On Breaking Deep Generative Model-based Defenses and Beyond

Yanzhi Chen [1] **Renjie Xie** [2] **Zhanxing Zhu** [3]

Abstract

Deep neural networks have been proven to be vulnerable to the so-called adversarial attacks. Recently there have been efforts to defend such attacks with deep generative models. These defenses often predict by inverting the deep generative models rather than simple feedforward propagation. Such defenses are difficult to attack due to the obfuscated gradients caused by inversion. In this work, we propose a new white-box attack to break these defenses. The idea is to view the inversion phase as a dynamical system, through which we extract the gradient w.r.t the image by backtracking its trajectory. An amortized strategy is also developed to accelerate the attack. Experiments show that our attack better breaks state-of-the-art defenses (e.g DefenseGAN, ABS) than other attacks (e.g BPDA). Additionally, our empirical results provide insights for understanding the weaknesses of deep generative model defenses.

1. Introduction

How to make deep neural network (DNN) more robust? This has been a hot research problem with a long history and has been recently re-emphasized due to the discovery of adversarial samples (Szegedy et al., 2014; Goodfellow et al., 2014). In short, adversarial samples are synthetic images that are perceptually similar to natural images, but can dramatically yield sub-optimal or even completely wrong decisions in a DNN. Such samples clearly reflect the frangibility of DNN and call for robustification solutions.

Many efforts have been devoted to defend DNN from adversarial samples. This includes regularization (Lyu et al., 2015; Ross & Doshi-Velez, 2018; Zhang et al., 2019), adversarial training (Goodfellow et al., 2014; Madry et al., 2018), feature denoising (Liao et al., 2018; Xie et al., 2019), randomized smoothing (Salman et al., 2019; Cohen et al., 2019), etc. Recently, a new line of research has focused on the use of deep generative model in defense (Samangouei et al., 2018; Schott et al., 2019; Ghosh et al., 2019). These defenses do not perform prediction by feedforward propagation. Rather, they first invert the generative model G to find the most plausible latent representation $\mathbf{z}^*$ that could have generated the image $\mathbf{x}$ (e.g by solving an optimization problem), then use $\mathbf{z}^*$ to predict. Compared to other defenses, deep generative model-based defenses are attractive as they provide a natural way to realize the *on-manifold* conjecture. However, they also bring new challenges to how to correctly evaluate their robustness, as the gradients in these defenses are obfuscated (Athalye et al., 2018) due to the inversion.

In this work, we propose a new white-box attack to break deep generative model-based defenses. We find that existing attacks for breaking these defenses like Backward Pass Differential Approximation (BPDA, (Athalye et al., 2018)) are indeed sub-optimal and might overestimate the robustness. We then develop a new attack based on a novel gradient estimation mechanism. Our contributions are two-folds:

- We develop a new white-box attack for breaking deep generative model defenses. Compared to BPDA, our attack can find adversarial samples with lower distortion and higher efficiency. Our attack can be applied to a wide range of deep generative model defenses including DefenseGAN (Samangouei et al., 2018), ABS (Schott et al., 2019) and (Ghosh et al., 2019; Lin et al., 2019).
- With our attack, we also analyze factors that cause deep generative model defenses to fail. For example, we find that there are 'holes' in the latent space of these defenses that can generate samples outside of the data distribution or even belonging to unseen classes, which make them vulnerable. The detection of these holes are difficult. We also provide some suggestions to improve these defenses.

2. Background

Throughout this paper, we denote the normal image by $\mathbf{x} \in \mathcal{X}$, the class label by $y \in \mathcal{Y}$ and the classifier by $F : \mathcal{X} \mapsto \mathcal{Y}$. In this work, we mainly consider the case where the classifier F (or part of F) is a deep neural network.

[1]School of Informatics, The University of Edinburgh, UK [2]School of Information Engineering, Southeast University, China [3]School of Mathematical Sciences, Peking University, China. Correspondence to: Zhanxing Zhu <zhanxing.zhu@pku.edu.cn>.

Proceedings of the 37^{th} International Conference on Machine Learning, Online, PMLR 119, 2020.

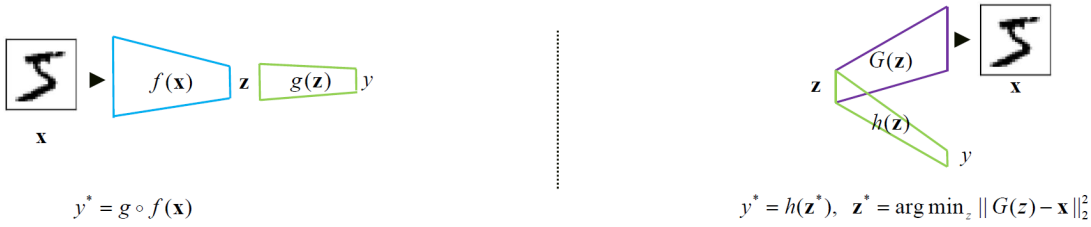


Figure 1. A comparison between conventional DNN and the deep generative model-based defenses. **Left**: conventional DNN, where classification is done by feedforward propagation in which the representation $\mathbf{z}$ and the label y are computed sequentially. **Right**: deep generative model-based defense, where we first invert the generative model to find $\mathbf{z}^*$ that best matches with $\mathbf{x}$, then uses $\mathbf{z}^*$ to predict y.

2.1. Adversarial sample

Given an image $\mathbf{x}$ with true label y, an adversarial sample $\mathbf{x}'$ is the minimal modified version of $\mathbf{x}$ that can cause the classifier F to misclassify:

$$\mathbf{x}' = \underset{\mathbf{x}':F(\mathbf{x}')\neq y}{\arg\min} \|\mathbf{x} - \mathbf{x}'\|_p \tag{1}$$

where $\|\mathbf{x} - \mathbf{x}'\|_p$ is the modification/distortion between $\mathbf{x}$ and $\mathbf{x}'$. Typical choices of p include $p = \infty$ or $p = 2$. Adversarial samples are often found by the so-called *adversarial attacks*, which are classified into two categories:

White-box attacks. This type of attack assumes full knowledge of the model. It typically finds $\mathbf{x}'$ by gradient descent method: $\mathbf{x}' \leftarrow \mathbf{x}' - \nabla_{\mathbf{x}'}\mathcal{L}(\mathbf{x}')$ with $\mathcal{L}$ being some classification loss. Well known instances in this attack include FGSM (Goodfellow et al., 2014), PGD, MIM variants (Dong et al., 2018) (Kurakin et al., 2016), CW (Carlini & Wagner, 2017), etc. White-box attacks are known to be fast, accurate and easy to implement. However, they can not be readily applied to cases where (a) part of the model is unknown or (b) the gradient in the model is not analytical (Athalye et al., 2018).

Black-box attacks. As complement to white-box attacks, this type of attack needs only partial access to the model. Generally speaking, it first samples a population of modifications $\epsilon_i \sim p(\epsilon_i)$ from a proposal distribution p, then modify the sample $\mathbf{x}$ according to $F(\mathbf{x} + \epsilon_i)$. Popular instances include boundary attack (Brendel et al., 2018), evolutionary strategy-based attack (Ilyas et al., 2018) and their extensions (Guo et al., 2019; Li et al., 2019). Black-box attacks are useful in that they not only need not to know the model fully, but can also be applied to cases without analytical gradients. However, they are typically less efficient and accurate.

A well-known hypothesis about adversarial samples is the *on-manifold* conjecture, which states that normal data lies on a low-dimensional manifold but adversarial samples are off this manifold. If this hypothesis were to hold true, we might resist adversarial attacks with deep generative models, as they naturally provide us with a way to parameterize the data manifold (Samangouei et al., 2018; Schott et al., 2019). Below, we review a line of defenses building upon this idea.

2.2. Deep generative model-based defenses

To defend adversarial attacks, one recent line of research has utilized the power of a deep generative model G in defense. These defenses assume that the clean data $\mathbf{x}$ is generated by some latent representation $\mathbf{z}$: $\mathbf{x} = G(\mathbf{z}), \mathbf{z} \sim p(\mathbf{z})$. To make prediction, they first invert G to find the representation $\mathbf{z}^*$ that best matches with the input $\mathbf{x}$, then use $\mathbf{z}^*$ to predict:

$$y^* = h(\mathbf{z}^*; \mathbf{x}), \qquad \mathbf{z}^* = \underset{\mathbf{z}}{\arg\min}\, l_G(\mathbf{z}; \mathbf{x}) \tag{2}$$

where h is a classification function, l_G is a loss function measuring how well $\mathbf{x}$ and $\mathbf{z}$ matches with each other. Typical choice of l_G is the reconstruction error: $l_G(\mathbf{z}; \mathbf{x}) = \|G(\mathbf{z}) - \mathbf{x}\|_2^2$. We revisit three representative works below.

DefenseGAN (Samangouei et al., 2018). This GAN-based approach works by first projecting the input $\mathbf{x}$ back to the range of GAN before classification. In this setting, the loss l_G is $l_G(\mathbf{z}; \mathbf{x}) = \|G(\mathbf{z}) - \mathbf{x}\|_2^2$, and the function h is $h(\mathbf{z}^*; \mathbf{x}) = C(G(\mathbf{z}^*))$ with C a conventional deep classifier.

Analysis by Synthetics (ABS) (Schott et al., 2019). This VAE-based approach classifies by inferring which class y could have generated the input $\mathbf{x}$. It first trains a set of VAE: $\{G_1, ...G_y\}$ for each class y, then seeks which G_y can best reconstruct $\mathbf{x}$. In this setting, the loss l_G is $l_{G_y}(\mathbf{z}; \mathbf{x}) = \frac{1}{\sigma^2}\|G_y(\mathbf{z}) - \mathbf{x}\|_2^2 + KL[\mathcal{N}(\mathbf{z}, \mathbf{I}); \mathcal{N}(\mathbf{0}, \mathbf{I})]$, and the function h is $h = \arg\max_y h_y$ where $h_y(\mathbf{z}^*; \mathbf{x}) = l_{G_y}(\mathbf{z}^*; \mathbf{x})$.

MoG-VAE (Ghosh et al., 2019). This VAE-based approach utilizes a VAE whose latent distribution $p(\mathbf{z})$ is modelled as a mixture of Gaussian (MoG), with each of its mode corresponding to one class. Classification is done by seeking which mode $\mathbf{z}^*$ belongs to the most likely. In this setting, the loss l_G is $l_G(\mathbf{z}; \mathbf{x}) = \|G(\mathbf{z}) - \mathbf{x}\|_2^2$, and the function h is $h = \arg\max_y h_y$ where $h_y(\mathbf{z}^*; \mathbf{x}) = \mathcal{N}(\mathbf{z}^*; \mu_y, \Sigma_y)$.

As one can see, due to the $\arg\min$ operator in (2), the gradient *w.r.t* $\mathbf{x}$ in the classification routine is no more analytical — a phenomenon known as *obfuscated gradient* (Athalye et al., 2018). This makes evaluating the robustness of these defenses by conventional attacks like PGD or CW difficult. Next, we develop a novel attack for breaking such defenses.

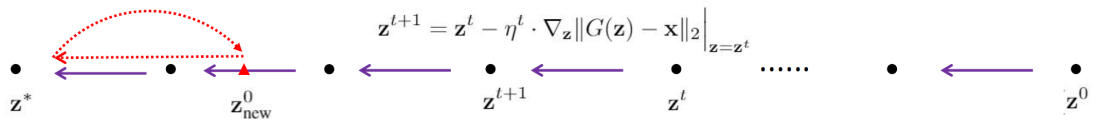


Figure 2. Demonstrating the proposed inversion attack for breaking deep generative model-based defense.

3. Attack

In this section, we develop a new attack tailored to break the above deep generative model-based defenses. Without loss of generality we assume that the clean image $\mathbf{x}$ belongs to class y, and denote the prediction scores for class y and class y' (here $y' \neq y$) by $h_y(\mathbf{z}^*; \mathbf{x})$ and $h_{y'}(\mathbf{z}^*; \mathbf{x})$ respectively. Here $\mathbf{z}^* = \arg\min_{\mathbf{z}} l_G(\mathbf{z}; \mathbf{x}')$ as previously defined in (2).

3.1. Main method

Objective function. We begin by setting up the objective for seeking $\mathbf{x}'$, the adversarial example for $\mathbf{x}$. Our new attack takes a CW-like form (Carlini & Wagner, 2017):

$$\mathcal{L}(\mathbf{x}') = \|\mathbf{x} - \mathbf{x}'\|_2^2 + \lambda \cdot \text{ReLU}(h_y(\mathbf{z}^*; \mathbf{x}') - h_{y'}(\mathbf{z}^*; \mathbf{x}')) \tag{3}$$

The first term in this objective aims to find the adversarial sample with the minimal distortion whereas the second term in the objective aims to cause a mis-classification. We can in principle calculate the gradient to $\mathbf{x}'$ in this objective as:

$$\nabla_{\mathbf{x}'} h_y(\mathbf{z}^*; \mathbf{x}') = \nabla_{\mathbf{z}^*} h_y(\mathbf{z}^*; \mathbf{x}')\Big|_{\mathbf{x}'} \nabla_{\mathbf{x}'} \mathbf{z}^* + \nabla_{\mathbf{x}'} h_y(\mathbf{z}^*; \mathbf{x}')\Big|_{\mathbf{z}^*} \tag{4}$$

Unfortunately, due to the absence of the analytical form for $\mathbf{z}^*$, direct calculation of (4) is impossible. One way to calculate the gradient $\nabla_{\mathbf{x}'} \mathbf{z}^*$ indirectly is by *implicit differentiation* (Gould et al., 2016; Colson et al., 2007), where one can compute $\nabla_{\mathbf{x}'} \mathbf{z}^*$ as $\nabla_{\mathbf{x}'} \mathbf{z}^* = -\mathbf{H}^{-1}\mathbf{g}$ with $\mathbf{H} = \nabla^2_{\mathbf{z}^*\mathbf{z}^*} l_G(\mathbf{z}^*, \mathbf{x}')$ and $\mathbf{g} = \nabla_{\mathbf{x}'} \nabla_{\mathbf{z}^*} l_G(\mathbf{z}^*, \mathbf{x}')$. However, this method requires to invert the Hessian $\mathbf{H}$ per every update to $\mathbf{x}'$, which is computationally heavy if (a) the dimensionality of $\mathbf{z}$ is high and (b) the number of updating steps required to find $\mathbf{x}'$ is large. We propose an alternative light-weighted method to approximate $\nabla_{\mathbf{x}'} \mathbf{z}^*$ below.

Gradient approximation. While we have no closed form solution for $\mathbf{z}^*$, it can be approximated by T gradient steps:

$$\begin{aligned} \mathbf{z}^* &\approx \mathbf{z}^T, \\ \mathbf{z}^{t+1} &= \mathbf{z}^t - \eta \cdot \nabla_{\mathbf{z}} l_G(\mathbf{z}; \mathbf{x}')|_{\mathbf{z}=\mathbf{z}^t} \end{aligned} \tag{5}$$

which can be seen as a dynamical system with parameter $\mathbf{x}'$:

$$\begin{aligned} \dot{\mathbf{z}} &= q_{\mathbf{x}'}(\mathbf{z}), \\ q_{\mathbf{x}'}(\mathbf{z}) &= \mathbf{z} - \eta \cdot \nabla_{\mathbf{z}} l_G(\mathbf{z}; \mathbf{x}') \end{aligned} \tag{6}$$

with which we can now express $\mathbf{z}^*$ as the output of a deterministic function $Q(\mathbf{x}')$:

$$\begin{aligned} \mathbf{z}^* &\approx Q(\mathbf{x}'), \\ Q(\mathbf{x}') := Q_{\mathbf{x}'}(\mathbf{z}^0) &= \underbrace{q_{\mathbf{x}'} \circ q_{\mathbf{x}'} \ \ldots \ \circ q_{\mathbf{x}'}(\mathbf{z}^0)}_{T} \end{aligned} \tag{7}$$

where $\mathbf{z}^0$ is the initial value of $\mathbf{z}$. Note that we have swapped the roles of parameter and variable in the definition of $Q(\mathbf{x}')$. Now, we could have been able to substitute $\mathbf{z}^* \approx Q(\mathbf{x}')$ back to (3) to solve $\mathbf{x}'$. However, evaluating the gradient *w.r.t.* $\mathbf{x}'$ through expanding the expression for Q is still impractical even for moderately large T (e.g $T = 50$) due to the high computational complexity. And even if we could expand Q, the gradient $\nabla_{\mathbf{x}'} Q(\mathbf{x}')$ in it is still likely to vanish due to the long sequence when applying back-propagation — an issue similar to that in recurrent neural networks.

Upon a closer look, the number of required gradient steps T indeed depends on the initial value of $\mathbf{z}^0$. If $\mathbf{z}^0$ is far away from $\mathbf{z}^*$, then we may need a larger T to reach $\mathbf{z}^*$. Conversely, if $\mathbf{z}^0$ has been near around $\mathbf{z}$, a few steps could have been enough. This inspires us to use a *backtracking* strategy for reducing T. More specifically, we first find $\mathbf{z}^*$ via standard optimizer like Adam (purple routine in Figure 2). After that, we conduct $T' \ll T$ gradient *ascent* steps to revert the optimization process and treat the result as the new $\mathbf{z}^0$. We then re-implement gradient *descent* from this new $\mathbf{z}^0$ to reach $\mathbf{z}^*$ again (red routine in Figure 2). Since the new $\mathbf{z}^0$ is much closer to $\mathbf{z}^*$, we can expect that we would approach $\mathbf{z}^*$ with small T, addressing the large T problem.

In practice, we implement the above backtracking strategy with $T' = 5$ backtracking steps under learning rate η', which as we confirm empirically well balances between accuracy and efficiency. η' is determined as:

$$\eta' = \arg\max_{\eta} \frac{\left| l_G(\mathbf{z}^0_{\text{new},\eta}; \mathbf{x}') - l_G(\mathbf{z}^*; \mathbf{x}') \right|}{\left| l_G(\mathbf{z}^*_{\text{new},\eta}; \mathbf{x}') - l_G(\mathbf{z}^*; \mathbf{x}') + \tau \right|} \tag{8}$$

where $\mathbf{z}^0_{\text{new},\eta}$ is the result of gradient ascent under learning rate η (i.e. the new $\mathbf{z}^0$) and $\mathbf{z}^*_{\text{new},\eta}$ is the result that we re-implement gradient descent under η (i.e. the approximated $\mathbf{z}^*$). Therefore we determine η' to be the largest learning rate that would allow us to return to $\mathbf{z}^*$ in the backtracking procedure. The reason why we prefer a larger η' is that this would allow us to visit the loss landscape of $l_G(\mathbf{z}; \mathbf{x}')$ as much as possible within the T' steps, which provides us with

more information about $\mathbf{x}'$ (remark that $\mathbf{x}'$ is the parameter of the loss function $l_G(\mathbf{z}; \mathbf{x}')$). Note that we have introduced a small constant $\tau = 10^{-4}$ in (8) to avoid dividing by zero. The determination of η' can typically be done by a pilot run, and its value could vary for different $\mathbf{x}'$.

Algorithm. With the above gradient approximation mechanism, we now elaborate our algorithm for finding $\mathbf{x}'$. The algorithm iterates between the following two steps:

- $\mathbf{z}$-*step*. Fix $\mathbf{x}'$, find the function $\mathbf{z}^* \approx Q(\mathbf{x}')$ with the gradient approximation technique above;
- $\mathbf{x}$-*step*. With the found $\mathbf{z}^* \approx Q(\mathbf{x}')$, update $\mathbf{x}'$ by projected gradient descent: $\mathbf{x}' \leftarrow \text{proj}(\mathbf{x}' - \delta \cdot \nabla_{\mathbf{x}'} \mathcal{L}(\mathbf{x}'))$.

we call our method *inversion attack* as it can be applied to defenses involving the inversion of deep generative models.

For the initial value of $\mathbf{x}'$, we use two initializations: (a) $\mathbf{x}' = \mathbf{x}$; (b) $\mathbf{x}' = G(\mathbf{u}^*)$ with $\mathbf{u}^*$ being the minimizer of:

$$\|\mathbf{x} - G(\mathbf{u})\|_2^2 + \lambda \cdot \text{ReLU}(h_y(\mathbf{z}^*; G(\mathbf{u})) - h_{y'}(\mathbf{z}^*; G(\mathbf{u}))) \tag{9}$$

This objective is a variant of (3) with the adversarial sample $\mathbf{x}'$ lying exactly on the generative range of G. We now update these two different initializations respectively, and take the final $\mathbf{x}'$ as the one with smaller distortion.

To find $\mathbf{x}'$ with the minimal distortion, we further apply the same binary search strategy as in CW attack to determine λ. The whole attack algorithm is summarized in Algorithm 1.

Algorithm 1 Inversion attack

Input: clean image $\mathbf{x}$, generative model G
Output: adversarial sample $\mathbf{x}'$
Hyperparam: learning rate δ, range of λ: $[\lambda_{\min}, \lambda_{\max}]$

Initialize $\mathbf{x}'$ as described above;
repeat
  for $k = 1$ **to** K **do**
    determine η' for $\mathbf{x}'$ as in (8) via pilot run;
    $\mathbf{z}$-step: find the expression $\mathbf{z}^* \approx Q(\mathbf{x}')$ with η';
    $\mathbf{x}$-step: with Q, set $\mathbf{x}' \leftarrow \text{proj}(\mathbf{x}' - \delta \cdot \nabla_{\mathbf{x}'} \mathcal{L}(\mathbf{x}'))$;
  end for
  if attack success **then**
    $\lambda \leftarrow \frac{1}{2}(\lambda + \lambda_{\min})$
  else
    $\lambda \leftarrow \frac{1}{2}(\lambda + \lambda_{\max})$
  end if
until convergence
return $\mathbf{x}'$

Finally, we have the following theorem telling that under mild conditions, the approximated gradient $\nabla_{\mathbf{x}'} Q(\mathbf{x}')$ in our attack would well approximate the true gradient $\nabla_{\mathbf{x}'} \mathbf{z}^*$.

Theorem 1. *Let T' be the number of backtracking steps and $\mathbf{H} = \nabla^2_{\mathbf{z}^*\mathbf{z}^*} l_G$ be the Hessian of the function l_G at $(\mathbf{z}^*, \mathbf{x}')$. Assuming that (a) the solved $\mathbf{z}^*$ is optimal in each $\mathbf{z}$-step (b) the new learning rate η' satisfies $\eta' \|\mathbf{H}\| \leq 1$ and (c) the new $\mathbf{z}^0$ in the backtracking procedure namely $\mathbf{z}^0_{new}$ is not far away from $\mathbf{z}^*$. Then $\nabla_{\mathbf{x}'} Q(\mathbf{x}')$ is approximately a T'-order approximation to $\nabla_{\mathbf{x}'} \mathbf{z}^*$:*

$$\nabla_{\mathbf{x}'} \mathbf{z}^* = \sum_{t=0}^{\infty} \mathbf{B}\mathbf{A}^t, \qquad \nabla_{\mathbf{x}'} Q(\mathbf{x}') \approx \sum_{t=0}^{T'} \mathbf{B}\mathbf{A}^t$$

where $\mathbf{A}$ and $\mathbf{B}$ are some matrices with $\|\mathbf{A}\| \leq 1$.

Proof: see Appendix A. The proof is partially inspired by (Shaban et al., 2019). □

3.2. Amortized extension

In this part, we further develop an amortized inference-inspired strategy to accelerate the attack. During our attack, we need to execute $\mathbf{z}$-step for K times if we are to perform K updates to find $\mathbf{x}'$ (see Algorithm 1). This is computationally expensive if the underlying steps T required to reach $\mathbf{z}^*$ in each $\mathbf{z}$-step is large (e.g $T = 500$). To reduce the computational expense, we propose to train an auxiliary encoder network $E(\cdot)$:

$$E = \arg\min_{E \in \mathcal{E}} \mathbb{E}_{\mathbf{x} \sim p_{\text{mix}}(\mathbf{x})} \Big[l_G(E(\mathbf{x}); \mathbf{x}) \Big] \tag{10}$$

where the distribution $p_{\text{mix}}(\mathbf{x})$ is (implicitly) defined as:

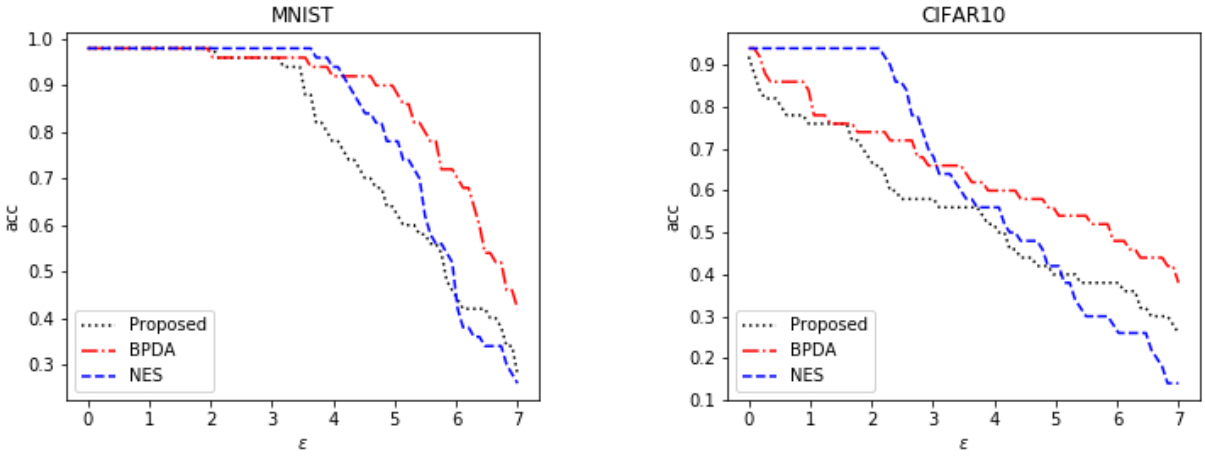
$$\begin{gathered} \mathbf{x} \sim p_{\text{mix}}(\mathbf{x}) \;\; \Leftrightarrow \;\; \mathbf{x} = G(\mathbf{z}_{\text{mix}}), \\ \mathbf{z}_{\text{mix}} = \beta \mathbf{z}_1 + (1 - \beta)\mathbf{z}_2 \\ \mathbf{z}_1 \sim p(\mathbf{z}), \qquad \beta \sim \mathcal{U}(0, 1), \qquad \mathbf{z}_2 \sim p(\mathbf{z}). \end{gathered} \tag{11}$$

That is, we randomly perform interpolation in the latent space and train E with the samples generated from these interpolations. One can imagine that with sufficient training samples and powerful E, we can obtain a good initial value for $\mathbf{z}^*$ by taking

$$\mathbf{z}^0 = E(\mathbf{x}) \tag{12}$$

We found this strategy particularly useful for reducing the number of required steps T to reach $\mathbf{z}^*$. We call this strategy *amortized attack* because E could help us to partially solve $\mathbf{z}^*$ for different $\mathbf{x}$, avoiding us to solve $\mathbf{z}^*$ from scratch. The training of E can typically be done offline before we attack.

Remark. One may wonder whether we could alternatively take $\mathbf{z}^* \approx E(\mathbf{x}')$ rather than $\mathbf{z}^* \approx Q(\mathbf{x}')$ in Algorithm 1. However, such a strategy would yield inaccurate gradient $\nabla_{\mathbf{x}'} \mathbf{z}^*$ since $E(\mathbf{x}')$ is typically not as close to $\mathbf{z}^*$ as $Q(\mathbf{x}')$ (i.e $\|Q(\mathbf{x}') - \mathbf{z}^*\|_2 \leq \|E(\mathbf{x}') - \mathbf{z}^*\|_2$). This could cause the attack to fail. Nonetheless, using $E(\mathbf{x}')$ as only an initialization of $\mathbf{z}^0$ is safe and would not weaken the attack.



(a) MNIST (b) CIFAR10

Figure 3. Comparing different attacks when applied to DefenseGAN. The horizontal axis denotes the l_2 distortion and the vertical axis denotes the classification accuracy. Theses results are reported on 100 images. Note that each pixel in the image is in the range $[-1, 1]$.

4. Comparison to other attacks

Some attacks in the field were also applied to estimate the gradients in deep generative model-based defenses. Here we discuss two of them that are most relevant to our work.

BPDA (Athalye et al., 2018). This white-box attack estimates the gradient by replacing the non-differentiable part in the defense with its differentiable approximation. When applied to deep generative model-based defenses, it first solves $\mathbf{z}^*$ as usual, then treats it as a constant and estimates the gradient *w.r.t* $\mathbf{x}'$ by that *w.r.t* $G(\mathbf{z}^*)$:

$$\nabla_{\mathbf{x}'} F(\mathbf{x}') \approx \nabla_{G(\mathbf{z})} F(G(\mathbf{z}))\Big|_{\mathbf{z}=\mathbf{z}^*}$$

and there is now no need to compute $\nabla_{\mathbf{x}'}\mathbf{z}^*$ which is hard to compute. However, such a method indeed induces a strong assumption on $\mathbf{x}'$ that $\mathbf{x}' \approx G(\mathbf{z}^*)$ i.e the adversarial sample must live near the range of the deep generative model. As we will see later in the experiments, this assumption is indeed impractical and might yield sub-optimal adversarial samples. Our attack, by contrast, makes no assumption about $\mathbf{x}'$.

NES (Ilyas et al., 2018). This black-box attack estimates the gradient $\nabla_{\mathbf{x}'} F(\mathbf{x}')$ from a set of random proposals:

$$\nabla_{\mathbf{x}'} F(\mathbf{x}') \approx \frac{1}{n\sigma^2} \sum_{i=1}^{n} \epsilon_i F(\mathbf{x}' + \epsilon_i), \quad \epsilon_i \sim \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I})$$

where σ^2 is the variance of the proposal distribution. While this method provides us with a derivative-free way for estimating gradient, it generally requires a large n to work well (e.g n=50), which is prohibitive in deep generative model-based defense as we would have to solve $\mathbf{z}^*$ for n time per each update to $\mathbf{x}'$. Our attack only needs to solve $\mathbf{z}^*$ once.

5. Experiments

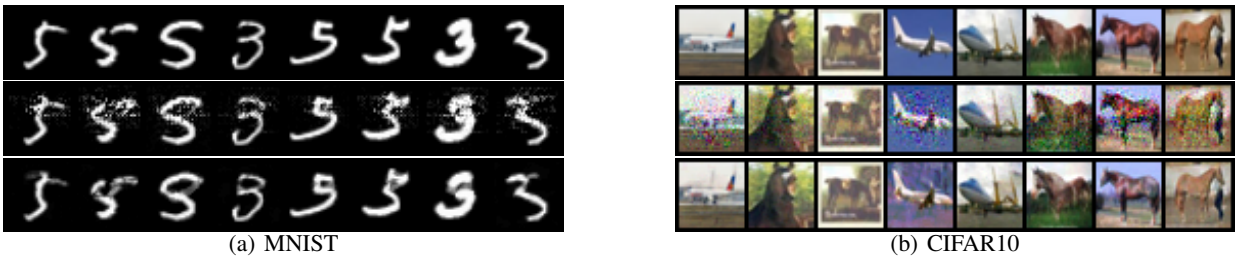
In this section, we apply our attack to re-assess the robustness of recent deep generative model-based defenses. Using our attack, we also discover many interesting properties of these defenses not identified by previous works. Codes are available at https://github.com/cyz-ai/attack_DGM.

Defenses. We focus on two state-of-the-art deep generative model defenses in the field: DefenseGAN (Samangouei et al., 2018) and Analysis-by-Synthesis (Schott et al., 2019).

Baselines. We compare our attack with the two attacks mentioned in Section 4: the black-box NES attack, and the white-box BPDA attack. For NES we use the default setting as in the original literature (Ilyas et al., 2018). For BPDA attack we use the version adapted to each defense (see Section 5.4.2 in (Athalye et al., 2018) for DefenseGAN and Section 5 'latent descent attack' in (Schott et al., 2019) for Analysis-by-Synthesis). For both attacks, we use the same CW-like objective as in our attack to achieve smallest distortion. We perform 75 gradient steps to $\mathbf{x}'$ in all attacks.

Dataset. We conduct experiments on two common datasets: MNIST and CIFAR10. For simplicity, we mainly consider the binary classification scenario: digit 3 vs 5 in MNIST, and plane vs horse in CIFAR10, however our analysis could easily be extended to multi-class cases. All images $\mathbf{x} \in [-1, 1]^{C\times 32\times 32}$ with C being the number of channels.

Detailed network settings. For all defenses, we adopt the same convolutional/deconvolutional neural network as in DCGAN (Radford et al., 2015), a popular deep generative model in the field. We use a 32-dim latent representation for MNIST and 64-dim latent representation for CIFAR10.



(a) MNIST (b) CIFAR10

Figure 4. A comparison between the adversarial samples found by BPDA attack (Athalye et al., 2018) and the proposed attack when applied to DefenseGAN (Samangouei et al., 2018). **Top row**: clean sample. **Middle row**: adversarial sample found by BPDA. **Bottom row**: adversarial sample found by our attack. Adversarial samples found by our attack are clearly more close to the original sample.

5.1. DefenseGAN

Effectiveness of the attack. In Figure 3, we show the accuracy-distortion curve of DefenseGAN under the various attacks. One can see that on both datasets, the proposed attack yields a lower model accuracy under the same distortion. Figure 4 further shows some adversarial samples found by our attack and that by BPDA, from which one can confirm that adversarial samples found by our attack are visually closer to natural images (especially on CIFAR10).

To further illustrate the advantage of our attack over BPDA, we also compare the distortion achieved by our attack and that by BPDA under various latent dimensionality d. Higher d corresponds to a more powerful generative model and hence better satisfying the assumption $G(\mathbf{z}^*) \approx \mathbf{x}'$ on which BPDA relies to work well. Table 1 shows the results. Unsurprisingly, as d decreases, the distortion of BPDA increases sharply. This is because the assumption $G(\mathbf{z}^*) \approx \mathbf{x}'$ no more holds in low d cases, which leads to biased gradients in BPDA and hence sub-optimal adversarial sample. Our attack, by contrast, is not sensitive to d and the power of G. These results suggest that our attack might be a better tool for evaluating the true robustness of DefenseGAN (the high efficacy of BPDA reported in (Athalye et al., 2018) for DefenseGAN might be due to the very powerful generative model used in the experiments where d was set to be 128).

Table 1. The average l_2 distortion of DefenseGAN on MNIST under different latent dimensionality d.

	$d = 8$	$d = 16$	$d = 32$
PROPOSED	5.76	5.64	5.57
BPDA	11.60	9.24	7.13

In Table 2, we compare the computational expense of different attacks when applied to DefenseGAN. Compared to NES, our attack only need to solve $\mathbf{z}^*$ for one time per each update to $\mathbf{x}'$, whereas in NES we need to solve $\mathbf{z}^*$ 50 times. Compared to BPDA, the number of gradient steps required to find $\mathbf{z}^*$ in our attack is just half of that in BPDA owing to the use of the amortized attack strategy. The overall cost in our attack is approximately 1/2 of BPDA and 1/100 of NES.

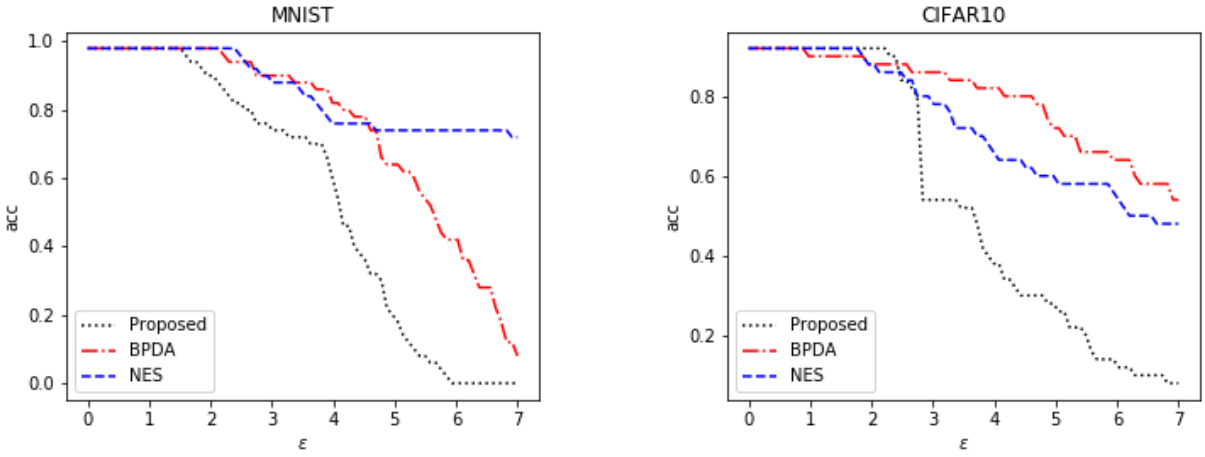
Table 2. Comparing the computational expense of different attacks per each update step to $\mathbf{x}'$ when applied to attack DefenseGAN.

	PROPOSED	BPDA	NES
# WE NEED TO SOLVE $\mathbf{z}^*$	1	1	50
# GD STEPS TO REACH $\mathbf{z}^*$	150	300	300

Weakness of DefenseGAN. Leveraging our attack, we further study what makes DefenseGAN vulnerable. Since DefenseGAN works by first projecting the input image back to the range of GAN, we look into the projected samples i.e $G(\mathbf{z}^*)$ of those adversarial samples found by our attack. Ideally, if the on-manifold conjecture does be well realized in DefenseGAN, the projected samples should highly resemble normal samples. We discover that this is not the case. As shown in Figure 5, many of the projected samples are semantically ambiguous and do not belong to any class. Such ambiguous samples can clearly cause a misclassification easily. This key fact indicates that DefenseGAN indeed does not well achieve the on-manifold conjecture and may explain why it is still vulnerable to our attack.

Figure 5. Visualizing some projected samples in DefenseGAN.

We now provide an explanation for why DefenseGAN does not well realize the on-manifold conjecture. As the GAN used in this defense is trained under the prior $\mathbf{z} \sim \mathcal{N}(\mathbf{z}; \mathbf{0}, \mathbf{I})$, it could be the case that for some specific $\mathbf{z}$, the generated sample $\mathbf{x} = G(\mathbf{z})$ would not look like normal



(a) MNIST (b) CIFAR10

Figure 6. Comparing different attacks when applied to ABS. The horizontal axis denotes the l_2 distortion and the vertical axis denotes the corresponding classification accuracy. Theses results are reported on 100 images. Note that each pixel in the image is in the range $[-1, 1]$.

sample. In other words, there may be some 'holes' in the latent space of DefenseGAN that can be exploited by our attack to generate off-manifold sample. One possible way to evade these holes is to add a regularization term to $l_G(\mathbf{z}; \mathbf{x})$ when solving $\mathbf{z}^*$:

$$l_G(\mathbf{z}; \mathbf{x}) = \|G(\mathbf{z}) - \mathbf{x}\|_2^2 + \beta \cdot MMD(\mathbf{z}, \mathbf{u}),$$
$$\mathbf{u} \sim \mathcal{N}(\mathbf{u}; \mathbf{0}, \mathbf{I})$$

where MMD denotes *maximum mean discrepancy* (Gretton et al., 2012). This regularization term ensures that the solved $\mathbf{z}^*$ would look like as if it were sampled from the prior (remark that if $\mathbf{z} \sim \mathcal{N}(\mathbf{z}; \mathbf{0}, \mathbf{I})$, each dimension of $\mathbf{z}$ namely $\{z_1, z_2, ...z_d\}$ would form samples of a 1D standard normal distribution). However, even under this strong restriction, we found that the generative model in this defense can still generate off-manifold samples. We conclude that the holes may be ubiquitous in the latent space of vanilla GAN; these holes could not be easily detected and avoided. Nonetheless this defense still provide a protection comparable to adversarial training (Madry et al., 2018) (see Appendix B3).

5.2. Analysis-by-Synthesis (ABS)

Effectiveness of the attack. In Figure 6, we show the accuracy-distortion curve of ABS under various attacks. Again, we see that the distortion in our attack is much smaller than that in other attacks. In Figure 8, we further show some adversarial samples found by our attack and that by BPDA. One can see that the adversarial samples found by our attack are visually more closer to normal images. (the unrealistic/blurry adversarial samples in BPDA might be due to its assumption $G(\mathbf{z}^*) \approx \mathbf{x}'$ with G being a vanilla VAE whose reconstruction images are well-known blurry).

In Table 3, we further compare the distortion achieved by our attack and BPDA under various latent dimensionality d. Expectedly our attack yields a lower distortion under all d.

Table 3. The average l_2 distortion of ABS on MNIST under different latent dimensionality d.

	$d = 8$	$d = 16$	$d = 32$
PROPOSED	4.85	4.14	3.90
BPDA	6.16	5.65	5.34

In Table 4, we compare the computational cost of different attacks when applied to ABS, where one can see that the overall cost of our attack is 4/7 of BPDA and 2/175 of NES.

Table 4. Comparing the computation expense of different attacks per each update step to $\mathbf{x}'$ when applied to attack ABS.

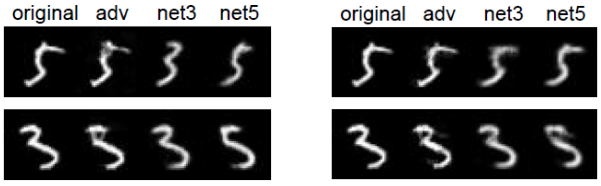
	PROPOSED	BPDA	NES
# WE NEED TO SOLVE $\mathbf{z}^*$	1	1	50
# GD STEPS TO REACH $\mathbf{z}^*$	200	350	350

Weakness of ABS. As in the previous experiment, we further leverage our attack to study what cause the weakness of ABS. From Table 3, we observe an interesting phenomenon: while the efficacy of our attack does not depend on the power of the deep generative model, we do see ABS tend to be more robust in low-dimensionality settings. To investigate this, we visualize the adversarial samples found by our attack as well as their best reconstructions in each VAE

(a) MNIST (b) CIFAR10

Figure 8. A comparison between the adversarial samples found by BPDA attack (Athalye et al., 2018) and the proposed backtracking attack when applied to ABS (Schott et al., 2019). **Top row**: the clean sample. **Middle row**: adversarial sample found by BPDA. **Bottom row**: adversarial sample found by our attack. Adversarial samples found by our attack are clearly more close to the original sample.

(remark that ABS classifies $\mathbf{x}$ by seeking which VAE can best reconstruct $\mathbf{x}$). Figure 7 shows the results. From the figure, we discover that when d is small, the VAE trained on one class can not well reconstruct the image of another class, but when d is large this becomes possible. This explains why ABS tends to be less robust with large d: the deep generative model in such cases becomes so flexible that it can even generate samples in unseen class. These results again confirm us the existence of the 'holes' in the latent space of deep generative models, which could be exploited by an attack to generate off-manifold samples, especially for an overly powerful deep generative model.



(a) $d = 8$ (b) $d = 32$

Figure 7. Visualizing the adversarial samples found by our inversion attack and their best reconstruction in each VAE. Net3 denotes the best reconstruction from the VAE trained on class 3 and net5 denotes the best reconstruction from the VAE trained on class 5.

As in DefenseGAN, we also try to evade the holes in the latent space. One possible way is to restrict that the solved representation $\mathbf{z}^*$ for image $\mathbf{x}$ must be 'common'. For example, we can restrict that the density $p(\mathbf{z}^*)$ is high. This can be achieved by adding a regularization term to $l_{G_y}(\mathbf{z}; \mathbf{x})$:

$$l_{G_y}(\mathbf{z}; \mathbf{x}) = \|G_y(\mathbf{z}) - \mathbf{x}\|_2^2 + \beta \cdot \text{ReLU}(p - p(\mathbf{z})),$$
$$p(\mathbf{z}) = \mathcal{N}(\mathbf{z}; \mu_{\mathbf{z}}, \Sigma_{\mathbf{z}}).$$

where $p(\mathbf{z})$ is estimated from the training set. p is a threshold that ensures $p(\mathbf{z}^*)$ to be high for $\mathbf{z}^*$. Here, p is set to be the 90% quantile of the population of the $p(\mathbf{z})$ values in the training set. We hope this can get rid of some rare $\mathbf{z}^*$ that can generate off-manifold sample. Unfortunately, no significant robustness gain is observed with this constraint. We conclude that the holes are ubiquitous in the latent space of vanilla VAE and are difficult to detect. Despite this weakness, ABS still achieve a robustness comparable to adversarial training (Madry et al., 2018) (see Appendix B3).

6. Conclusion

In this work, we propose a new white-box attack for breaking deep generative model-based defenses. A major challenge in attacking these defenses is how to estimate the gradient both accurately and efficiently, as the gradient in these defenses is not analytical due to the inversion of deep generative model. We tackle this problem by a novel gradient approximation mechanism, which is further accelerated by an amortized attack strategy. The proposed attack outperforms state-of-the-art attack methods (e.g BPDA) largely in both terms of minimal distortion and computational efficiency. Our attack provides a generic tool for evaluating the true robustness of deep generative model-based defenses.

Leveraging our attack, we also investigate what cause the weakness of deep generative model-based defenses. We discover that the vulnerability of these defenses originates from the existence of the 'holes' in the latent spaces of the deep generative models, which can be exploited by an attack to generate off-manifold samples. These holes may not be easily avoided or detected in vanilla deep generative models. We further see that a more powerful deep generative model does not necessarily lead to better robustness and could even do harm to it, as could be seen from the case of ABS.

A potential way to improve deep generative model-based defenses is to restrict the power of the deep generative models appropriately. For example, we could choose the dimensionality of the latent space more carefully (e.g by setting it to be the intrinsic dimensionality), or add some regularizations to the latent representation like disentanglement (Chen et al., 2016; Mathieu et al., 2019) or sparsity (Tonolini et al., 2019; Zhou et al., 2020). By doing so, we might be able to prevent deep generative models from generating off-manifold samples or to detect them. We leave these to future works.

Acknowledgements

We thank the reviewers for their insightful comments. This project is supported by National Natural Science Foundation of China (No.61806009 and 61932001), PKU-Baidu Funding 2019BD005, Beijing Academy of Artificial Intelligence (BAAI) and Intelligent Manufacturing Action Plan of Industrial Solid Foundation Program (JCKY2018204C004).

References

Athalye, A., Carlini, N., and Wagner, D. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *International Conference on Machine Learning*, pp. 274–283, 2018.

Brendel, W., Rauber, J., and Bethge, M. Decision-based adversarial attacks: Reliable attacks against black-box machine learning models. In *ICLR*, 2018.

Carlini, N. and Wagner, D. Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy (SP)*, pp. 39–57. IEEE, 2017.

Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I., and Abbeel, P. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *NeurIPS*, pp. 2172–2180, 2016.

Cohen, J., Rosenfeld, E., and Kolter, Z. Certified adversarial robustness via randomized smoothing. In *International Conference on Machine Learning*, pp. 1310–1320, 2019.

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

Dong, Y., Liao, F., Pang, T., Su, H., Zhu, J., Hu, X., and Li, J. Boosting adversarial attacks with momentum. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 9185–9193, 2018.

Ghosh, P., Losalka, A., and Black, M. J. Resisting adversarial attacks using gaussian mixture variational autoencoders. In *Proceedings of the AAAI*, volume 33, pp. 541–548, 2019.

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

Gould, S., Fernando, B., Cherian, A., Anderson, P., Cruz, R. S., and Guo, E. On differentiating parameterized argmin and argmax problems with application to bi-level optimization. *arXiv preprint arXiv:1607.05447*, 2016.

Gretton, A., Borgwardt, K. M., Rasch, M. J., Schölkopf, B., and Smola, A. A kernel two-sample test. *Journal of Machine Learning Research*, 13(Mar):723–773, 2012.

Guo, C., Frank, J. S., and Weinberger, K. Q. Low frequency adversarial perturbation. In *UAI*, 2019.

Ilyas, A., Engstrom, L., Athalye, A., Lin, J., Athalye, A., Engstrom, L., Ilyas, A., and Kwok, K. Black-box adversarial attacks with limited queries and information. In *International Conference on Machine Learning*, 2018.

Kurakin, A., Goodfellow, I., and Bengio, S. Adversarial machine learning at scale. In *ICLR*, 2016.

Li, Y., Li, L., Wang, L., Zhang, T., and Gong, B. Nattack: Learning the distributions of adversarial examples for an improved black-box attack on deep neural networks. In *ICML*, pp. 3866–3876, 2019.

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

Lin, W.-A., Balaji, Y., Samangouei, P., and Chellappa, R. Invert and defend: Model-based approximate inversion of generative adversarial networks for secure inference. *arXiv preprint arXiv:1911.10291*, 2019.

Lyu, C., Huang, K., and Liang, H.-N. A unified gradient regularization family for adversarial examples. In *ICDM*, pp. 301–309. IEEE, 2015.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. Towards deep learning models resistant to adversarial attacks. In *ICLR*, 2018.

Mathieu, E., Rainforth, T., Siddharth, N., and Teh, Y. W. Disentangling disentanglement in variational autoencoders. 2019.

Radford, A., Metz, L., and Chintala, S. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

Ross, A. S. and Doshi-Velez, F. Improving the adversarial robustness and interpretability of deep neural networks by regularizing their input gradients. In *Thirty-second AAAI conference on artificial intelligence*, 2018.

Salman, H., Li, J., Razenshteyn, I., Zhang, P., Zhang, H., Bubeck, S., and Yang, G. Provably robust deep learning via adversarially trained smoothed classifiers. In *Advances in Neural Information Processing Systems*, pp. 11292–11303, 2019.

Samangouei, P., Kabkab, M., and Chellappa, R. Defense-gan: Protecting classifiers against adversarial attacks using generative models. In *ICLR*, 2018.

Schott, L., Rauber, J., Bethge, M., and Brendel, W. Towards the first adversarially robust neural network model on mnist. In *ICLR*, pp. 1–16, 2019.

Shaban, A., Cheng, C.-A., Hatch, N., and Boots, B. Truncated back-propagation for bilevel optimization. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pp. 1723–1732, 2019.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. Intriguing properties of neural networks. In *ICLR*, 2014.

Tonolini, F., Jensen, B. S., and Murray-Smith, R. Variational sparse coding. In *UAI*, 2019.

Xie, C., Wu, Y., Maaten, L. v. d., Yuille, A. L., and He, K. Feature denoising for improving adversarial robustness. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 501–509, 2019.

Zhang, H., Yu, Y., Jiao, J., Xing, E., El Ghaoui, L., and Jordan, M. Theoretically principled trade-off between robustness and accuracy. In *International Conference on Machine Learning*, pp. 7472–7482, 2019.

Zhou, K., Gao, S., Cheng, J., Gu, Z., Fu, H., Tu, Z., Yang, J., Zhao, Y., and Liu, J. Sparse-gan: Sparsity-constrained generative adversarial network for anomaly detection in retinal oct image. In *2020 IEEE 17th International Symposium on Biomedical Imaging (ISBI)*, pp. 1227–1231. IEEE, 2020.