PIXEL REDRAWN FOR A ROBUST ADVERSARIAL DEFENSE

Anonymous authors
Paper under double-blind review

ABSTRACT

Recently, an adversarial example becomes a serious problem to be aware of because it can fool trained neural networks easily. To prevent the issue, many researchers have proposed several defense techniques such as adversarial training, input transformation, stochastic activation pruning, etc. In this paper, we propose a novel defense technique, Pixel Redrawn (PR) method, which redraws every pixel of training images to convert them into distorted images. The motivation for our PR method is from the observation that the adversarial attacks have redrawn some pixels of the original image with the known parameters of the trained neural network. Mimicking these attacks, our PR method redraws the image without any knowledge of the trained neural network. This method can be similar to the adversarial training method but our PR method can be used to prevent future attacks. Experimental results on several benchmark datasets indicate our PR method not only relieves the over-fitting issue when we train neural networks with a large number of epochs, but it also boosts the robustness of the neural network.

1 INTRODUCTION

Deep neural networks have exhibited very accurate results, but, are also vulnerable to adversarial examples (Szegedy et al., 2013). Adversarial example is a generated output that induces a trained machine to misclassify a test datum with a high probability. In other words, the adversarial example can fool the trained machine to fail its task (Dhillon et al., 2018b; Goodfellow et al., 2014; Szegedy et al., 2013). We call the method that generates adversarial examples as an attack technique. For instance, in image classification, an attack technique is creating an adversarial image by perturbing some pixels of the original image in which the adversarial image is remained to be perceptible by the human. Obviously, the lesser the pixels to be perturbed, the better the attack technique it is. Defense techniques to prevent those attacks usually need to produce a robust neural network model which can detect adversarial examples or label the adversarial examples correctly. There are many researchers who have studied both attacking and defense techniques, and it is generally believed that the defense techniques are more challenging than the attack techniques.

In this paper, we focus on the defense technique that generates a robust neural network. For this purpose, we have to understand the fundamental idea of generating the adversarial example. In image classification, usually, an adversarial image is generated by perturbing its pixels' value (for example, L_2 or L_∞ attack). If we regard the perturbed pixels to be formed by adding some noises, then the technique of creating the noises can be recognized as a dropout (Srivastava et al., 2014) (if the perturbed pixel value is zero) or other regularization techniques (Adeli & Wu, 1998; Krogh & Hertz, 1992; Nowlan & Hinton, 1992). The neural network with the regularization method usually shows higher performance than the one without the regularization method (normal testing). From another viewpoint, we can regard adversarial examples as a subset from the training dataset universe. Due to the difficulty of collecting all possible examples in the world, we usually just use the "clean" examples as the representative of our training dataset.

In order to solve the problem, we propose Pixel Redrawn (PR) method, which is a pre-processing method that redraws a pixel value of the original image into a different pixel value. The PR method is motivated by two observations. The first observation is based on the lesson provided by Carlini and Wagner in (Carlini & Wagner, 2017), which the randomization can increase the required distortion. The paper has mentioned that the most effective defense technique so far is the dropout

randomization which has increased nearly five times more the difficulty by generating the adversarial examples on CIFAR dataset. We agree that the randomization method makes it difficult to compute the derivation during the back-propagation and it could replace the perturbed pixel with another value by some random chances.

The second observation is that humans usually can recognize images regardless of the colors in which the image has been drawn (e.g. in various colors or in gray level). This indicates that we usually (human) may not need to have a colorful image just for image classification. We discuss this in more detail with our proposed method in Section 2.

In this study, we focus on the following:

- We analyze the influence of our PR method toward a normal deep neural network without any adversarial machine learning.
- We evaluate the effectiveness of our PR method applied in the deep neural network against the adversarial example.
- And, we compare our PR method with a random noise injection.

Our research contributions can be summarized as follows:

- Our PR method effectively generates adversarial training images that are not covered in the original dataset.
- Hence, our PR method increases the robustness in neural network training.

To promote reproducible research, we release the implementation of our defense. [1]

2 PIXEL REDRAWN METHOD

As aforementioned, PR is a pre-processing method which reproduces a new image by redrawing the pixels of the original image. In neural network architecture, we can denote PR as an activation function for the input layer like the other activation functions (i.e. Sigmoid activation function, Softmax activation function, etc.), because PR transforms an pixel value of an image into a random value within a certain interval. In other words, this resulting value of the pixel is within an accepted range determined from the original value of the pixel, with some exceptions which will be discussed in Section 2.2. For example, in a graylevel image (e.g. MNIST dataset (LeCun et al., 1998)) as shown in Fig. 1, the normalized value of the black color is 0 and that of the white color is 1. PR method assigns a random value between 0.0 and 0.5 for the black colored pixel, and a random value between 0.5 and 1.0 for the white colored pixel. Note that the range for producing a random value is specified by a user.

From Fig. 1, after applying the PR method (right), we still able to notice the image is a "C" shape although it is not as clear as before (left). We also depict the examples from a real dataset (i.e. MNIST dataset) in Fig. 2. The intuition of redrawing each pixel with a certain value in an accepted interval is that, for a certain training image, the training dataset does not include all possible combination of the training image with human-acceptably varying pixel values. With the PR method, we try to include those missing data in the training phase.

For a color image (e.g. CIFAR-10 dataset (Krizhevsky et al., 2010)), each pixel of the image is represented by red, green and blue channels. If a pixel is a red color, the value in the channel format is (1.0, 0.0, 0.0). Note that the value of each channel is between 0.0 and 1.0. For a blue color, the value in the channel is (0.0, 1.0, 0.0). And for a green color, the value in the channel is (0.0, 0.0, 1.0). Consider the PR method, with the same accepted range as in the previous example, applied to the color image with three channels. A red color pixel might be assigned to a random value between 0.5 and 1.0 in the red channel and a value between 0.0 and 0.5 for the green and blue channels, respectively (0.5-1.0, 0.0-0.5, 0.0-0.5). This shows that our PR method assigns the random value for each pixel more effectively than naïve approaches that assign the random values generated from full range $[0.0..1.0]$ to the pixel. Another advantage of using the PR method is that we can generate

[1] We modify and extend the scripts from CleverHans. Our scripts are available at https://github.com/Jiacang/pixel-redrawn

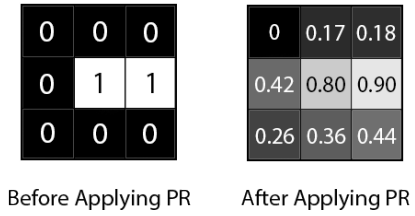

Figure 1: Pixel Redrawn Concept in a grayscale image.

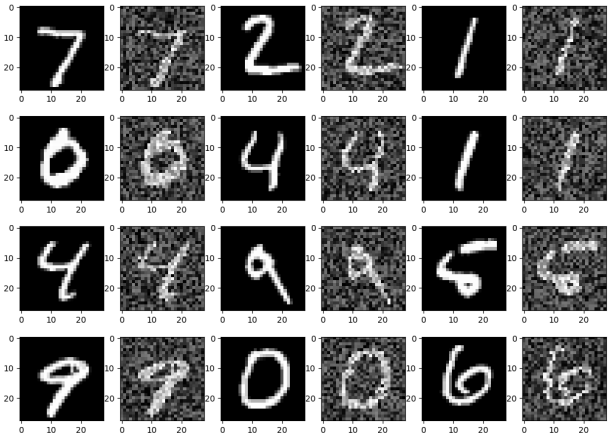

Figure 2: Example MNIST images before and after applying the Pixel Redrawn method. From a pair of an image, the left-hand side is the original image and the right-hand side is the image after applying the PR method.

more accurate neural network models (discussed in more detail in Section 2.2). In the following subsections, we explain Pixel Redrawn model (PR model) for effectively generating new pixel redrawn images by our PR method.

2.1 Generating Pixel Redrawn Model

Given a pixel value from the original training image, PR model predicts the output pixel value, perturbed within an accepted range, for the corresponding pixel location of the new image. Since we redraw each pixel of the image (either grayscale or color) independently which the pixel is the only input for the PR model, and we need the PR model to produce several outputs, then single-

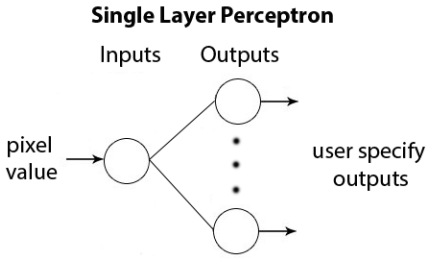

Figure 3: Pixel Redrawn Architecture.

layer Perceptron is qualified in this case as illustrated in Fig. 3. Firstly, we train the PR model by some random examples with the corresponding labels initialized with the accepted range values. After the PR model is converged, we predict each pixel of the original image with the newly trained PR model. Lastly, we assign a predicted output value of the winning node at the last layer in the trained PR model to the corresponding pixel location of the new image. Note that the output value is within the accepted range which is initialized by the user. The final output should look similar as the right-hand side image in Fig. 1 (or the right-hand side image of a pair image in Fig. 2).

2.2 Extension of the Pixel Redrawn Model

One direct extension is *k* PR models. We train several PR models and predict each pixel of the original image with one of the trained PR models. In other words, we randomly select one PR model out of *k* PR models to predict a pixel of the original image. Then, we repeat the same steps until all pixels are redrawn.

Another extension is partially converged PR models (less accurate) to redraw a pixel to a totally different color value. The partially converged PR model is generated when we prematurely stop the training phase on purpose after a few epochs so that the model is not fully trained. For example, if a black and white PR model has 70% accuracy to predict a pixel either black or white color, then a black color pixel can have 30% chance to be converted into a white color. Note that this may or may not influence the content of the image depending on the number of the pixels that are different from the original input. In a color image, however, we may able to change an eye color from a black color to a blue color, for example, by using the partially converged (color) PR models. The motivation of using the partially converged PR models is because the adversarial example changes a pixel value into a different value which the changed value is far from the original value (L_∞ attack).

In summary, using several PR models (including some partially converged PR models) increase the robustness of the neural network.

$$x_i = \begin{cases} x_i \pm \beta_f, & \text{for the fully converged PR model} \\ x_i \pm \beta_p, & \text{for the partially converged PR model} \end{cases} \tag{1}$$

where β_f is a value that is in the accepted range and β_p is a value that is outside of the accepted range.

2.3 Pseudo-code of the Pixel Redrawn Method

In order to have a clear description of our proposed method, we provide the pseudo-code of the PR method in Algorithm 1. First, we initialize the settings of the PR models, which include the range of each color value, the number of the input and output, the number of the epoch/iteration, and some random weights. Then, we train the PR model by generating some random inputs with a corresponding label which the label is based on the range of each color value. After we train the PR model, we use the model to redraw all pixels of the original image. Next, we use the newly generated image as the input of the neural network during the training phase and testing phase. In the end, the trained network show the output label of the test data.

Algorithm 1 The pseudo-code of the Pixel Redrawn Method

```
Input: Image dataset.
Initialize the range of each color value, the # of colors as the # of output nodes of PR model, random weights
Preprocessing phase:
Train a PR model based on the type of the image dataset (grayscale or color image)
Predict each pixel of the original image (training and test data) with the trained PR model
Assign a new pixel value based on the output of the trained PR model
Training phase:
Train a neural network model with the newly generated training data (normal training)
Testing phase:
Classify the newly generated test data with the trained neural network model (normal testing)
Output: The class label of the test image
```

Table 1: Configuration of the CNN model for MNIST, Fashion MNIST, and CIFAR-10 (color & grayscale) datasets.

NO.	LAYER NAME
1	CONVOLUTIONAL LAYER
2	RELU ACTIVATION
3	CONVOLUTIONAL LAYER
4	RELU ACTIVATION
5	CONVOLUTIONAL LAYER
6	RELU ACTIVATION
7	FULLY CONNECTED LAYER
8	SOFTMAX ACTIVATION

3 EXPERIMENTAL SETTINGS

3.1 DATASET

For experimental analysis of our propose methods, we use three public benchmark datasets in this study. The datasets include MNIST, Fashion MNIST (Xiao et al., 2017) and CIFAR-10. For the CIFAR-10 dataset, we generate another grayscale CIFAR-10 for the purpose of analyzing the influence of our proposed method on the color image. MNIST and Fashion MNIST have 60,000 training images and 10,000 test images associated with a label from ten classes. The size of each image is 28×28 grayscale. CIFAR-10, however, has 50,000 training images and 10,000 test images with ten classes. Each image is 32×32 color image.

We use a basic convolutional neural network (CNN) (Krizhevsky et al., 2012) as the neural network architecture for all datasets. The neural network architecture is shown in Table 1. We use three convolutional layers and one fully connected layer. The convolutional layers are followed by the Rectifier Linear Unit (ReLU) activation function. The fully connected layer is followed by the Softmax activation function.

3.2 ATTACK TECHNIQUE

There are two kinds of attacks, which are white-box attack and black-box attack. In the white-box attack, attackers know all parameters of the attacked model, whereas in the black-box attack, attackers have no knowledge about the parameters of the attacked model. In this experiment, we run the white-box attack because it is the hardest to defend. The attack can be targeted attack or untargeted attack. In the targeted attack, the attacker tries to deceive the trained network to misclassify the datum as the targeted label. The untargeted attack, on the other hand, is fooling the trained model to misclassify the datum as any label except the true label. We choose the untargeted attack in the experiment because it is easier for the attacker than the targeted attack.

We use several state-of-the-art attack techniques for the evaluation of our proposed method. The attack techniques include Fast Gradient Sign Method (FGSM) (Goodfellow et al., 2014), basic iterative method (BIM) (Kurakin et al., 2016), momentum iterative method (MIM) (Dong et al., 2018) and Carlini & Wagner's (CW) (Carlini & Wagner, 2016) attack.

FGSM It is a fast and simple attack technique to generate the adversarial example which has proposed by Goodfellow et al. (2014). We set this technique as the baseline of attack techniques. The equation of FGSM is computed by

$$X^{adv} = X + \epsilon \; sign(\nabla_X L(X, y)) \tag{2}$$

where ϵ is the maximum perturbation allowed for each pixel and $L(X, y)$ is a loss function.

BIM It is an extension of FGSM by applying multiple iterations with small step size in order to obtain the least perturbations of the image. This technique is proposed by Kurakin et al. (2016). The computation of the technique is shown in Eq. 3

$$X_0^{adv} = X, \; X_{N+1}^{adv} = Clip_{X,\epsilon}\{X_N^{adv} + \alpha \; sign(\nabla_X L(X_N^{adv}, y))\} \tag{3}$$

where $Clip_{X,\epsilon}$ is a function to clip the output image to be within the ϵ-ball of X.

MIM It is more advanced than BIM with momentum algorithm. This technique is proposed by Dong et al. (2018). The computation of the technique is shown in Eq. 4

$$X_{t+1}^{adv} = X_t^{adv} + \alpha \cdot \frac{g_{t+1}}{||g_{t+1}||_2} \tag{4}$$

where g_{t+1} is shown in Eq. 5.

$$g_{t+1} = \mu \cdot g_t + \frac{J(x_t^{adv}, y)}{||\nabla_x J(x_t^{adv}, y)||_1} \tag{5}$$

CW It is an efficient attack technique in finding the adversarial example with the smallest perturbations. The equation is shown as follow.

$$minimize \parallel \frac{1}{2}(tanh(w) + 1) - x \parallel_2^2 + c \cdot f(\frac{1}{2}(tanh(w) + 1)) \tag{6}$$

During the experiments, we set $\epsilon = 0.3$ for the MNIST dataset and $\epsilon = \frac{8}{256}$ for the Fashion MNIST and CIFAR-10 datasets when we apply with the FGSM, BIM, and MIM attacks.

3.3 Defense Technique

We categorize the defense techniques into two different categories. They are white-box defense and black-box defense. In the white-box defense, defenders use only the known attack techniques to generate adversarial examples to be included in a training dataset (e.g. adversarial training). In the black-box defense, on the other hand, defenders have no knowledge about the attack techniques and they try to generate a robust neural network model (e.g. stochastic activation pruning (SAP) (Dhillon et al., 2018a), input transformations (Guo et al., 2018), etc.). The white-box defense is usually performed better than the black-box defense because it can use the state-of-the-art attack technique to create adversarial examples for the training purpose and then the trained model can prevent the attacks with the similar level (or weaker levels) of the known attack technique. However, unlike the black-box defense, the white-box defense might not be able to defend strong attacks devised in the future. Due to the disability of the white-box defense in defending future attacks, the black-box defense is more reliable to be studied. Therefore, most state-of-the-art defense techniques are the black-box defense.

3.4 Case Study with Different Attack Scenarios

In this paper, we evaluate our method in several cases with different attack scenarios. The case studies are included as follows:

- **Normal:** No attack technique is used. Test the neural networks with a legitimate datum.
- **Case A:** The attackers have no knowledge of the PR method but they knows the parameters of the trained neural networks. The attackers create an adversarial image from an image with the parameters of the trained neural networks. The defense mechanism receives the adversarial image as the input for the PR model and then generate a new image so-called "PR image" for the input of the trained neural networks.

  Clean image → Adversarial image → PR image.
- **Case B:** No PR method is used during the testing phase. The attackers have no knowledge of the PR method but they knows the parameters of the trained neural networks. The attackers generate an adversarial image from an image with the parameters of the trained neural networks. The trained neural networks use the adversarial image as the input without being pre-processed by the PR method.

  Clean image → Adversarial image.
- **Case C:** The attackers know both the PR method and the trained neural networks. The attackers produce an PR image from an image with the PR method. Then they create the adversarial image from the PR image with the parameters of the trained neural network.

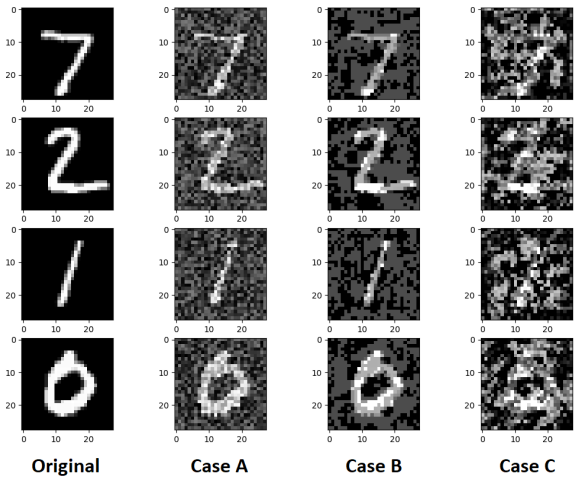

Figure 4: The adversarial images generated from the MNIST dataset with the FGSM technique ($\epsilon = 0.3$) in four cases.

The defense mechanism receives the adversarial image as the input for the PR model and the newly created image is used as the input of the trained neural networks.

Clean image $\rightarrow$ PR image $\rightarrow$ Adversarial image.

We illustrate some adversarial images generated from the MNIST dataset for each case in Fig. 4. Note that the adversarial example generated from the case C can increase the L_0, L_2 and L_∞ distances because the attacker has used a distorted image instead of a clean image. In other words, the perturbed pixels from the adversarial image can be perceptible by the human.

4 Experimental Results

In this study, we perform several experiments based on the following questions:

1. What is the influence of applying the PR method in different phases with a legitimate test datum?
2. What is the influence of using the PR method in different case studies?
3. What is the difference between multiple PR models and a single PR model applying in the neural networks?
4. What is the difference between the PR method and the random noise injection?

4.1 Experiment to answer Question # 1

The objective of performing these experiments is to demonstrate the use of the PR method has more advantages than common training methods. From Table 2, we produce four different results by applying the PR method in different phases. These include applying no PR method (non-PR or common training method) and only applying the PR method during the training phase, testing phase or both phases. We analyze that applying the PR method (either during the training phase or both phases) has less difference with the non-PR in terms of the accuracy. This means that the PR method has almost the same performance as the non-PR. From Table 2, the non-PR provides less robustness in the neural networks when we only apply the PR method during the testing phase. In other words, we use the PR method as an attack technique to fool the trained neural networks. The accuracy for the MNIST and Fashion MNIST datasets are greatly decreased in Table 2 could be caused by the over-fitting issue when a large number of the epoch is used. However, our method can relieve the

over-fitting issue as shown in Table 2 when we apply the PR method in the training phase only. Our method is able to classify the "clean" image correctly after we train the neural networks with PR method. We conjecture the PR method has run as a regularization method while training the neural networks. Compared to the common regularization (e.g. dropout, weight decay, etc.) method, they usually apply in the neural networks architecture, but the PR method executes on the data itself.

Table 2: The comparison results of applying pixel redrawn method during the different phase.

Dataset	Epoch	Accuracy			
		Non-PR	Training Phase	Testing Phase	Both Phases
MNIST	50	0.9929	0.9905	0.8097	0.9893
	500	0.9878	0.9824	0.4964	0.9879
	1000	0.9835	0.9671	0.2765	0.9865
Fashion MNIST	50	0.9107	0.8879	0.3538	0.8724
	500	0.8984	0.8795	0.3018	0.8670
	1000	0.8842	0.8788	0.2550	0.8632
CIFAR-10	50	0.6179	0.5703	0.5144	0.5386
	500	0.5989	0.5557	0.4864	0.5323
	1000	0.5873	0.5495	0.4739	0.5204
CIFAR-10 (grayscale)	50	0.5697	0.5099	0.3764	0.4845
	500	0.5466	0.4864	0.3429	0.4609
	1000	0.5369	0.4761	0.3370	0.4569

4.2 Experiment to answer Question # 2

The goal of operating these experiments is to display the PR method can be used as a defense technique against the state-of-the-art attack techniques. We conduct four cases that we have discussed in Section 3.4 in this sub-section. From Table 3, the PR method can defend most attacks in the case A. One interesting observation that we have found in the case B is that the trained neural networks are robust to defend most attacks (except the CW attack) while we do not apply the PR method during the testing phase. This shows that the PR method boosts the robustness of the neural networks. For the case C, the performance is greatly reduced due to the L_p distance is greatly increased. As we know, the case C is not practical because the L_p distance has to be as low as possible in order to create an adversarial example. We also analyze the PR method performs better in the grayscale dataset from the comparison between CIFAR-10 and CIFAR-10 (grayscale) dataset.

Table 3: Results for several datasets in the case A, B and C with 1,000 epochs. Note that in the case A, the adversarial image is generated from the original image before applying the PR method. In the case B, the adversarial image is produced from the original image without the PR method. In the case C, the adversarial image is created from the image after applying the PR method.

Dataset	Attack	Accuracy			
		Normal	Case A	Case B	Case C
MNIST, $\epsilon = 0.300$	FGSM	0.9865	0.9721	0.8985	0.5858
	BIM		0.9855	0.7234	0.4801
	MIM		0.9597	0.6953	0.3956
	CW		0.9885	0.3561	0.4270
Fashion MNIST, $\epsilon = 0.031$	FGSM	0.8632	0.8503	0.7976	0.7338
	BIM		0.8489	0.7655	0.6622
	MIM		0.8446	0.7310	0.6145
	CW		0.8444	0.1049	0.0934
CIFAR-10, $\epsilon = 0.031$	FGSM	0.5204	0.3343	0.2265	0.2155
	BIM		0.3612	0.2104	0.2048
	MIM		0.3070	0.1904	0.1875
	CW		0.4664	0.1728	0.1728
CIFAR-10 (grayscale), $\epsilon = 0.031$	FGSM	0.4569	0.3346	0.2217	0.1966
	BIM		0.3451	0.2079	0.1895
	MIM		0.3158	0.1888	0.1808
	CW		0.4288	0.1844	0.1727

4.3 EXPERIMENT TO ANSWER QUESTION # 3

The target of implementing these experiments is to present the benefit of using the multiple PR models rather than the single PR model. As aforementioned, the case C is not practical since the case C will cause high L_p distance. Hence, we do not include case C in this sub-section. From Table 4, the one with the multiple PR models outperforms the one with a single PR model. In average of the accuracy, the multiple PR models produce 1.7% and 3% higher than the single PR model for case A and B respectively. This shows the multiple PR models produce a stronger defense than the single PR model.

Table 4: The comparison of using a PR model and multiple PR models in several datasets with 1,000 epochs for the case A and B.

Dataset	Attack	Accuracy			
		single PR		multiple PR	
		Case A	Case B	Case A	Case B
MNIST, $\epsilon = 0.300$	FGSM	0.9721	0.8985	**0.9752**	**0.9371**
	BIM	0.9855	0.7234	**0.9865**	**0.8003**
	MIM	0.9597	0.6953	**0.9706**	**0.7693**
	CW	**0.9885**	0.3561	0.9875	**0.5767**
Fashion MNIST, $\epsilon = 0.031$	FGSM	0.8503	0.7976	**0.8564**	**0.8059**
	BIM	0.8489	0.7655	**0.8555**	**0.7703**
	MIM	0.8446	0.7310	**0.8501**	**0.7422**
	CW	0.8444	0.1049	**0.8476**	**0.1615**
CIFAR-10, $\epsilon = 0.031$	FGSM	0.3343	**0.2265**	**0.3630**	0.2181
	BIM	0.3612	**0.2104**	**0.3859**	0.2040
	MIM	0.3070	**0.1904**	**0.3353**	0.1853
	CW	0.4664	**0.1728**	**0.4964**	0.1717
CIFAR-10 (grayscale), $\epsilon = 0.031$	FGSM	0.3346	0.2217	**0.3655**	**0.2222**
	BIM	0.3451	0.2079	**0.3812**	**0.2118**
	MIM	0.3158	0.1888	**0.3532**	**0.1948**
	CW	0.4288	**0.1844**	**0.4526**	0.1795

4.4 EXPERIMENT TO ANSWER QUESTION # 4

The aim of conducting these experiments are to distinguish between PR model and random noise injection. Note that the value of the random noise that we used in this sub-section is between -1.0 and 1.0 with the normal distribution and we clip the value of the pixel between 0.0 and 1.0. We depict several outputs with the use of the PR method and random noise injection in Fig. 5.

Besides that, we show the experiment results for the random noises injection with the legitimate data in Table 5. We use a similar settings in Section 4.1 to only apply the random noises injection during the training phase, testing phase or both phases. We inspect the PR model (from Table 2) has higher performance than the random noises injection (from Table 5) in the case that the method is applied either during the training phase only or both the training and testing phases. Although the random noise injection has high performance in fooling the trained neural networks (in the case when we apply the random noise injection during the testing phase only), the L_2 distance is larger than our method since the image generated with the random noise injection is not able to be recognized by the human easily (shown in Fig. 5). In summary, the PR method is different from the random noise injection in terms of the performance in maintaining the accuracy when the number of the epoch is increased. Moreover, the PR method also remains the newly generated image perceptible by the human.

4.5 SUMMARY OF PR METHOD

From the experiment results, we summarize the contribution of using the PR method as follows:

- The PR method relieves the over-fitting issue by generating different possible output of the image. (Q1)
- The PR method increases the robustness of the neural networks.(Q2)

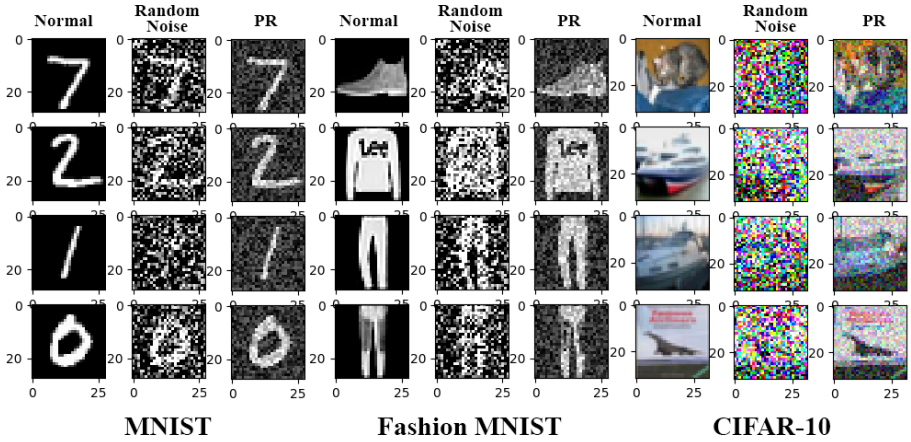

Figure 5: The example of using PR method and some random noises in MNIST, Fashion MNIST and CIFAR-10 datasets. From three images, the left most image is the original image, the middle one is the image after adding some random noises, and the right most image is the image generated after applying the PR method.

Table 5: The comparison results of applying the some random noises on the image during the different phase.

Dataset	Epoch	Accuracy		
		Training Phase	Testing Phase	Both Phases
MNIST	50	0.9871	0.1650	0.9596
	500	0.2789	0.1958	0.9705
	1000	0.1021	0.1695	0.9758
Fashion MNIST	50	0.8275	0.1435	0.8161
	500	0.7308	0.1357	0.8446
	1000	0.7263	0.1297	0.8449
CIFAR-10	50	0.5112	0.2205	0.4779
	500	0.4296	0.1883	0.4674
	1000	0.4194	0.4739	0.4718

- The defense level for the one with the multiple PR models is stronger than the one with the single PR model. (Q3)
- The PR model has higher performance than the random noise injection. (Q4)

5 Conclusion

In this study, we propose a novel method, namely Pixel Redrawn method which regenerates the image to all possible forms without compromising human perception. We address four questions in our experiments: **Q1** What is the influence of applying the PR method in different phases with a legitimate test datum?, **Q2** What is the influence of using the PR method in different case studies?, **Q3** What is the difference between multiple PR models and a single PR model applying in the neural networks?, and **Q4** What is the difference between the PR method and the random noise injection? From the experimental results, our method shows that it can prevent the over-fitting problem while maintaining the performance of the neural network, it also improves the robustness of the neural network. Furthermore, our method is more reliable than using some random noises.

References

Hojjat Adeli and Mingyang Wu. Regularization neural network for construction cost estimation. *Journal of construction engineering and management*, 124(1):18–24, 1998.

Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. *arXiv preprint arXiv:1608.04644*, 2016.

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, 2017.

Guneet S. Dhillon, Kamyar Azizzadenesheli, Jeremy D. Bernstein, Jean Kossaifi, Aran Khanna, Zachary C. Lipton, and Animashree Anandkumar. Stochastic activation pruning for robust adversarial defense. In *International Conference on Learning Representations*, 2018a. URL `https://openreview.net/forum?id=H1uR4GZRZ`.

Guneet S Dhillon, Kamyar Azizzadenesheli, Zachary C Lipton, Jeremy Bernstein, Jean Kossaifi, Aran Khanna, and Anima Anandkumar. Stochastic activation pruning for robust adversarial defense. *arXiv preprint arXiv:1803.01442*, 2018b.

Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Hang Su, Jun Zhu, Xiaolin Hu, and Jianguo Li. Boosting adversarial attacks with momentum. *arXiv preprint*, 2018.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

Chuan Guo, Mayank Rana, Moustapha Cisse, and Laurens van der Maaten. Countering adversarial images using input transformations. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=SyJ7ClWCb`.

Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-10 (canadian institute for advanced research). *URL http://www. cs. toronto. edu/kriz/cifar. html*, 2010.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pp. 1097–1105, 2012.

Anders Krogh and John A Hertz. A simple weight decay can improve generalization. In *Advances in neural information processing systems*, pp. 950–957, 1992.

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

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Steven J Nowlan and Geoffrey E Hinton. Simplifying neural networks by soft weight-sharing. *Neural computation*, 4(4):473–493, 1992.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

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.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

A APPENDIX

A.1 EXPERIMENTS TO ANSWER QUESTION #2

In this appendix, we provide Table 6, which is the extended version of Table 3 in Section 4.2. Note that more results with 50 and 500 epochs are added.

Table 6: Results for several datasets in the case A, B and C with 50, 500, and 1000 epochs respectively. Note that the case A is the adversarial image generated from the original image before applying the PR method. The case B is the adversarial image produced from the original image without the PR method. The case C is the adversarial image created from the image after applying the PR method.

Dataset	Attack	Epoch	Accuracy			
			Normal	Case A	Case B	Case C
MNIST, $\epsilon = 0.300$	FGSM	50	0.9893	0.9621	0.8728	0.4893
		500	0.9879	0.9738	0.9269	0.5563
		1000	0.9865	0.9721	0.8985	0.5858
	BIM	50	0.9893	0.9564	0.5964	0.1079
		500	0.9879	0.9804	0.8167	0.3772
		1000	0.9865	0.9855	0.7234	0.4801
	MIM	50	0.9893	0.9441	0.5711	0.0835
		500	0.9879	0.9589	0.7590	0.2520
		1000	0.9865	0.9597	0.6953	0.3956
	CW	50	0.9893	0.9874	0.4060	0.0248
		500	0.9879	0.9871	0.2392	0.5104
		1000	0.9865	0.9885	0.3561	0.4270
Fashion MNIST, $\epsilon = 0.031$	FGSM	50	0.8724	0.8363	0.7546	0.6248
		500	0.8670	0.8524	0.7899	0.7213
		1000	0.8632	0.8503	0.7976	0.7338
	BIM	50	0.8724	0.8299	0.7054	0.5073
		500	0.8670	0.8472	0.7558	0.6528
		1000	0.8632	0.8489	0.7655	0.6622
	MIM	50	0.8724	0.8260	0.6811	0.4877
		500	0.8670	0.8429	0.7231	0.6117
		1000	0.8632	0.8446	0.7310	0.6145
	CW	50	0.8724	0.8364	0.0770	0.0862
		500	0.8670	0.8462	0.0826	0.0829
		1000	0.8632	0.8444	0.1049	0.0934
CIFAR-10, $\epsilon = 0.031$	FGSM	50	0.5386	0.2968	0.1931	0.1893
		500	0.5323	0.3256	0.2053	0.1984
		1000	0.5204	0.3343	0.2265	0.2155
	BIM	50	0.5386	0.3126	0.1876	0.1865
		500	0.5323	0.3403	0.1932	0.1876
		1000	0.5204	0.3612	0.2104	0.2048
	MIM	50	0.5386	0.2772	0.1783	0.1815
		500	0.5323	0.2918	0.1775	0.1782
		1000	0.5204	0.3070	0.1904	0.1875
	CW	50	0.5386	0.4706	0.1646	0.1647
		500	0.5323	0.4732	0.1626	0.1683
		1000	0.5204	0.4664	0.1728	0.1728
CIFAR-10 (grayscale), $\epsilon = 0.031$	FGSM	50	0.4845	0.3417	0.2216	0.2014
		500	0.4609	0.3312	0.2098	0.1988
		1000	0.4569	0.3346	0.2217	0.1966
	BIM	50	0.4845	0.3518	0.2112	0.1910
		500	0.4609	0.3494	0.2005	0.1924
		1000	0.4569	0.3451	0.2079	0.1895
	MIM	50	0.4845	0.3270	0.1971	0.1851
		500	0.4609	0.3181	0.1834	0.1819
		1000	0.4569	0.3158	0.1888	0.1808
	CW	50	0.4845	0.4406	0.1817	0.1788
		500	0.4609	0.4361	0.1727	0.1726
		1000	0.4569	0.4288	0.1844	0.1727

A.2 EXPERIMENTS TO ANSWER QUESTION #3

In this appendix, we show Table 7 as the extended version of Table 4 in Section 4.3 with the results of the multiple PR models and the single model with two epochs (i.e. 500 and 1,000 epochs).

Table 7: The comparison of using a PR model and multiple PR models in several datasets with 500 and 1,000 epochs respectively, for case A and case B.

Dataset	Attack	Epoch	Accuracy			
			single PR		multiple PR	
			Case A	Case B	Case A	Case B
MNIST, $\epsilon = 0.300$	FGSM	500	**0.9738**	0.9269	0.9724	**0.9350**
		1000	0.9721	0.8985	**0.9752**	**0.9371**
	BIM	500	0.9804	0.8167	**0.9816**	**0.8224**
		1000	0.9855	0.7234	**0.9865**	**0.8003**
	MIM	500	0.9589	0.7590	**0.9645**	**0.7806**
		1000	0.9597	0.6953	**0.9706**	**0.7693**
	CW	500	0.9871	0.2392	**0.9872**	**0.3273**
		1000	**0.9885**	0.3561	0.9875	**0.5767**
Fashion MNIST, $\epsilon = 0.031$	FGSM	500	0.8524	0.7899	**0.8582**	**0.7903**
		1000	0.8503	0.7976	**0.8564**	**0.8059**
	BIM	500	0.8472	0.7558	**0.8527**	**0.7575**
		1000	0.8489	0.7655	**0.8555**	**0.7703**
	MIM	500	0.8429	0.7231	**0.8503**	**0.7310**
		1000	0.8446	0.7310	**0.8501**	**0.7422**
	CW	500	0.8462	0.0826	**0.8487**	**0.1432**
		1000	0.8444	0.1049	**0.8476**	**0.1615**
CIFAR-10, $\epsilon = 0.031$	FGSM	500	0.3256	0.2053	**0.3470**	**0.2104**
		1000	0.3343	0.2265	**0.3630**	**0.2181**
	BIM	500	0.3403	0.1932	**0.3689**	**0.2013**
		1000	0.3612	**0.2104**	**0.3859**	0.2040
	MIM	500	0.2918	0.1775	**0.3189**	**0.1847**
		1000	0.3070	**0.1904**	**0.3353**	0.1853
	CW	500	0.4732	0.1626	**0.4947**	**0.1709**
		1000	0.4664	**0.1728**	**0.4964**	0.1717
CIFAR-10 (grayscale), $\epsilon = 0.031$	FGSM	500	0.3312	0.2098	**0.3665**	**0.2174**
		1000	0.3346	0.2217	**0.3655**	**0.2222**
	BIM	500	0.3494	0.2005	**0.3750**	**0.2083**
		1000	0.3451	0.2079	**0.3812**	**0.2118**
	MIM	500	0.3181	0.1834	**0.3526**	**0.1917**
		1000	0.3158	0.1888	**0.3532**	**0.1948**
	CW	500	0.4361	0.1727	**0.4483**	**0.1829**
		1000	0.4288	**0.1844**	**0.4526**	0.1795