GAT: GENERATIVE ADVERSARIAL TRAINING FOR ADVERSARIAL EXAMPLE DETECTION AND ROBUST CLASSIFICATION

Xuwang Yin  
Department of Electrical Engineering  
University of Virginia  
xy4cm@virginia.edu

Soheil Kolouri  
Information and Systems Sciences Laboratory  
HRL Laboratories, LLC.  
skolouri@hrl.com

Gustavo K. Rohde  
Department of Electrical Engineering  
University of Virginia  
gustavo@virginia.edu

ABSTRACT

The vulnerabilities of deep neural networks against adversarial examples have become a significant concern for deploying these models in sensitive domains. Devising a definitive defense against such attacks is proven to be challenging, and the methods relying on detecting adversarial samples are only valid when the attacker is oblivious to the detection mechanism. In this paper we first present an adversarial example detection method that provides performance guarantee to norm constrained adversaries. The method is based on the idea of training adversarial robust subspace detectors using generative adversarial training (GAT). The novel GAT objective presents a minimax problem similar to that of GANs; it has the same convergence property, and consequently supports the learning of class conditional distributions. We first demonstrate that the minimax problem could be reasonably solved by PGD attack, and then use the learned class conditional generative models to define generative detection/classification models that are both robust and more interpretable. We provide comprehensive evaluations of the above methods, and demonstrate their competitive performances and compelling properties on adversarial detection and robust classification problems. Code is available at https://github.com/xuwangyin/GAT-Generative-Adversarial-Training.

1 INTRODUCTION

Deep neural networks have become the staple of modern machine learning pipelines, achieving state-of-the-art performance on extremely difficult tasks in various applications such as computer vision (He et al., 2016), speech recognition (Amodei et al., 2016), machine translation (Vaswani et al., 2017), robotics (Levine et al., 2016), and biomedical image analysis (Shen et al., 2017). Despite their outstanding performance, these networks are shown to be vulnerable against various types of adversarial attacks, including evasion attacks (aka, inference or perturbation attacks) (Szegedy et al., 2013; Goodfellow et al., 2014b; Carlini & Wagner, 2017b; Su et al., 2019) and poisoning attacks (Liu et al., 2017; Shafahi et al., 2018). These vulnerabilities in deep neural networks hinder their deployment in sensitive domains including, but not limited to, health care, finances, autonomous driving, and defense-related applications and have become a major security concern.

Due to the mentioned vulnerabilities, there has been a recent surge toward designing defense mechanisms against adversarial attacks (Gu & Rigazio, 2014; Jin et al., 2015; Papernot et al., 2016b; Bastani et al., 2016; Madry et al., 2017; Sinha et al., 2018), which has in turn motivated the design of stronger attacks that defeat the proposed defenses (Goodfellow et al., 2014b; Kurakin et al., 2016b; Carlini & Wagner, 2017b; Xiao et al., 2018; Athalye et al., 2018; Chen et al., 2018; He
et al., 2018). Besides, the proposed defenses have been shown to be limited and often not effective and easy to overcome (Athalye et al., 2018). Alternatively, a large body of work has focused on detection of adversarial examples (Bhagoji et al., 2017; Feinman et al., 2017; Gong et al., 2017; Grosse et al., 2017; Metzen et al., 2017; Hendrycks & Gimpel, 2017; Li & Li, 2017; Xu et al., 2017; Pang et al., 2018; Roth et al., 2019; Bahat et al., 2019; Ma et al., 2018; Zheng & Hong, 2018; Tian et al., 2018). While training robust classifiers focuses on maintaining performance in presence of adversarial examples, adversarial detection only cares for detecting such examples.

The majority of the current detection mechanisms focus on non-adaptive threats, for which the attacks are not specifically tuned/tailored to bypass the detection mechanism, and the attacker is oblivious to the detection mechanism. In fact, Carlini & Wagner (2017a) and Athalye et al. (2018) showed that the detection methods presented in (Bhagoji et al., 2017; Feinman et al., 2017; Gong et al., 2017; Grosse et al., 2017; Metzen et al., 2017; Hendrycks & Gimpel, 2017; Li & Li, 2017; Ma et al., 2018), are significantly less effective than their claims under adaptive attacks. Overall, current solutions are mostly heuristic approaches that cannot provide performance guarantees to the adversary they considered.

In this paper we present a detection mechanism that can withstand adaptive attacks. The idea is to partition the input space into subspaces based on the classification system’s output and perform adversarial/natural sample classification in these subspaces. The partition allows us to drop the adversarial constrain and employ a novel generative adversarial training (GAT) objective to train robust binary classifiers in subspaces. The robustness could be further explained by the fact that GAT objective presents a minimax problem that supports the learning of class conditional distributions.

Our specific contributions are:

- We develop adversarial example detection techniques that provide performance guarantees to norm constrained adversaries. Empirically, our best models improve previous state-of-the-art mean $L_2$ distortion from 3.68 to 5.65 on MNIST dataset, and from 1.1 to 1.5 on CIFAR10 dataset.
- We study powerful and versatile generative classification models derived from our detection framework and demonstrate their competitive performances over discriminative robust classifiers. While defense mechanisms based on ordinary adversarial training are vulnerable to rubbish examples, inputs that cause confident predictions of our models have human-understandable semantic meanings.
- We propose the GAT objective, that presents a minimax problem similar to that of GANs, and supports the learning of class conditional distributions. We used PGD attack (Madry et al., 2017) to solve the optimization problem, and obtained competitive generative modeling results on CIFAR10, ImageNet, and synthetic datasets.

2 RELATED WORKS

Adversarial attacks. Since the pioneering work of Szegedy et al. (2013), a large body of work has focused on designing algorithms that achieve successful attacks on neural networks (Goodfellow et al., 2014b; Moosavi-Dezfooli et al., 2016; Kurakin et al., 2016b; Chen et al., 2018; Papernot et al., 2016a; Carlini & Wagner, 2017b). More recently, iterative projected gradient descent (PGD), has been empirically identified as the most effective approach for performing norm constrained attacks, and the attack reasonably approximates the optimal attack (Madry et al., 2017).

Adversarial detection techniques. The majority of the methods developed for detecting adversarial attacks are based on the following core idea: given a trained $K$-class classifier, $f: \mathbb{R}^d \rightarrow \{1...K\}$, and its corresponding natural training samples, $D = \{x_i \in \mathbb{R}^d\}_{i=1}^{N}$, generate a set of adversarially attacked samples $D' = \{x'_j \in \mathbb{R}^d\}_{j=1}^{M}$, and devise a mechanism to discriminate $D$ from $D'$. For
instance, Gong et al. (2017) use this exact idea and learn a binary classifier to distinguish the natural and adversarially perturbed sets. Similarly, Grosse et al. (2017) append a new “attacked” class to the classifier, \( f \), and re-train a secured network that classifies natural images, \( x \in \mathcal{D} \), into the \( K \) classes and all attacked images, \( x' \in \mathcal{D}' \), to the \((K+1)\)-th class. In contrast to Gong et al. (2017); Grosse et al. (2017), which aim at detecting adversarial examples directly from the image content, Metzen et al. (2017) trained a binary classifier that receives as input the intermediate layer features extracted from the classifier network \( f \), and distinguished \( \mathcal{D} \) from \( \mathcal{D}' \) based on such input features. More importantly, Metzen et al. (2017) considered the so-called case of adaptive/dynamic adversary and proposed to harden the detector against such attacks using a similar adversarial training approach as in Goodfellow et al. (2014b). Unfortunately, the mentioned detection methods are significantly less effective under an adaptive adversary equipped with a strong attack (Carlini & Wagner, 2017a; Athalye et al., 2018).

3 Adversarial Detection Methods

3.1 Integrated adversarial detection

For a \( K (K \geq 2) \) class classification problem, given a dataset of natural samples \( \mathcal{D} = \{x_i\}_{i=1}^N, x_i \in \mathbb{R}^d \), along with labels \( \{y_i\}_{i=1}^N, y_i \in \{1...K\} \), let \( f: \mathbb{R}^d \rightarrow \{1...K\} \) be a classifier that is used to do classification on \( \mathcal{D} \). With the labels and predicted labels the dataset respectively forms the partition \( \mathcal{D} = \bigcup \mathcal{D}_k \) and \( \mathcal{D}' = \bigcup \mathcal{D}'_k \), where \( \mathcal{D}_k = \{x : y = k, x \in \mathcal{D}\} \), and \( \mathcal{D}'_k = \{x : f(x) = k, x \in \mathcal{D}'\} \). Let \( \mathcal{H} = \{h_k\}_{k=1}^K, h_k: \mathbb{R}^d \rightarrow \{0, 1\} \) be a set of binary classifiers (detectors), in which \( h_k \) is trained to discriminate natural samples classified as \( k \), from adversarial samples that fool to be classified as \( k \). Also, let \( \mathcal{D}' \) be a set of \( L_p \) norm bounded adversarial examples crafted from \( \mathcal{D} \): \( \mathcal{D}' = \{x + \delta : f(x + \delta) \neq y, f(x) = y, x \in \mathcal{D}, \delta \in \mathcal{S}\} \), \( \mathcal{S} = \{\delta \in \mathbb{R}^d \mid \|\delta\|_p \leq \epsilon\} \). Consider the following procedure to determine whether a sample \( x \) in \( \mathcal{D} \cup \mathcal{D}' \) is an adversarial example:

First obtain the estimated class label \( k = f(x) \), then use the \( k \)-th detector to predict: if \( h_k(x) = 1 \) then categorize \( x \) as a natural sample, otherwise categorize it as an adversarial example.

The detection accuracy of the algorithm is given by

\[
\sum_{k=1}^K \left| \left\{ x : h_k(x) = 1, x \in \mathcal{D}'_k \right\} \right| + \left| \left\{ x : h_k(x) = 0, x \in \mathcal{D}'_k \right\} \right| \frac{1}{|\mathcal{D}| + |\mathcal{D}'|},
\]

where \( \mathcal{D}'_k = \{x : f(x) = k, x \in \mathcal{D}'\} \). Hence minimizing the algorithm’s classification error is equivalent to minimizing classification error of individual detectors. Employing empirical risk minimization, detector \( k \), parameterized by \( \theta_k \), is trained by

\[
\theta_k^* = \arg \min_{\theta_k} \mathbb{E}_{x \sim \mathcal{D}'_k} \left[ L(h_k(x; \theta_k), 0) \right] + \mathbb{E}_{x \sim \mathcal{D}'_k} \left[ L(h_k(x; \theta_k), 1) \right],
\]

where \( L \) is a loss function that measures the distance between \( h_k \)’s output and the supplied label (e.g., the binary cross-entropy loss).

In the case of adaptive attacks, when the adversary aims to fool both the classifier and detectors, the accuracy of a naively trained detector could be significantly reduced. In order to be robust to adaptive attacks, by taking a similar approach as Madry et al. (2017) we incorporate the attack into the training objective:

\[
\min_{\theta_k} \rho(\theta_k), \quad \rho(\theta_k) = \mathbb{E}_{x \sim \mathcal{D}'_k} \left[ \max_{\delta \in \mathcal{S}, f(x+\delta) = k} L(h_k(x+\delta; \theta_k), 0) \right] + \mathbb{E}_{x \sim \mathcal{D}'_k} \left[ L(h_k(x; \theta_k), 1) \right],
\]

where \( \mathcal{D}'_k = \{x : f(x) \neq k, y \neq k, x \in \mathcal{D}\} \), and we assume that perturbation budget is large enough such that \( \forall x \in \mathcal{D}'_k, \exists \delta \in \mathcal{S}, \text{s.t. } f(x+\delta) = k \). Now by dropping the \( f(x+\delta) = k \) constrain we could derive an upper bound for the first loss term:

\[
\max_{\delta \in \mathcal{S}, f(x+\delta) = k} L(h_k(x+\delta; \theta_k), 0) \leq \max_{\delta \in \mathcal{S}} L(h_k(x+\delta; \theta_k), 0).
\]
The detector could instead be trained by minimizing this upper bound using the following unconstrained objective,

$$
\rho(\theta_k) = \mathbb{E}_{x \sim \mathcal{D}_X} \left[ \max_{\delta \in \mathcal{S}} L(h_k(x + \delta; \theta_k), 0) \right] + \mathbb{E}_{x \sim \mathcal{D}_X} \left[ L(h_k(x; \theta_k), 1) \right].
$$

(4)

Further, we use the fact that when \( \mathcal{D} \) is used as the training set, \( f \) could overfit on \( \mathcal{D} \) such that \( \mathcal{D}_{\setminus k} = \{ x_i : y_i \neq k \} \) and \( \mathcal{D}_k \) are respectively good approximations of \( \mathcal{D}_{\setminus k} \) and \( \mathcal{D}_k \). This leads to our proposed generative adversarial training (GAT) objective:

$$
\min_{\theta_k} \rho(\theta_k), \quad \text{where} \quad \rho(\theta_k) = \mathbb{E}_{x \sim \mathcal{D}_X} \left[ \max_{\delta \in \mathcal{S}} L(h_k(x + \delta; \theta_k), 0) \right] + \mathbb{E}_{x \sim \mathcal{D}_X} \left[ L(h_k(x; \theta_k), 1) \right].
$$

(5)

In a nutshell, each detector is trained using in-class natural samples and detector-adversarial examples crafted from out-of-class samples. We use iterative PGD attack (Madry et al., 2017) to solve the inner maximization. The outer minimization is solved using regular gradient descent based optimization methods.

3.2 Generative Adversarial Detection/Classification

If we assume \( L \) to be the binary cross-entropy loss, the minimax problem presented in objective 5 is then similar to that of GANs, and it is straightforward to use GANs’ convergence theorem to show convergence of the out-of-class distribution to the target class distribution (assuming large enough perturbation budget). The perturbation limit provides a mechanism for constraining the optimization such that the discriminator (detector) could retain density information and not converge to a degenerate uniform solution as in the case of GANs (Goodfellow et al., 2014a; Dai et al., 2017). In Appendix F we provide experimental results on 1D and 2D benchmark datasets to demonstrate GAT’s capability to do density estimation.

Unfortunately, the detectors don’t define explicit density functions. Under the energy-based learning framework (LeCun et al., 2006), we could, however, obtain the joint probability of the input and a class category using the Gibbs distribution: \( p(x, k) = \frac{\exp(-E_{\theta_k}(x))}{Z_{\theta}}, \) where \( E_{\theta_k}(x) = -z(h_k(x)) \), and \( Z_{\theta} = \int \exp(-E_{\theta_k}(x))dx \) is an intractable normalizing constant known as the partition function. We could then apply the Bayes classification rule to obtain a generative classifier: \( H(x) = \arg \max_k p(x, k) = \arg \max_k z(h_k(x)) \). In addition, we could base on \( p(x, k) \) to reject low probability inputs. We implement the reject option by thresholding \( \hat{k} \)-th detector’s logit output, where \( \hat{k} \) is the predicted class. In the context of adversarial example detection, rejected samples are considered as adversarial examples.

4 Evaluation Methodology

4.1 Robustness Test

We first test the robustness of individual detectors. We show that, once a detector is trained with an adequately configured PGD attack, its performance cannot be significantly reduced by an adversary with much stronger configurations. We note that although the PGD attack has been shown (Madry et al., 2017) to be able to induce robustness in ordinary adversarial training, the test is necessary as it is not clear whether the optimization landscape of the generative objective is the same as its discriminative counterpart. For instance, we found that the step size used by Madry et al. (2017) to train their CIFAR10 robust classifier would not induce robustness to our detectors (see Appendix C.2.2). Furthermore, we use adversarial finetuning on CIFAR10 and ImageNet to speedup detector training. With the robustness test, we show that PGD attack also introduces robustness within this new training paradigm.

We use AUC (area under the ROC Curve) to measure detection performances. The metric could be interpreted as the probability that the detector assigns a higher score to a random positive sample than to a random negative example. While true positive rate and false positive rate are the commonly used metrics for measuring detection performances, they require a detection threshold to be specified. AUC, however, is an aggregated measurement of detection performance across a range of thresholds, and we found it to be a more stable and reliable metric. For the \( k \)-th detector \( h_k \), its AUC is computed
on the set \( \{ (x, 0) : x \in D'_k \} \cup \{ (x, 1) : x \in D'_k \} \), where \( D'_k = \{ \arg \max_{x+i} L(h_k(x + \delta; \theta_k), 0) : x \in D'_k \} \) (refer to loss 4).

4.2 Detection Performance

Having validated the robustness of individual detectors, we evaluate the overall performance of our integrated detection system. Recalling our detection rule, we first obtain the estimated class label having validated the robustness of individual detectors, we evaluate the overall performance of our

\[ \delta \in \{ 1, \ldots, K \} \]

\( T_k \) on the set \( \{ (x, y) : x \in D \} \). We define as

\[ D = \{ (x_i, y_i) \}_{i=1}^N \]

\( D' = \{ (x_i + \delta, y_i) \}_{i=1}^N \) to denote the test set that contains natural samples, and \( D' = \{ (x_i + \delta, y_i) \}_{i=1}^N \) to denote the corresponding perturbed test set. For a given threshold \( T_k \), we compute the true positive rate (TPR) on \( D \) and false positive rate (FPR) on \( D' \). These two metrics are respectively defined as

\[ \text{TPR} = \frac{1}{N} \left| \{ x : z(h_k(x)) \geq T, k = f(x), x \in D \} \right|, \]

\[ \text{FPR} = \frac{1}{N} \left| \{ x : z(h_k(x)) \geq T, k = f(x), f(x) \neq y, (x, y) \in D' \} \right|. \]

In the FPR definition we use \( f(x) \neq y \) to constrain that only true adversarial examples are counted as false positives. This constraint is necessary, as we found that for the norm ball constraint we considered in the experiments, not all perturbed samples are adversarial examples that cause misclassification on \( f \).

In order to craft the perturbed dataset \( D' \), we consider three attacking scenarios.

**Classifier attack.** This attack corresponds to the scenario where the adversary is oblivious to the detection mechanism. For a given natural sample \( x \) and its label \( y \), the perturbed sample \( x' \) is computed by minimizing the loss,

\[ L(x') = z(f(x'))_y - \max_{i \neq y} z(f(x'))_i, \]

where \( z(f(x')) \) is the classifier’s logit outputs. This objective is derived from the CW attack (Carlini & Wagner, 2017b) and used in MadryLab (b;a) to perform untargeted attacks.

**Detectors attack.** In this scenario adversarial examples are produced by attacking only the detectors. We construct a single detection function \( H \) by using the \( i \)-th detector’s logit output as its \( i \)-th logit output: \( z(H(x))_i = z(h_i(x)) \). \( H \) is then recognized as a single network, and the perturbed sample \( x' \) for a given input \( (x, y) \) is computed by minimizing the loss

\[ L(x') = -\max_{i \neq y} z(H(x'))_i. \]

According to our detection rule, a low value of the detector’s logit output indicates detection of an adversarial example, thus by minimizing the negative of logit output we make the perturbed example harder to detect. \( H \) could also be fed directly to the CW loss 8 or to the cross-entropy loss, but we found the attack based on the loss in 9 to be significantly more effective.

**Combined attack.** With the goal of fooling both the classifier and detectors, perturbed samples are produced by attacking the integrated detection system. We consider two loss functions for realizing the combined attack. The first is based on the combined loss function (Carlini & Wagner, 2017a) that has been shown to be effective against existing detection methods. Given a natural example \( x \) and its label \( y \), same as the detectors-attack scenario, we first construct a single detection function \( H \) by aggregating the logit outputs of individual detectors: \( z(H(x))_i = z(h_i(x)) \). We then use the aggregated detector’s largest logit output \( \max_{i \neq y} z(H(x))_i \) (low value of this quantity indicates detection of an adversarial example) and the classifier logit outputs \( z(f(x)) \) to construct a surrogate classifier \( g \), with its logit outputs being

\[ z(g(x))_i = \begin{cases} z(f(x))_i, & \text{if } i \leq K, \\ \left(-\max_{j \neq y} z(H(x))_j + 1\right) \cdot \max_{j} z(f(x))_j, & \text{if } i = K + 1. \end{cases} \]
A perturbed example $x'$ is then computed by minimizing the loss function
\[
L(x') = \max_i z(g(x'))_i - \max_{i \neq y} z(f(x'))_i.
\] (11)

In practice we observe that the optimization of this loss tends to stuck at the point where $\max_{i \neq y} z(f(x'))_i$ keeps changing signs while $\max_{j \neq y} z(H(x))_j$ stays as a large negative number (which indicates detection). To derive a more effective attack we consider a simple combination of loss 8 and loss 9:
\[
L(x') = \begin{cases} 
  z(f(x'))_y - \max_{i \neq y} z(f(x'))_i, & \text{if } z(f(x'))_y \geq \max_{i \neq y} z(f(x'))_i, \\
  -\max_{i \neq y} z(H(x'))_i, & \text{else.}
\end{cases}
\] (12)

The objective is straightforward: if $x'$ is not yet an adversarial example on $f$, optimize it for that goal; otherwise optimize it for fooling the aggregated detector.

We mention briefly here that we perform the same performance analysis of the generative detection method as detailed in Section 3.2, by computing TPR on $D$ and FPR on $D'$, where $D'$ is computed using loss 9, loss 8 and the cross-entropy loss.

4.3 Robust Classification Performance

Integrated classification. In addition to the generative classifier proposed in Section 3.2, we introduce another classification scheme that comes with a reject option. The scheme is based on a combination of the naive classifier $f$ and the detectors: for a given input $x$ and its prediction label $k = f(x)$, if $z(h_k(x)) < T$, then $x$ is rejected, otherwise it’s classified as $k$. We respectively use loss 12 and 9 to attack the integrated classifier and the generative classifier.

Performance metric. In the context of robust classification, the performance of a robust classifier is measured using standard accuracy and robust accuracy — accuracies respectively computed on the natural dataset and perturbed dataset. On the natural dataset $D = \{(x_i, y_i)\}_{i=1}^N$, we compute the accuracy as the fraction of samples that are correctly classified ($f(x) = y$) and at the same time not rejected ($z(h_k(x)) \geq T$):
\[
\text{Accuracy} = \frac{1}{N} |\{x : z(h_k(x)) \geq T, k = f(x), f(x) = y, (x, y) \in D\}|.
\] (13)

On the perturbed dataset $D' = \{(x_i + \delta_i, y_i)\}_{i=1}^N$ we compute the error as the fraction of samples that are misclassified ($f(x) \neq y$) and at the same time not rejected:
\[
\text{Error} = \frac{1}{N} |\{x : z(h_k(x)) \geq T, k = f(x), f(x) \neq y, (x, y) \in D'\}|.
\] (14)

Note that in this case the error is no longer a complement of the accuracy. For a classification system with a reject option, any perturbed samples that are rejected should be considered as properly handled, regardless of whether they are misclassified. Thus on the perturbed dataset, the error, which is the fraction of misclassified and not rejected samples, is a more proper notion of such system’s performance. For a standard softmax robust classifier that comes without an reject option, its perturbed set error is computed as the complement of its accuracy on the perturbed set.

5 Experiments

5.1 MNIST

Using different $p$-norm and maximum perturbation $\epsilon$ constrains we trained four detection systems (each has 10 base detectors), with training and validation adversarial examples optimized using PGD attacks of different steps and step sizes (see Table 6). At each step of PGD attack we use the Adam optimizer to perform gradient descent, both for $L_2$ and $L_\infty$ constrained scenarios. Appendix A.1 provides more training details.

Robustness results. The robustness test results in Table 1 confirm that the base detectors trained with objective 5 are able to withstand much stronger PGD attacks, for both $L_2$ and $L_\infty$ scenarios. In Table 8 we present the robustness test result using normalized steepest descent based attack. We
Table 1: AUC scores of the first two detectors \((k = 0, 1)\) tested with different strengths of Adam based PGD attacks.

<table>
<thead>
<tr>
<th>PGD attack steps, step size</th>
<th>(L_\infty \epsilon = 0.3) detectors</th>
<th>(L_\infty \epsilon = 0.5) detectors</th>
</tr>
</thead>
<tbody>
<tr>
<td>200, 0.01</td>
<td>0.99959</td>
<td>0.99971</td>
</tr>
<tr>
<td>2000, 0.005</td>
<td>0.99958</td>
<td>0.99971</td>
</tr>
</tbody>
</table>

Table 2: MNIST mean \(L_2\) distortion (higher is better) of perturbed samples when the detection method has 1.0 FPR on perturbed set and 0.95 TPR on natural set.

<table>
<thead>
<tr>
<th>Detection method</th>
<th>Mean (L_2) distortion</th>
</tr>
</thead>
<tbody>
<tr>
<td>State-of-the-art (Carlini &amp; Wagner, 2017a)</td>
<td>3.68</td>
</tr>
<tr>
<td>Ours (use (L_\infty \epsilon = 0.3) trained base detectors)</td>
<td>4.40</td>
</tr>
<tr>
<td>Ours (use (L_\infty \epsilon = 0.5) trained base detectors)</td>
<td>5.65</td>
</tr>
</tbody>
</table>

also performed other types of tests, including cross-norm test, cross-perturbation test, and random restart test. The results of these tests, along with the complete robustness test results on \(L_\infty \epsilon = 0.3\) and \(L_\infty \epsilon = 0.5\) trained detectors, are presented in Appendix C.1.

**Detection results.** Figure 2a shows that generative detection outperforms integrated detection (compare their performances under their respective most effective attacks). Combined attack, as expected, is the most effective attack against integrated detection. Attack based on loss 9 is the most effective attack against generative detection (see Figure 7). Notably, the red curve that overlaps the y-axis indicates that integrated detection is able to perfectly detect adversarial examples crafted by attacking only the classifier (using objective 8).

In Table 2 we list the performances of the generative detection method along with the state-of-the-art detection method as identified by Carlini & Wagner (2017a). Our method, apart from being able to provide performance guarantee, outperforms the state-of-the-art method by large margins. (Please refer to Appendix B for a description of how we compute the mean \(L_2\) distortions.)

**Classification results.** In Figure 2b we plot the robust classification performance of our classification methods and a state-of-the-art discriminative robust classifier (Madry et al., 2017). Our methods provide reject options that allow the user to find a balance between standard accuracy and robust error by adjusting the rejection threshold. We observe that a stronger attack \((\epsilon = 0.4)\) breaks the robust classifier (as indicated by the right red cross), while the generative classifier still exhibits robustness, even though both systems are trained with \(L_\infty \epsilon = 0.3\) constrain.

Figure 2: (a) Performances of integrated detection and generative detection under \(L_\infty \epsilon = 0.3\) constrained attack. (b) Performances of the integrated classifier (discussed in Section 4.3) and generative classifier under \(L_\infty \epsilon = 0.3\) constrained and \(L_\infty \epsilon = 0.4\) constrained attacks. The performances of the robust classifier (Madry et al., 2017) (accuracy 0.984, error 0.08 at \(\epsilon = 0.3\), and accuracy 0.984, error 0.941 at \(\epsilon = 0.4\)) are indicated with red cross marks. PGD attack steps 100, step size 0.01.

Figure 3 shows perturbed samples produced by performing targeted attacks against the generative classifier and robust classifier. The generative classifier’s perturbation samples have distinguishable
visible features of the target class, indicating that the base detectors, from which the generative classifier is built, have learned the class conditional distributions, and the perturbations have to change the semantics for a successful attack. In contrast, perturbations introduced by attacking the robust classifier are not interpretable, even though they could cause high logit output of the target classes (see Figure 8 for the logit outputs distribution). Following this path and with a larger perturbation limit, it is straightforward to generate completely unrecognizable inputs that fool the discriminative robust classifier.

Figure 3: Natural samples and corresponding perturbed samples produced by performing a targeted attack against the generative classifier and robust classifier (Madry et al., 2017). Targets from top row to bottom row are digit class from 0 to 9. We perform the targeted attack by maximizing the logit output of the targeted class, using $L_\infty \epsilon = 0.4$ constrained PGD attack of steps 100 and step size 0.01. Both classifiers are trained with $L_\infty \epsilon = 0.3$ constraint.

5.2 CIFAR10

On CIFAR10 we train the base detectors using $L_\infty \epsilon = 8$ constrain PGD attack of steps 40 and step size 0.5. Note that the scale of $\epsilon$ and step size here is 0-255 (rather than 0-1 as in the case of MNIST). The robust classifier (Madry et al., 2017) that we compare with is trained with the same $L_\infty \epsilon = 8$ constraint but with a different step size (see Appendix C.2.2 for a discussion of the effects of step size). Appendix A.2 provides the training details.

Robustness results. Table 3 shows that the base detector models can withstand attacks that are significantly stronger than the training attack. In Appendix C.2.1 we report random restart test results, cross-norm and cross-perturbation test results, and robustness test result for $L_2$ based models.

Table 3: AUC scores of the first two $L_\infty \epsilon = 8$ base detectors under different strengths of the $L_\infty \epsilon = 8$ constrained PGD attack.

<table>
<thead>
<tr>
<th>$L_\infty \epsilon = 8$ detectors</th>
<th>$L_\infty$ attack steps and step size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base detector $k = 0$</td>
<td>0.9224 0.9234 0.9231 0.9205 0.9203</td>
</tr>
<tr>
<td>Base detector $k = 1$</td>
<td>0.9533 0.9553 0.9550 0.9504 0.9500</td>
</tr>
</tbody>
</table>

Detection results. Consistent with the MNIST results, in Figure 4a combined attack is the most effective method against integrated detection. Generative detection outperforms integrated detection when the detection threshold is low (i.e., when TPR is high). In this figure we use loss 9 to attack generative detection, and in Figure 9 we show that this loss is more effective than cross-entropy loss and CW loss. In Table 4 our method outperforms the state-of-the-art adversarial detection method.

Classification results. Contrary to MNIST’s result, we did not observe a dramatic decrease in the robust classifier’s performance when we increase the perturbation limit to $\epsilon = 12$ (Figure 4b). Integrated classification can reach the standard accuracy of a regular classifier, but at the cost of significantly increased error on the perturbed set. Figure 5 shows some perturbed samples produced by attacking the generative classifier and robust classifier. While these two classifiers have similar errors on the perturbed set, samples produced by attacking the generative classifier have more visible
Table 4: CIFAR10 mean $L_2$ distortion (higher is better) of perturbed samples when the detection method has 1.0 FPR on perturbed set and 0.95 TPR on natural set. Appendix B provides details about how the mean $L_2$ distances are computed.

<table>
<thead>
<tr>
<th>Detection method</th>
<th>Mean $L_2$ distortion (0-1 scale)</th>
</tr>
</thead>
<tbody>
<tr>
<td>State-of-the-art (Carlini &amp; Wagner, 2017a)</td>
<td>1.1</td>
</tr>
<tr>
<td>Ours (use $L_\infty$ $\epsilon = 8.0$ trained models)</td>
<td>1.5</td>
</tr>
</tbody>
</table>

Figure 4: (a) Performances of generative detection and integrated detection under $L_\infty$ $\epsilon = 8$ attack. (b) Performances of integrated classifier (discussed in Section 4.3) and generative classifier under $L_\infty$ $\epsilon = 8$ constrained and $L_\infty$ $\epsilon = 12$ constrained attacks. The performances of the robust classifier (Madry et al., 2017) (accuracy 0.8735, error 0.5311 at $\epsilon = 8$, and accuracy 0.8735, error 0.7087 at $\epsilon = 12$) are annotated. PGD attack step size 2.0, steps 20 for $\epsilon = 8$, and 30 for $\epsilon = 12$.

features of the targets, suggesting that the adversary has to change more semantic to cause the same error.

Figures 6 and 10 demonstrate that hard to recognize images are able to cause high logit outputs of the robust classifier. Such examples highlight a major defect of the defense mechanisms based on ordinary adversary training: they could be easily fooled by unrecognizable inputs (Nguyen et al., 2015; Goodfellow et al., 2014b; Schott et al., 2018). In contrast, samples that cause high logit outputs of the generative classifier all have clear semantic meaning. In Figure 13 we present image synthesis results using $L_\infty$ $\epsilon = 16$ constrained detectors. These results conform to the theory that GAT supports the learning of class conditional distributions. (See Appendix D for Gaussian noise attack results and a discussion about the interpretability of the generative classification approach.)

Figure 5: Natural samples and corresponding perturbed samples by performing targeted attack against the generative classifier and robust classifier (Madry et al., 2017). The targeted attack is performed by maximizing the logit output of the targeted class. We use $L_\infty$ $\epsilon = 12$ constrained PGD attack of steps 30 and step size 2.0 to produce these samples.
Figure 6: Images generated from class conditional Gaussian noise by performing targeted attack against the generative classifier and robust classifier. We use PGD attack of steps 60 and step size $0.5 \times 255$ to perform $L_2 \epsilon = 30 \times 255$ constrained attack (same as Santurkar et al. (2019)). The Gaussian noise inputs from which these two plots are generated are the same. Samples not selected.

5.3 IMAGE NET

On ImageNet we show GAT induces detection robustness and supports the learning of class conditional distributions. Our experiment is based on Restricted ImageNet (Tsipras et al., 2018), a subset of ImageNet that has its samples reorganized into customized categories. The dog category consists of images of different breeds collected from ImageNet class from 151 to 268. We trained a dog class detector by finetuning a pre-trained ResNet50 (He et al., 2016) model. The dog category covers a range of ImageNet classes, with each one having its logit output. We use the subnetwork defined by the logit output of class 151 as the detector (in principle logit output of other classes in the range should also work). Due to computational resource constraints, we only validated the robustness of a $L_\infty \epsilon = 0.02$ trained detector (trained with PGD attack of steps 40 and step size $0.001$), and we present the result in Table 5. (On Restricted ImageNet in the case of $L_\infty$ scenario Tsipras et al. (2018) only demonstrates the robustness of a $\epsilon = 0.005$ constrained model). Please refer to Appendix C.3 for more results of adversarial examples, image synthesis, and image enhancement.

Table 5: AUC scores of the dog detector under different strengths of $L_\infty \epsilon = 0.02$ constrained PGD attacks

<table>
<thead>
<tr>
<th>Attack steps, step size</th>
<th>40, 0.001</th>
<th>100, 0.001</th>
<th>200, 0.001</th>
<th>40, 0.002</th>
<th>200, 0.002</th>
<th>200, 0.0005</th>
</tr>
</thead>
<tbody>
<tr>
<td>AUC</td>
<td>0.9720</td>
<td>0.9698</td>
<td>0.9692</td>
<td>0.9703</td>
<td>0.9690</td>
<td>0.9698</td>
</tr>
</tbody>
</table>

6 CONCLUSION

We studied the problem of adversarial example detection under the robust optimization framework and proposed a novel detection method based on input space partitioning. Our formulation leads to a new generative modeling technique which we called generative adversarial training (GAT). GAT's capability to learn class conditional distributions further gives rise to generative detection/classification approaches that show competitive performance and improved interpretability. In particular, our generative classification method is more resistant to “rubbish examples”, a major threat to even the most successful defense mechanisms. High computational cost (see Appendix E for more discussion) is a major drawback of our methods, and in the future, we will explore the idea of shared computation between detectors.
REFERENCES


Shibani Santurkar, Dimitris Tsipras, Brandon Tran, Andrew Ilyas, Logan Engstrom, and Aleksander Madry. Image synthesis with a single (robust) classifier, 2019.


A Training Details

A.1 MNIST Training

We use 50K samples from the original training set for training and the rest 10K samples for validation, and report test performances based on the epoch-saved checkpoint that gives the best validation performance. All base detectors are trained using a network consisting of two max-pooled convolutional layers each with 32 and 64 filters, and a fully connected layer of size 1024: same as the one used in Madry et al. (2017). At each iteration we sample a batch of 320 samples, from which in-class samples are used as positive samples, and out-of-class samples are used for crafting adversarial examples that will be used as negative samples. To balance positive and negative examples at each batch, we resample the out-of-class set to have same number of samples as in-class set. All base detectors are trained for 100 epochs.
Table 6: MNIST dataset PGD attack steps and step sizes for base detector training and validation.

<table>
<thead>
<tr>
<th></th>
<th>$L_2$ models</th>
<th>$L_\infty$ models</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\epsilon$</td>
<td>$\epsilon$</td>
<td>$\epsilon$</td>
</tr>
<tr>
<td>Train</td>
<td>100, 0.1</td>
<td>100, 0.01</td>
</tr>
<tr>
<td>Validation</td>
<td>200, 0.1</td>
<td>200, 0.01</td>
</tr>
</tbody>
</table>

A.2 CIFAR10 training

We train CIFAR10 base detectors using a ResNet model (same as the one used by Madry et al. (2017); MadryLab (a)). To speedup training, we take advantage of a natural trained classifier: the subnetwork of $f$ that defines the output logit $z(f(\cdot))_k$ is essentially a “detector”, that would output high values for samples of class $k$, and low values for others. The base detector is then trained by finetuning the subnetwork using objective 5. The pretrained classifier has a test accuracy of 95.01% (fetched from MadryLab (a)).

At each iteration of training we sample a batch of 300 samples, from which in-class samples are used as positive samples, while an equal number of out-of-class samples are used for crafting adversarial examples. Adversarial examples for training $L_2$ and $L_\infty$ models are both optimized using normalized steepest descent based PGD attacks (MadryLab, b). We report results based on the best performances on the CIFAR10 test set (thus don’t claim generalization performance of the proposed method).

B Computing mean $L_2$ distance

We first find the detection threshold $T$ with which the detection system has 0.95 TPR. We construct a new loss function by adding a weighted loss term that measures perturbation size to objective 9

$$L(x’) = -\max_{i \neq y} z(H(x’))_i + c \cdot \|x’ - x\|_2^2$$  \hspace{1cm} (15)

We then use unconstrained PGD attack to optimize $L(x’)$. We use binary search to find the optimal $c$, where in each bsearch attempt if $x’$ is a false positive ($\max_i z(H(x’))_i \neq y$ and $\max_{i \neq y} z(H(x’))_i > T$) we consider the current $c$ as effective and continue with a larger $c$. The configurations for performing binary search and PGD attack are detailed in Table 7. The $c$ upper bound is established such that with this upper bound, no samples except those that are inherently misclassified by the generative classifier, could be perturbed as a false positive. With these settings, our MNIST $L_\infty \epsilon = 0.3$ and $L_\infty \epsilon = 0.5$ generative detection models respective reached 1.0 FPR and 0.9455 FPR, and CIFAR10 generative model reached 0.9995 FPR.

Due to the technical and computational difficulties of finding the optimal $c$ in loss 15 using binary search, performances based on mean $L_2$ distortion are not precise, and we encourage future work to measure detection performances based on norm constrained attacks.

Table 7: Binary search and PGD attack configurations on MNIST and CIFAR10 dataset

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Initial $c$</th>
<th>$\epsilon$ lower bound</th>
<th>$\epsilon$ upper bound</th>
<th>bsearch depth</th>
<th>PGD steps</th>
<th>PGD step size</th>
<th>PGD optimizer</th>
</tr>
</thead>
<tbody>
<tr>
<td>MNIST</td>
<td>0.0</td>
<td>0.0</td>
<td>8.0</td>
<td>20</td>
<td>1000</td>
<td>1.0 (0-1 scale)</td>
<td>Adam</td>
</tr>
<tr>
<td>CIFAR10</td>
<td>0.0</td>
<td>0.0</td>
<td>1.0</td>
<td>20</td>
<td>100</td>
<td>2.56 (0-255 scale)</td>
<td>$L_2$ normalized steepest descent</td>
</tr>
</tbody>
</table>

C More experimental results

C.1 More MNIST results
Table 8: AUC scores of the first two base detectors under different strengths of normalized steepest descent based PGD attacks. The gradient descent rules for $L_2$ and $L_\infty$ constrained attacks are respectively $x_{n+1} = x_n - \gamma \nabla f(x_n)$ and $x_{n+1} = x_n - \gamma \cdot \text{sign}(\nabla f(x_n))$.

<table>
<thead>
<tr>
<th>PGD attack steps, step size</th>
<th>$L_\infty$ $\epsilon = 0.3$ base detector</th>
<th>$L_\infty$ $\epsilon = 0.5$ base detector</th>
<th>PGD attack steps, step size</th>
<th>$L_2$ $\epsilon = 2.5$ base detector</th>
<th>$L_2$ $\epsilon = 5.0$ base detector</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$k = 0$</td>
<td>$k = 1$</td>
<td></td>
<td>$k = 0$</td>
<td>$k = 1$</td>
</tr>
<tr>
<td>200, 0.01</td>
<td>0.99962</td>
<td>0.99973</td>
<td>200, 0.01</td>
<td>0.99906</td>
<td>0.99997</td>
</tr>
<tr>
<td>2000, 0.005</td>
<td>0.99959</td>
<td>0.99971</td>
<td>2000, 0.05</td>
<td>0.99855</td>
<td>0.99983</td>
</tr>
</tbody>
</table>

Table 9: AUC scores of the first two base detectors under cross-norm and cross-perturbation attacks. $L_\infty$ based attacks use steps 200 and step size 0.01, and $L_2$ based attacks uses steps 200 and step size 0.1.

<table>
<thead>
<tr>
<th>Attack</th>
<th>$L_\infty$ $\epsilon = 0.3$</th>
<th>$L_\infty$ $\epsilon = 0.5$</th>
<th>$L_2$ $\epsilon = 2.5$</th>
<th>$L_2$ $\epsilon = 5.0$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$L_\infty$ $\epsilon = 0.3$</td>
<td>0.99959</td>
<td>0.99966</td>
<td>0.99927</td>
<td>0.99925</td>
</tr>
<tr>
<td>$L_\infty$ $\epsilon = 0.5$</td>
<td>0.99436</td>
<td>0.9983</td>
<td>0.99339</td>
<td>0.99767</td>
</tr>
<tr>
<td>$L_2$ $\epsilon = 2.5$</td>
<td>0.99974</td>
<td>0.99969</td>
<td>0.99962</td>
<td>0.99944</td>
</tr>
<tr>
<td>$L_2$ $\epsilon = 5.0$</td>
<td>0.96421</td>
<td>0.98816</td>
<td>0.97747</td>
<td>0.99577</td>
</tr>
</tbody>
</table>

Table 10: AUC scores of the first MNIST base detector under fixed start and multiple random restarts attacks. These two tests use the same attack configuration: the $L_\infty$ $\epsilon = 0.5$ trained base detector is attacked using $L_\infty$ $\epsilon = 0.5$ constrained PGD attack of steps 200 and step size 0.01, and the $L_2$ $\epsilon = 5.0$ trained base detector is attacked using $L_2$ $\epsilon = 5.0$ constrained PGD attack of steps 200 and step size 0.1.

<table>
<thead>
<tr>
<th>Attack</th>
<th>MNIST $k = 0$ base detector</th>
<th>$L_\infty$ $\epsilon = 0.5$ trained</th>
<th>$L_2$ $\epsilon = 5.0$ trained</th>
</tr>
</thead>
<tbody>
<tr>
<td>fixed start</td>
<td>0.99830</td>
<td>0.99578</td>
<td></td>
</tr>
<tr>
<td>50 random restarts</td>
<td>0.99776</td>
<td>0.99501</td>
<td></td>
</tr>
</tbody>
</table>

Table 11: AUC scores of all $L_\infty$ $\epsilon = 0.3$ trained base detectors. Tested with $L_\infty$ $\epsilon = 0.3$ constrained PGD attacks of steps 200 and step size 0.01.

<table>
<thead>
<tr>
<th>Base detector</th>
<th>$k = 0$</th>
<th>$k = 1$</th>
<th>$k = 2$</th>
<th>$k = 3$</th>
<th>$k = 4$</th>
<th>$k = 5$</th>
<th>$k = 6$</th>
<th>$k = 7$</th>
<th>$k = 8$</th>
<th>$k = 9$</th>
</tr>
</thead>
<tbody>
<tr>
<td>AUC</td>
<td>0.99959</td>
<td>0.99971</td>
<td>0.99876</td>
<td>0.99861</td>
<td>0.99859</td>
<td>0.99795</td>
<td>0.99863</td>
<td>0.99687</td>
<td>0.99418</td>
<td></td>
</tr>
</tbody>
</table>

Table 12: AUC scores of all $L_\infty$ $\epsilon = 0.5$ trained base detectors. Tested with $L_\infty$ $\epsilon = 0.5$ constrained PGD attacks of steps 200 and step size 0.01.

<table>
<thead>
<tr>
<th>Base detector</th>
<th>$k = 0$</th>
<th>$k = 1$</th>
<th>$k = 2$</th>
<th>$k = 3$</th>
<th>$k = 4$</th>
<th>$k = 5$</th>
<th>$k = 6$</th>
<th>$k = 7$</th>
<th>$k = 8$</th>
<th>$k = 9$</th>
</tr>
</thead>
<tbody>
<tr>
<td>AUC</td>
<td>0.99830</td>
<td>0.99869</td>
<td>0.99327</td>
<td>0.99355</td>
<td>0.99314</td>
<td>0.99228</td>
<td>0.99424</td>
<td>0.99439</td>
<td>0.97875</td>
<td>0.9769</td>
</tr>
</tbody>
</table>
Figure 7: Performance of generative detection (a) and generative classification (b) on MNIST dataset under attacks with different loss functions. Please refer to MadryLab (b) for the implementations of cross-entropy loss and CW loss based attacks.

Figure 8: Distributions of class 1’s logit outputs of natural samples from class 1 and perturbed samples from the first row of Figure 3 (MNIST dataset).
C.2 MORE CIFAR10 RESULTS

C.2.1 MORE ROBUSTNESS TEST RESULTS

Table 13: AUC scores of the first ($k = 0$) CIFAR10 base detector under fixed start and multiple random restarts attacks. The $L_\infty \epsilon = 2.0$ base detector is attacked using PGD attack of steps 10 and step size 0.5, and the $L_\infty \epsilon = 8.0$ base detector is attacked using PGD attack of steps 40 and step size 0.5.

<table>
<thead>
<tr>
<th>Attack</th>
<th>CIFAR10 $k = 0$ base detector</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$L_\infty \epsilon = 2.0$ trained</td>
</tr>
<tr>
<td>fixed start</td>
<td>0.9866</td>
</tr>
<tr>
<td>10 random starts</td>
<td>0.9866</td>
</tr>
</tbody>
</table>

Table 14: AUC scores of the first ($k = 0$) $L_\infty \epsilon = 8$ trained CIFAR10 base detector under cross-norm and cross-perturbation attack.

<table>
<thead>
<tr>
<th>Attack</th>
<th>AUC</th>
</tr>
</thead>
<tbody>
<tr>
<td>$L_2 \epsilon = 80$, steps 20, step size 10</td>
<td>0.9814</td>
</tr>
<tr>
<td>$L_\infty \epsilon = 2$, steps 10, step size 0.5</td>
<td>0.9841</td>
</tr>
</tbody>
</table>

Table 15: AUC scores of the first two CIFAR10 $L_2 \epsilon = 80$ trained base detectors under different strengths of $L_2$ based PGD attacks. These two models are trained with $L_2$ based PGD attack of steps 20 and step size 10.

<table>
<thead>
<tr>
<th>$L_2$ attack steps, step size</th>
<th>$L_2 \epsilon = 80$ models</th>
</tr>
</thead>
<tbody>
<tr>
<td>k = 0</td>
<td>0.9839</td>
</tr>
<tr>
<td>k = 1</td>
<td>0.9924</td>
</tr>
<tr>
<td></td>
<td>0.9837</td>
</tr>
<tr>
<td></td>
<td>0.9922</td>
</tr>
</tbody>
</table>

Table 16: AUC scores of $L_\infty \epsilon = 2.0$ trained base detectors under $L_\infty \epsilon = 2.0$ constrained PGD attack of steps 10 and step size 0.5.

<table>
<thead>
<tr>
<th>Base detector</th>
<th>k = 0</th>
<th>k = 1</th>
<th>k = 2</th>
<th>k = 3</th>
<th>k = 4</th>
<th>k = 5</th>
<th>k = 6</th>
<th>k = 7</th>
<th>k = 8</th>
<th>k = 9</th>
</tr>
</thead>
<tbody>
<tr>
<td>AUC</td>
<td>0.9866</td>
<td>0.9926</td>
<td>0.9721</td>
<td>0.9501</td>
<td>0.9773</td>
<td>0.9636</td>
<td>0.9859</td>
<td>0.9908</td>
<td>0.9930</td>
<td>0.9916</td>
</tr>
</tbody>
</table>

Table 17: AUC scores of $L_\infty \epsilon = 8.0$ trained base detectors under $L_\infty \epsilon = 8.0$ constrained PGD attack of steps 40 and step size 0.5.

<table>
<thead>
<tr>
<th>Base detector</th>
<th>k = 0</th>
<th>k = 1</th>
<th>k = 2</th>
<th>k = 3</th>
<th>k = 4</th>
<th>k = 5</th>
<th>k = 6</th>
<th>k = 7</th>
<th>k = 8</th>
<th>k = 9</th>
</tr>
</thead>
<tbody>
<tr>
<td>AUC</td>
<td>0.9234</td>
<td>0.9553</td>
<td>0.8393</td>
<td>0.7893</td>
<td>0.8494</td>
<td>0.8557</td>
<td>0.9071</td>
<td>0.9276</td>
<td>0.9548</td>
<td>0.9370</td>
</tr>
</tbody>
</table>
Figure 9: Performance of generative detection (a) and generative classification (b) on CIFAR10 dataset under attacks with different loss functions. Cross-entropy and CW loss is only able to out-performs loss 9 when detection threshold is low (over 0.9 TPR). Please refer to MadryLab (a) for the implementations of cross-entropy loss and CW loss based attacks.

Figure 10: Distributions of class 1’s logit outputs of natural samples of class 1 and generated samples from the first row of Figure 6 (CIFAR10 dataset).
C.2.2 TRAINING STEP SIZE AND ROBUSTNESS

We found training with adversarial examples optimized with a sufficiently small step size to be essential for detection robustness. In Table 18 we tested two $L_\infty \epsilon = 2.0$ base detectors respectively trained with 0.5 and 1.0 step size. The step size 1.0 model is not robust when tested with a much smaller step size. We observe that when training the step size 1.0 model, training set adv AUC reached 1.0 in less than one hundred iterations, but test set natural AUC plummeted to around 0.95 and couldn’t recover thereafter. (Please refer to Figure 11 for the definitions of adv AUC and nat AUC.) This suggests that naturally occurring data, and perturbed data produced using a large step size, which are essentially noise images, live in two easily separable spaces, and training on such data is not beneficial for model performance on natural data or adversarial examples.

Table 18: AUC scores of two $L_\infty \epsilon = 2.0$ base detectors trained with different steps and step sizes.

<table>
<thead>
<tr>
<th>Attack steps, stepsize</th>
<th>Training steps, stepsize</th>
<th>10.0</th>
<th>10.1.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>10, 0.5</td>
<td>0.9866</td>
<td>0.9965</td>
<td></td>
</tr>
<tr>
<td>40, 0.1</td>
<td>0.9892</td>
<td>0.8848</td>
<td></td>
</tr>
</tbody>
</table>

C.2.3 EFFECTS OF PERTURBATION LIMIT

To study the effects of perturbation limit on generative adversarial training, we analyse the training dynamics of one $L_\infty \epsilon = 2.0$ constrained and one $L_\infty \epsilon = 8.0$ constrained base detector. In Figure 11 we show the training and testing history of these two models. The $\epsilon = 2.0$ model history shows that by adversarial finetuning the model reaches robustness in just a few thousands of iterations, and the performance on natural samples is preserved (test natural AUC begins at 0.9971, and ends at 0.9981). Adversarial finetuning on the $\epsilon = 8.0$ model didn’t converge after an extended 20K iterations of training. The gap between train adv AUC and test adv AUC of the $\epsilon = 8.0$ model is more pronounced, and we observed a decrease of test natural AUC from 0.9971 to 0.9909.

These results suggest that training with larger perturbation limit is more time and resource consuming, and could lead to performance decrease on natural samples. The benefit is that the detector is pushed to a better approximation of the target data distribution. As an illustration, in Figure 12, perturbations generated by attacking the naturally trained classifier (corresponds to 0 perturbation limit) don’t have clear semantics, while perturbed samples of the $L_\infty \epsilon = 8.0$ model are completely recognizable.

![Figure 11: Training and testing AUC histories of two base detectors.](image)

Figure 11: Training and testing AUC histories of two base detectors. Adv AUC is the AUC score computed on $\{(x, 0) : x \in D^f_k\} \cup \{(x, 1) : x \in D^f_k\}$, and nat AUC is the score computed on $\{(x, 0) : x \in D^f_k\} \cup \{(x, 1) : x \in D^f_k\}$.
Figure 12: Perturbed samples produced by attacking the $k = 0$ (airplane) detectors and the natural trained classifier’s 1st logit output. All samples reached the same $L_2$ perturbation of 1200 (produced using PGD attacks of step size 10.0).

Figure 13: Images generated from class conditional Gaussian noise by attacking $L_\infty \epsilon = 16$ constrained CIFAR10 detectors. We use $L_2 \epsilon = 100 \times 255$ constrained PGD attack of steps 200 and step size $0.5 \times 255$. Samples not selected.
C.3 More ImageNet results

(a) $L_2 \epsilon = 3.5$ trained robust classifier with $L_2 \epsilon = 40$ constrained PGD attack of steps 60 and step size 1.0 (Santurkar et al., 2019).

(b) $L_\infty \epsilon = 0.02$ trained detector with $L_2 \epsilon = 40$ constrained PGD attack of steps 60 and step size 1.0.

(c) $L_\infty \epsilon = 0.05$ trained detector with $L_2 \epsilon = 40$ constrained PGD attack of steps 60 and step size 1.0.

(d) $L_\infty \epsilon = 0.1$ trained detector with $L_2 \epsilon = 40$ constrained PGD attack of steps 60 and step size 1.0.

(e) $L_\infty \epsilon = 0.1$ trained detector with $L_2 \epsilon = 100$ constrained PGD attack of steps 10 and step size 10.0.

(f) $L_\infty \epsilon = 0.3$ trained detector with $L_2 \epsilon = 100$ constrained PGD attack of steps 100 and step size 10.0.

Figure 14: ImageNet $224 \times 224 \times 3$ random samples generated from class conditional Gaussian noise by attacking robust classifier and detector models trained with different constrains. Note than large perturbation models didn’t reach robustness. Please refer to Santurkar et al. (2019) for the detail about how the class conditional Gaussian is estimated.
Figure 15: Perturbed samples produced by attacking the $L_\infty \epsilon = 0.3$ trained dog detector using $L_2 \epsilon = 30$ constrained PGD attack of steps 100 and step size 5. Top rows are original images, and second rows are attacked images.

Figure 16: Dog face retouching by attacking the $L_\infty \epsilon = 0.3$ trained dog detector using $L_2 \epsilon = 30$ constrained PGD attack of steps 100 and step size 5. Top rows are original images, and second rows are attacked images.
Figure 17

Figure 18: More $224 \times 224 \times 3$ random samples generated by attacking the $L_\infty \epsilon = 0.3$ trained detector with $L_2 \epsilon = 100$ constrained PGD attack of steps 100 and step size 10.0.
In this section we use Gaussian noise attack experiment to motivate a comparative analysis of the interpretabilities of our generative classification approach and discriminative robust classification approach (Madry et al., 2017).

We first discuss how these two approaches determine the posterior class probabilities. For the discriminative classifier, the posterior probabilities are computed from the logit outputs of the classifier using the softmax function

\[
p(k|x) = \frac{\exp(z(f(x)_k))}{\sum_{j=1}^{K} \exp(z(f(x)_j))}.
\]

For the generative classifier, the posterior probabilities are computed in two steps: in the first, we train the base detectors, which is the process of solving the inference problem of determining the joint probability \(p(x, k)\), and in the second, we use Bayes rule to compute the posterior probability

\[
p(k|x) = \frac{p(x,k)}{p(x)} = \frac{\exp(z(h_k(x)))}{\sum_{j=1}^{K} \exp(z(h_j(x)))}.
\]

Coincidentally, the formulas for computing the posterior probabilities take the same form. But in our approach, the exponential of the logit output of a detector (i.e., \(\exp(z(h_k(x)))\)) has a clear probabilistic interpretation: it’s the unnormalized joint probability of the input and the corresponding class category. We use Gaussian noise attack to demonstrate that this probabilistic interpretation is consistent with visual perception.

We start from a Gaussian noise image, and gradually perturb it to cause higher and higher logit outputs. This is implemented by targeted PGD attack against logit outputs of these two classification models. The resulting images in Figure 19 show that, in our model, the logit output increase direction, i.e. the join probability increase direction, indicates the class semantic changing direction; while for the discriminative robust model, the perturbed image computed by increasing logit outputs are not as clearly interpretable. In particular, the perturbed images that cause high logit outputs of the robust classifiers are not recognizable.

In summary, as a generative classification approach that explicitly models class conditional distributions, our system offers a probabilistic view of the decision making process of the classification problem; adversarial attacks that rely on imperceptible or uninterpretable noises are not effective against such a system.

### E COMPUTATIONAL COST ISSUE

In this section we provide an analysis of the computational cost of our generative classification approach. In terms of memory requirements, if we assume the softmax classifier (i.e., the discriminative robust classifier) and the detectors use the same architecture (i.e., only defer in the final layer) then the detector based generative classifier is approximately \(K\) times more expensive than the \(K\)-class softmax classifier. This also means that the computational graph of the generative classifier is \(K\) times larger than the softmax classifier. Indeed, in the CIFAR10 task, on our Quadro M6000 24GB GPU (TensorFlow 1.13.1), the inference speed of the generative classifier is roughly ten times slower than the softmax classifier.

We next benchmark the training speed of these two types of classifiers.

The generative classifier has \(K\) logit outputs, with each one defined by the logit output of a detector. Same with the softmax classifier, except that the \(K\) outputs share the parameters in the convolutional base. Now consider ordinary adversarial training on the softmax classifier and generative adversarial training on the generative classifier. To train the softmax classifier, we use batches of \(N\) samples. For the generative classifier, we train each detector with batches of \(2 \times M\) samples \((M\) positive samples and \(M\) negative samples\). At each iteration, we need to respectively compute \(N\) and \(M \times K\) adversarial examples for these two classifiers. Now we test the speed of the following two scenarios: 1) compute the gradient w.r.t. to \(N\) samples on a single computational graph, and 2) compute the gradient w.r.t. to \(M \times K\) samples on \(K\) computational graphs, with each graph working on \(M\) samples. We assume that in scenario 2 all the computational graphs are loaded to GPUs, and thus their computations are in parallel.

In our CIFAR10 experiment, we used batches consisting of 30 positive samples and 30 negative samples to train each ResNet50 based detectors. In Madry et al. (2017), the softmax classifier was trained with batches of 128 samples. In this case, \(K = 10\), \(M = 30\), and \(N = 128\). On our GPU, scenario 1 took 683 ms \(\pm 6.76\) ms per loop, while scenario 2 took 1.85 s \(\pm 42.7\) ms per loop. In
Figure 19: Image generated by attacking the generative classifier (based on $L_\infty$ $\epsilon = 16$ trained detectors) and discriminative robust classifier (Madry et al., 2017) using (the same) Gaussian noise image. We used unconstrained $L_2$ PGD attack of step size $0.5*255$. The five columns corresponding to the perturbed images at step 0, 50, 100, 150, and 200.

In this case, we could expect generative adversarial training to be about 2.7 times slower than ordinary adversarial training, if not considering parameter gradient computation.

In practice, large batch size is almost always preferred. And our method won’t compare as favorably if we choose to use one.

F Density estimation on synthetic datasets

While ordinary discriminative training only learns a good decision boundary, GAT is able to learn the underlying density functions that generate the training data. Results on 1D (Figure 20) and 2D benchmark datasets (Figure 21) show that through properly configured generative adversarial training, detectors’ output recover target density functions.
Figure 20: Ordinary discriminative training and generative adversarial training on real 1D data. The positive class data (blue points) are sampled from a mixture of Gaussians (mean 0.4 with std 0.01, and mean 0.6 with std 0.005, each with 250 samples). Both the blue and red data has 500 samples. The estimated density function is computed using Gibbs distribution and network logit outputs. PGD attack steps 20, step size 0.05, and perturbation limit $\epsilon = 0.3$.

Figure 21: 2D datasets (top row, blue points are class 1 data, and red points are class 0 data, both have 1000 data points) and sigmoid outputs of GAT trained models (bottom row). The architecture of the MLP model for solving these tasks is 2-500-500-500-500-500-1. PGD attack steps 10, step size 0.05, and perturbation limit $L_{\infty} \epsilon = 0.5$. 