One Man’s Trash is Another Man’s Treasure: Resisting Adversarial Examples by Adversarial Examples

Chang Xiao  
Columbia University  
New York, 10027  
{chang, cxz}@cs.columbia.edu

Abstract

Modern image classification systems are often built on deep neural networks, which suffer from adversarial examples—images with deliberately crafted, imperceptible noise to mislead the network’s classification. To defend against adversarial examples, a plausible idea is to obfuscate the network’s gradient with respect to the input image. This general idea has inspired a long line of defense methods. Yet, almost all of them have proven vulnerable.

We revisit this seemingly flawed idea from a radically different perspective. We embrace the omnipresence of adversarial examples and the numerical procedure of crafting them, and turn this harmful attacking process into a useful defense mechanism. Our defense method is conceptually simple: before feeding an input image for classification, transform it by finding an adversarial example on a pre-trained external model. We evaluate our method against a wide range of possible attacks. On both CIFAR-10 and Tiny ImageNet datasets, our method is significantly more robust than state-of-the-art methods. Particularly, in comparison to adversarial training, our method offers lower training cost as well as stronger robustness.

1. Introduction

Deep neural networks have vastly improved the performance of image classification systems. Yet they are prone to adversarial examples. Those are natural images with deliberately crafted, imperceptible noise, aiming to mislead the network’s decision entirely [5, 41]. In numerous applications, from face recognition authorization to autonomous cars [36, 43], the vulnerability caused by adversarial examples gives rise to serious security concerns and presses for efficient defense mechanisms.

The defense, unfortunately, remains grim. Recent studies [34, 45, 12] suggest that the prevalence of adversarial examples may be an inherent property of high-dimensional natural data distributions. Facing this intrinsic difficulty of eliminating adversarial examples, a plausible thought is to conceal them—making them hard to find. Indeed, a long line of works aims to obfuscate the network model’s gradient with respect to its input [50, 14, 49, 7, 39, 33], motivated by the fact that the gradient information is essential for crafting adversarial examples: the gradient indicates how to perturb the input to alter the network’s decision.

Yet, almost all these gradient obfuscation based defenses have proven vulnerable. In their recent seminal work, Athalye et al. [2] presented a suite of strategies for estimating network gradients in the presence of gradient obfuscation. Adversarial examples crafted by their method have successfully fooled many existing defense models, some of which even yield 0% accuracy under their attack.

We revisit the idea of gradient obfuscation but take a radically different approach. Instead of expelling adversarial examples, we embrace them. Instead of obstructing the way of finding adversarial examples on a model, we exploit it to strengthen the robustness of another model.

Our defense is conceptually simple: before feeding an input image to a classification model, we transform it through the process of finding adversarial examples on a pre-trained simple model $f_b$, deliberately adding strong adversarial noise. The transformed image is then fed into another model $f_a$ for classification. The same pipeline is applied in both training and inference.

Figure 1. A simple and effective defense mechanism. Given an input image, our defense method first transforms it through the process of crafting adversarial examples on a pre-trained simple model $f_b$, deliberately adding strong adversarial noise. The transformed image is then fed into another model $f_a$ for classification. The same pipeline is applied in both training and inference.
The robustness of our defense model $f(g(x))$ stems from the fundamental difficulties of estimating the gradient of $g(x)$ with respect to $x$. Finding an adversarial example amounts to searching for a local minimum on a highly fluctuated objective landscape [26]. As a result, $g(x)$ is not an analytic function, not smooth, not deterministic, but an iterative procedure with random initialization and non-differentiable operators. We show that all these traits together constitute a highly robust defense mechanism.

We play devil’s advocate in attacking our defense model thoroughly. We examine a wide range of possible attacks, including those having successfully circumvented many previous defenses [2]. Under these attacks, we compare the worst-case robustness of our method with state-of-the-art defense methods on both CIFAR-10 and Tiny ImageNet datasets. Our defense demonstrates superior robustness over those methods. Particularly, in comparison to models optimized with adversarial training—by far the most effective defense against white-box attacks—our method offers simultaneously lower training cost and stronger robustness.

2. Related Work

Adversarial attack. The seminal work of Biggio et al. [5] and Szegedy et al. [41] first suggested the existence of adversarial examples that can mislead deep neural networks. The latter also used a constrained L-BFGS to find adversarial examples. Goodfellow et al. [13] later introduced Fast Gradient Sign Method (FGSM) that generates adversarial examples more efficiently. Madry et al. [26] further formalized the problem of adversarial attacks and proposed Projected Gradient Descent (PGD) method, which further inspires many subsequent attacking methods [11, 8, 28, 19]. PGD-type methods are considered the strongest attacks based on first-order information, namely the network’s gradient with respect to the input [26]. To compute the gradients, the adversary must have full access to the network structure and parameters. This scenario is referred as the white-box attack.

When the adversary has no knowledge about the model, the attack, referred as black-box attack, is not as easy as the white-box attack. By far the most popular black-box attack is the so-called transfer attack, which uses adversarial examples generated on a known model (e.g., using PGD) to attack an unknown model [30]. Several methods (e.g., [44, 3, 52, 16]) are proposed to improve the transferability of the adversarial examples so that the adversarial examples generated on one model are more likely to fool another model. Another type of black-box attacking methods is query-based [6, 37, 9, 1, 20]: they evolve the model many times with different input in order to learn the behavior of the model and construct adversarial examples.

While our defense is motivated by attacks in white-box scenarios, we evaluate our method under a wide range of possibilities, including both white-box and black-box attacks.

Adversarial defense. The threat of adversarial examples has motivated active studies of defense mechanisms. By far the most successful defense against white-box attacks is adversarial training [26, 13, 38], and a rich set of methods has been proposed to accelerate its training speed or further improve its robustness [51, 54, 21, 13, 46, 35, 53, 29, 27]. In comparison to adversarial training, our method offers both stronger robustness and lower training cost.

To defend against gradient-based attacks (such as the PGD attack), a natural idea is to obfuscate (or mask) network gradients [30, 44]. To this end, there exist a long line of works that apply random transformation to input images [50, 14], or employ stochastic activation functions [10] and non-differentiable operators in the model [49, 7, 39, 33].

Unfortunately, many of these methods have proven vulnerable by Athalye et al. [2], who introduced a set of attacking strategies, including a method called Backward Pass Differentiable Approximation (BPDA), to circumvent gradient obfuscation (see further discussion in Sec. 3.1 and 3.3). Since then, a few other gradient obfuscation based defenses have been proposed [23, 31, 17, 42, 22]. But those works either report degraded robustness under BPDA attacks [23, 31] or neglected the evaluation against BPDA attacks [17, 42, 22]. Thus far, gradient obfuscation is generally considered vulnerable (and at least incomplete) [2]. We revisit gradient obfuscation, and our defense demonstrates unprecedented robustness against BPDA and other possible attacks.

3. Defense via Adversarial Transformation

We now present a simple approach to defend against adversarial attacks. We will first motivate and describe our adversarial transformation (Sec. 3.1 and 3.2), and then provide the rationale of why it improves adversarial robustness (Sec. 3.3 and 3.4), backed by empirical evidence (Sec. 4).

3.1. Motivation: Input Transformation

An attempt that has been explored in adversarial defense—albeit unsuccessfully so far—is the defense via input transformation. Consider a neural network model $f_a$ that classifies the input image $x$ (i.e., evaluating $f_a(x)$). Instead of feeding $x$ into $f_a$ directly, this defense approach transforms the input image through an operator $g$ before presenting it to the classification model (i.e., evaluating $f_a(g(x))$).

The transformation $g$ is applied in both training and inference. Provided a training dataset $X$, the network weights $\theta$ are optimized by solving

$$\theta^* = \arg \min_{\theta} \mathbb{E}_{(x,y) \in X} [\ell(f_a(g(x); \theta), y)],$$

where $x$ and $y$ are respectively the image and its corresponding label drawn from the training dataset, and $\ell$ is the loss function (such as cross entropy for classification tasks). Correspondingly, at inference time, the model predicts the label of an input image $x$ by evaluating $f_a(g(x))$. 

413
Input transformation $g(\cdot)$ offers an opportunity to implement the idea of gradient obfuscation. For example, by transforming the input image with certain randomness such as random resizing and padding [50], the network gradients become hard to estimate.

Another use of $g(\cdot)$ for defense is to remove the noise (or perturbations) in adversarial examples. For instance, $g(\cdot)$ has been used to restore a natural image from a potentially adversarial input, by projecting it on a GAN- or PixelCNN-represented image manifold [33, 39] or regularizing the input image through total variation minimization [14].

These input-transformation-based defense mechanisms seem plausible. Yet they are all fragile. As demonstrated by Athalye et al. [2], with random input transformation, adversarial examples can still be found using Expectation over Transformation [3], which estimates the network gradient by taking the average over multiple trials (more details in Sec. 3.3). The noise-removal transformation is also ineffective. One can use Backward Pass Differentiable Approximation [2] to easily construct effective adversarial examples. In short, the current consensus is that input transformation as a defense mechanism remains vulnerable.

We challenge this consensus. We now present a new input transformation method for gradient obfuscation, followed by the explanation of why it is able to avoid the shortcomings of prior work and offer stronger adversarial robustness.

3.2. Adversarial Transformation

Our input transformation operation takes an approach opposite to the intuition behind previous methods [14, 39, 33]. In contrast to those aiming to purge input images of the adversarial noise, we embrace adversarial noise. As we will show, our transformation injects noticeably strong adversarial noise into the input image. This seemingly counter-intuitive operation is able to strengthen the network model in training, making it more robust.

Our transformation operation relies on another network model $f_b$, whose choice will be discussed later in Sec. 3.4. The model $f_b$ is pre-trained to perform the same task as $f_a$. Then, given an input image $x$, the transformation operator $g(\cdot)$ is defined as the process that finds the adversarial example nearby $x$ to fool $f_b$. Formally, this process is meant to reach a local minimum of the optimization problem,

$$g(x) = \arg \min_{x' \in \Delta_x} \ell(f_b(x'), y_L),$$

(2)

where $\ell(\cdot)$ is the loss function as used in network training (1); and $y_L$ is the adversarial target, setting to be the input $x'$'s least likely class predicted by $f_b$. The adversarial examples are restricted in $\Delta_x$, an $L_\infty$-ball at $x$, defined as $\|x' - x\|_\infty < \Delta$. The perturbation range $\Delta$ is a hyperparameter.

Transformation $g(x)$ defined in (2) can be implemented using any gradient-based attacking methods (such as Deepfool [28] and C&W [8]). We choose to use the least-likely class projected gradient descent (LL-PGD) method [19, 44]. LL-PGD is an iterative process, wherein each iteration updates the adversarial example by the rule,

$$x'_t = \Pi_{x' \in \Delta_x} \left[ x'_{t-1} - \epsilon \cdot \text{sgn}(\nabla_{x'} \ell(f_b(x'_{t-1}), y_L)) \right].$$

(3)

Here $x'_{t}$ denotes the adversarial example after $t$ iterations; $\text{sgn}(\cdot)$ is the sign function, and $\Pi_{x' \in \Delta_x} [\cdot]$ projects the image back into the allowed perturbation ball $\Delta_x$. This iterative process starts from a random perturbation of input image $x$, namely $x + \delta$, where each element (pixel) in $\delta$ is uniformly drawn from $[-\Delta, \Delta]$. The output $x'_N$ (after $N$ iterations) is the transformed version of $x$. In other words, $g(x) = x'_N$, which we refer as adversarial transformation.

After defining the adversarial transformation $g(\cdot)$ based on the pre-trained model $f_b$, we use $g(\cdot)$ to train the model $f_a$ as described in (1). At inference time, the label of an image $x$ is predicted as $f_a(g(x))$. Figure 1 illustrates the pipeline of our method.

Differences from adversarial training. With adversarial transformation, our training process superficially resembles the adversarial training, because both training processes need to search for adversarial examples of the input training data. But fundamental differences exist. In adversarial training, a single model $f_a$ is used for crafting adversarial examples and evolving itself at each epoch, whereas our method involves two models: the model $f_b$ is pre-trained and stays fixed during both the training of $f_a$ and the inference using $f_a$.

The consequence of using a fixed external model $f_b$ for adversarial transformation is substantial. As we will discuss in Sec. 3.4, $f_b$ can be chosen much simpler than $f_a$. As a result, crafting the adversarial examples on $f_b$ has lower cost than that on $f_a$, and thus our training process is faster than adversarial training (see experiments in Sec. 5). More remarkably, the adversarial transformation using $f_b$ makes the model $f_a$ much harder to attack, as explained next.

3.3. Rationale behind Adversarial Transformation

Embracing adversarial noise. Given an input image $x$, our adversarial transformation effectively adds perturbation
noise to $x$. The perturbation range $\Delta$ controls how much noise is added. Normally, in adversarial attacks, $\Delta$ is set small to generate adversarial examples perceptually similar to the input image. But when we use adversarial attacks (on $f_b$) as a means of input transformation for training $f_a$, we have the freedom to use a much larger $\Delta$, thereby adding noticeably stronger adversarial noise (see Fig. 2-c).

At training time, the excessively strong adversarial noise forces the network $f_a$ to learn how to classify robustly. This is because perturbations crafted on an external model can approximate the adversarial examples of the model under training (an insight inspired the prior work [44]). This reason, although valid, can not explain how our method is able to avoid the deficiencies of prior defense methods. There exist deeper reasons:

**Randomness.** The adversarial noise added by our $g(x)$ is randomized, since the update rule (3) always starts from the input image with a random perturbation (i.e., $x + \delta$ with uniformly sampled $\delta_i \sim [-\Delta, \Delta]$). Randomization is not new; prior defenses also employ randomized transformations to the input. But they have been circumvented by Expectation Over Transformation (EOT) [2, 3]. EOT attack first estimates the gradient of expected $f(\tilde{g}(x))$ with respect to $x$ using the relationship $\nabla \tilde{g}_{\sim T} f(\tilde{g}(x)) = \mathbb{E}_{\sim T} \nabla f(\tilde{g}(x))$, where $\tilde{g}(\cdot)$ is a deterministic version of $g(\cdot)$ sampled from the distribution of randomized transformations $\mathcal{T}$. It then uses the estimated gradients in PGD-type attacks to generate adversarial examples. Thus, the feasibility of EOT hinges on a reliable estimation of $\nabla f(\tilde{g}(x))$. In our method, $\tilde{g}(x)$ corresponds to solving the optimization problem (2) starting from a particular sample $x + \delta$.

In what follows, we examine a range of strategies that have been successfully used to estimate $\nabla f(\tilde{g}(x))$ in prior defense methods (and thus break them), and show that our method is robust against all those attacking strategies.

**Automatic differentiation.** By chain rule, the estimation of $\nabla f(\tilde{g}(x))$ requires the knowledge of $\tilde{g}(\cdot)$’s Jacobian (first-order derivatives) $D\tilde{g}(x)$. A straightforward attempt to this end is by unrolling the iterative steps (3) and using automatic differentiation (AD) [47] to compute $D\tilde{g}(x)$. Yet, this is infeasible. As shown in (3), the iterative steps involves non-differentiable operators including $\text{sgn}(\cdot)$ and $\Pi_{x' \in \Delta_x}[\cdot]$. Thus, directly applying AD leads to erroneous estimation of $D\tilde{g}(x)$, which in turn obstructs the search for adversarial examples. Our early experiments indeed show that virtually no adversarial examples crafted using AD can fool our model.

**Backward Pass Differentiable Approximation (BPDA).** To circumvent the defense using non-differentiable operators, Athalye et al. [2] introduced a strategy called Backward Pass Differentiable Approximation (BPDA) to estimate the defense model’s gradients. The idea is to replace the non-differentiable operators in $\tilde{g}$ with differentiable approximations, and estimate the derivatives $D\tilde{g}(x)$ in AD by computing the AD’s forward pass using the original $\tilde{g}$ and computing its backward pass using $\tilde{g}$’s differentiable approximation. BPDA has succeed in gradient-based attacks (such as PGD and C&W [8]) toward many prior defenses, allowing the adversary to craft efficient adversarial examples.

When applying BPDA to estimate the gradients of our defense model, we replace $\text{sgn}(\cdot)$ and $\Pi_{x' \in \Delta_x}[\cdot]$ in (3) with their differentiable approximations (see Sec. 4.1 for details). We found that if the number of iterations (LL-PGD steps) for applying (3) is low (i.e., $\leq 2$), BPDA indeed enables the adversary to find valid adversarial examples. But when the number of iterations is set moderately high (i.e., $> 4$), BPDA is greatly thwarted; the adversary can hardly find any valid adversarial example (see Sec. 4.1). This is because the differentiable approximations must be applied in each iteration, and as the number of iterations increases, the approximation error accumulates rapidly.

**Finite difference gradients.** Another strategy for estimating $D\tilde{g}(x)$ is the classic finite difference estimation. Each element in the Jacobian matrix $D\tilde{g}(x)$ can be estimated using

$$\frac{\partial \tilde{g}_i(x)}{\partial x_j} \approx \frac{1}{2h} [\tilde{g}_i(x + h_j) - \tilde{g}_i(x - h_j)],$$

where $\tilde{g}_i(x)$ indicates the $i$-th element (or pixel) of the transformed image, and $h_j$ is a vector with all zeros except the $j$-th element (or pixel) which has a value $h$.

Our defense inherently thwart this attacking strategy. It causes the adversary to suffer from either exploding or vanishing gradients [2]. Figure 3-left shows a 1D depiction illustrating this phenomenon in our method. Indeed, our experiments confirm that it is too unreliable to estimate derivatives using (4) (see Fig. 3-right).

**Reparameterization.** Vanishing and exploding gradients have been exploited as a defense mechanism [39, 33]. Yet
those defenses have been proven vulnerable under a reparameterization strategy [2]. This strategy aims to find some differentiable function $h(\cdot)$ for a change-of-variable $x = h(z)$ such that $\tilde{g}(h(z)) \approx h(z)$. If such a function $h(\cdot)$ can be found, then one can compute the gradient of the differentiable function $f(h(z))$ to launch adversarial attack.

To break our defense using this strategy, one must find an $h(\cdot)$ that constructs the adversarial examples of $f_b$ directly (so that $\tilde{g}(h(\cdot)) = h(\cdot)$), without solving the optimization problem (2). We argue that finding such an $h(\cdot)$ is extremely hard. If $h(\cdot)$ could be constructed, we would have a direct way of crafting adversarial examples; PGD-type iterations would not be needed; and the entire territory of adversarial learning would be redefined—which are unlikely to happen.

Indeed, we implemented this strategy by training a neural network model $h_a$ that aims to minimize $\|h_a(x) - \tilde{g}(x)\|_2$ over the natural image distribution. This attempt is futile. Our experiments show that the generalization error of the trained $h_a$ is too high to launch any valid adversarial attack (see Appendix A.2). This conclusion also echoes the prior studies [4, 48], which show that learning-based adversarial attacks usually perform worse than gradient-based attacks.

**Identity mapping approximation.** Some prior defense methods also use an optimization process to transform the input image—for example, the optimization that aims to erase adversarial noise from the input image [14, 33]. In those defenses, the transformed image $g(x)$ remain similar to the input $x$. Consequently, as shown in [2], those defenses can be easily circumvented by replacing $g(\cdot)$ with the identity mapping in the backward pass of BPDA attack.

Similarly, in our defense, if the perturbation range $\Delta$ in (3) for defining $g(\cdot)$ were set small, $g(x)$ (the adversarial example of $f_b$) would be close to $x$, and our defense would be at risk. To prevent this vulnerability, we must ensure that $g(\cdot)$ be far from the identity mapping. This requires us to set a relatively large $\Delta$. In practice, we use $\Delta = 0.2$ for pixel values ranging in $[0, 1]$ (see details in Sec. 4.1).

It turns out that a relatively large $\Delta$ is necessary but not sufficient. The choice of the network model $f_b$ also affects how far $g(x)$ is from $x$ statistically, as we will discuss next.

### 3.4. Choosing Pre-trained Model $f_b$

A large perturbation range $\Delta$ allows our adversarial transformation $g(\cdot)$ to output an image far from the input. Yet, because of the randomness in $g(\cdot)$, a large $\Delta_b$ can not guarantee that $g(x)$ is statistically different from $x$. If the expectation over the transformation $\mathbb{E}_{\tilde{g} \sim \mathcal{T}} \tilde{g}(x)$ remains close to $x$, our defense method may still suffer from the aforementioned BPDA attack, in which identity mapping can be used to approximate $g(\cdot)$ in the backward pass.

This intuition is supported by an empirical discovery. We experimented with an input transformation $g(\cdot)$ constructed using an *untrained* model $f_b$ whose weights are assigned randomly. As shown in Fig. 4 and the first column in Table 1, the expectation over transformation $\mathbb{E}_{\tilde{g} \sim \mathcal{T}} \tilde{g}(x)$ is indeed close to $x$ (in $L_\infty$ norm), and the BPDA attack with identity mapping approximation can easily fool this defense model.

Next, we train a series of models $f_b^{(i)}$, each obtained with an increasing number of training epochs. We found that as the number of training epochs increases, the expectation over transformation $\mathbb{E}_{\tilde{g} \sim \mathcal{T}} \tilde{g}(x)$ resulted by using each of these $f_b^{(i)}$ models drifts further away from $x$, that is, $\|x - \mathbb{E}_{\tilde{g} \sim \mathcal{T}} \tilde{g}(x)\|_\infty$ increases. Meanwhile, the defense model $f(g(\cdot))$ trained with the corresponding $f_b^{(i)}$ becomes more robust, yielding increasingly better robust accuracy under the BPDA attack (see Table 1).

Remarkably, we discover that an even larger distance $\|x - \mathbb{E}_{\tilde{g} \sim \mathcal{T}} \tilde{g}(x)\|_\infty$ can be obtained, if the model $f_b$ is *adversarially* trained. When used in $g(\cdot)$, the adversarially trained model $f_b$ further improves the robustness of our defense model. This discovery confirms our intuition.

**Computational performance.** The choice of $f_b$ also affects the computational cost of our defense method. A complex network structure of $f_b$ makes $g(\cdot)$ expensive, which in turn imposes a large performance overhead on both the training of $f_b$ and the inference using $f_b$. Therefore, a simple network structure is preferred.

The freedom of choosing a simple network $f_b$ brings our method a performance advantage over adversarial training. In adversarial training, adversarial examples are crafted on the classification network $f_a$ for each input image at every
epoch. In our training, however, by choosing a model \( f_b \) simpler than \( f_a \), it becomes faster to find adversarial examples. As shown in our experiments (in Sec. 5), in comparison to adversarial training, our defense requires shorter training time, and at the same time offers stronger robustness.

Guiding rules. In summary, we present two guiding rules for choosing \( f_b \). 1) \( f_b \) should be chosen to yield a large \( \|x - \mathbb{E}_{g \sim \mathcal{T}} \hat{g}(x)\|_\infty \) value. Given an \( f_b \)’s network structure, adversarial training on \( f_b \) (in pre-training step) is preferred. 2) Meanwhile, the structure of \( f_b \) should be as simple as possible. In Appendix A.1, we report \( f_b \)’s network structure that we use in our experiments.

4. Devil’s Advocate

We now play devil’s advocate in attacking our defense method. In our defense, the network gradient with respect to the input (i.e., \( \nabla f_a(g(x)) \)) is intentionally undefined. Thus one can not craft adversarial examples by directly applying PGD-type methods on our defense (recall Sec. 2). We therefore evaluate our defense against a range of other possible attacks, including those discussed in Sec. 3.3. Later in Sec. 5, we will compare the worst-case robustness of our defense under these attacks with various recently proposed defense methods. We also release our code for public evaluation1.

Common experiment setups. Experiments in this section are conducted on CIFAR-10 dataset [18] with standard training/test split. We use ResNet18 [15] as the classification model \( f_a \) and a small VGG-style network for \( f_b \), whose details are given in Appendix A.1. All models are trained for 80 epochs using Stochastic Gradient Descent (SGD) (constant learning rate=0.1, momentum=0.9). Our adversarial transformation \( g(\cdot) \) performs LL-PGD update (3) for 13 iterations, each with a stepsize \( \epsilon = \Delta/6 \). The perturbation range \( \Delta \) varies in individual experiments, and will be reported therein.

Metric. Following prior work, our evaluation uses an accuracy measure defined as the ratio of the number of correctly classified images to the total number of tested images. We refer to this measure as standard accuracy if the tested images include only clean images, and as robust accuracy if the tested images consist of adversarially crafted images.

<table>
<thead>
<tr>
<th></th>
<th>Untrained</th>
<th>1 epoch</th>
<th>2 epochs</th>
<th>5 epochs</th>
<th>10 epochs</th>
<th>15 epochs</th>
<th>20 epochs</th>
<th>Adv. trained</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard Acc.</td>
<td>81.8%</td>
<td>81.6%</td>
<td>82.4%</td>
<td>83.0%</td>
<td>82.4%</td>
<td>82.2%</td>
<td>82.7%</td>
<td>82.9%</td>
</tr>
<tr>
<td>BPDA-I Acc.</td>
<td>30.6%</td>
<td>30.7%</td>
<td>40.0%</td>
<td>46.4%</td>
<td>62.3%</td>
<td>63.1%</td>
<td>62.9%</td>
<td>80.5%</td>
</tr>
<tr>
<td>Avg. ( L_\infty ) dist.</td>
<td>0.005</td>
<td>0.033</td>
<td>0.036</td>
<td>0.067</td>
<td>0.130</td>
<td>0.133</td>
<td>0.136</td>
<td>0.272</td>
</tr>
</tbody>
</table>

Table 1. Discovery for choosing \( f_b \). Corresponding to individual columns are \( f_b \) models untrained, trained with an increasing number of epochs, and adversarially trained. For the defense model \( f_a(g(\cdot)) \) equipped with each \( f_b \), we evaluate its standard accuracy (first row) and robust accuracy (second row) under the BPDA-I attack (see Sec. 4.1). The third row shows \( \|x - \mathbb{E}_{g \sim \mathcal{T}} \hat{g}(x)\|_\infty \) where \( \mathbb{E}_{g \sim \mathcal{T}} \hat{g}(x) \) is estimated using 5000 samples. Notice the correlation between the increase of the \( L_\infty \) distance and the increase of adversarial robustness.

4.1. BPDA Attack and the Variants

BPDA attack [2], as reviewed in Sec. 3.3, is a powerful way to estimate network gradients that are obfuscated by defense methods. The estimated gradients are then used in PGD-type methods (if the defense is deterministic) or the EOT method (if the defense is randomized) for crafting adversarial examples. BPDA has circumvented a handful of recent defense techniques [14, 50, 25, 39, 7] that implement gradient obfuscation, in many defenses resulting in 0% robust accuracy. We therefore evaluate our defense against it and its possible variants.

Differentiable approximation on backward pass. The update rule (3) in our adversarial transformation involves two non-differentiable operators, namely, \( \text{sgn}(\cdot) \) and \( \Pi_{x' \in \Delta_x}(\cdot) \), whose specific forms are given in Appendix A.3. To launch BPDA attack, we need to replace them with differentiable operators and compute their derivatives. We experimented with two different smooth approximations of \( \text{sgn}(\cdot) \): the soft sign \( \frac{\hat{x}}{1+|\hat{x}|} \) and \( \text{tanh} \) function \( \frac{e^x-e^{-x}}{e^x+e^{-x}} \). The smooth approximation of \( \Pi_{x' \in \Delta_x}(\cdot) \) is not explicitly defined. Instead, we directly approximate its derivative using

\[
\frac{d}{dx} \Pi_{x' \in \Delta_x}(x) \approx \begin{cases} 
1, & \text{if } |x| < \Delta, \\
\frac{1}{1+|x|}, & \text{otherwise.}
\end{cases}
\]

Reported in Fig. 5, the experiments show that our defense is robust to this attack, as long as the number of LL-PGD steps \( N \) in \( g(\cdot) \) is not too small (i.e., \( N > 5 \)).

If the number of LL-PGD steps \( N \) is set too small, BPDA attack is indeed able to find adversarial examples (Fig. 5). Therefore, another attempt one may ponder is to craft adversarial examples on a model trained with a small \( N \) (\( N \leq 3 \)), and use them to transfer attack our defense (which is trained with a larger \( N \)). This attack remains ineffective (see details in Appendix A.3). We conjecture that this is because the adversarial examples for the model with a small \( N \) have a different distribution from that with a larger \( N \) [30, 40].

Identity mapping approximation. Another possible attack is by replacing the input transformation \( g(\cdot) \) with the identity mapping for gradient estimation in BPDA backward pass (recall discussion in Sec. 3.3). We refer this attack as BPDA-I attack. Under this attack, several previous defenses (e.g., [33, 39, 14]) have been nullified.

---

1https://github.com/a554b554/DefenseByAttack
We applied BPDA-I attack on our defense. The attack setup and results are summarized in Fig. 6. As the perturbation size $\Delta$ in the adversarial transformation increases, the robust accuracy of our method increases. The robust accuracy is always upper bounded by the standard accuracy, which decreases gradually as $\Delta$ increases. If $\Delta$ is too large, the excessive perturbations to the input make the network $f_a$ harder to learn and thus lower the standard and robust accuracies. Empirically, $\Delta = 0.2$ offers the best performance.

Reparameterization. In Sec. 3.3, we described another BPDA strategy, one that uses reparameterization to smoothly approximate our adversarial transformation $g(\cdot)$. As discussed therein, it is extremely hard to directly derive the reparameterization function. Instead, we attempted to train a Fully Convolutional Network [24] to represent $h(x)$. We denote this network as $h(x; \theta)$, whose weights are optimized with the loss function, $\ell(\theta) = \mathbb{E}_{x \in \mathcal{X}} \| h(x; \theta) - g(x) \|^2$. Here, $\mathcal{X}$ represents the distribution of natural images (we use CIFAR-10 as the training dataset).

Our experiment shows that although we can reach a low loss value in training $h(x; \theta)$, the loss on test dataset always stays high, indicating that $h(x; \theta)$ is always overfitted. As a result, the adversarial examples resulted in this way have almost no effect on our defense model—the accuracy drop under this attack is within 1% from the standard accuracy. See Appendix A.2 for the details of this experiment.

4.2. Gradient-Free Attacks

Several attacking methods require no gradient information of the model, and they can be employed to potentially threaten our defense. As discussed in Sec. 2, these attacks fall into two categories: transfer attack and query-based attack. Against both types of attacks we evaluate our defense.

White-box transfer attack. In white-box setting, the adversary has full knowledge of our defense model. A tempting idea is to generate adversarial examples on the classifier model $f_a$, and use them to transfer attack our defense model $f_a(g(\cdot))$. Note that this differs from BPDA-I attack in Sec. 4.1, where $f_a$ is used only in the backward pass for gradient estimation while the forward pass still uses the full model $f_a(g(\cdot))$. Here, in contrast, adversarial examples are generated solely on $f_a$. We refer this attack as White-box Transfer (WT) attack, and report the robust accuracies of our defense in Fig. 6, along with the results under BPDA-I attack. We found that our model’s robustness performances under both attacks are similar, and $\Delta = 0.2$ is the best choice.

One may realize another attacking possibility by noticing the way we choose $f_b$ (on which we perform adversarial transformation). In Sec. 3.4, we present that $f_b$ should be chosen such that for a given natural image $x$ the average transformation $\mathbb{E}_{\tilde{g} \sim T} \tilde{g}(x)$ stays far from $x$. Thus, it seems plausible to first generate adversarial examples on $f_a$ using PGD attack starting from the average transformation $\mathbb{E}_{\tilde{g} \sim T} \tilde{g}(x)$, and use them to attack our full model $f_a(g(\cdot))$. However, thanks to the large perturbation range $\Delta_x$ we use (recall Sec. 3.3), $\mathbb{E}_{\tilde{g} \sim T} \tilde{g}(x)$ is always far from a natural image (see the image in the red box of Fig. 4). Thus the adversarial examples generated in this way all have easily noticeable artifacts; they are not valid.

Black-box attacks. We also evaluate our defense against the black-box attacks, including the black-box transfer attack [30] and two most recently introduced query-based attacks, HopSkipJumpAttack (HSJ) [9] and GenAttack (GA) [11]. In Table 2, we summarize the robust accuracies of our model under these attacks, along with two baselines from the same classification model ($f_a$) optimized respectively using standard training and adversarial training.
### 5. Comparisons and Further Evaluation

#### Comparisons
We now compare the robustness of our method with other state-of-the-art defense methods under white-box attacks. Unlike many others that can be attacked using PGD-type methods, our defense model is inherently non-differentiable, immune to direct PGD attacks. Therefore it is not possible to compare all these defense methods under exactly the same attacks. Instead, we compare the worst-case robustness of our method under all the attacks described in Sec. 4 with other methods.

The comparison results on CIFAR-10 dataset are summarized in Table 3, where $A_{\text{std}}$ is the standard accuracy tested with clean images, and $A_{\text{rob}}$ is the worst-case robust accuracy under all tested attacks. The methods indicated by a star (*) are those circumvented by Athalye et al. [2].

We include their results therein as a reference. The other methods, including ours, all use ResNet18 as their classification model, trained with SGD (learning rate=0.1, momentum=0.9) for 80 epochs.

On CIFAR-10 dataset, the most effective attack on our method is the black-box transfer attack (see Sec. A.4 for detail), the robust accuracy under this attack is 73.4%. The worst-case robust accuracy of our method under BPDA attack and its variants (Sec. 4.1) is 80.2%. Nevertheless, our robustness performance is significantly better than the state-of-the-art methods, as shown in Table 3.

We also performed the comparisons on Tiny ImageNet dataset, and our method demonstrates significantly stronger robustness as well, improving robust accuracy from 18% to 40.2%. The results are reported in details in Appendix A.5.

#### Sufficiency of EOT samples
Our defense is randomized, and when using EOT to attack our method we take 500 samples of $\tilde{g}(\cdot)$ (recall experiments in Fig. 6). Here we conduct additional experiments to ensure the sufficiency of using 500 samples for estimating the expectation. As shown in Fig. 7, when the perturbation size $\Delta$ is small and the number of samples is also small (e.g., < 100), increasing sample size indeed allows EOT to better attack our method. However, when the perturbation size is set to 0.2, the value we consistently use throughout all our evaluations, EOT attacks became persistently inefficient, regardless of the sample size. Therefore, we conclude that 500 samples in EOT allow thor-

<table>
<thead>
<tr>
<th>Defense Model</th>
<th>Clean</th>
<th>Transfer</th>
<th>HSJ</th>
<th>GA</th>
</tr>
</thead>
<tbody>
<tr>
<td>No defense</td>
<td>92.9%</td>
<td>1.3%</td>
<td>3.6%</td>
<td>5.8%</td>
</tr>
<tr>
<td>Madry et al. [26]</td>
<td>81.7%</td>
<td>77.5%</td>
<td>72.1%</td>
<td>78.4%</td>
</tr>
<tr>
<td>Ours</td>
<td>82.9%</td>
<td>78.1%</td>
<td>82.0%</td>
<td>81.9%</td>
</tr>
</tbody>
</table>

Table 2. **Robustness under black-box attacks.** In the transfer attack, adversarial examples are crafted on an independently trained ResNet18 model. The query-based attacks are performed using a third-party library `foolbox` [32] with default parameters to launch these attacks. All adversarial examples are restricted in the $L_{\infty}$ ball with a perturbation size of 0.031. The last column indicates our worst-case robustness under all attacks.

<table>
<thead>
<tr>
<th>Method</th>
<th>$A_{\text{std}}$</th>
<th>$A_{\text{rob}}$</th>
<th>Attack Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>No defense</td>
<td>92.9%</td>
<td>0.0%</td>
<td>PGD</td>
</tr>
<tr>
<td>Madry et al. [26]</td>
<td>81.7%</td>
<td>42.7%</td>
<td>PGD</td>
</tr>
<tr>
<td>Zhang et al. [54]</td>
<td>80.4%</td>
<td>44.6%</td>
<td>PGD</td>
</tr>
<tr>
<td>Xie et al. [51]</td>
<td>83.8%</td>
<td>45.2%</td>
<td>PGD</td>
</tr>
<tr>
<td>Guo et al.* [14]</td>
<td>89.8%</td>
<td>0.0%</td>
<td>BPDA</td>
</tr>
<tr>
<td>Buckman et al.* [7]</td>
<td>81.1%</td>
<td>0.0%</td>
<td>BPDA</td>
</tr>
<tr>
<td>Dhillon et al.* [10]</td>
<td>82.0%</td>
<td>5.0%</td>
<td>BPDA</td>
</tr>
<tr>
<td>Song et al.* [39]</td>
<td>82.9%</td>
<td>80.2%</td>
<td>BPDA</td>
</tr>
<tr>
<td>Ours (under BPDA)</td>
<td>82.9%</td>
<td>73.4%</td>
<td>Transfer</td>
</tr>
</tbody>
</table>

Table 3. **Comparisons on CIFAR-10.** Methods indicated by * are those circumvented in [2]. We evaluate other methods using the code provided in the original papers, training them using the same network and hyperparameters as our method. The perturbation range of all adversarial examples is $\Delta = 0.031$. The last column indicates the most efficient attacking method that produces the worst robustness. The second last row indicates the worst-case robustness of our method under all BPDA-type attacks, while the last row indicates our worst-case robustness under all attacks.

Figure 7. **Robustness w.r.t. EOT samples.** When using EOT to attack our method, we sample our adversarial transformation with different random starts $x + \delta$ to estimate the expected $\nabla f_a(g(x))$. When $\Delta = 0.1$ (blue curve), increasing the sample size allows EOT to better attack our defense, until it plateaus. But when $\Delta = 0.2$ (orange curve), EOT becomes persistently inefficient.

### 6. Conclusion
We have presented a simple defense mechanism against adversarial attacks. Our method takes advantage of the numerical recipe that searches for adversarial examples, and turns this harmful process into a useful input transformation for better robustness of a network model. On CIFAR-10 and Tiny ImageNet datasets, it demonstrates state-of-the-art worst-case robustness under a wide range of attacks. We hope our work can offer other researchers a new perspective to study the adversarial defense mechanisms. In the future, we would like to better understand the theoretical properties of our adversarial transformation and their connections to stronger adversarial robustness.

**Acknowledgements.** This work was supported in part by the National Science Foundation (CAREER-1453101, 1910839, 1717178, 1816041), a Columbia SEAS CKGSB Fellowship, and SoftBank Group.
References


Simen Thys, Wiebe Van Ranst, and Toon Goedemé. Fooling automated surveillance cameras: adversarial patches to attack person detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, pages 0–0, 2019. 1


Cihang Xie, Jianyu Wang, Zhishuai Zhang, Zhou Ren, and Alan Yuille. Mitigating adversarial effects through randomization. In International Conference on Learning Representations, 2018. 1, 2, 3, 6


Cihang Xie, Zhishuai Zhang, Yuan Zhou, Jianyu Wang, Zhou Ren, and Alan L. Yuille. Improving transferability of adversarial examples with input diversity. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2019. 2
