$$

(Sketch of proof) Similarly to the analysis in [36], we first Taylor expand $\boldsymbol{r}(\boldsymbol{x} + \boldsymbol{\nu})$ around $\boldsymbol{x}$, and write the sDAE objective similar to the contrastive autoencoder [56, 57] objective (The objective contains a higher order term than in [36] since we do not assume that $\boldsymbol{r}(\boldsymbol{x}) = \boldsymbol{x} + o(1)$). After that, applying the second order Euler-Lagrange equation gives Eq.(10) as a stationary condition. The complete proof is given in Appendix A. □

If the label distribution is flat (or equivalently the number of training samples for all classes is the same), i.e., $p(\boldsymbol{y}) = 1/K$, the residual of sDAE gives

$$\boldsymbol{r}(\boldsymbol{x}) - \boldsymbol{x} = \sigma^2 \mathbb{E}_{p(\boldsymbol{y}|\boldsymbol{x})} \left[\boldsymbol{\nabla}_{\boldsymbol{x}} \log \widetilde{p}(\boldsymbol{x}|\boldsymbol{y}) \right] + O(\sigma^3).$$

The first term is the gradient of the (estimated) log-*conditional*-distribution on the label, where the label is estimated from the prior knowledge (the expectation is taken over the training distribution of the label, given $\boldsymbol{x}$). If the number of training samples are non-uniform over the classes, the weight (or the step size) should be adjusted so that all classes contribute equally to the sDAE training.

3.4. MALA with sDAE for Defense (MALADE)

As discussed in Section 1, MALA drives the input into high density regions but not necessarily to the cluster sharing the same label with the original image (see Figure 1). To overcome this drawback, we propose MALA for defence (MALADE), which drives samples into high density regions of the *conditional* training distribution $p(\boldsymbol{x}|\boldsymbol{y})$, instead of the marginal $p(\boldsymbol{x})$. More specifically, sampling is performed by

$$\boldsymbol{x}_{t+1} = \boldsymbol{x}_t + \alpha \mathbb{E}_{p(\boldsymbol{y}|\boldsymbol{x}_t)} \left[\boldsymbol{\nabla}_{\boldsymbol{x}} \log p(\boldsymbol{x}_t|\boldsymbol{y}) \right] + \boldsymbol{\kappa}. \tag{11}$$

MALADE generates samples at every step using the score function $\boldsymbol{\nabla}_{\boldsymbol{x}} \log p(\boldsymbol{x}_t|\boldsymbol{y})$, for which an estimator is provided by a sDAE. α is the step size which describes the stride to be taken at every step and $\boldsymbol{\nu}$ is the noise term. Figure 3 shows a typical example, where MALADE (top-row) successfully drives the adversarial sample to the correct cluster, while MALA (bottom-row) drives it to a wrong cluster.

3.5. MALADE for Detection while Defending

Adversarial samples, especially with large perturbations, lie off the manifold and can have a higher norm of the score function compared to the clean samples. Such samples are easier to *detect* (and then reject) rather than to *fix* by defense methods. In most practice applications, it is not necessary to fix all adversarial samples, as long as they can be identified as adversarial and rejected.

The Magnet [28] defense was proposed with a detection procedure, where the samples with the norm of the score function estimator (by DAE) larger than a threshold θ, i.e.,

$$\|\boldsymbol{\nabla}_{\boldsymbol{x}} \log p(\boldsymbol{x}_0)\| > \theta, \tag{12}$$

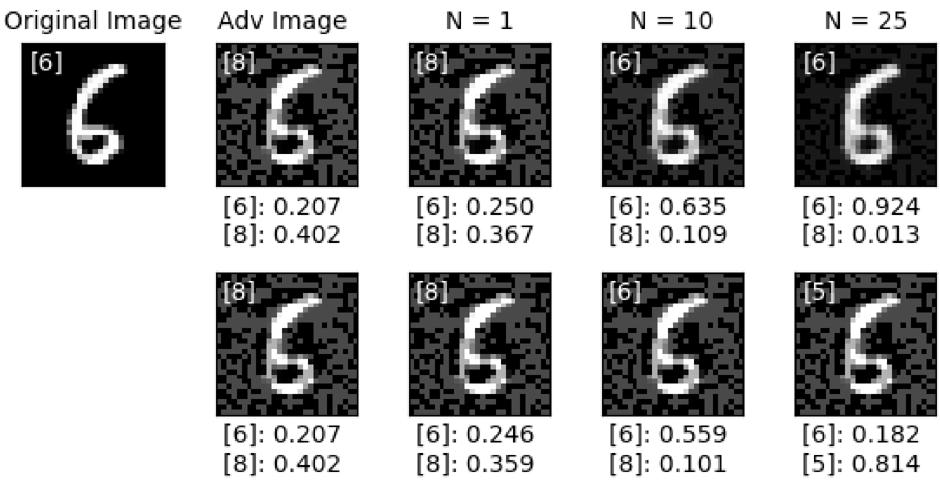


Figure 3: The top-left is the original image, from which the adversarial image (second column) was crafted. The third to the fifth columns show the images after $N = 1, 10$, and 25 steps of MALADE (top-row) and of MALA with the marginal distribution (bottom-row). Below each image, the prediction output $\widehat{y}_{k^*}$ for the original label $k^* =$"6" and that for the label with the highest output, i.e., $k = \mathrm{argmax}_{k \neq k*} \widehat{y}_k$, are shown. In this example, MALADE, trained with perceptional information, drives the adversarial sample towards the right cluster with the original label "6". On the other hand, although MALA successfully removed the adversarial pattern for "8", it brought the sample into a neighboring cluster with a wrong label "5".

are identified as adversarial, and thus rejected. In a similar fashion, we identify a sample as adversarial, if the score function estimator by sDAE is larger than a threshold:

$$\|\nabla_{\boldsymbol{x}} \log \widetilde{p}(\boldsymbol{x}_0|\boldsymbol{y})\| > \theta. \tag{13}$$

The threshold θ is set so that the false positive rate is controlled. We show in Section 4.5 that this combined strategy of detection and defense is highly useful when defense alone is not robust enough.

4. Experiments

In this section, we empirically evaluate our proposed MALADE against various attacking strategies, and compare it with the state-of-the-art baseline defense strategies.

4.1. Datasets

We conduct experiments on the following datasets:

MNIST: MNIST consists of handwritten digits from 0-9. The dataset is split into training, validation and test set with 50,000, 10,000 and 10,000 images, respectively. MNIST, in spite of being a small dataset, remains to be considered as adversarially robust.

ImageNet: ImageNet dataset consisting of 1,000 classes [58]. Each class contains around 1,300 training images and 50,000 test images in total. The images have a resolution of $224 \times 224 \times 3$. For the purpose of evaluation, we used the *NIPS 2017: Adversarial Learning Development Set*[8]. This dataset

[8]`https://www.kaggle.com/google-brain/nips-2017-adversarial-learning-development-set`

Table 2: Summary of classification performance on MNIST. For each attacking scenario, i.e., whitebox/blackbox and bounds for the amplitude of adversarial patterns, the lowest row gives the *worst case* result over the considered attacking strategies.

Setting	Condition	Attack	CNN	Madry	CNN + MALADE	Madry + MALADE
whitebox	L_∞ $\varepsilon = 0.3$	FGSM	11.77	97.52	93.54	95.59
		PGD	0.00	93.71	94.22	95.76
		R+PGD	-	-	92.65	93.51
		BPDA	-	-	84.74	94.70
		BPDAwEOT	-	-	82.48	91.54
		MIM	0.00	97.66	94.32	94.53
		worst case	0.00	**93.71**	82.48	91.54
	L_∞ $\varepsilon = 0.4$	PGD	0.00	0.02	93.51	92.18
		BPDA	-	-	66.16	83.90
		BPDAwEOT	-	-	62.15	80.65
		worst case	0.00	0.02	62.15	**80.65**
	L_2 $\varepsilon = 4$	FGM	30.79	97.68	94.68	96.05
		PGD	0.01	92.68	95.91	96.76
		CW	0.00	85.53	90.07	91.14
		worst case	0.0	85.53	90.07	**91.14**
	L_2-L_1 $\beta = 0.01$ and $c = 0.01$	EAD	0.00	0.01	**31.00**	30.59
blackbox	$\rho = 0.25$ $T = 5,000$	SaltnPepper	36.49	41.61	80.41	80.72
		Boundary Attack	32.39	1.10	93.79	95.80
		Transfer Attack	9.19	63.47	76.87	71.40
		worst case	9.19	1.10	**76.87**	71.40

was introduced in the NIPS 2017 Adversarial Learning challenges containing 1000 ImageNet-like images, and their corresponding labels to be used in the competition.

4.2. Attacking Strategies

We explore traditional whitebox attacking strategies, PGD, CW, MIM and EAD, as well as adaptive strategies, R+PGD, BPDA and EOT. We also include blackbox attacking strategies in our evaluation – Boundary Attack, SaltnPepper and Transfer Attack. We also explore possible attacking methods suitable against our proposed MALADE by combining BPDA and EOT to counter the randomness and significant dispersion inherent in MALADE. We first make sure that the attacking methods are applied with appropriate parameter setting and pay careful attention to the *fairness* of the presented evaluation [59].

4.3. Baseline Defense Strategies

We choose the following state-of-the-art methods for comparison against MALADE,

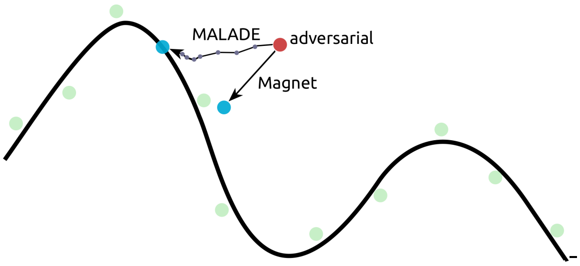


Figure 4: Magnet vs MALADE: Magnet [28] is a projection method which makes use of an autoencoder as a preprocessing step. MALADE on the other hand, is a random sampling algorithm which uses an sDAE only for estimating the conditional gradient at every time step. The significant dispersion of MALADE makes it hard for attackers to accurately align adversarial samples to targeted spots.

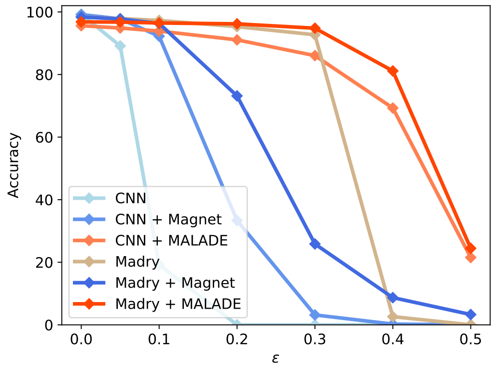


Figure 5: Classification accuracy against BPDA with EOT on MNIST. Madry breaks down for adversarial samples which have a perturbation higher than 0.3. MALADE boosts the robustness when applied to either CNN or Madry, much more than Magnet does.

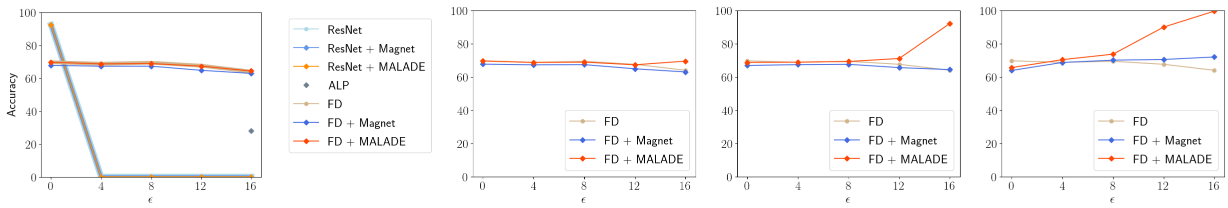
Madry [17] and FD [20] (adversarial training) Madry and FD are the state-of-the-art adversarial training methods, which showed best performance, respectively, on MNIST and on ImageNet. As discussed in Section 2.2, both are considered to be essentially the same method. We used a pretrained Madry model[9] in the MNIST experiment, and a pretrained FD model[10] in the ImageNet experiment.

Magnet [28] Magnet can be considered as a special case of MALA – with the step size $\alpha = \sigma^2$ and the number of steps $N = 1$ with no Gaussian noise added (see Figure 4). Similar to MALADE, Magnet can be applied to adversarial training methods, possibly providing state-of-the-art baselines.

4.4. Results on MNIST

We first show our extensive experiments on MNIST. Table 2 shows classification accuracy of the original convolutional neural network (CNN) classifier, the CNN classifier protected by MALADE (CNN+MALADE), the Madry classifier, i.e., the classifier trained with adversarial samples, and the Madry classifier protected by MALADE (Madry+MALADE). Each classifier was attacked with several strategies including different L_p-norms. The Madry classifier was trained on the adversarial samples created by PGD-L_∞ for $\varepsilon = 0.3$. As expected, Madry is robust against PGD-L_∞ up to $\varepsilon \leq 0.3$. Consistently with the author's report [17], it is also robust against PGD-L_2 up to $\varepsilon \leq 4.5$. However, Madry is broken down by PGD-L_∞ for $\varepsilon > 0.4$ and PGD-L_2 for $\varepsilon > 6$. Furthermore, EAD method completely breaks down Madry. On the other hand, our proposed MALADE is robust against PGD-L_∞ and PGD-L_2 in a wide range of ε, and is not completely broken down by EAD.

[9] `https://github.com/MadryLab/mnist_challenge`
[10] `https://github.com/facebookresearch/ImageNet-Adversarial-Training`



(a) FPR = 0% (b) FPR = 0.1% (c) FPR = 1% (d) FPR = 5%

Figure 6: Classification accuracy on ImageNet. The panel (a) shows pure defense performance, while (b)–(d) show defense with detection performance for different false positive rate. While the pure (FPR=0%) defense strategy with MALADE shows a marginal gain, combining detection with defense (FPR>0%) significantly improves the performance. The plot for ALP in the panel (a) indicates the reported value in ALP [18], which outperformed all previous classifiers on ImageNet but was later shown to be outperformed by FD.

We also investigated the robustness against attacks adapted to MALADE, i.e., R, BPDA, EOT, and their combinations (see Section 2.1.2). We see in Table 2 that the elaborated attacks reduce the accuracy of Madry+MALADE to some extent, but their effect is limited. Table 2 also shows results under the blackbox scenario with the state-of-the-art attacking strategies – SaltnPepper, Boundary attack and Transfer Attacks. The table clearly shows high robustness of MALADE against those attacks, while Madry exhibits vulnerability against them. Especially, for larger ε and different norm bounds, Madry+MALADE, as well as CNN+MALADE, significantly outperforms Madry and CNN.

Figure 5 depicts the classification accuracy as a function of ε against BPDA with EOT, the most effective attack against MALADE. We see that MALADE applied both to CNN and Madry significantly improves the performance, for high epsilons. Figure 5 also shows the performance of Magnet applied to CNN (CNN + Magnet), as well as Madry (Madry + Magnet), as other baseline methods. We see that Magnet fails to improve the performance of Madry, and Madry without MALADE breaks down for perturbations higher than 0.3.

4.5. Results on ImageNet

Next we show experimental results on ImageNet. Figure 6a shows the defense accuracy on ImageNet in the same format as in Figure 5: the curves corresponds to the plain classifier (ResNet152) and the adversarially trained classifier (FD) with and without Magnet and MALADE. Unfortunately, ResNet152 is completely broken down (the accuracy is zero for $\epsilon > 0$) and neither Magnet nor MALADE improve the performance. It is known that, when the data space is high dimensional as of ImageNet, the decision boundary of normal classifiers (that are not trained with adversarial samples) tend to be highly complicated and have large untrained spots [8, 15, 60, 61, 62, 63]. Consequently, there is no existing projection method that can protect them, to the best of our knowledge.

On the other hand, adversarially trained classifiers such as FD tend to have smoother decision boundary [63, 64, 65, 66, 67]. We observe that FD is not broken down, and MALADE contributes to improve the robustness. The performance by MALADE is enhanced when it is combined with adversarial detection (see Section 3.5). We determine the threshold θ in Eq.(13) so that the false positive rate, i.e., the proportion of the original test samples identified as adversarial, is equal to target values. Figure 6 shows the classification accuracy for different false positive rates. Compared with FD and FD + Magnet, FD + MALADE shows significant improvement. Note that, in the scenario where defender is equipped with a detector, the attacker optimizes samples so that they fool both the detector and the defense systems, which can be achieved by minimizing the detection amplitude, i.e., the left-hand side of Eq.(13), in addition to the (negative) cross entropy loss.

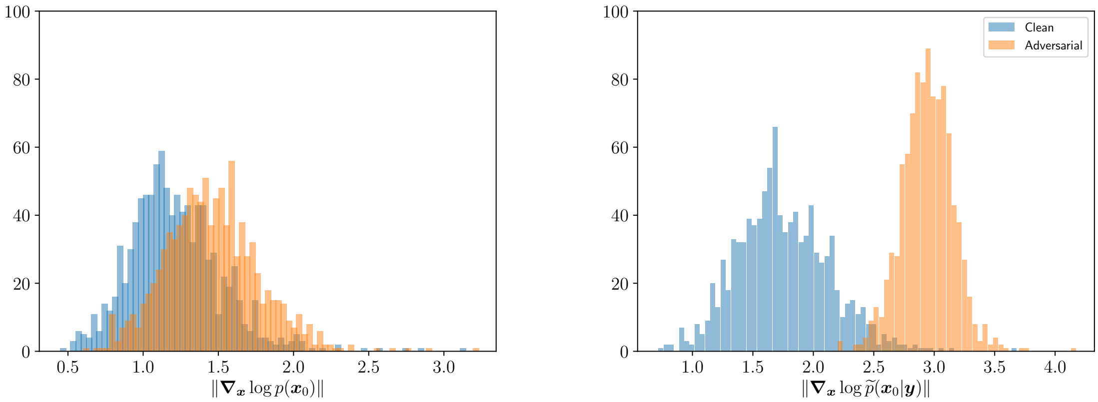
In Figures 6b–6d, the accuracy of FD + Magnet and FD + MALADE decreases on non-adversarial samples, i.e., at $\epsilon = 0$, as FPR increases. This is because we counted the false positive samples as "failures," and therefore the accuracy at $= 0$ is upper-bounded by $(1 - \mathrm{FPR})$. This amounts to treating the costs of the false positives and the false negatives equally. However, in the adversarial detection and defense scenario [28], the cost of false positives is considered to be much lower than the cost of the false negatives, because the system can ask for a new sample or human interaction for the non-adversarial samples detected as adversarial (false positives), while the adversarial samples that fool both the detector and the defended classifier (false negatives) can lead to serious consequences, e.g., fatal accidents resulting in injury or death in the self-driving application. To treat the false positives and false negatives separately, [28] proposed a definition of "correct decisions," where the false positive samples are not counted as "failures," and no accuracy drop at $\epsilon = 0$ would be observed. Although this definition is appropriate, we did not adopt it in Figures 6b–6d, because the definition makes the accuracy non-decreasing as FPR increases, and therefore makes the comparison unfair with FD that is not equipped with adversarial detection.

The contribution of adversarial detection to the final accuracy is indirect. When the defender uses a detector, the attacker needs to balance between fooling the classifier and hiding from the detector. As a result, the attacking samples are easier to detect, if the defense method is stronger. This way the difference in defence performance between Magnet and MALADE can be enhanced. Figure 7 shows the histograms of the score function norms of clean and adversarial samples, where the adversarial samples were generated against FD + Magnet (a) and against FD + MALADE (b), respectively. As observed, the separation between the clean and the adversarial samples is clear for FD + MALADE, which implies that fooling MALADE is harder, and adversarial samples against FD + MALADE are easier to detect.

5. Concluding Discussion

The threat of adversarial sample still remains an unresolved issue, even on a small toy dataset like MNIST. State-of-the-art robust methods do not scale well to larger data or models.

In this work, we have proposed to use the Metropolis-adjusted Langevin algorithm (MALA) which is guided through a supervised DAE—MALA for DEfense (MALADE). This framework allows us to drive adversarial samples towards the underlying data manifold and thus towards the high density regions of the data generating distribution, where the nonlinear learning machine is trained well with sufficient training data. In this process, the gradient is computed not based on the *marginal* input distribution but on the *conditional* input distribution given an output, and it is estimated by a supervised DAE with the weights for balancing reconstruction and supervision appropriately set. This prevents MALADE from driving samples into a neighboring cluster with a wrong label, and gives rise to high generalization performance



(a) FD + Magnet

(b) FD + MALADE

Figure 7: The histograms of the norm of the score function of the clean samples and the adversarial samples against FD + Magnet (a) and those against FD + MALADE (b). The separation is clearer in (b), because the attacker has to pull the sample far away from the data manifold to fool the FD + MALADE classifier than the FD + Magnet classifier. As a result, the adversarial samples against FD+MALADE are easier to detect than those against FD + Magnet.

that significantly reduces the effect of adversarial attacks. We have shown that the MALADE improves the robustness of the state-of-the-art methods in countering adversarial samples on not just small datasets – MNIST [28, 17] but also on larger datasets – ImageNet [20].

Let us briefly reflect on the fundamental changes that we have proposed in this work as they may hold value beyond their excellent practical results and may hold a wider applicability. First, broad projection of the samples by MALADE onto the high density region of the manifold adds further resilience against attacks into the picture as attackers cannot easily adapt to MALADE even in a whitebox scenario. Second establishing the conditional estimate through the sDAE helps use class information for projecting back into the relevant high density areas. Also this concept may serve as a blueprint for other estimators. Finally, decomposing defense into detection and Langevin dynamics steps turn out helpful. This is because attackers have to control two points distant from each other, in order to avoid from being detected at the input point and to fool the classifier after many Langevin steps.

Future work on MALADE includes stabilizing the prediction by majority voting from a collection of the generated samples after burn-in and in developing tools which estimate the gradient more accurately in high dimensional space. Other future work includes analyzing the attacks and defenses using interpretation methods [68, 69], and applying the supervised DAE to other applications such as federated or distributed learning [70, 71, 72].

Acknowledgments

KRM was supported in part by the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea Government (No. 2019-0-00079, Artificial Intelligence Graduate School Program, Korea University), and was partly supported by the German Ministry for Education and Research (BMBF) under Grants 01IS14013A-E, 01GQ1115, 01GQ0850, 01IS18025A, 031L0207D and 01IS18037A; the German Research Foundation (DFG) under Grant Math+, EXC 2046/1, Project ID 390685689. WS and SN acknowledge financial support by the German Ministry for Education and Research (BMBF) for the Berlin Institute for the Foundations of Learning and Data (BIFOLD) (ref. 01IS18037I and 01IS18025A).

References

[1] A. Krizhevsky, I. Sutskever, G. E. Hinton, Imagenet classification with deep convolutional neural networks, in: Advances in Neural Information Processing Systems, 2012, pp. 1097–1105.

[2] Y. Lecun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proceedings of the IEEE 86 (11) (1998) 2278–2324.

[3] C. Szegedy, Wei Liu, Yangqing Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, Going deeper with convolutions, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015, pp. 1–9.

[4] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, in: International Conference on Learning Representations (ICLR), 2015.

[5] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 770–778.

[6] I. J. Goodfellow, J. Shlens, C. Szegedy, Explaining and harnessing adversarial examples, in: International Conference on Learning Representations (ICLR), 2015.

[7] N. Papernot, P. D. McDaniel, I. J. Goodfellow, S. Jha, Z. B. Celik, A. Swami, Practical black-box attacks against machine learning, in: Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security, ACM, 2017, pp. 506–519.

[8] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. J. Goodfellow, R. Fergus, Intriguing properties of neural networks, in: International Conference on Learning Representations (ICLR), 2014.

[9] A. Nguyen, J. Yosinski, J. Clune, Deep neural networks are easily fooled: High confidence predictions for unrecognizable images, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015, pp. 427–436.

[10] I. Evtimov, K. Eykholt, E. Fernandes, T. Kohno, B. Li, A. Prakash, A. Rahmati, D. Song, Robust physical-world attacks on machine learning models, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018, pp. 1625–1634.

[11] A. Athalye, L. Engstrom, A. Ilyas, K. Kwok, Synthesizing robust adversarial examples, in: International Conference on Machine Learning (ICML), Vol. 80 of Proceedings of Machine Learning Research, PMLR, 2018, pp. 284–293.

[12] F. Tramèr, A. Kurakin, N. Papernot, D. Boneh, P. McDaniel, Ensemble adversarial training: Attacks and defenses, in: International Conference on Learning Representations (ICLR), 2018.

[13] N. Papernot, P. D. McDaniel, X. Wu, S. Jha, A. Swami, Distillation as a defense to adversarial perturbations against deep neural networks, in: IEEE Symposium on Security and Privacy, (SP), 2016, pp. 582–597.

[14] T. Strauss, M. Hanselmann, A. Junginger, H. Ulmer, Ensemble methods as a defense to adversarial perturbations against deep neural networks, arXiv preprint arXiv:1709.03423.

[15] S. Gu, L. Rigazio, Towards deep neural network architectures robust to adversarial examples, in: International Conference on Learning Representations (ICLR) Workshop Track Proceedings, 2015.

[16] A. Lamb, J. Binas, A. Goyal, D. Serdyuk, S. Subramanian, I. Mitliagkas, Y. Bengio, Fortified networks: Improving the robustness of deep networks by modeling the manifold of hidden representations, arXiv preprint arXiv:1804.02485.

[17] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, A. Vladu, Towards deep learning models resistant to adversarial attacks, in: International Conference on Learning Representations (ICLR), 2018.

[18] H. Kannan, A. Kurakin, I. Goodfellow, Adversarial logit pairing, arXiv preprint arXiv:1803.06373.

[19] X. Liu, Y. Li, C. Wu, C.-J. Hsieh, Adv-BNN: Improved adversarial defense through robust bayesian neural network, in: International Conference on Learning Representations (ICLR), 2018.

[20] C. Xie, Y. Wu, L. v. d. Maaten, A. L. Yuille, K. He, Feature denoising for improving adversarial robustness, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2019, pp. 501–509.

[21] L. Schott, J. Rauber, M. Bethge, W. Brendel, Towards the first adversarially robust neural network model on MNIST, in: International Conference on Learning Representations (ICLR), 2018.

[22] Y. Song, T. Kim, S. Nowozin, S. Ermon, N. Kushman, Pixeldefend: Leveraging generative models to understand and defend against adversarial examples, in: International Conference on Learning Representations (ICLR), 2018.

[23] P. Samangouei, M. Kabkab, R. Chellappa, Defense-GAN: Protecting classifiers against adversarial attacks using generative models, in: International Conference on Learning Representations (ICLR), 2018.

[24] A. Ilyas, A. Jalal, E. Asteri, C. Daskalakis, A. G. Dimakis, The robust manifold defense: Adversarial training using generative models, arXiv preprint arXiv:1712.09196.

[25] G. Jin, S. Shen, D. Zhang, F. Dai, Y. Zhang, APE-GAN: adversarial perturbation elimination with GAN, in: IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2019, pp. 3842–3846.

[26] C. Guo, M. Rana, M. Cissé, L. van der Maaten, Countering adversarial images using input transformations, in: International Conference on Learning Representations (ICLR), 2018.

[27] F. Liao, M. Liang, Y. Dong, T. Pang, X. Hu, J. Zhu, Defense against adversarial attacks using high-level representation guided denoiser, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018, pp. 1778–1787.

[28] D. Meng, H. Chen, MagNet: A two-pronged defense against adversarial examples, in: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS '17, Association for Computing Machinery, 2017, p. 135–147.

[29] C. Xie, J. Wang, Z. Zhang, Z. Ren, A. Yuille, Mitigating adversarial effects through randomization, in: International Conference on Learning Representations (ICLR), 2018.

[30] E. Wong, Z. Kolter, Provable defenses against adversarial examples via the convex outer adversarial polytope, in: International Conference on Machine Learning (ICML), Vol. 80 of Proceedings of Machine Learning Research, PMLR, 2018, pp. 5286–5295.

[31] E. Wong, F. Schmidt, J. H. Metzen, J. Z. Kolter, Scaling provable adversarial defenses, in: Advances in Neural Information Processing Systems, 2018, pp. 8400–8409.

[32] A. Raghunathan, J. Steinhardt, P. Liang, Certified defenses against adversarial examples, in: International Conference on Learning Representations (ICLR), 2018.

[33] K. Dvijotham, R. Stanforth, S. Gowal, T. A. Mann, P. Kohli, A dual approach to scalable verification of deep networks, in: Proceedings of the Thirty-Fourth Conference on Uncertainty in Artificial Intelligence, (UAI), 2018, pp. 550–559.

[34] G. O. Roberts, J. S. Rosenthal, Optimal scaling of discrete approximations to langevin diffusions, Journal of the Royal Statistical Society: Series B (Statistical Methodology) 60 (1) (1998) 255–268.

[35] G. O. Roberts, R. L. Tweedie, et al., Exponential convergence of langevin distributions and their discrete approximations, Bernoulli 2 (4) (1996) 341–363.

[36] G. Alain, Y. Bengio, What regularized auto-encoders learn from the data-generating distribution, The Journal of Machine Learning Research 15 (1) (2014) 3563–3593.

[37] P. Vincent, H. Larochelle, Y. Bengio, P. Manzagol, Extracting and composing robust features with denoising autoencoders, in: International Conference on Machine Learning (ICML), Vol. 307, ACM, 2008, pp. 1096–1103.

[38] J. Lee, E. Mansimov, K. Cho, Deterministic non-autoregressive neural sequence modeling by iterative refinement, Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (2018) 1173–1182.

[39] E. P. Lehman, R. G. Krishnan, X. Zhao, R. G. Mark, L.-w. H. Lehman, Representation learning approaches to detect false arrhythmia alarms from ecg dynamics, Vol. 85 of Proceedings of Machine Learning Research, PMLR, 2018, pp. 571–586.

[40] A. Athalye, N. Carlini, D. Wagner, Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples, in: International Conference on Machine Learning (ICML), Vol. 80 of Proceedings of Machine Learning Research, PMLR, 2018, pp. 274–283.

[41] N. Carlini, D. Wagner, Magnet and" efficient defenses against adversarial attacks" are not robust to adversarial examples, arXiv preprint arXiv:1711.08478.

[42] Y. Dong, F. Liao, T. Pang, H. Su, J. Zhu, X. Hu, J. Li, Boosting adversarial attacks with momentum, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018, pp. 9185–9193.

[43] N. Carlini, D. Wagner, Towards evaluating the robustness of neural networks, in: 2017 IEEE Symposium on Security and Privacy (SP), 2017, pp. 39–57.

[44] P. Chen, Y. Sharma, H. Zhang, J. Yi, C. Hsieh, EAD: elastic-net attacks to deep neural networks via adversarial examples, in: Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, 2018, pp. 10–17.

[45] Y. Sharma, P. Chen, Attacking the madry defense model with L_1-based adversarial examples, in: International Conference on Learning Representations (ICLR) Workshop, 2018.

[46] N. Frosst, S. Sabour, G. Hinton, DARCCC: Detecting adversaries by reconstruction from class conditional capsules, arXiv preprint arXiv:1811.06969.

[47] N. Papernot, P. McDaniel, I. Goodfellow, Transferability in machine learning: from phenomena to black-box attacks using adversarial samples, arXiv preprint arXiv:1605.07277.

[48] W. Brendel, J. Rauber, M. Bethge, Decision-based adversarial attacks: Reliable attacks against black-box machine learning models, in: International Conference on Learning Representations (ICLR), 2018.

[49] F. Tramèr, N. Papernot, I. Goodfellow, D. Boneh, P. McDaniel, The space of transferable adversarial examples, arXiv preprint arXiv:1704.03453.

[50] A. Kurakin, I. J. Goodfellow, S. Bengio, Adversarial machine learning at scale, in: International Conference on Learning Representations (ICLR), 2017.

[51] L. Engstrom, A. Ilyas, A. Athalye, Evaluating and understanding the robustness of adversarial logit pairing, arXiv preprint arXiv:1807.10272.

[52] A. Sinha, H. Namkoong, J. Duchi, Certifying some distributional robustness with principled adversarial training, in: International Conference on Learning Representations (ICLR), 2018.

[53] Y. Bengio, L. Yao, G. Alain, P. Vincent, Generalized denoising auto-encoders as generative models, in: Advances in Neural Information Processing Systems, 2013, pp. 899–907.

[54] Unknown, Learning by denoising part 2. connection between data distribution and denoising function (2016).
URL `http://www.aihelsinki.com/connection-to-g/`

[55] A. Nguyen, J. Clune, Y. Bengio, A. Dosovitskiy, J. Yosinski, Plug & play generative networks: Conditional iterative generation of images in latent space, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 4467–4477.

[56] S. Rifai, P. Vincent, X. Muller, X. Glorot, Y. Bengio, Contracting auto-encoders: Explicit invariance during feature extraction, in: International Conference on Machine Learning (ICML), 2011, pp. 833–840.

[57] S. Rifai, X. Muller, X. Glorot, G. Mesnil, Y. Bengio, P. Vincent, Learning invariant features through local space contraction, arXiv preprint arXiv:1104.4153.

[58] J. Deng, W. Dong, R. Socher, L. Li, K. Li, F. Li, Imagenet: A large-scale hierarchical image database, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2009, pp. 248–255.

[59] N. Carlini, A. Athalye, N. Papernot, W. Brendel, J. Rauber, D. Tsipras, I. Goodfellow, A. Madry, On evaluating adversarial robustness, arXiv preprint arXiv:1902.06705.

[60] A. Fawzi, S.-M. Moosavi-Dezfooli, P. Frossard, Robustness of classifiers: from adversarial to random noise, in: Advances in Neural Information Processing Systems, 2016, pp. 1632–1640.

[61] A.-K. Dombrowski, M. Alber, C. Anders, M. Ackermann, K.-R. Müller, P. Kessel, Explanations can be manipulated and geometry is to blame, in: Advances in Neural Information Processing Systems, 2019, pp. 13589–13600.

[62] G. Elsayed, D. Krishnan, H. Mobahi, K. Regan, S. Bengio, Large margin deep networks for classification, in: Advances in Neural Information Processing Systems, 2018, pp. 842–852.

[63] S.-M. Moosavi-Dezfooli, A. Fawzi, J. Uesato, P. Frossard, Robustness via curvature regularization, and vice versa, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2019, pp. 9078–9086.

[64] S.-M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, P. Frossard, S. Soatto, Robustness of classifiers to universal perturbations: A geometric perspective, in: International Conference on Learning Representations (ICLR), 2018.

[65] T. Miyato, S. Maeda, M. Koyama, K. Nakae, S. Ishii, Distributional smoothing with virtual adversarial training, in: International Conference on Learning Representations (ICLR), 2016.

[66] Y. Yang, R. Khanna, Y. Yu, A. Gholami, K. Keutzer, J. E. Gonzalez, K. Ramchandran, M. W. Mahoney, Boundary thickness and robustness in learning models, arXiv preprint arXiv:2007.05086.

[67] H. Zhang, Y. Yu, J. Jiao, E. Xing, L. E. Ghaoui, M. Jordan, Theoretically principled trade-off between robustness and accuracy, in: International Conference on Machine Learning (ICML), Vol. 97 of Proceedings of Machine Learning Research, PMLR, 2019, pp. 7472–7482.

[68] G. Montavon, W. Samek, K.-R. Müller, Methods for interpreting and understanding deep neural networks, Digital Signal Processing 73 (2018) 1–15.

[69] S. Lapuschkin, S. Wäldchen, A. Binder, G. Montavon, W. Samek, K.-R. Müller, Unmasking clever hans predictors and assessing what machines really learn, Nature communications 10 (1) (2019) 1096.

[70] B. McMahan, E. Moore, D. Ramage, S. Hampson, B. A. y Arcas, Communication-efficient learning of deep networks from decentralized data, in: Artificial Intelligence and Statistics, Vol. 54 of Proceedings of Machine Learning Research, PMLR, 2017, pp. 1273–1282.

[71] F. Sattler, S. Wiedemann, K.-R. Müller, W. Samek, Sparse binary compression: Towards distributed deep learning with minimal communication, in: Proceedings of the IEEE International Joint Conference on Neural Networks (IJCNN), 2019, pp. 1–8.

[72] F. Sattler, S. Wiedemann, K.-R. Müller, W. Samek, Robust and communication-efficient federated learning from non-iid data, IEEE Transactions on Neural Networks and Learning Systems 31 (9) (2020) 772–785.

[73] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al., Tensorflow: a system for large-scale machine learning, in: Proceedings of the 12th USENIX conference on Operating Systems Design and Implementation (OSDI), Vol. 16, 2016, pp. 265–283.

[74] M. Girolami, B. Calderhead, Riemann manifold langevin and hamiltonian monte carlo methods, Journal of the Royal Statistical Society: Series B (Statistical Methodology) 73 (2) (2011) 123–214.

[75] S. Jégou, M. Drozdzal, D. Vazquez, A. Romero, Y. Bengio, The one hundred layers tiramisu: Fully convolutional densenets for semantic segmentation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, 2017, pp. 11–19.

Appendix A. Proof of Theorem 1

sDAE is trained so that the following functional is minimized with respect to the function $\boldsymbol{r}:\mathbb{R}^L \mapsto \mathbb{R}^L$:

$$\mathbb{E}_{p'(\boldsymbol{x},\boldsymbol{y})p'(\boldsymbol{\nu})}\left[\|\boldsymbol{r}(\boldsymbol{x}+\boldsymbol{\nu})-\boldsymbol{x}\|^2+2\sigma^2 J\left(\boldsymbol{r}(\boldsymbol{x}+\boldsymbol{\nu}),\boldsymbol{y}\right)\right], \tag{A.1}$$

which is a finite sample approximation to the true objective

$$g(\boldsymbol{r})=\int\left(\|\boldsymbol{r}(\boldsymbol{x}+\boldsymbol{\nu})-\boldsymbol{x}\|^2-2\sigma^2\mathbb{E}_{p(\boldsymbol{y}|\boldsymbol{x})}\left[\boldsymbol{y}^\top\log\widehat{\boldsymbol{y}}(\boldsymbol{r}(\boldsymbol{x}+\boldsymbol{\nu}))\right]\right)p(\boldsymbol{x})\mathcal{N}_L(\boldsymbol{\nu};\boldsymbol{0},\sigma^2\boldsymbol{I})d\boldsymbol{x}d\boldsymbol{\nu}. \tag{A.2}$$

For small σ^2, the Taylor expansion of the l-th component of $\boldsymbol{r}$ around $\boldsymbol{x}$ gives

$$r_l(\boldsymbol{x}+\boldsymbol{\nu})=r_l(\boldsymbol{x})+\boldsymbol{\nu}^\top\frac{\partial r_l}{\partial\boldsymbol{x}}+\frac{1}{2}\boldsymbol{\nu}^\top\frac{\partial^2 r_l}{\partial\boldsymbol{x}\partial\boldsymbol{x}}\boldsymbol{\nu}+O(\sigma^3),$$

where $\frac{\partial^2 f}{\partial\boldsymbol{x}\partial\boldsymbol{x}}$ is the Hessian of a function $f(\boldsymbol{x})$. Substituting this into Eq.(A.2), we have

$$\begin{aligned}
g(\boldsymbol{r})&=\int\Bigg\{\sum_{l=1}^{L}\left(r_l(\boldsymbol{x})+\boldsymbol{\nu}^\top\frac{\partial r_l}{\partial\boldsymbol{x}}+\frac{1}{2}\boldsymbol{\nu}^\top\frac{\partial^2 r_l}{\partial\boldsymbol{x}\partial\boldsymbol{x}}\boldsymbol{\nu}-x_l\right)^2\\
&\qquad\qquad-2\sigma^2\mathbb{E}_{p(\boldsymbol{y}|\boldsymbol{x})}\left[\boldsymbol{y}^\top\log\widehat{\boldsymbol{y}}(\boldsymbol{r}(\boldsymbol{x}))\right]\Bigg\}p(\boldsymbol{x})d\boldsymbol{x}\mathcal{N}_L(\boldsymbol{\nu};\boldsymbol{0},\sigma^2\boldsymbol{I})d\boldsymbol{\nu}+O(\sigma^3)\\
&=\int\Bigg\{\sum_{l=1}^{L}\left((r_l(\boldsymbol{x})-x_l)^2+(r_l(\boldsymbol{x})-x_l)\,\boldsymbol{\nu}^\top\frac{\partial^2 r_l}{\partial\boldsymbol{x}\partial\boldsymbol{x}}\boldsymbol{\nu}+\frac{\partial r_l}{\partial\boldsymbol{x}}^\top\boldsymbol{\nu}\boldsymbol{\nu}^\top\frac{\partial r_l}{\partial\boldsymbol{x}}\right)\\
&\qquad\qquad-2\sigma^2\mathbb{E}_{p(\boldsymbol{y}|\boldsymbol{x})}\left[\boldsymbol{y}^\top\log\widehat{\boldsymbol{y}}(\boldsymbol{r}(\boldsymbol{x}))\right]\Bigg\}p(\boldsymbol{x})d\boldsymbol{x}\mathcal{N}_L(\boldsymbol{\nu};\boldsymbol{0},\sigma^2\boldsymbol{I})d\boldsymbol{\nu}+O(\sigma^3)\\
&=\int\Bigg\{\sum_{l=1}^{L}\left((r_l(\boldsymbol{x})-x_l)^2+\sigma^2\,(r_l(\boldsymbol{x})-x_l)\operatorname{tr}\left(\frac{\partial^2 r_l}{\partial\boldsymbol{x}\partial\boldsymbol{x}}\right)+\sigma^2\left\|\frac{\partial r_l}{\partial\boldsymbol{x}}\right\|^2\right)\\
&\qquad\qquad-2\sigma^2\mathbb{E}_{p(\boldsymbol{y}|\boldsymbol{x})}\left[\boldsymbol{y}^\top\log\widehat{\boldsymbol{y}}(\boldsymbol{r}(\boldsymbol{x}))\right]\Bigg\}p(\boldsymbol{x})d\boldsymbol{x}+O(\sigma^3).
\end{aligned} \tag{A.3}$$

Thus, the objective functional Eq.(A.2) can be written as

$$g(\boldsymbol{r})=\int G d\boldsymbol{x}+O(\sigma^3), \tag{A.4}$$

where

$$\begin{aligned}
G=\Bigg\{\sum_{l=1}^{L}&\left(\left(r_l(\boldsymbol{x})-x_l+\sigma^2\operatorname{tr}\left(\frac{\partial^2 r_l}{\partial\boldsymbol{x}\partial\boldsymbol{x}}\right)\right)(r_l(\boldsymbol{x})-x_l)+\sigma^2\left\|\frac{\partial r_l}{\partial\boldsymbol{x}}\right\|^2\right)\\
&\qquad-2\sigma^2\mathbb{E}_{p(\boldsymbol{y}|\boldsymbol{x})}\left[\boldsymbol{y}^\top\log\widehat{\boldsymbol{y}}(\boldsymbol{r}(\boldsymbol{x}))\right]\Bigg\}p(\boldsymbol{x}).
\end{aligned} \tag{A.5}$$

We can find the optimal function minimizing the functional Eq.(A.4) by using *calculus of variations*. The optimal function satisfies the following Euler-Lagrange equation: for each $l=1,\dots,L$,

$$\frac{\partial G}{\partial r_l}-\sum_{m=1}^{L}\frac{\partial}{\partial x_m}\frac{\partial G}{\partial(\boldsymbol{r}'_l)_m}+\sum_{m=1}^{L}\sum_{m'=m+1}^{L}\frac{\partial^2}{\partial x_m\partial x_{m'}}\frac{\partial G}{\partial(\boldsymbol{R}''_l)_{m,m'}}=0, \tag{A.6}$$

where $\boldsymbol{r}'_l=\frac{\partial r_l}{\partial\boldsymbol{x}}\in\mathbb{R}^L$ is the gradient (of r_l with respect to $\boldsymbol{x}$) and $\boldsymbol{R}''_l=\frac{\partial r_l}{\partial\boldsymbol{x}\partial\boldsymbol{x}}\in\mathbb{R}^{L\times L}$ is the Hessian.

We have

$$\frac{\partial G}{\partial r_l} = \left\{ 2\left(r_l(\boldsymbol{x}) - x_l\right) + \sigma^2 \mathrm{tr}\left(\frac{\partial^2 r_l}{\partial \boldsymbol{x} \partial \boldsymbol{x}}\right) - 2\sigma^2 \frac{\partial \mathbb{E}_{p(\boldsymbol{y}|\boldsymbol{x})}\left[\boldsymbol{y}^\top \log \widehat{\boldsymbol{y}}(\boldsymbol{r}(\boldsymbol{x}))\right]}{\partial r_l} \right\} p(\boldsymbol{x}),$$

$$\frac{\partial G}{\partial (\boldsymbol{r}_l')_m} = 2\sigma^2 \frac{\partial r_l}{\partial x_m} p(\boldsymbol{x}),$$

$$\frac{\partial G}{\partial (\boldsymbol{R}_l'')_{m,m'}} = \delta_{m,m'} \sigma^2 \left(r_l(\boldsymbol{x}) - x_l\right) p(\boldsymbol{x}),$$

and therefore

$$\frac{\partial}{\partial x_m} \frac{\partial G}{\partial (\boldsymbol{r}_l')_m} = 2\sigma^2 \left(\frac{\partial^2 r_l}{\partial x_m^2} p(\boldsymbol{x}) + \frac{\partial r_l}{\partial x_m} \frac{\partial p(\boldsymbol{x})}{\partial x_m} \right),$$

$$\begin{aligned}\frac{\partial^2}{\partial x_m \partial x_{m'}} \frac{\partial G}{\partial (\boldsymbol{R}_l'')_{m,m'}} &= \sigma^2 \delta_{m,m'} \frac{\partial}{\partial x_m} \left(\left(\frac{\partial r_l}{\partial x_{m'}} - \delta_{l,m'} \right) p(\boldsymbol{x}) + \left(r_l(\boldsymbol{x}) - x_l\right) \frac{\partial p(\boldsymbol{x})}{\partial x_{m'}} \right) \\ &= \sigma^2 \delta_{m,m'} \left(\frac{\partial^2 r_l}{\partial x_m^2} p(\boldsymbol{x}) + 2\left(\frac{\partial r_l}{\partial x_m} - \delta_{l,m} \right) \frac{\partial p(\boldsymbol{x})}{\partial x_m} + \left(r_l(\boldsymbol{x}) - x_l\right) \frac{\partial^2 p(\boldsymbol{x})}{\partial x_m^2} \right),\end{aligned}$$

where $\delta_{m,m'}$ is the Kronecker delta. Substituting the above into Eq.(A.6), we have

$$\begin{aligned}&\left\{ 2\left(r_l(\boldsymbol{x}) - x_l\right) + \sigma^2 \mathrm{tr}\left(\tfrac{\partial^2 r_l}{\partial \boldsymbol{x} \partial \boldsymbol{x}}\right) - 2\sigma^2 \tfrac{\partial \mathbb{E}_{p(\boldsymbol{y}|\boldsymbol{x})}\left[\boldsymbol{y}^\top \log \widehat{\boldsymbol{y}}(\boldsymbol{r}(\boldsymbol{x}))\right]}{\partial r_l} \right\} p(\boldsymbol{x}) \\ &\quad + \sigma^2 \textstyle\sum_{m=1}^L \sum_{m'=m+1}^L \delta_{m,m'} \left(\tfrac{\partial^2 r_l}{\partial x_m^2} p(\boldsymbol{x}) + 2\left(\tfrac{\partial r_l}{\partial x_m} - \delta_{l,m} \right) \tfrac{\partial p(\boldsymbol{x})}{\partial x_m} + \left(r_l(\boldsymbol{x}) - x_l\right) \tfrac{\partial^2 p(\boldsymbol{x})}{\partial x_m^2} \right) \\ &\qquad - 2\sigma^2 \textstyle\sum_{m=1}^L \left(\tfrac{\partial^2 r_l}{\partial x_m^2} p(\boldsymbol{x}) + \tfrac{\partial r_l}{\partial x_m} \tfrac{\partial p(\boldsymbol{x})}{\partial x_m} \right) = 0,\end{aligned}$$

and therefore

$$\left(r_l(\boldsymbol{x}) - x_l - \sigma^2 \frac{\partial \mathbb{E}_{p(\boldsymbol{y}|\boldsymbol{x})}\left[\boldsymbol{y}^\top \log \widehat{\boldsymbol{y}}(\boldsymbol{r}(\boldsymbol{x}))\right]}{\partial r_l} \right) \left(1 + \frac{\sigma^2}{2p(\boldsymbol{x})} \mathrm{tr}\left(\frac{\partial^2 p(\boldsymbol{x})}{\partial \boldsymbol{x} \partial \boldsymbol{x}} \right) \right) - \sigma^2 \frac{1}{p(\boldsymbol{x})} \frac{\partial p(\boldsymbol{x})}{\partial x_l} = 0. \quad \text{(A.7)}$$

It holds that

$$\frac{\partial \log p(\boldsymbol{x})}{\partial x_m} = \frac{1}{p(\boldsymbol{x})} \frac{\partial p(\boldsymbol{x})}{\partial x_m}, \quad \text{(A.8)}$$

$$\begin{aligned}\frac{\partial^2 \log p(\boldsymbol{x})}{\partial x_m \partial x_{m'}} &= \frac{\partial}{\partial x_{m'}} \left(\frac{1}{p(\boldsymbol{x})} \frac{\partial p(\boldsymbol{x})}{\partial x_m} \right) \\ &= -\frac{\partial \log p(\boldsymbol{x})}{\partial x_{m'}} \frac{\partial \log p(\boldsymbol{x})}{\partial x_m} + \frac{1}{p(\boldsymbol{x})} \frac{\partial^2 p(\boldsymbol{x})}{\partial x_m \partial x_{m'}}, \quad \text{(A.9)}\end{aligned}$$

$$\begin{aligned}\frac{\partial \mathbb{E}_{p(\boldsymbol{y}|\boldsymbol{x})}\left[\boldsymbol{y}^\top \log \widehat{\boldsymbol{y}}(\boldsymbol{r}(\boldsymbol{x}))\right]}{\partial r_l} &= \frac{\partial \sum_{k=1}^K p(y_k|\boldsymbol{x}) \log \widehat{y}_k(\boldsymbol{r}(\boldsymbol{x}))}{\partial r_l} \\ &= \textstyle\sum_{k=1}^K p(y_k|\boldsymbol{x}) \dfrac{\partial \log \widehat{y}_k(\boldsymbol{r}(\boldsymbol{x}))}{\partial r_l} \\ &= \textstyle\sum_{k=1}^K p(y_k|\boldsymbol{x}) \dfrac{\partial \log \widehat{y}_k(\boldsymbol{x})}{\partial x_l}\Bigg|_{\boldsymbol{x}=\boldsymbol{r}(\boldsymbol{x})} \\ &= \mathbb{E}_{p(\boldsymbol{y}|\boldsymbol{x})}\left[\boldsymbol{y}^\top \frac{\partial \log \widehat{\boldsymbol{y}}(\boldsymbol{x})}{\partial x_l}\Bigg|_{\boldsymbol{x}=\boldsymbol{r}(\boldsymbol{x})} \right]. \quad \text{(A.10)}\end{aligned}$$

By substituting Eqs.(A.8)–(A.10) into Eq.(A.7), we have

$$\begin{aligned}&\left(r_l(\boldsymbol{x}) - x_l - \sigma^2 \mathbb{E}_{p(\boldsymbol{y}|\boldsymbol{x})}\left[\boldsymbol{y}^\top \frac{\partial \log \widehat{\boldsymbol{y}}(\boldsymbol{x})}{\partial x_l}\Bigg|_{\boldsymbol{x}=\boldsymbol{r}(\boldsymbol{x})} \right] \right) \left(1 + \frac{\sigma^2}{2} \left(\mathrm{tr}\left(\frac{\partial^2 \log p(\boldsymbol{x})}{\partial \boldsymbol{x} \partial \boldsymbol{x}} \right) + \left\| \frac{\partial \log p(\boldsymbol{x})}{\partial \boldsymbol{x}} \right\|^2 \right) \right) \\ &\qquad - \sigma^2 \tfrac{\partial \log p(\boldsymbol{x})}{\partial x_l} = 0,\end{aligned}$$

and therefore

$$r_l(\boldsymbol{x}) - x_l = \sigma^2 \mathbb{E}_{p(\boldsymbol{y}|\boldsymbol{x})} \left[\boldsymbol{y}^\top \frac{\partial \log \widehat{\boldsymbol{y}}(\boldsymbol{x})}{\partial x_l} \Bigg|_{\boldsymbol{x}=\boldsymbol{r}(\boldsymbol{x})} \right]$$
$$+ \sigma^2 \frac{\partial \log p(\boldsymbol{x})}{\partial x_l} \left(1 + \frac{\sigma^2}{2} \left(\mathrm{tr} \left(\frac{\partial^2 \log p(\boldsymbol{x})}{\partial \boldsymbol{x} \partial \boldsymbol{x}} \right) + \left\| \frac{\partial \log p(\boldsymbol{x})}{\partial \boldsymbol{x}} \right\|^2 \right) \right)^{-1}$$
$$= \sigma^2 \mathbb{E}_{p(\boldsymbol{y}|\boldsymbol{x})} \left[\boldsymbol{y}^\top \frac{\partial \log \widehat{\boldsymbol{y}}(\boldsymbol{x})}{\partial x_l} \Bigg|_{\boldsymbol{x}=\boldsymbol{r}(\boldsymbol{x})} \right] + \sigma^2 \frac{\partial \log p(\boldsymbol{x})}{\partial x_l} + O(\sigma^4).$$

Taking the asymptotic term in Eq.(A.4) into account, we have

$$r_l(\boldsymbol{x}) - x_l = \sigma^2 \mathbb{E}_{p(\boldsymbol{y}|\boldsymbol{x})} \left[\boldsymbol{y}^\top \frac{\partial \log \widehat{\boldsymbol{y}}(\boldsymbol{x})}{\partial x_l} \Bigg|_{\boldsymbol{x}=\boldsymbol{r}(\boldsymbol{x})} \right] + \sigma^2 \frac{\partial \log p(\boldsymbol{x})}{\partial x_l} + O(\sigma^3),$$

which implies that $\boldsymbol{r}(\boldsymbol{x}) = \boldsymbol{x} + O(\sigma^2)$. Thus, we conclude that

$$r_l(\boldsymbol{x}) - x_l = \sigma^2 \mathbb{E}_{p(\boldsymbol{y}|\boldsymbol{x})} \left[\frac{\partial \log \widetilde{p}\,(\boldsymbol{y}|\boldsymbol{x})}{\partial x_l} \right] + \sigma^2 \frac{\partial \log p(\boldsymbol{x})}{\partial x_l} + O(\sigma^3)$$
$$= \sigma^2 \mathbb{E}_{p(\boldsymbol{y}|\boldsymbol{x})} \left[\frac{\partial}{\partial x_l} \log \widetilde{p}(\boldsymbol{x}, \boldsymbol{y}) \right] + O(\sigma^3), \tag{A.11}$$

where we used

$$\mathbb{E}_{p(\boldsymbol{y}|\boldsymbol{x})} \left[\boldsymbol{y}^\top \frac{\partial \log \widehat{\boldsymbol{y}}(\boldsymbol{x})}{\partial x_l} \Bigg|_{\boldsymbol{x}=\boldsymbol{r}(\boldsymbol{x})} \right] = \mathbb{E}_{p(\boldsymbol{y}|\boldsymbol{x})} \left[\boldsymbol{y}^\top \frac{\partial \log \widehat{\boldsymbol{y}}(\boldsymbol{x})}{\partial x_l} \right] + O(\sigma^2)$$
$$= \mathbb{E}_{p(\boldsymbol{y}|\boldsymbol{x})} \left[\frac{\partial \log \boldsymbol{y}^\top \widehat{\boldsymbol{y}}(\boldsymbol{x})}{\partial x_l} \right] + O(\sigma^2),$$

and the notation in Eq.(9). This completes the proof of Theorem 1.

Appendix B. Implementation Details

We implemented our attacks with the help of repositories including *cleverhans*[11], Madry[12], Magnet[13], FD[14] among others. The code was written in Tensorflow [73]

Appendix B.1. Hyper-parameter Settings for the Attacks

Appendix B.1.1. PGD

PGD attack was an untargeted attack as it is the simplest strategy for attacking. Learning rate of 0.01 was found to provide for a strong attack while number of iterations was tested for different values and fixed at N = 1000. Similar hyper-parameters were used for MIM attack.

Appendix B.1.2. BPDAwEOT

The attack strategy against MALADE was tested for its effectiveness by varying the number of steps of EOT to be computed. N = 30 was found to be sufficient for optimal convergence of the attack.

[11] https://github.com/tensorflow/cleverhans
[12] https://github.com/MadryLab/mnist_challenge
[13] https://github.com/carlini/MagNet
[14] https://github.com/facebookresearch/ImageNet-Adversarial-Training

Appendix B.1.3. CW

Hyper-parameters tuned include learning rate = 0.1 and number of iterations N = 1000, initial constant $c = 100$, binary search step = 1 and confidence = 0. The optimizer used here was Adam optimizer.

Appendix B.1.4. EAD

Hyper-parameters tuned for attacking MALADE include learning rate = 0.01 and number of iterations N = 100, initial constant $c = 0.01$, binary search step = 9 and confidence = 0. Increasing the number of iterations did not show any increase in the strength of the attack. On the other hand, increasing the number of iterations proved useful for attacking [17], although with all the hyper-parameters being the same. The Adam optimizer was investigated for this attack as recommended by [44, 45], however Gradient Descent optimizer proved better in the convergence of the attack.

Appendix B.2. Score Function Estimation

Appendix B.2.1. Training sDAE

The score function $\nabla_{\boldsymbol{x}} \log p(\boldsymbol{x})$ provided by DAE is dependent on the noise σ^2 added to the input while training the DAE. While too small values for σ^2 make the score function highly unstable, too large values blur the score. The same is true for the score function $\nabla_{\boldsymbol{x}} \log p(\boldsymbol{x}|\boldsymbol{y})$ provided by MALADE. Here in our experiments on MNIST, we trained the DAE as well as the sDAE with $\sigma^2 = 0.15$. Such a large noise is beneficial for reliable estimation of the score function [55]. On the other hand, for ImageNet, we used $\sigma^2 = 0.1$.

Appendix B.2.2. Step Size for Malade

The score function provided by MALADE drives the generated sample towards high density regions in the data generating distributions. With the direction provided by the score function, α controls the distance to move agt each step. With large α, there is possibility of jumping out of the data manifold. While annealing α and δ^2 would provides best results as the samples move towards high density region [34, 74]. In our experiments, we train the sDAE (or DAE) first, followed by searching for good parameters for the α and number of steps on the training or validation set. This reasonable procedure allows for manually finetuning the step size and number of steps based on the difficulty of the dataset, such that they samples are driven to the nearest high density region of the correct label in polynomial time. In the case of adversarial robustness, it also important to select the hyperparameters such that the off-manifold adversarial examples are returned to the data manifold of the correct label in the given number of steps. These parameters are then fixed and evaluated on the test set for each of the datasets.

Appendix B.3. Model Architecture

Appendix B.3.1. Classifier and DAE (and sDAE) Architectures

On MNIST, we used a classifier with two convolution layers and two fully connected layers for the CNN architecture. The classifier was trained from scratch. For Madry, we used the code provided on the

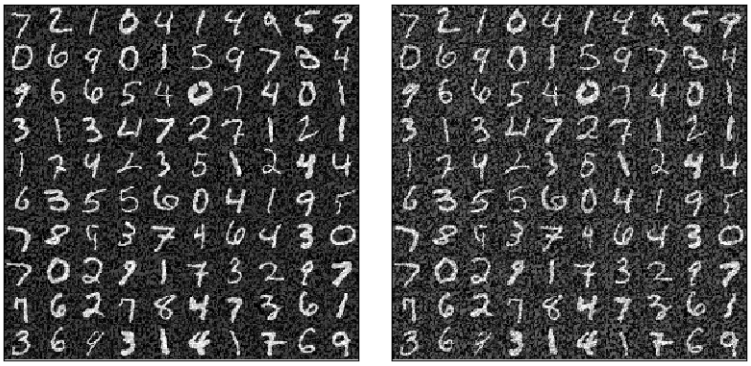
GitHub page[15]. The sDAE (and DAE) for MNIST had two convolution layers on the encoder along with two deconvolution layers and one final convolution layer on the decoder side. The same architecture was used for training the autoencoder used in Magnet.

FD classifier and pretrained weights were used as per the instructions provided on the GitHub Page[16]. For sDAE as well as for the autoencoder in Magnet, we used a Tiramisu network [75].

[15]https://github.com/MadryLab/mnist_challenge

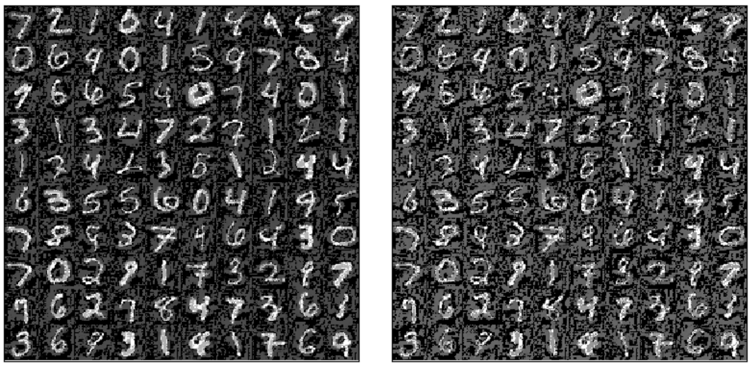
[16]https://github.com/facebookresearch/ImageNet-Adversarial-Training

Appendix C. Adversarial Samples



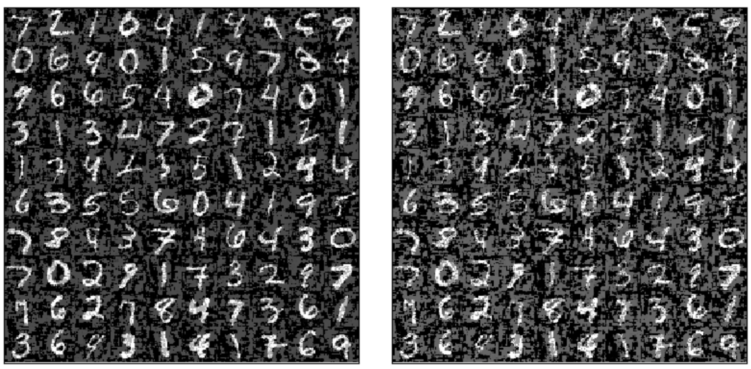
(a) $\varepsilon = 0.3$ (b) $\varepsilon = 0.4$

Figure C.8: Sample adversarial images crafted by a PGD attack with L_∞ norm are shown here.



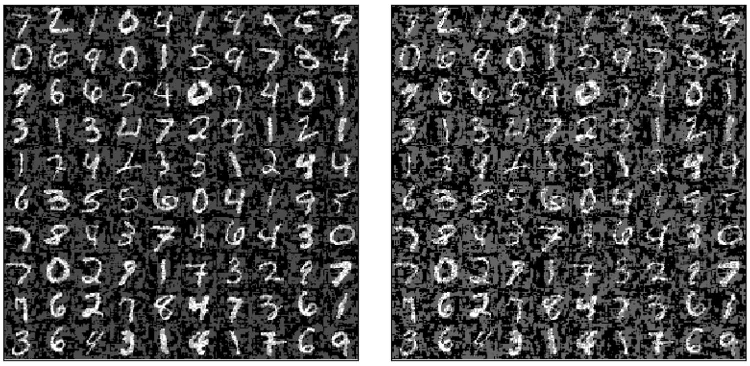
(a) $\varepsilon = 0.3$ (b) $\varepsilon = 0.4$

Figure C.9: Sample adversarial images crafted by R+PGD attack with L_∞ norm are shown here.



(a) $\varepsilon = 0.3$ (b) $\varepsilon = 0.4$

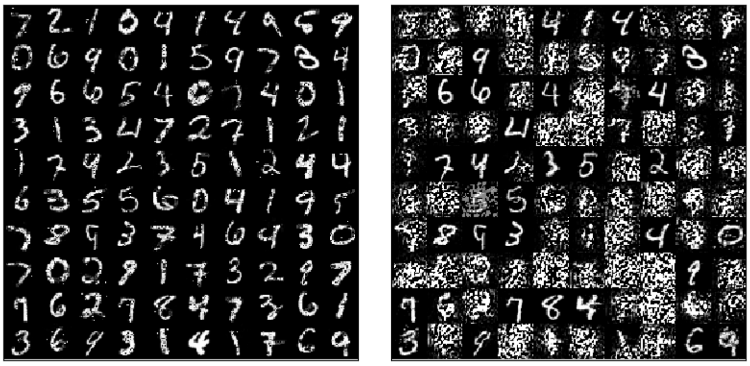
Figure C.10: Sample adversarial images crafted by a BPDA attack with L_∞ norm are shown here.



(a) $\varepsilon = 0.3$

(b) $\varepsilon = 0.4$

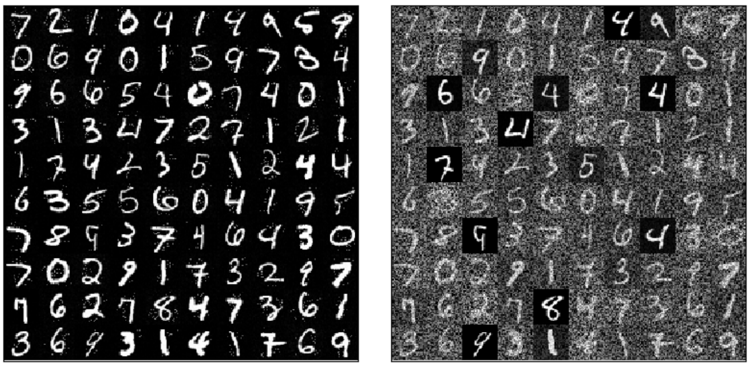
Figure C.11: Sample adversarial images crafted by a BPDAwEOT attack with L_∞ norm are shown here.



(a) Adversarial Examples against Madry with $\varepsilon_{L_\infty} = 0.93, \varepsilon_{L_1} = 125.39, \varepsilon_{L_2} = 8.18$

(b) Adversarial Examples against Madry + MALADE with $\varepsilon_{L_\infty} = 0.83, \varepsilon_{L_1} = 22.64, \varepsilon_{L_2} = 3.55$

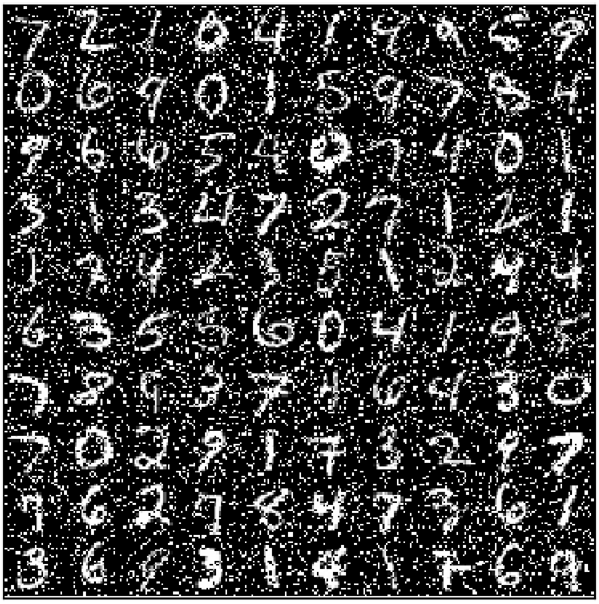
Figure C.12: Sample adversarial images crafted by EAD attack are shown here. The attack fails to converge for many images despite our best effort to finetune the algorithm.



(a) Adversarial Examples against Madry with $\varepsilon_{L_\infty} = 0.56, \varepsilon_{L_1} = 11.36, \varepsilon_{L_2} = 1.38$

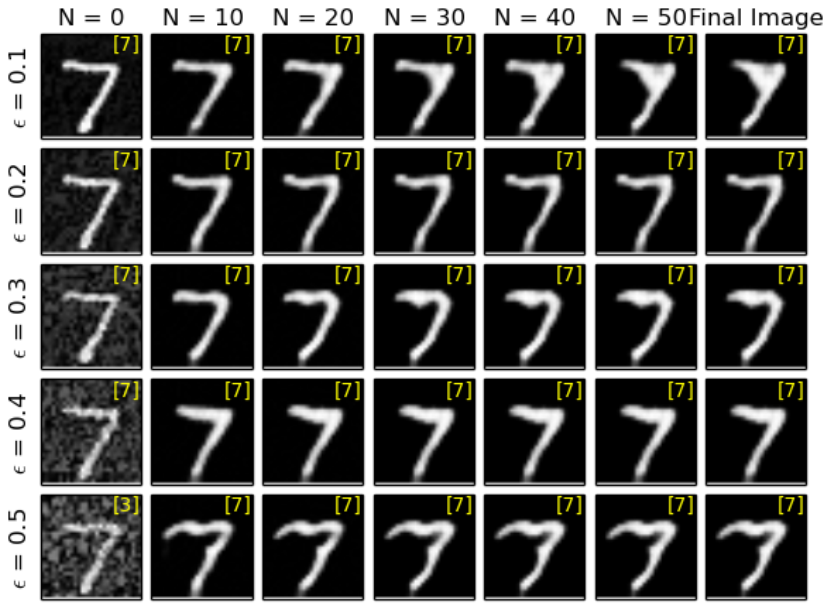
(b) Adversarial Examples against Madry + MALADE with $\varepsilon_{L_\infty} = 0.54, \varepsilon_{L_1} = 200.46, \varepsilon_{L_2} = 8.34$

Figure C.13: Sample adversarial images crafted by a Boundary attack are shown here. Some images fail to become adversarial during the initialization of the algorithm with random uniform noise and hence are retained as the original image.

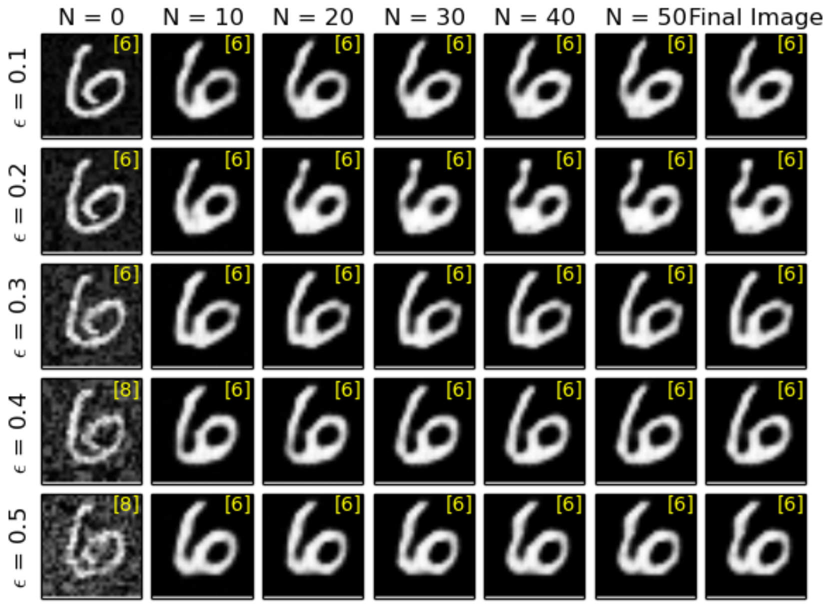


(a) Images corrupted by Salt and Pepper Noise Attack with $\varepsilon_{L_\infty} = 0.99, \varepsilon_{L_1} = 98.02, \varepsilon_{L_2} = 9.70$

Figure C.14: Sample adversarial images crafted by adding salt and pepper noise are shown here. The mean of the magnitude of the perturbations over the entire test dataset for each distance measure is given below each image.

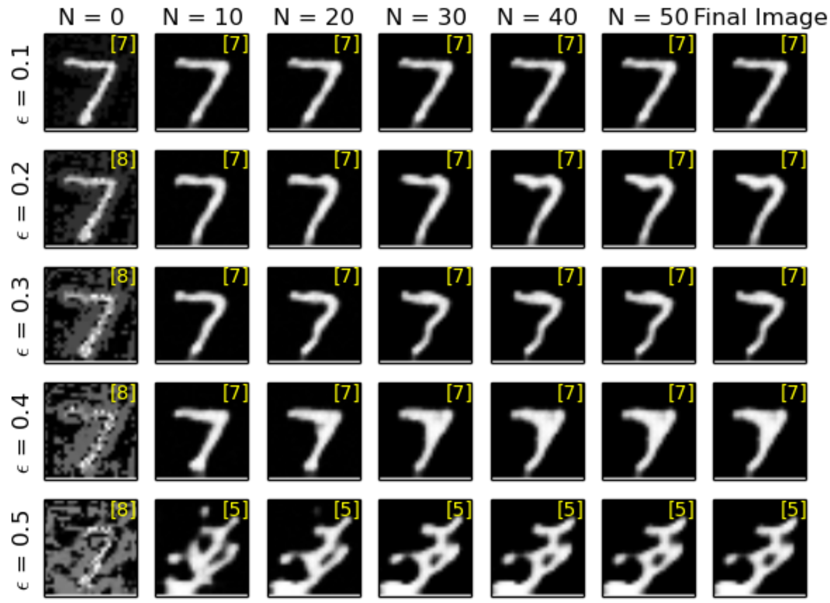


(a)

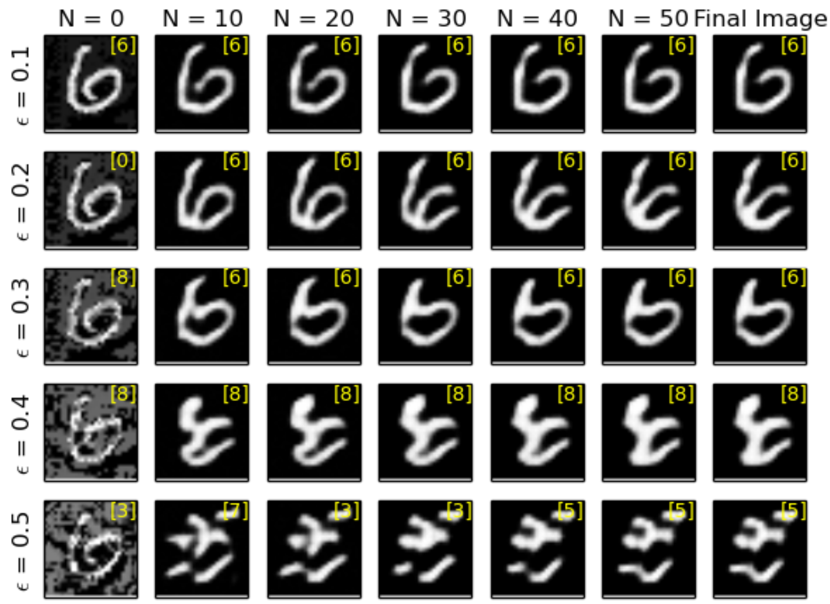


(b)

Figure C.15: Sample images for the MALADE algorithm against a PGD attack with L_∞ norm are shown here. The rows indicate the norm of the perturbation used by the attacked while the columns indicate the intermittent steps taken by MALADE to defend the attack. The classifier's decision is displayed in yellow in the top right corner of each image. Due to gradient obfuscation, the adversarial sampels are not very strong and hence MALADE is very robust here.

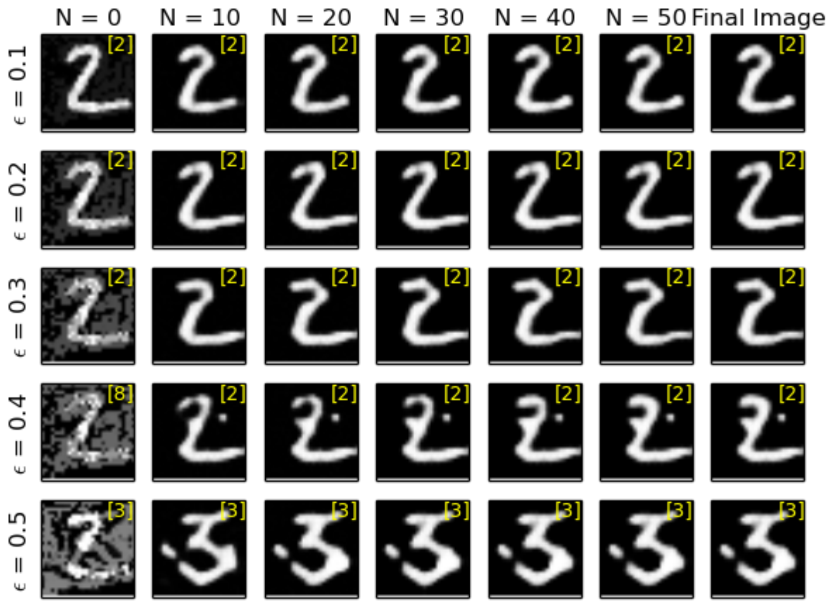


(a)

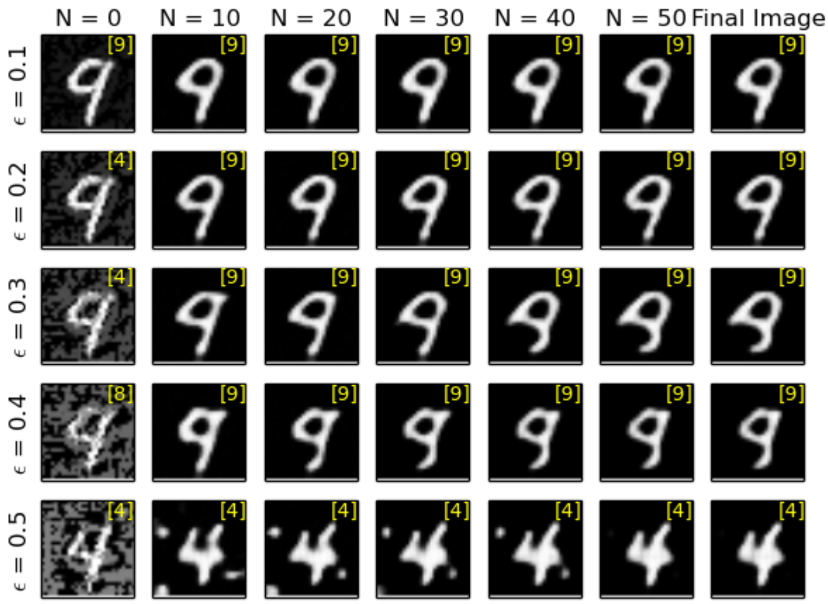


(b)

Figure C.16: Sample images for the MALADE algorithm against a BPDAwEOT attack with L_∞ norm are shown here. The rows indicate the norm of the perturbation used by the attacked while the columns indicate the intermittent steps taken by MALADE to defend the attack. The classifier's decision is displayed in yellow in the top right corner of each image. Since the gradients are computed only until the input to the classifier, there is no gradient obfuscation and hence the attack is strong.

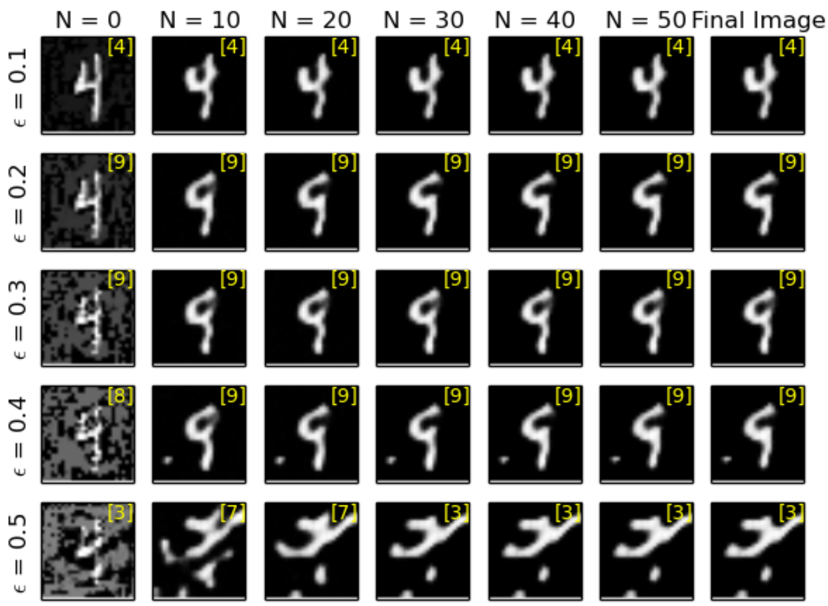


(a)

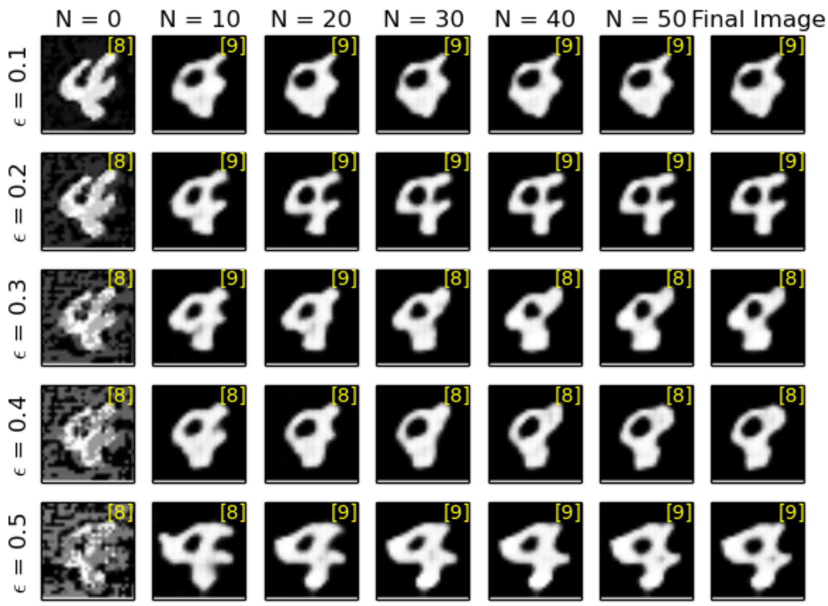


(b)

Figure C.17: Sample images for the MALADE algorithm against a BPDAwEOT attack with L_∞ norm are shown here. The rows indicate the norm of the perturbation used by the attacked while the columns indicate the intermittent steps taken by MALADE to defend the attack. The classifier's decision is displayed in yellow in the top right corner of each image.



(a)



(b)

Figure C.18: Sample images for the MALADE algorithm against a BPDAwEOT attack with L_∞ norm are shown here. The rows indicate the norm of the perturbation used by the attacked while the columns indicate the intermittent steps taken by MALADE to defend the attack. The classifier's decision is displayed in yellow in the top right corner of each image.