ADVERSARIAL EXAMPLE DETECTION AND CLASSIFICATION WITH ASYMMETRICAL ADVERSARIAL TRAINING

Anonymous authors
Paper under double-blind review

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 consider the adversarial detection problem under the robust optimization framework. We partition the input space into subspaces and train adversarial robust subspace detectors using asymmetrical adversarial training (AAT). The integration of the classifier and detectors presents a detection mechanism that provides a performance guarantee to the adversary it considered. We demonstrate that AAT promotes the learning of class-conditional distributions, which further gives rise to generative detection/classification approaches 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.

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;a; 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 these 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 claimed performances under *adaptive* attacks. The current solutions are mostly heuristic approaches that cannot provide performance guarantees to the adversary they considered.

In this paper, we are interested in detection mechanisms for adversarial examples that can withstand adaptive attacks. Unlike previous approaches that assume adversarial and natural samples coming from different distributions, thus rely on using a single classifier to distinguish between them, we instead partition the input space into subspaces based on the classification system's output and perform adversarial/natural sample classification in these subspaces. Importantly, the mentioned partitions allow us to drop the adversarial constrain and employ a novel *asymmetrical adversarial training* (AAT) objective to train robust binary classifiers in the subspaces. Figure 1 demonstrates our idea of space partitioning and robust detector training. Our qualitative results show that AAT supports detectors to learn class-conditional distributions, which further motivates generative detection/classification solutions that are both robust and interpretable.

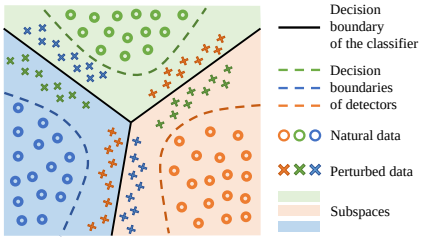


Figure 1: A conceptual visualization of our integrated adversarial detection mechanism.

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 4.47 on the MNIST dataset, and from 1.1 to 1.5 on the CIFAR10 dataset.
- We study powerful and versatile generative classification models derived from our detection framework and demonstrate their competitive performances over discriminatively robust classifiers. While defense mechanisms based on ordinary adversarial training are vulnerable to unrecognizable inputs (e.g., rubbish examples), inputs that cause confident predictions of our models have human-understandable semantic meanings.
- We demonstrate that asymmetrical adversarial training (ATT) not only induces model robustness as ordinary adversarial training methods do but also promotes the learning of class-conditional distributions. The proposed objective does not require auxiliary network and is stable to train. On CIFAR10 and ImageNet, our models can generate realistic images with qualities comparable to state-of-the-art generative models.

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), initially proposed by Kurakin et al. (2016b), has been empirically identified as the most effective approach for performing norm ball 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, $f : \mathbb{R}^d \to \{1...K\}$, and its corresponding natural training samples, $\mathcal{D} = \{x_i \in \mathbb{R}^d\}_{i=1}^N$, generate a set of adversarially attacked samples $\mathcal{D}' = \{x'_j \in \mathbb{R}^d\}_{j=1}^M$, and devise a mechanism to discriminate $\mathcal{D}$ from $\mathcal{D}'$. 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 $f : \mathbb{R}^d \to \{1...K\}$ be the 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 $\mathcal{D}^f = \bigcup \mathcal{D}_k^f$, where $\mathcal{D}_k = \{x : y = k, x \in \mathcal{D}\}$, and $\mathcal{D}_k^f = \{x : f(x) = k, x \in \mathcal{D}\}$. Let $\mathcal{H} = \{h_k\}_{k=1}^K, h_k : \mathbb{R}^d \to \{0, 1\}$ 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* the network, $f(\cdot)$, to be classified as k. Also, let $\mathcal{D}'$ be a set of ℓ_p norm bounded adversarial examples crafted from $\mathcal{D}$: $\mathcal{D}' = \{x + \delta : f(x + \delta) \neq y, f(x) = y, x \in \mathcal{D}, \delta \in \mathcal{S}\}, \mathcal{S} = \{\delta \in \mathbb{R}^d \,|\, \|\delta\|_p \leq \epsilon\}$. 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 x a natural sample, otherwise it's an adversarial sample.*

The detection accuracy of the algorithm is given by

$$\frac{\sum_{k=1}^K |\{x : h_k(x) = 1, x \in \mathcal{D}_k^f\}| + |\{x : h_k(x) = 0, x \in \mathcal{D}'^f_k\}|}{|\mathcal{D}| + |\mathcal{D}'|}, \tag{1}$$

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

$$\theta_k^* = \arg\min_{\theta_k} \mathbb{E}_{x \sim \mathcal{D}'^f_k} \Big[L(h_k(x; \theta_k), 0) \Big] + \mathbb{E}_{x \sim \mathcal{D}_k^f} \Big[L(h_k(x; \theta_k), 1) \Big], \tag{2}$$

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, inspired by the idea of robust optimization (Madry et al., 2017), we incorporate the attack into the training objective:

$$\min_{\theta_k} \rho(\theta_k), \quad \text{where} \quad \rho(\theta_k) = \mathbb{E}_{x \sim \mathcal{D}^f_{\setminus k}} \Big[\max_{\delta \in \mathcal{S}, f(x+\delta)=k} L(h_k(x+\delta; \theta_k), 0) \Big] + \mathbb{E}_{x \sim \mathcal{D}_k^f} \Big[L(h_k(x; \theta_k), 1) \Big], \tag{3}$$

where $\mathcal{D}^f_{\setminus k} = \{x : f(x) \neq k, y \neq k, x \in \mathcal{D}\}$, and we assume that perturbation budget is large enough such that $\forall x \in \mathcal{D}^f_{\setminus k}$, $\exists \delta \in \mathcal{S}$, $s.t.\ f(x + \delta) = k$. 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,

$$\rho(\theta_k) = \mathbb{E}_{x \sim \mathcal{D}^f_{\setminus k}} \Big[\max_{\delta \in \mathcal{S}} L(h_k(x + \delta; \theta_k), 0) \Big] + \mathbb{E}_{x \sim \mathcal{D}_k^f} \Big[L(h_k(x; \theta_k), 1) \Big]. \tag{4}$$

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 $\mathcal{D}^f_{\setminus k}$ and $\mathcal{D}^f_k$. This leads to our proposed *asymmetrical adversarial training (AAT)* objective:

$$\min_{\theta_k} \rho(\theta_k), \quad \text{where} \quad \rho(\theta_k) = \mathbb{E}_{x\sim\mathcal{D}_{\setminus k}}\Big[\max_{\delta\in\mathcal{S}} L(h_k(x+\delta;\theta_k),0)\Big] + \mathbb{E}_{x\sim\mathcal{D}_k}\Big[L(h_k(x;\theta_k),1)\Big]. \tag{5}$$

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.

3.2 Generative adversarial detection/classification

Because of the integrated adversary, objective 5 is no longer a straightforward discriminative objective. Our investigations showed that AAT promotes detectors to learn conditional data distributions. Similar to GANs' objective (Goodfellow et al., 2014a), the AAT objective presents a min-max problem, where the adversary tries to generate perturbed samples that look like the target class data, and the detector is trained by discriminating between target class data and perturbed data.

We now assume that through properly configured adversary and training procedure, the k'th detector could be trained to model the class-conditional distribution $P(X|\mathcal{C}_k)$. An application of the Bayes classification rule leads to the *generative classifier*:

$$H(x) = \arg\max_k p(\mathcal{C}_k)p(x|\mathcal{C}_k), \tag{6}$$

where conditional density $p(x|\mathcal{C}_k)$ is given by the sigmoid output of the k-th detector, and prior $p(\mathcal{C}_k)$ could be simply estimated from the fraction of training data of class k. Because we explicitly model class-conditional distributions, we could use the model to detect and reject low probability inputs. We provide the reject option by thresholding the predicted class $\hat{k}$'s likelihood $p(x|\mathcal{C}_{\hat{k}})$ (or equivalently, by thresholding the logit output of the $\hat{k}$-th detector). 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 we train a detector with an adequately configured PGD attack, its performance cannot be significantly reduced by an adversary with much stronger configurations (stronger in terms of steps and step-size). Although the PGD attack (Madry et al., 2017) can reasonably solve the inner maximization in objective 5, it is not clear whether the optimization landscape of the asymmetrical objective is the same as its symmetrical counterparts. 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). We also face a unique challenge when training with objective 5: the number of positive and negative samples are highly imbalanced. Our solution is to use re-sampling to balance positive and negative classes. Furthermore, we use adversarial finetuning on CIFAR10 and ImageNet to speed up the training of our detectors. With the robustness test, we show that robust optimization 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 the true-positive and the false-positive rates are the commonly used metrics for measuring the detection performance, 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 $\{(x,0) : x \in \mathcal{D}'^f_{\setminus k}\} \cup \{(x,1) : x \in \mathcal{D}^f_k\}$, where $\mathcal{D}'^f_{\setminus k} = \{\arg\max_{x+\delta} L(h_k(x+\delta;\theta_k),0) : x \in \mathcal{D}^f_{\setminus k})\}$ (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 $k := f(x)$, then use the k-th detector's logit output $z(h_k(x))$ to predict: if $z(h_k(x)) \geq T_k$, then x is a natural sample, otherwise it is an adversarially perturbed sample. For the sake of this evaluation, we use a universal threshold for all the detectors: $\forall k \in \{1...K\}\, T_k = T$, and report detection performance at a range of universal thresholds. In practice, however, the optimal value of each detector's detection threshold T_k should be determined by optimizing a utility function.

We use $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$ to denote the test set that contains natural samples, and $\mathcal{D}' = \{(x_i + \delta_i, y_i)\}_{i=1}^N$ to denote the corresponding perturbed test set. For a given threshold T, we compute the true positive rate (TPR) on $\mathcal{D}$ and false positive rate (FPR) on $\mathcal{D}'$. These two metrics are respectively defined as

$$\text{TPR} = \frac{1}{N}|\{x : z(h_k(x)) \geq T, k := f(x), x \in \mathcal{D}\}|, \tag{7}$$

and

$$\text{FPR} = \frac{1}{N}|\{x : z(h_k(x)) \geq T, k := f(x), f(x) \neq y, (x, y) \in \mathcal{D}'\}|. \tag{8}$$

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 $\mathcal{D}'$, 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,

$$L(x') = z(f(x'))_y - \max_{i \neq y} z(f(x'))_i, \tag{9}$$

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) and MadryLab (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 treated as a single network, and the perturbed sample x' for a given input (x, y) is computed by minimizing the loss

$$L(x') = -\max_{i \neq y} z(H(x'))_i. \tag{10}$$

Note that, 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 9 or to cross-entropy loss, but we found the attack based on the loss in 10 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 an array of 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 $\max_{k \neq y} z(H(x))_k$ (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

$$z(g(x))_i = \begin{cases} z(f(x))_i & \text{if } i \leq K, \\ (-\max_{j \neq y} z(H(x))_j + 1) \cdot \max_j z(f(x))_j & \text{if } i = K + 1. \end{cases} \tag{11}$$

A perturbed example x' is then computed by minimizing the loss function

$$L(x') = \max_i z(g(x'))_i - \max_{i \neq y} z(f(x'))_i. \tag{12}$$

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 9 and loss 10:

$$L(x') = \begin{cases} z(f(x'))_y - \max_{i\neq y} z(f(x'))_i & \text{if } z(f(x'))_y \geq \max_{i\neq y} z(f(x'))_i, \\ -\max_{i\neq y} z(H(x'))_i & \text{else.} \end{cases} \tag{13}$$

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 our generative detection method (as detailed in Section 3.2) by computing TPR on $\mathcal{D}$ and FPR on $\mathcal{D}'$. We use the loss 10 to perform attacks against the generative detection method, but also provide results of attacks based on cross-entropy loss and CW loss 9.

4.3 Robust Classification Performance

Integrated classification. In addition to the generative classifier proposed in Section 3.2, we introduce another classification scheme that provides a reject option. The scheme is based an integration of the naive classifier f and the detectors: for a given input x and its prediction label $k := f(x)$, if $z(h_k(x)) < T$, x is rejected, otherwise it's classified as k. We respectively use loss 13 and 10 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. We provide a similar performance analysis of the above classification models. On the natural dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$, we compute the accuracy as the fraction of samples that are correctly classified ($f(x) = y$) and at the same time not rejected ($z(h_k(x)) \geq T$):

$$\text{Accuracy} = \frac{1}{N}|\{x : z(h_k(x)) \geq T, k := f(x), f(x) = y, (x, y) \in \mathcal{D}\}|. \tag{14}$$

On the perturbed dataset $\mathcal{D}' = \{(x_i + \delta_i, y_i)\}_{i=1}^N$ we compute the *error* as the fraction of samples that are misclassified ($f(x) \neq y$) and at the same time not rejected:

$$\text{Error} = \frac{1}{N}|\{x : z(h_k(x)) \geq T, k := f(x), f(x) \neq y, (x, y) \in \mathcal{D}'\}|. \tag{15}$$

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 robust classifier, 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 ϵ 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-size (see Table 1). At each step of PGD attack we use the Adam optimizer to perform gradient descent, both for L_2 and L_∞ constrained scenarios. Appendix A.1 provides more training details.

Table 1: PGD attack steps and step-sizes for base detector training and validation.

	L_2 models		L_∞ models	
	$\epsilon = 2.5$	$\epsilon = 5.0$	$\epsilon = 0.3$	$\epsilon = 0.5$
Train	100, 0.1	200, 0.1	100, 0.01	100, 0.01
Validation	200, 0.1	200, 0.1	200, 0.01	200, 0.01

Robustness results. The robustness test results in Table 2 confirm that the base detectors trained with objective 5 are able to withstand much stronger PGD attacks, for both L_2 and L_∞ scenarios.

Normalized steepest descent is another popular choice for performing PGD attack (Madry et al., 2017; MadryLab, b;a), for which we got similar robustness results (Table 8). Further results on the complete list of performances of the L_∞ $\epsilon = 0.3$ and L_∞ $\epsilon = 0.5$ trained detectors, cross-norm and cross-perturbation test results, and random restart test results are included in Appendix C.1.

Table 2: AUC scores of the first two detectors ($k = 0, 1$) tested with different strengths of PGD attacks using Adam optimizer.

PGD attack steps, step-size	L_∞ $\epsilon = 0.3$ detectors $k=0$	$k=1$	L_∞ $\epsilon = 0.5$ detectors $k=0$	$k=1$
200, 0.01	0.99959	0.99971	0.99830	0.99869
2000, 0.005	0.99958	0.99971	0.99796	0.99861

PGD attack steps, step-size	L_2 $\epsilon = 2.5$ detectors $k=0$	$k=1$	L_2 $\epsilon = 5.0$ detectors $k=0$	$k=1$
200, 0.1	0.99962	0.99968	0.99578	0.99987
2000, 0.05	0.99927	0.99900	0.99529	0.99918

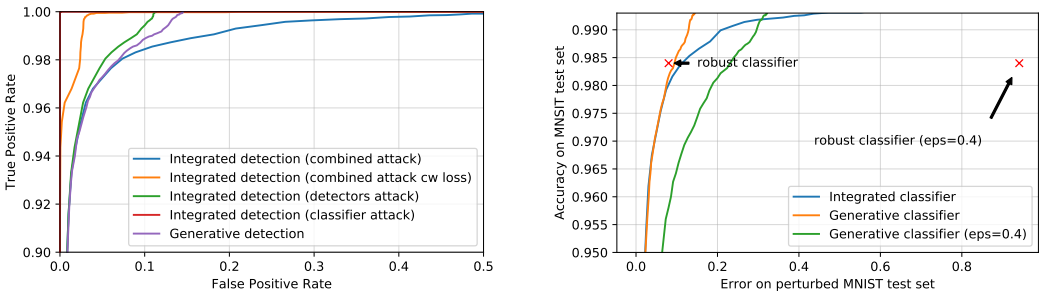
Table 3: MNIST mean L_2 distortion (higher is better) of perturbed samples when the detection method has 1.0 FPR on perturbed set and 0.95 TPR on natural set.

Detection method	Mean L_2 distortion
State-of-the-art (Carlini & Wagner, 2017a)	3.68
Ours (use L_∞ $\epsilon = 0.3$ trained base detectors)	3.80
Ours (use L_∞ $\epsilon = 0.5$ trained base detectors)	4.47

Detection results. Figure 2a shows that the combined attack is the most effective attack against integrated detection. Generative detection (attacked using loss 10) outperforms integrated detection, especially when the detection threshold is low (the region where TPR is high). In Figure 7 we confirm that loss 10 is more effective than CW loss and cross-entropy loss for attacking generative detection. Notably, the red curve that overlaps the y-axis shows that integrated detection can perfectly detect adversarial examples crafted by attacking only the classifier (using objective 9).

In Table 3 we compare the performances of our generative detection method with the state-of-the-art detection method as identified by Carlini & Wagner (2017a). Our method using L_∞ $\epsilon = 0.5$ trained base detectors is able to outperform the state-of-the-art method by a large margin. Appendix B describes the procedure we used to compute the mean L_2 distortion of our method.

Classification results. In Figure 2b, we compare the robust classification performance of our methods and a state-of-the-art robust classifier. While the performance of the robust classifier is fixed, by using different rejection thresholds, our classification methods provide the option to balance standard accuracy and robust error. The generative classifier outperforms the integrated classifier when the rejection threshold is low (i.e., when the perturbed set error is high). We observe that a stronger attack ($\epsilon = 0.4$) breaks the robust classifier, while the generative classifier still exhibits robustness, even though both systems are trained with the same L_∞ $\epsilon = 0.3$ constrain.



(a) (b)

Figure 2: (a) Performances of integrated detection and generative detection under L_∞ $\epsilon = 0.3$ constrained attack. (b) Performances of the integrated classifier (discussed in Section 4.3) and generative classifier under L_∞ $\epsilon = 0.3$ constrained and L_∞ $\epsilon = 0.4$ 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 annotated. PGD attack steps 100, step-size 0.01.

Figure 3 shows perturbed samples by performing targeted attacks against the generative classifier and robust classifier. We observe that perturbed samples produced by attacking the generative classifier have distinguishably visible features of the target class, indicating that the base detectors, from which the generative classifier is built, have successfully learned the class-conditional distributions, and the perturbation has to change the semantics of the underlying sample 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 use a larger perturbation limit, it is straightforward to generate unrecognizable images that cause highly confident predictions of the robust classifier.

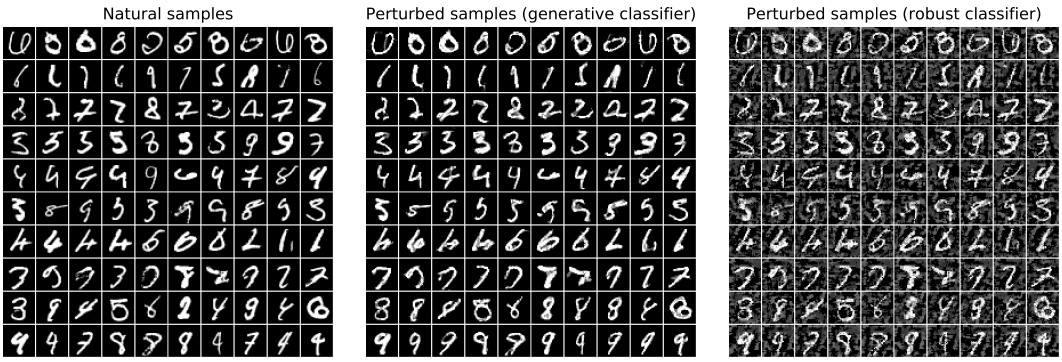


Figure 3: Natural samples and corresponding perturbed samples produced by performing a targeted attack against the generative classifier and robust classifier (Madry et al., 2017). Targets from top row to bottom row are digit class from 0 to 9. We perform the targeted attack by maximizing the logit output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

5.2 CIFAR10

On CIFAR10 we train the base detectors using L_∞ $\epsilon = 8$ constrain PGD attack of steps 40 and step size 0.5. Note that the scale of ϵ 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 will compare with is trained with the same L_∞ $\epsilon = 8$ constraint but with a different step-size (see Appendix C.2.2 for a discussion of step-sizes). Appendix A.2 provides the training details.

Robustness results. Table 4 shows that the base detector models can withstand attacks that are significantly stronger than the training attack. In Appendix C.2.1 we present random restart test results, cross-norm and cross-perturbation test results, and robustness test result for L_2 based models.

Table 4: AUC scores of the first two L_∞ $\epsilon = 8$ base detectors under different strengths of the L_∞ $\epsilon = 8$ constrained PGD attack.

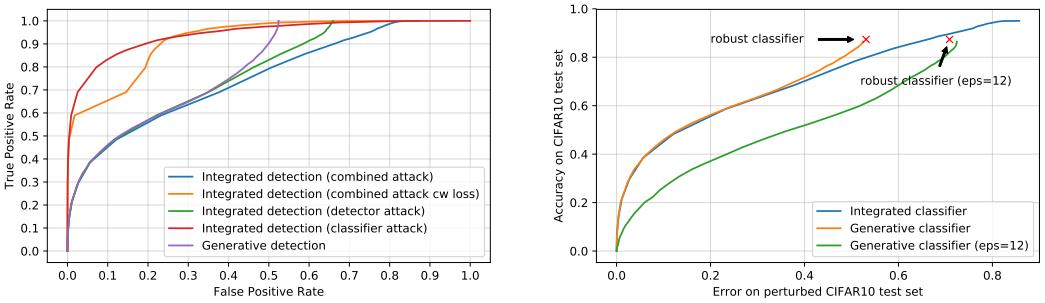
L_∞ $\epsilon = 8$ detectors	L_∞ attack steps and step-size				
	20, 2.0	40, 0.5	200, 0.1	200, 0.5	500, 0.5
Base detector $k = 0$	0.9224	0.9234	0.9231	0.9205	0.9203
Base detector $k = 1$	0.9533	0.9553	0.9550	0.9504	0.9500

Detection results. Consistent with the MNIST results, in Figure 4a combined attack is the most effective method against integrated detection. Similarly, the generative detection outperforms integrated detection when the detection threshold is low (i.e., where TPR is high). In this figure we use loss 10 to attack generative detection, and in Figure 9 we show that it's more effective than attack based on cross-entropy loss and CW loss. In Table 5 our method outperforms the state-of-the-art adversarial detection method.

Classification results. In Figure 4b, we did not observe a dramatic decrease in the robust classifier's performance when we increase the perturbation limit to $\epsilon = 12$. Integrated classification can reach the standard accuracy of a regular classifier, but at the cost of significantly increasing error

Table 5: 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.

Detection method	Mean L_2 distortion (0-1 scale)
State-of-the-art (Carlini & Wagner, 2017a)	1.1
Ours (use L_∞ $\epsilon = 8.0$ trained models)	1.5



(a) (b)

Figure 4: (a) Performances of generative detection and integrated detection under L_∞ $\epsilon = 8$ attack. (b) Performances of integrated classifier (discussed in Section 4.3) and generative classifier under L_∞ $\epsilon = 8$ constrained and L_∞ $\epsilon = 12$ 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 $\epsilon = 12$.

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 features of the targets, which indicates that the adversary has to change more semantic in order 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. Since both classifiers are trained with L_∞ $\epsilon = 8$ constrain, these results indicate that AAT significantly improves robust and interpretable feature learning. The visual similarity between generated samples in Figure 6 and real samples further suggests that detectors have successfully learned the conditional data distributions.

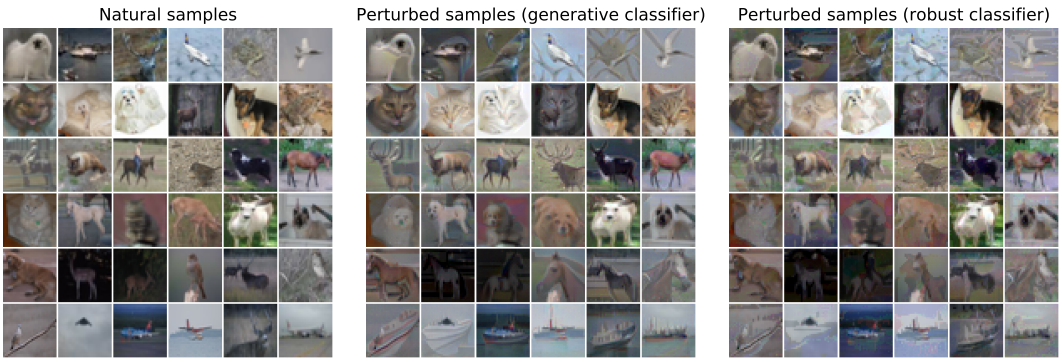


Figure 5: Natural samples and corresponding perturbed samples by performing targeted attack against the generative classifier and robust classifier (Madry et al., 2017). The targeted attack is performed by maximizing the logit output of the targeted class. We use L_∞ $\epsilon = 12$ constrained PGD attack 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 classier. we use PGD attack of steps 60 and step-size 0.5×255 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 IMAGENET

Similarly, on ImageNet, we show asymmetrical adversarial training induces detection robustness and supports the learning of class-conditional distributions. We use the Restricted ImageNet (Tsipras et al., 2018) in our experiment, which is a subset of ImageNet that has its samples reorganized into customized categories. The dog category contains images of different dog breeds collected from ImageNet class between 151 and 268. We trained a dog class detector by finetuning a pre-trained ResNet50 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 constraints, we only validated the robustness of a L_∞ $\epsilon = 0.02$ trained detector (trained with PGD attack of steps 40 and step-size 0.001), and we present the result in Table 6. (On Restricted ImageNet in the case of L_∞ 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, including perturbed samples and generated samples.

Table 6: AUC of the dog detector under different strengths of L_∞ $\epsilon = 0.02$ constrained PGD attacks

Attack steps, step-size	40, 0.001	100, 0.001	200, 0.001	40, 0.002	200, 0.002	200, 0.0005
AUC	0.9720	0.9698	0.9692	0.9703	0.9690	0.9698

6 CONCLUSION

In this paper, we studied the problem of adversarial detection under the robust optimization framework and proposed a novel adversarial detection scheme based on input space partitioning. Our formulation leads to a new generative modeling technique which we called asymmetrical adversarial training (AAT). AAT's capability to learn class-conditional distributions further gives rise to generative detection/classification methods that show competitive performance and improved interpretability. In particular, our generative classifier is more resistant to "rubbish examples", a significant threat to even the most successful defense mechanisms. High computational cost is a major drawback of our methods, and in the future, we will explore the idea of shared computation between detectors.

REFERENCES

Dario Amodei, Sundaram Ananthanarayanan, Rishita Anubhai, Jingliang Bai, Eric Battenberg, Carl Case, Jared Casper, Bryan Catanzaro, Qiang Cheng, Guoliang Chen, et al. Deep speech 2: End-to-end speech recognition in english and mandarin. In *International conference on machine learning*, pp. 173–182, 2016.

Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. *arXiv preprint arXiv:1802.00420*, 2018. URL https://arxiv.org/abs/1802.00420.

Yuval Bahat, Michal Irani, and Gregory Shakhnarovich. Natural and adversarial error detection using invariance to image transformations. *arXiv preprint arXiv:1902.00236*, 2019.

Osbert Bastani, Yani Ioannou, Leonidas Lampropoulos, Dimitrios Vytiniotis, Aditya Nori, and Antonio Criminisi. Measuring neural net robustness with constraints. In *Advances in neural information processing systems*, pp. 2613–2621, 2016.

Arjun Nitin Bhagoji, Daniel Cullina, and Prateek Mittal. Dimensionality reduction as a defense against evasion attacks on machine learning classifiers. *arXiv preprint arXiv:1704.02654*, 2017.

Nicholas Carlini and David Wagner. Adversarial examples are not easily detected: Bypassing ten detection methods. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pp. 3–14. ACM, 2017a. URL https://arxiv.org/abs/1705.07263.

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

Pin-Yu Chen, Yash Sharma, Huan Zhang, Jinfeng Yi, and Cho-Jui Hsieh. Ead: elastic-net attacks to deep neural networks via adversarial examples. In *Thirty-second AAAI conference on artificial intelligence*, 2018.

Reuben Feinman, Ryan R Curtin, Saurabh Shintre, and Andrew B Gardner. Detecting adversarial samples from artifacts. *arXiv preprint arXiv:1703.00410*, 2017. URL https://arxiv.org/abs/1703.00410.

Zhitao Gong, Wenlu Wang, and Wei-Shinn Ku. Adversarial and clean data are not twins. *arXiv preprint arXiv:1704.04960*, 2017. URL https://arxiv.org/abs/1704.04960.

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

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014b. URL https://arxiv.org/abs/1412.6572.

Kathrin Grosse, Praveen Manoharan, Nicolas Papernot, Michael Backes, and Patrick McDaniel. On the (statistical) detection of adversarial examples. *CoRR*, 2017.

Shixiang Gu and Luca Rigazio. Towards deep neural network architectures robust to adversarial examples. *arXiv preprint arXiv:1412.5068*, 2014.

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

Warren He, Bo Li, and Dawn Song. Decision boundary analysis of adversarial examples. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=BkpiPMbA-.

Dan Hendrycks and Kevin Gimpel. Early methods for detecting adversarial images. In *ICLR*, 2017. URL https://arxiv.org/abs/1608.00530.

Jonghoon Jin, Aysegul Dundar, and Eugenio Culurciello. Robust convolutional neural networks under adversarial noise. *arXiv preprint arXiv:1511.06306*, 2015.

Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial examples in the physical world. *arXiv preprint arXiv:1607.02533*, 2016a. URL https://arxiv.org/abs/1607.02533.

Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial machine learning at scale. *arXiv preprint arXiv:1611.01236*, 2016b.

S Levine, C Finn, T Darrell, and P Abbeel. End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research*, 17:1334–1373, 2016.

Xin Li and Fuxin Li. Adversarial examples detection in deep networks with convolutional filter statistics. *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 5775–5783, 2017. URL https://arxiv.org/abs/1612.07767.

Yuntao Liu, Yang Xie, and Ankur Srivastava. Neural trojans. In *2017 IEEE International Conference on Computer Design (ICCD)*, pp. 45–48. IEEE, 2017.

Xingjun Ma, Bo Li, Yisen Wang, Sarah M Erfani, Sudanthi Wijewickrema, Grant Schoenebeck, Dawn Song, Michael E Houle, and James Bailey. Characterizing adversarial subspaces using local intrinsic dimensionality. *arXiv preprint arXiv:1801.02613*, 2018. URL https://arxiv.org/abs/1801.02613.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017. URL https://arxiv.org/abs/1706.06083.

MadryLab. CIFAR10 Adversarial Examples Challenge. https://github.com/MadryLab/cifar10_challenge, a.

MadryLab. MNIST Adversarial Examples Challenge. https://github.com/MadryLab/mnist_challenge, b.

Jan Hendrik Metzen, Tim Genewein, Volker Fischer, and Bastian Bischoff. On detecting adversarial perturbations. *CoRR*, abs/1702.04267, 2017. URL https://arxiv.org/abs/1702.04267.

Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2574–2582, 2016.

Anh Nguyen, Jason Yosinski, and Jeff Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 427–436, 2015.

Tianyu Pang, Chao Du, Yinpeng Dong, and Jun Zhu. Towards robust detection of adversarial examples. In *Advances in Neural Information Processing Systems*, pp. 4579–4589, 2018. URL https://arxiv.org/abs/1706.00633.

Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. In *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*, pp. 372–387. IEEE, 2016a.

Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a defense to adversarial perturbations against deep neural networks. In *2016 IEEE Symposium on Security and Privacy (SP)*, pp. 582–597. IEEE, 2016b.

Kevin Roth, Yannic Kilcher, and Thomas Hofmann. The odds are odd: A statistical test for detecting adversarial examples. *arXiv preprint arXiv:1902.04818*, 2019.

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

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

Ali Shafahi, W Ronny Huang, Mahyar Najibi, Octavian Suciu, Christoph Studer, Tudor Dumitras, and Tom Goldstein. Poison frogs! targeted clean-label poisoning attacks on neural networks. In *Advances in Neural Information Processing Systems*, pp. 6103–6113, 2018.

Dinggang Shen, Guorong Wu, and Heung-Il Suk. Deep learning in medical image analysis. *Annual review of biomedical engineering*, 19:221–248, 2017.

Aman Sinha, Hongseok Namkoong, and John Duchi. Certifiable distributional robustness with principled adversarial training. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=Hk6kPgZA-.

Jiawei Su, Danilo Vasconcellos Vargas, and Kouichi Sakurai. One pixel attack for fooling deep neural networks. *IEEE Transactions on Evolutionary Computation*, 2019.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013. URL https://arxiv.org/abs/1312.6199.

Shixin Tian, Guolei Yang, and Ying Cai. Detecting adversarial examples through image transformation. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. Robustness may be at odds with accuracy. *stat*, 1050:11, 2018. URL https://arxiv.org/abs/1805.12152.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pp. 5998–6008, 2017.

Chaowei Xiao, Bo Li, Jun-Yan Zhu, Warren He, Mingyan Liu, and Dawn Song. Generating adversarial examples with adversarial networks. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pp. 3905–3911. AAAI Press, 2018.

Weilin Xu, David Evans, and Yanjun Qi. Feature squeezing: Detecting adversarial examples in deep neural networks. *arXiv preprint arXiv:1704.01155*, 2017.

Zhihao Zheng and Pengyu Hong. Robust detection of adversarial attacks by modeling the intrinsic properties of deep neural networks. In *Advances in Neural Information Processing Systems*, pp. 7913–7922, 2018.

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 as the source for 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.

A.2 CIFAR10 TRAINING

We train our CIFAR10 base detectors using the ResNet50 model (He et al., 2016; Madry et al., 2017). 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. Our detector is then trained by finetuning the subnetwork using objective 5. Our pretrained classifier has a test accuracy of 95.01% (fetched from the CIFAR10 adversarial challenge (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 as sources for adversarial examples. Adversarial examples for training L_2 and L_∞ models are both optimized using PGD attack with normalized steepest descent (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 10

$$L(x') = -\max_{i \neq y} z(H(x'))_i + c \cdot \|x' - x\|_2^2. \tag{16}$$

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 generative detector with L_∞ $\epsilon = 0.3$ base detectors reached 0.9962 FPR, generative detector with L_∞ $\epsilon = 0.5$ base detectors reached 0.9936 FPR, and CIFAR10 generative detector reached 0.9995 FPR.

We note that it's very difficult to find the optimal c in loss 16 using binary search, hence performance based on mean L_2 distortion is not precise, and we encourage future work to measure detection performances based on norm constrained attacks (as in Figure 2a).

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

Dataset	Initial c	c lower bound	c upper bound	bsearch depth	PGD steps	PGD step-size	Threshold	PGD optimizer
MNIST	0.0	0.0	8.0	20	500	0.1	3.6	Adam
CIFAR10	0.0	0.0	1.0	20	100	2.56 (0-255 scale)	-5.0	L_2 normalized steepest descent

C MORE EXPERIMENTAL RESULTS

C.1 MORE MNIST RESULTS

Table 8: AUC scores of the first two base detectors under different strengths of PGD attacks using normalized steepest descent. The gradient descent rules for L_2 and L_∞ constrained attacks are respectively $x_{n+1} = x_n - \gamma \frac{\nabla f(x_n)}{\|\nabla f(x_n)\|_2}$ and $x_{n+1} = x_n - \gamma \cdot \text{sign}(\nabla f(x_n))$.

PGD attack steps, step-size	L_∞ $\epsilon = 0.3$ base detector		L_∞ $\epsilon = 0.5$ base detector	
	$k = 0$	$k = 1$	$k = 0$	$k = 1$
200, 0.01	0.99962	0.99973	0.99820	0.99901
2000, 0.005	0.99959	0.99971	0.99795	0.99872

PGD attack steps, step-size	L_2 $\epsilon = 2.5$ base detector		L_2 $\epsilon = 5.0$ base detector	
	$k = 0$	$k = 1$	$k = 0$	$k = 1$
200, 0.1	0.99906	0.99916	0.99960	0.99997
2000, 0.05	0.99855	0.99883	0.99237	0.99994

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

	$k=0$ base detector				$k=1$ base detector			
Attack	$L_\infty\ \epsilon=0.3$	$L_\infty\ \epsilon=0.5$	$L_2\ \epsilon=2.5$	$L_2\ \epsilon=5.0$	$L_\infty\ \epsilon=0.3$	$L_\infty\ \epsilon=0.5$	$L_2\ \epsilon=2.5$	$L_2\ \epsilon=5.0$
$L_\infty\ \epsilon=0.3$	0.99959	0.99966	0.99927	0.99925	0.99971	0.99967	0.99949	0.99984
$L_\infty\ \epsilon=0.5$	0.99436	0.9983	0.99339	0.99767	0.99778	0.99869	0.99397	0.99961
$L_2\ \epsilon=2.5$	0.99974	0.99969	0.99962	0.99944	0.99965	0.99955	0.99968	0.99987
$L_2\ \epsilon=5.0$	0.96421	0.98816	0.97747	0.99577	0.98268	0.98687	0.98117	0.99986

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.

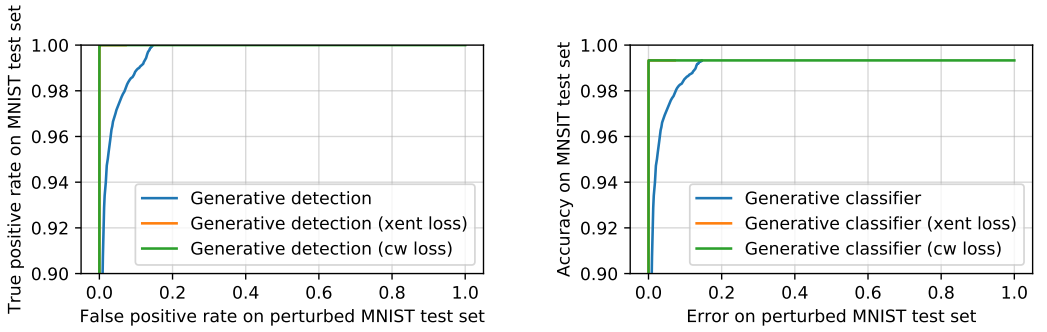
	MNIST $k=0$ base detector	
Attack	$L_\infty\ \epsilon=0.5$ trained	$L_2\ \epsilon=5.0$ trained
fixed start	0.99830	0.99578
50 random restarts	0.99776	0.99501

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.

Base detector	$k=0$	$k=1$	$k=2$	$k=3$	$k=4$	$k=5$	$k=6$	$k=7$	$k=8$	$k=9$
AUC	0.99959	0.99971	0.99876	0.99861	0.99859	0.99861	0.99795	0.99863	0.99687	0.99418

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.

Base detector	$k=0$	$k=1$	$k=2$	$k=3$	$k=4$	$k=5$	$k=6$	$k=7$	$k=8$	$k=9$
AUC	0.99830	0.99869	0.99327	0.99355	0.99314	0.99228	0.99424	0.99439	0.97875	0.9769



(a) (b)

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.

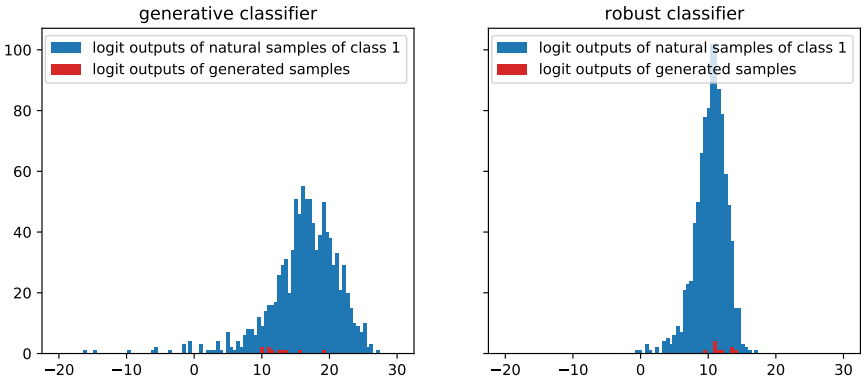


Figure 8: Distributions of class 1's logit outputs of natural samples from class 1 and perturbed samples from the first row of Figure 3 (MNIST dataset).

C.2 MORE CIFAR10 RESULTS

C.2.1 MORE ROBUSTNESS TEST RESULTS

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

	CIFAR10 $k = 0$ base detector	
Attack	L_∞ $\epsilon = 2.0$ trained	L_∞ $\epsilon = 8.0$ trained
fixed start	0.9866	0.9234
10 random starts	0.9866	0.9233

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

Attack	AUC
L_2 $\epsilon = 80$, steps 20, step-size 10	0.9814
L_∞ $\epsilon = 2$, steps 10, step-size 0.5	0.9841

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.

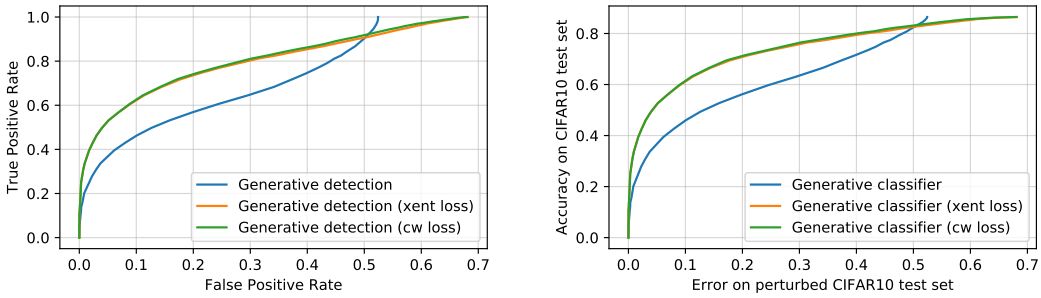
L_2 attack steps, step-size	L_2 $\epsilon = 80$ models	
	$k = 0$	$k = 1$
20, 10	0.9839	0.9924
50, 5.0	0.9837	0.9922

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

Base detector	$k = 0$	$k = 1$	$k = 2$	$k = 3$	$k = 4$	$k = 5$	$k = 6$	$k = 7$	$k = 8$	$k = 9$
AUC	0.9866	0.9926	0.9721	0.9501	0.9773	0.9636	0.9859	0.9908	0.9930	0.9916

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

Base detector	$k = 0$	$k = 1$	$k = 2$	$k = 3$	$k = 4$	$k = 5$	$k = 6$	$k = 7$	$k = 8$	$k = 9$
AUC	0.9234	0.9553	0.8393	0.7893	0.8494	0.8557	0.9071	0.9276	0.9548	0.9370



(a) (b)

Figure 9: Performance of generative detection (a) and generative classification (b) on CIFAR10 dataset under attacks with different loss function. Cross-entropy and CW loss is only able to outperforms loss 10 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.

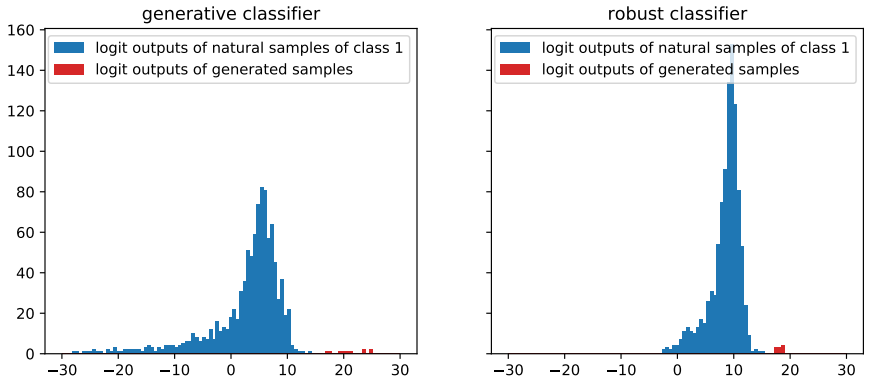


Figure 10: Distributions of class 1's logit outputs of natural samples of class 1 and generated samples from the first row of Figure 6 (CIFAR10 dataset).

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_∞ $\epsilon = 2.0$ 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 samples, and adversarial examples produced using a large step-size, live in two quite different data spaces — training a classifier to separate these two kinds of data is easy, but the performance won't generalize to real attacks. While Madry et al. (2017) was able to train their CIFAR10 robust classifier using step-size 2, we found this step-size not working in our case.

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

Attack steps, stepsize	Training steps, stepsize	
	10, 0.5	10, 1.0
10, 0.5	0.9866	0.9965
40, 0.1	0.9892	0.8848

C.2.3 EFFECTS OF PERTURBATION LIMIT

To study the effects of perturbation limit on asymmetrical adversarial training, we compare one L_∞ $\epsilon = 2.0$ trained and one L_∞ $\epsilon = 8.0$ trained base detector. In Figure 11 we show the training and testing history of these two models. The $\epsilon = 2.0$ model history show that by adversarial finetuning the model reach 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.

The comparison shows that training with larger perturbation limit is more difficult and time-consuming, and could lead to performance decrease on natural samples. The benefit is that the model learns more interpretable features. In Figure 12, perturbations generated by attacking the naturally trained classifier (corresponds to 0 perturbation limit) don't have clear semantic. In contrast, perturbed samples of the L_∞ $\epsilon = 8$ model are completely legible.

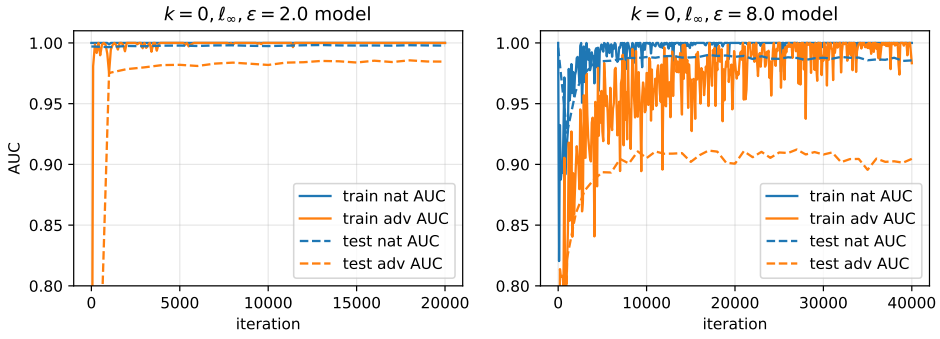


Figure 11: Training and testing AUC histories of two base detectors. Adv AUC is the AUC score computed on $\{(x, 0) : x \in \mathcal{D}'^{f}_{\setminus k}\} \cup \{(x, 1) : x \in \mathcal{D}^{f}_{k}\}$, and nat AUC is the score computed on $\{(x, 0) : x \in \mathcal{D}^{f}_{\setminus k}\} \cup \{(x, 1) : x \in \mathcal{D}^{f}_{k}\}$.

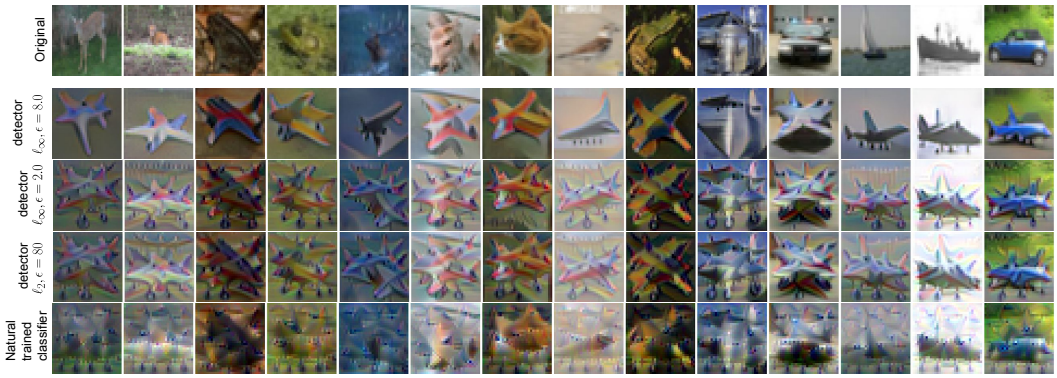
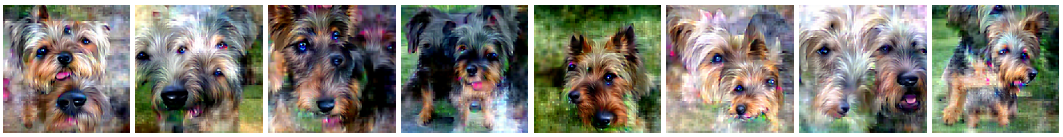


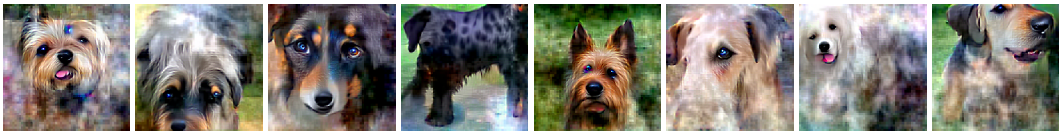
Figure 12: Perturbed samples produced by attacking the $k = 0$ (airplane) detectors and the natural trained classifier's 1st logit output. All samples reached the same L_2 perturbation of 1200 (produced using PGD attacks of step-size 10.0).

C.3 MORE IMAGENET RESULTS

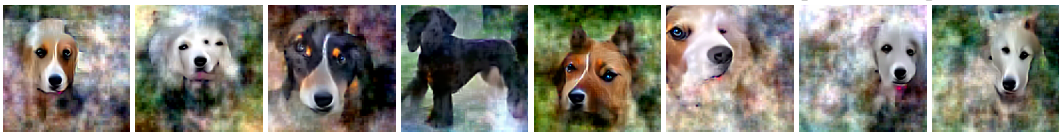
(a) L_2 $\epsilon = 3.5$ trained robust classifier with L_2 $\epsilon = 40$ constrained PGD attack of steps 60 and step-size 1.0 (Santurkar et al., 2019).



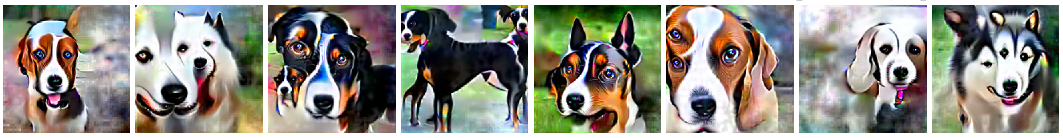
(b) L_∞ $\epsilon = 0.02$ trained detector with L_2 $\epsilon = 40$ constrained PGD attack of steps 60 and step-size 1.0.



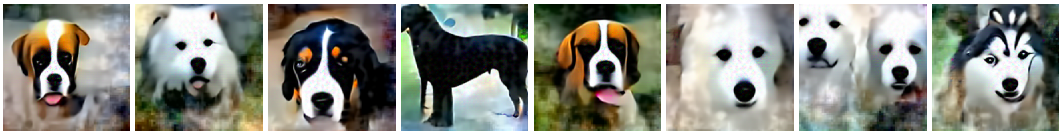
(c) L_∞ $\epsilon = 0.05$ trained detector with L_2 $\epsilon = 40$ constrained PGD attack of steps 60 and step-size 1.0.



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



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



(f) L_∞ $\epsilon = 0.3$ trained detector with L_2 $\epsilon = 100$ constrained PGD attack of steps 10 and step-size 10.0.

Figure 13: 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 are still under training and haven't reached robustness. Please refer to Santurkar et al. (2019) for the detail about how the class-conditional Gaussian is estimated.

(a)

(b)

Figure 14: Perturbed samples produced by attacking the L_∞ $\epsilon = 0.3$ 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.

(a)

(b)

Figure 15: Dog image retouching by attacking the L_∞ $\epsilon = 0.3$ 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

Figure 17: More $224 \times 224 \times 3$ random samples generated by attacking the L_∞ $\epsilon = 0.3$ trained detector with L_2 $\epsilon = 100$ constrained PGD attack of steps 10 and step-size 10.0.