# MITIGATING ADVERSARIAL EFFECTS THROUGH RANDOMIZATION

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Convolutional neural networks have demonstrated their powerful ability on various tasks in recent years. However, they are extremely vulnerable to adversarial examples. I.e., clean images, with imperceptible perturbations added, can easily cause convolutional neural networks to fail. In this paper, we propose to utilize randomization to mitigate adversarial effects. Specifically, we use two randomization operations: random resizing, which resizes the input images to a random size, and random padding, which pads zeros around the input images in a random manner. Extensive experiments demonstrate that the proposed randomization method is very effective at defending against both single-step and iterative attacks. Our method also enjoys the following advantages: 1) no additional training or fine-tuning, 2) very few additional computations, 3) compatible with other adversarial defense methods. By combining the proposed randomization method with an adversarially trained model, it get a normalized score of $0.921$ in a public challenge on defending against adversarial examples, which is far better than using adversarial training alone with a normalized score of $0.830$.

## 1 INTRODUCTION

Convolutional Neural Networks (CNNs) have been successfully applied to a wide range of vision tasks, including image classification (Krizhevsky et al., 2012; Simonyan & Zisserman, 2015; He et al., 2016a), object detection (Girshick, 2015; Ren et al., 2015), semantic segmentation (Long et al., 2015; Chen et al., 2017), etc. However, recent works show that CNNs are extremely vulnerable to small perturbations to the input image. I.e., adding visually imperceptible perturbations to the original image can result in failures for image classification (Szegedy et al., 2014; Goodfellow et al., 2015), object detection (Xie et al., 2017) and semantic segmentation (Xie et al., 2017; Fischer et al., 2017; Cisse et al., 2017). These perturbed images are called adversarial examples and figure 1 gives an example. Adversarial examples pose a great security danger to the deployment of commercial machine learning systems. Thus, making CNNs more robust to adversarial examples is a very important yet challenging problem. Recent works (Papernot et al., 2016b; Kurakin et al., 2017; Tramèr et al., 2017; Metzen et al., 2017; Feinman et al., 2017; Meng & Chen, 2017) are making progress on this line of research.

Adversarial attacks can be divided into two categories: single-step attacks, which perform only one step of gradient computation, and iterative attacks, which perform multiple steps. Intuitively, the perturbation generated by iterative methods may easily get over-fitted to the specific network parameters, and thus be less transferable. On the other hand, single-step methods may not be strong enough to fool the network. For examples, it has been demonstrated that single-step attacks, like Fast Gradient Sign Method (FGSM) (Goodfellow et al., 2015), have better transferability but weaker attack rate than iterative attacks, like DeepFool (Moosavi-Dezfooli et al., 2016).

Due to the weak generalization of iterative attacks, low-level image transformations, e.g., resizing, padding, compression, etc, may probably destroy the specific structure of adversarial perturbations, thus making it a good defense. It can even defend against white-box iterative attacks if random transformations are applied. This is because each test image goes through random transformations and the attacker does not know the specific transformation when generating adversarial noise. Recently, adversarial training (Kurakin et al., 2017; Tramèr et al., 2017) was developed to defend against single-step attacks. Thus by adding the proposed random transformations as additional layers to

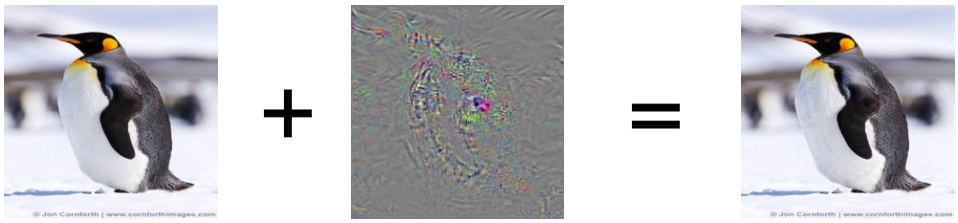


Figure 1: This is an adversarial example crafted for VGG (Simonyan & Zisserman, 2015). The left image is classified correctly as king penguin, the center image is the adversarial perturbation (magnified by 10 and enlarged by 128 for better visualization), and the right image is the adversarial example misclassfied as chihuahua.

an adversarially trained model (Tramèr et al., 2017), it is expected that the method is able to effectively defend against both single-step and iterative attacks, including both black-box and white-box settings.

Based on the above reasoning, in this paper, we propose a defense method by randomization, i.e., random resizing and random padding, to mitigate adversarial effects. To the best of our knowledge, this is the first work that demonstrates the effectiveness of randomization on mitigating adversarial effects on large-scale dataset, e.g., ImageNet (Deng et al., 2009). The proposed method enjoys the following advantages:

- Randomization makes the network much more robust to adversarial images, especially iterative attacks (both white-box and black box), but hardly hurts the performance on clean (non-adversarial) images. Experiments on section 4.2 support this argument.
- There is no additional training or fine-tuning required which is easy for implementation.
- Very few computations are required by adding the two randomization layers, thus there is nearly no run time increase.
- Randomization layers are compatible to different network structures and adversarial defense methods, which can serve as a basic network module for adversarial defense.

We conduct comprehensive experiments to test the effectiveness of our defense method, using different network structures, against different attack methods, and under different attack scenarios. The results in Section 4 demonstrate than the proposed randomization layers can significantly mitigate adversarial effects, especially for iterative attack methods. Moreover, we submitted the model, which combines the proposed randomization layers and an adversarially trained model (Tramèr et al., 2017), to a public challenge on defending against adversarial examples, it reached a normalized score of $0.921$, which is far better than just using adversarial training (Tramèr et al., 2017) alone with a normalized score of $0.830$.

## 2 Related Work

### 2.1 Generating Adversarial Examples

Generating adversarial examples has been extensively studied recently. (Szegedy et al., 2014) first showed that adversarial examples, computed by adding visually imperceptible perturbations to the original images, make CNNs predict wrong labels with high confidence. (Goodfellow et al., 2015) proposed the fast gradient sign method to generate adversarial examples based on the linear nature of CNNs, and also proposed adversarial training for defense. (Moosavi-Dezfooli et al., 2016) generated adversarial examples by assuming that the loss function can be linearized around the current data point at each iteration. (Carlini & Wagner, 2017) developed a stronger attack to find adversarial perturbations by introducing auxiliary variables which incooperate the pixel value constrain, e.g., pixel intensity must be within the range [0,255], naturally into the loss function and make the optimization process easier. (Liu et al., 2017) proposed an ensemble-based approaches to generate adversarial examples with stronger transferability. Unlike the works above, (Biggio & Laskov,

2012; Koh & Liang, 2017) showed that manipulating only a small fraction of the training data can significantly increase the number of misclassified samples at test time for learning algorithms, and such attacks are called poisoning attacks.

### 2.2 Defending Against Adversarial Examples

Opposite to generating adversarial examples, there is also progress on reducing the effects of adversarial examples. (Papernot et al., 2016b) showed networks trained using defensive distillation can effectively defend against adversarial examples. (Kurakin et al., 2017) proposed to replace the original clean images with a mixture of clean images and corresponding adversarial images in each training batch to improve the network robustness. (Tramèr et al., 2017) improved the robustness further by training the network on an ensemble of adversarial images generated from the trained model itself and from a number of other pre-trained models. (Metzen et al., 2017) trained a detector on the inner layer of the classifier to detect adversarial examples. (Feinman et al., 2017) detected adversarial examples by looking at the Bayesian uncertainty estimates of the input images in dropout neural networks and by performing density estimation in the subspace of deep features learned by the model. MagNet (Meng & Chen, 2017) detected adversarial examples with large perturbation using detector networks, and pushed adversarial examples with small perturbation towards the manifold of clean images.

## 3 Approach

### 3.1 An Overview of Generating Adversarial Examples

Before introducing the proposed adversarial defense method, we give an overview of generating adversarial examples. Let $X_n$ denote the $n$-th image in a dataset containing $N$ images, and let $y_n^{\text{true}}$ denote the corresponding ground-truth label. We use $\theta$ to denote the network parameters, and $L(X_n, y_n^{\text{true}}; \theta)$ to denote the loss. For the adversarial example generation, the goal is to maximize the loss $L(X_n + r_n, y_n^{\text{true}}; \theta)$ for each image $X_n$, under the constraint that the generated adversarial example $X_n^{\text{adv}} = X_n + r_n$ should look visually similar to the original image $X_n$, i.e., $||r_n|| \leq \epsilon$, and the corresponding predicted label $y_n^{\text{adv}} \neq y_n^{\text{true}}$.

In our experiment, we consider three different attack methods, including one single-step attack method and two iterative attack methods. We use the cleverhans library (Papernot et al., 2016a) to generate adversarial examples, where all these attacks have been implemented in TensorFlow.

- FGSM: FGSM (Goodfellow et al., 2015) is a single-step attack method. It finds the adversarial perturbation that yields the highest increase of the linear cost function under $l_\infty$-norm. The update equation is

$$x_n^{\text{adv}} = x_n + \epsilon \cdot sign(\nabla_{x_n} L(X_n, y_n^{\text{true}}; \theta)), \tag{1}$$

  where $\epsilon$ controls the magnitude of adversarial perturbation. In the experiment, we choose $\epsilon = \{2, 5, 10\}$, which corresponds to small, medium and high magnitude of adversarial perturbations, respectively.

- DeepFool: DeepFool (Moosavi-Dezfooli et al., 2016) is an iterative attack method which finds the minimal perturbation to cross the decision boundary based on the linearization of the classifier at each iteration. Any $l_p$-norm can be used with DeepFool, and we choose $l_2$-norm for the study in this paper.

- Carlini & Wagner (C&W): C&W (Carlini & Wagner, 2017) is a stronger iterative attack method proposed recently. It finds the adversarial perturbation $r_n$ by using an auxiliary variable $\omega_n$ as

$$r_n = \frac{1}{2}(tanh(\omega_n + 1)) - x_n. \tag{2}$$

  Then the loss function optimizes the auxiliary variable $\omega_n$

$$\min_{\omega_n} ||\frac{1}{2}(tanh(\omega_n) + 1) - x_n|| + c \cdot f(\frac{1}{2}(tanh(\omega_n) + 1)). \tag{3}$$

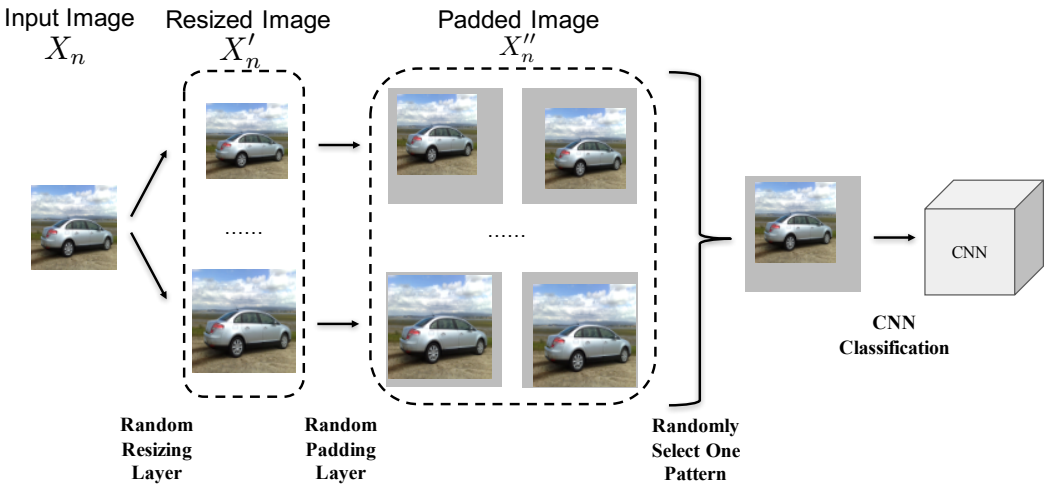


Figure 2: The pipeline of the proposed defense mechanism by randomization. The input image $X_n$ first goes through the random resizing layer with a random scale applied. Then the random padding layer pads the resized image $X'_n$ in a random manner. The resulting padded image $X''_n$ is used for classification.

The function $f(\cdot)$ is defined as

$$f(x) = \max(Z(x)_{y^{\text{true}}} - \max\{Z(x)_i : i \neq y^{\text{true}}\}, -k), \tag{4}$$

where $Z(x)_i$ is the logits output for class $i$, and $k$ controls the confidence gap between the adversarial class and true class. C&W can also work with various $l_p$-norm, and we choose $l_2$-norm in the experiments.

### 3.2 Defending Against Adversarial Examples

The goal of defense is to build a network that is robust to adversarial examples, i.e., it can classify adversarial images correctly while with little performance loss on non-adversarial images. Towards this goal, we propose a randomization based mechanism, as show in figure 2, which adds a random resizing layer and a random padding layer to the beginning of the networks.

#### 3.2.1 Randomization Layers

The first randomization layer is a random resizing layer, which resizes the original image $X_n$ with the size $W \times H \times 3$ to a new image $X'_n$ with random size $W' \times H' \times 3$. Note that, $|W' - W|$ and $|H' - H|$ should be within a reasonablely small range, otherwise the network performance on non-adversarial images would significantly drop. Taking Inception-ResNet network (Szegedy et al., 2017) as an example, the original data input size is $299 \times 299 \times 3$. Empirically we found that the network performance hardly drops if we control the height and width of the resized image $X'_n$ to be within the range $[299, 331)$.

The second randomization layer is the random padding layer, which pads zeros around the resized image in a random manner. Specifically, by padding the resized image $X'_n$ into a new image $X''_n$ with the size $W'' \times H'' \times 3$, we can choose to pad $w$ zero pixels on the left, $W'' - W' - w$ zero pixels on the right, $h$ zero pixels on the top and $H'' - H' - h$ zero pixels on the bottom. This results in a total number of $(W'' - W' + 1) \times (H'' - H' + 1)$ different possible padding patterns.

During implementation, the original image first goes through two randomization layers, and then we pass the transformed image to the original CNN for classification. The pipeline is illustrated in figure 2.

#### 3.2.2 RANDOMIZATION LAYERS + ADVERSARIAL TRAINING

Note that our randomization-based method is good at defending against iterative attacks, and adversarial training (Kurakin et al., 2017; Tramèr et al., 2017) can effectively increase the robustness of neural networks to single-step attacks. Thus, to make the best of both worlds, we combine the proposed randomization layers and an adversarially trained model (Tramèr et al., 2017) together to defend against both single-step and iterative attacks.

## 4 EXPERIMENTS

### 4.1 EXPERIMENT SETUP

**Dataset.** It is less meaningful to attack the images that are already classified wrongly. Therefore, we randomly choose 5000 images from the ImageNet validation set that are classified correctly by all the models to form our test dataset. All these images are of the size $299 \times 299 \times 3$.‘

**Network Models.** We test with four networks, including *Inception-v3* (Szegedy et al., 2016), *ResNet-v2* (He et al., 2016b) of 101 layers, *Inception-ResNet-v2* (Szegedy et al., 2017), and *ens-adv-Inception-ResNet-v2* which applies the ensemble adversarial training (Tramèr et al., 2017) on *Inception-ResNet-v2*. All these models are publicly available[1][2].

**Details of Defense Method.** The defense models consist of the original models and two additional randomization layers. For the random resizing layer, it changes the input shape from $299 \times 299 \times 3$ to $rnd \times rnd \times 3$, where $rnd$ is a integer randomly sampled from the range $[299, 331)$. For the random padding layer, it pads the resized image to the shape of $331 \times 331 \times 3$ in a random manner. By applying these two randomization layers, we can create $\sum_{rnd=299}^{330} rnd \cdot (331 - rnd + 1)^2 = 12528$ different patterns for a single image. Since there exist small variance on model performance w.r.t. different random patterns, we run the defense model three times independently and report the average accuracy.

**Attack Scenarios.** The attacks on target models are all under the white-box setting, where the attackers know the exact structure and parameters. Note that, the target models and the defense models are exactly the same except randomization layers. Due to the large amount of possible patterns created by randomization layers, our defense model cannot be attacked directly in practice (otherwise the loss will be hard to optimize and extremely time-consuming as discussed in section 3.2.2). Thus we consider three different attack scenarios described below.

- vanilla attack: the attackers do not know the existence of the randomization layers, and the target model is the original model without randomization layers.
- single-pattern attack: the attackers know the existence of the randomization layers and mimic the structure of randomization layers using only one fixed pattern.
- ensemble-pattern attack: the attackers know the existence of the randomization layers and mimic the structure of randomization layers with an ensemble of patterns.

### 4.2 CLEAN IMAGES

Table 1 shows the top-1 accuracy of models with and without randomization layers on the test dataset. We can see that randomization layers introduce negligible performance degradation on clean images. Moreover, we can observe that: (1) models with more advanced architectures tend to have less performance degradation, e.g., *Inception-ResNet-v2* only has $0.7\%$ degradation while *Inception-v3* has $2.7\%$ degradation; (2) ensemble adversarial training brings nearly no performance degradation to the models, e.g., *Inception-ResNet-v2* and *ens-adv-Inception-ResNet-v2* have nearly the same performance degradation.

[1] https://github.com/tensorflow/models/tree/master/research/slim

[2] http://download.tensorflow.org/models/ens_adv_inception_resnet_v2_2017_08_18.tar.gz

Table 1: Top-1 classification accuracy of different models on the test dataset of 5000 images. We see that adding random resizing and random padding cause very little accuracy drop on clean (non-adversarial) images.

| Models | Inception-v3 | ResNet-v2-101 | Inception-ResNet-v2 | ens-adv-Inception-ResNet-v2 |
|---|---|---|---|---|
| w/o randomization | 100% | 100% | 100% | 100% |
| w randomization | 97.3% | 98.3% | 99.3% | 99.2% |

Table 2: Top-1 classification accuracy under vanilla attack scenario. We see that randomization layers effectively mitigate adversarial effects for all attacks and all network models. Particularly, combining randomization with ensemble adversarial training (*ens-adv-Inception-ResNet-v2*) performs very well on all attacks.

| Models | Inception-v3 | | ResNet-v2-101 | | Inception-ResNet-v2 | | ens-adv-Inception-ResNet-v2 | |
|---|---|---|---|---|---|---|---|---|
| | target model | defense model | target model | defense model | target model | defense model | target model | defense model |
| FGSM-2 | 33.2% | 65.1% | 26.3% | 71.8% | 65.3% | 81.0% | 84.4% | 95.7% |
| FGSM-5 | 31.1% | 54.5% | 20.4% | 54.3% | 61.7% | 74.1% | 87.4% | 94.5% |
| FGSM-10 | 33.0% | 52.4% | 20.4% | 46.1% | 61.2% | 71.3% | 90.2% | 94.3% |
| DeepFool | 0% | 98.3% | 0% | 97.7% | 0% | 98.2% | 0.2% | 99.1% |
| C&W | 0% | 96.9% | 0% | 97.1% | 0.3% | 97.7% | 0.9% | 98.8% |

### 4.3 Vanilla Attack

For the vanilla attack, since the attackers are not aware of the existence of randomization layers, the attacks on the defense models mostly rely on the transferability of adversarial examples to different resizing and padding. From the top-1 accuracy presented in table 2, we observe that randomization can mitigate the adversarial effects of both single-step and iterative attacks significantly. As for single-step attacks FGSM-$\epsilon$, larger $\epsilon$ indicates stronger transferability, thus making it harder to defend. However, we still get satisfactory accuracy on single-step attacks (even with large $\epsilon$) using *ens-adv-Inception-ResNet-v2* (94.3% top-1 accuracy). As for iterative attacks, they always reach a very high attack rate on target model, but have almost no impact on models after randomization layers are applied. This is because iterative attack methods are over-fitted to the target models thus have weak transferability.

### 4.4 Single-Pattern Attack

For the single-pattern attack, the attackers are aware of the existence of randomization layers and also the range of parameters for resizing and padding (i.e., starting from $299 \times 299$ to $331 \times 331$), but they do not know the specific randomization patterns. In order to generate adversarial examples, the attackers choose only **one** specific pattern to compute the gradient. In this experiment, the specific pattern that we use is to keep the original input $X_n$ at the center of the padded image $X_n''$, i.e., no resizing is applied, and 16 zeros pixels are padded on the left, right, top and bottom on the input images, respectively. Table 3 shows the top-1 accuracy of both target models and defense models, and similar results to vanilla attack are observed: (1) for single-step attacks, randomization layers are less effective on mitigating adversarial effects for a larger $\epsilon$, while the adversarially trained models are able to defend against such attacks; (2) for iterative attacks, they reach high attack rates on target models, while have nearly no impact on defense models.

### 4.5 Ensemble-Pattern Attack

For the ensemble-pattern attack, similar to single-pattern attack, the attackers are aware of the randomization layers and the parameters for resizing and padding (i.e., starting from $299 \times 299$ to $331 \times 331$), but they do not know the specific patterns. The attackers use a stronger attack method:

Table 3: Top-1 classification accuracy under single-pattern attack scenario. We see that randomization layers effectively mitigate adversarial effects for all attacks and all network models. Particularly, combining randomization with ensemble adversarial training (*ens-adv-Inception-ResNet-v2*) performs very well on all attacks.

| Models | Inception-v3 | | ResNet-v2-101 | | Inception-ResNet-v2 | | ens-adv-Inception-ResNet-v2 | |
|---|---|---|---|---|---|---|---|---|
| | target model | defense model | target model | defense model | target model | defense model | target model | defense model |
| FGSM-2 | 35.1% | 63.8% | 29.5% | 70.1% | 71.6% | 83.4% | 86.3% | 96.4% |
| FGSM-5 | 32.4% | 53.9% | 23.2% | 52.3% | 68.3% | 78.2% | 88.4% | 95.4% |
| FGSM-10 | 34.7% | 51.8% | 22.4% | 43.8% | 66.8% | 75.6% | 90.7% | 95.2% |
| DeepFool | 1.1% | 98.2% | 1.7% | 97.8% | 0.6% | 98.4% | 1.0% | 99.2% |
| C&W | 1.1% | 97.4% | 1.7% | 97.0% | 0.8% | 97.9% | 1.6% | 99.1% |

choose an ensemble of patterns to generate adversarial examples. The goal of the ensemble-pattern attack is to let all chosen patterns fail on classification. In this experiment, the specific patterns that we choose are: (1) first resize the input image to five different scales $\{299, 307, 315, 323, 331\}$; (2) then pad each resized image to five different patterns, where the resized image is at the top left, top right, bottom left, bottom right, and center of the padded image, respectively. Since there is only one padding pattern for the resized image with size 331, we can obtain $4 * 5 + 1 = 21$ patterns in total. Due to the large computations introduced by ensemble-pattern attack, we randomly choose $500$ images out of the entire test dataset for the experiment. The top-1 accuracy for the target model here is calculated by summing the number of correctly classified patterns for each image over the entire pattern number of all images. For the results presented in table 4, we can see that ensemble-pattern attack constructs much stronger adversarial examples. For single-step attacks, the generated adversarial examples can let the performance of the adversarially trained model drop around $8\%$ compared to the performance under vanilla attack and single-pattern attack scenarios, and drop much more on other defense models. For iterative attacks, we observe that the adversarial examples generated by C&W are stronger than those generated by DeepFool, e.g., *Inception-v3* has an accuracy of $81.3\%$ on DeepFool, while only has an accuracy of $62.9\%$ on C&W. We argue that this is due to the more advanced loss function (i.e., introduction of auxiliary variable for pixel value control) utilized by C&W than DeepFool. Additionally, the accuracy of dense model on C&W can be improved by utilizing more advanced architecture (e.g., *ResNet-v2-101* has higher accuracy than *Inception-v3*) and applying ensemble adversarial training (e.g., *ens-adv-Inception-ResNet-v2* has higher accuracy than *Inception-ResNet-v2*). For the best defense model that we have, *ens-adv-Inception-ResNet-v2* + randomization layers reaches the top-1 accuracy of $93.5\%$ on DeepFool and $86.1\%$ on C&W, respectively.

Note that ensemble with more patterns always brings a higher attack rate on defense models, but the computations introduced and GPU memory required grow nearly linear with respect to the number of patterns. This makes the ensemble attack with large number of patterns less practical. Therefore, in our experiment, we consider 21 patterns which is a good balance between computation and performance.

### 4.6 Diagnostic Experiment

Due to the large amount of possible patterns introduced by randomization layers, it is hard to analyze the effectiveness of random resizing layer and random padding layer precisely. In this section, we limit the freedom of randomization to be a small number (i.e., 4 in random padding and 1 in random resizing) and analyze the effectiveness of these two layers separately. The same $500$ images as we used in section 4.5 are used in this experiment. In addition, the input images for target models and defense models are of the size $330 \times 330 \times 3$.

#### 4.6.1 One Pixel Padding

For the random padding, there are only $4$ padding patterns when padding the input images from $330 \times 330 \times 3$ to $331 \times 331 \times 3$. In order to construct a stronger attack, we follow the experiment

Table 4: Top-1 classification accuracy under ensemble-pattern attack scenario. Similar to vanilla attack and single-pattern attack, we see that randomization layers increase the accuracy under all attacks and network models. This clearly demonstrates the effectiveness of randomization on defending against adversarial examples, even under this very strong attack scenario.

| Models | Inception-v3 | | ResNet-v2-101 | | Inception-ResNet-v2 | | ens-adv-Inception-ResNet-v2 | |
|---|---|---|---|---|---|---|---|---|
| | target model | defense model | target model | defense model | target model | defense model | target model | defense model |
| FGSM-2 | 37.3% | 41.2% | 39.2% | 44.9% | 71.5% | 74.3% | 86.2% | 88.9% |
| FGSM-5 | 31.7% | 34.0% | 24.6% | 29.7% | 65.2% | 67.3% | 85.8% | 87.5% |
| FGSM-10 | 30.4% | 32.8% | 18.6% | 21.7% | 62.9% | 64.5% | 86.6% | 87.9% |
| DeepFool | 0.6% | 81.3% | 0.9% | 80.5% | 0.9% | 69.4% | 1.6% | 93.5% |
| C&W | 0.6% | 62.9% | 1.0% | 74.3% | 1.6% | 68.3% | 5.8% | 86.1% |

Table 5: Top-1 classification accuracy under one pixel padding scenario. This table shows that creating different padding patterns (even 1-pixel padding) can effectively mitigate adversarial effects.

| Models | Inception-v3 | | ResNet-v2-101 | | Inception-ResNet-v2 | | ens-adv-Inception-ResNet-v2 | |
|---|---|---|---|---|---|---|---|---|
| | target model | defense model | target model | defense model | target model | defense model | target model | defense model |
| FGSM-2 | 36.4% | 39.6% | 29.8% | 34.4% | 71.3% | 74.0% | 88.2% | 94.8% |
| FGSM-5 | 33.5% | 36.2% | 22.2% | 26.2% | 68.4% | 71.0% | 92.1% | 94.4% |
| FGSM-10 | 34.5% | 38.8% | 21.3% | 23.6% | 67.4% | 70.4% | 93.7% | 94.0% |
| DeepFool | 0.9% | 97.2% | 0.9% | 95.2% | 0.9% | 87.6% | 1.5% | 99.2% |
| C&W | 0.8% | 70.2% | 0.9% | 76.8% | 1.0% | 79.4% | 2.4% | 98.2% |

setup in section 4.5 where 3 chosen patterns are ensembled. Specifically, the target model takes an ensemble pattern where the original images are at the top left, top right and bottom left (3 patterns) of the padded images, and the defense model takes the last pattern where the original images are at the bottom right of the padded images. Note that, since there is no randomization in the defense model, we only run the defense model once. Table 5 summaries the results, and we can see that: (1) adversarial examples generated by single-step attacks have strong transferability, but still cannot attack the adversarially trained model successfully (e.g., *ens-adv-Inception-ResNet-v2*); (2) adversarial examples generated by iterative attacks are much less transferable between different padding patterns even when only 4 different patterns exist. The results here support the effectiveness of mitigating adversarial effects by creating different padding patterns.

#### 4.6.2 One-Pixel Resizing

For the random resizing, there is only 1 pattern exist when the input images are resized from $330 \times 330 \times 3$ to $331 \times 331 \times 3$. The results in table 6 indicates that resizing the images at only 1 pixel range, can efficiently destroy the transferability of adversarial examples on both single-step and iterative attacks.

## 5 Public Challenge on Defending Against Adversarial Examples

We submitted our model to a public challenge on defending against adversarial examples for performance evaluation. The test dataset contains 1000 images which are all of the size $299 \times 299 \times 3$, and their corresponding labels are the same as the ImageNet 1000-class labels. Each defense method are run on all 1000 adversarial images generated against all adversarial attacks. For each correctly classified image, the defense method gets one point. The normalized score for each defense method

Table 6: Top-1 classification accuracy under one pixel resizing scenario. This table shows that resizing image to a different scale (even 1-pixel scale) can effectively mitigate adversarial effects.

| Models | Inception-v3 | | ResNet-v2-101 | | Inception-ResNet-v2 | | ens-adv-Inception-ResNet-v2 | |
|---|---|---|---|---|---|---|---|---|
| | target model | defense model | target model | defense model | target model | defense model | target model | defense model |
| FGSM-2 | 30.8% | 56.2% | 31.6% | 44.6% | 66.2% | 75.0% | 87.6% | 97.2% |
| FGSM-5 | 31.2% | 48.8% | 25.6% | 35.8% | 61.4% | 70.2% | 91.2% | 96.6% |
| FGSM-10 | 36.4% | 51.0% | 23.8% | 32.6% | 62.8% | 68.2% | 94.8% | 95.2% |
| DeepFool | 2.6% | 99.4% | 1.0% | 98.6% | 1.2% | 97.4% | 1.2% | 99.4% |
| C&W | 2.6% | 97.8% | 1.0% | 94.8% | 2.0% | 94.8% | 1.8% | 99.6% |

is computed using the following formula:

$$\text{score} = \frac{1}{M} \sum_{\text{attack} \in A} \sum_{n=1}^{1000} \left[ \text{defense}(\text{attack}(X_n)) = y_n^{\text{true}} \right], \tag{5}$$

where $A$ is the set of all attacks, $M$ is the total number of generated adversarial examples by all attacks, and the function $[\cdot]$ is the indicator function which equals to 1 when the prediction is true.

### 5.1 Challenge Results

The best defense model in our experiments, i.e., randomization layers + *ens-adv-Inception-Resnet-v2*, was submitted to the challenge. To increase the classification accuracy, we (1) changed the resizing range from $[299, 331)$ to $[310, 331)$; (2) averaged the prediction results over 30 randomization patterns for each image; (3) flipped the input image with probability $0.5$ for each randomization pattern.

By evaluating our model against $146$ different attacks, it reaches a normalized score of $0.921$, which is far better than using ensemble adversarial training (Tramèr et al., 2017) alone with a normalized score of $0.830$. This result indicates the additional robustness brought by the proposed randomization mechanism.

## 6 Conclusion

In this paper, we propose a randomization based mechanism to mitigate adversarial effects. We conduct comprehensive experiments to validate the effectiveness of our defense method, using different network structures, against different attack methods, and under different attack scenarios. The experimental results show that adversarial examples rarely transfer between different randomization patterns, especially for iterative attacks. In addition, the proposed randomization layers are compatible to different network structures and adversarial defense methods, which can serve as a basic network module for adversarial defense. By adding the proposed randomization layers to an adversarially trained model (Tramèr et al., 2017), it got a normalized score of $0.921$ in a public challenge on defending against adversarial examples, which is far better than using adversarial training alone with a normalized score of $0.830$.

## References

Battista Biggio and Pavel Laskov. Poisoning attacks against support vector machines. In *International Conference on Machine Learning*, 2012.

Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *IEEE Symposium on Security and Privacy*. IEEE, 2017.

Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

Moustapha Cisse, Yossi Adi, Natalia Neverova, and Joseph Keshet. Houdini: Fooling deep structured prediction models. *arXiv preprint arXiv:1707.05373*, 2017.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition*. IEEE, 2009.

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

Volker Fischer, Mummadi Chaithanya Kumar, Jan Hendrik Metzen, and Thomas Brox. Adversarial examples for semantic image segmentation. *arXiv preprint arXiv:1703.01101*, 2017.

Ross Girshick. Fast r-cnn. In *International Conference on Computer Vision*. IEEE, 2015.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*, 2015.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Computer Vision and Pattern Recognition*. IEEE, 2016a.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European Conference on Computer Vision*. Springer, 2016b.

Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. *arXiv preprint arXiv:1703.04730*, 2017.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, 2012.

Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial machine learning at scale. In *International Conference on Learning Representations*, 2017.

Yanpei Liu, Xinyun Chen, Chang Liu, and Dawn Song. Delving into transferable adversarial examples and black-box attacks. In *International Conference on Learning Representations*, 2017.

Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Computer Vision and Pattern Recognition*. IEEE, 2015.

Dongyu Meng and Hao Chen. Magnet: a two-pronged defense against adversarial examples. *arXiv preprint arXiv:1705.09064*, 2017.

Jan Hendrik Metzen, Tim Genewein, Volker Fischer, and Bastian Bischoff. On detecting adversarial perturbations. In *International Conference on Learning Representations*, 2017.

Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: a simple and accurate method to fool deep neural networks. In *Computer Vision and Pattern Recognition*. IEEE, 2016.

Nicolas Papernot, Ian Goodfellow, Ryan Sheatsley, Reuben Feinman, and Patrick McDaniel. cleverhans v1.0.0: an adversarial machine learning library. *arXiv preprint arXiv:1610.00768*, 2016a.

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

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems*, 2015.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *International Conference on Learning Representations*, 2014.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Computer Vision and Pattern Recognition*. IEEE, 2016.

Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *AAAI*, 2017.

Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Dan Boneh, and Patrick McDaniel. Ensemble adversarial training: Attacks and defenses. *arXiv preprint arXiv:1705.07204*, 2017.

Cihang Xie, Jianyu Wang, Zhishuai Zhang, Yuyin Zhou, Lingxi Xie, and Alan Yuille. Adversarial Examples for Semantic Segmentation and Object Detection. In *International Conference on Computer Vision*. IEEE, 2017.