Denoising Adversarial Examples Using CNN Models

Juntong Jing*
Applied Mathematics, University of California, Santa Barbara, Goleta, California, 93117, US
*Corresponding author’s e-mail: juntongjing@ucsb.edu

Abstract. It has always been a complicated problem to resolve adversarial attacks because figures with adversarial attacks look similar to the original figures so that models can be fooled. With deceptive data, adversarial attacks can be a threat to neural networks. There are various ways to generate adversarial attacks. For instance, they are using one-step perturbation and using multi-step perturbation. In both methods, noise is added to the images. Therefore, a question pops up: are adversarial attacks similar to normal random noise? This paper aims to find if there is anything in common between random noise and adversarial attacks. A normal denoising CNN model is trained with random noise. Then groups of adversarial examples are collected by training on LeNet. Next, the denoising CNN model has been used to denoise those adversarial examples. Finally, after denoising the adversarial examples with the CNN model trained on normal random noise, the classification accuracy increases. Thus, it is reasonable to conclude that normal random noise and adversarial tracks have some common patterns.

1. Introduction
Due to the imperfections of imaging systems, transmission media, and recording equipment, digital images are often interfered with by various noises during their formation, transmission, and recording processes which may affect the visual effects of images and even hinder people's standard recognition. Thus, image denoising, which reduces noise in digital images, is essential in various fields.

A mainstream image denoising model is Auto Encoder. The auto encoder is an unsupervised model that contains two parts, the encoder and the decoder [1]. The encoder encodes the original representation into a hidden layer representation. At the same time, the decoder decodes a hidden layer representation into the original representation. The training goal is to minimize the reconstruction error. The dimension of the hidden feature is generally lower than the dimension of the original feature. The encoder and the decoder can be implemented with Convolutional Neural Networks (CNN) [2]. The image denoising task can be completed by learning the mapping from the noisy picture to the original picture.

In recent years, a special kind of image noise has attracted researchers' attention, called the adversarial attack. Adversarial attacks can be generated through various methods, making the machine learning models vulnerable [3-5]. However, it is a complicated problem to resolve adversarial attacks for deep neural networks [6]. The features of adversarial examples remain to be mysterious. Therefore, it can be inspiring and necessary to compare the adversarial examples to the normal random noise to see if they have any patterns in common.

Using MNIST as the dataset, this paper aims to use the denoising model trained on random noise to eliminate the noise in the adversarial examples. Suppose it can successfully eliminate the noise in the adversarial examples. In that case, it may indicate that adversarial examples share some common
patterns with the random noise. Then based on the experiment results, analyze if the noise in the adversarial examples is different from ordinary random noise.

Specifically, a denoising CNN model is proposed and trained by MNIST with random noise. After successfully building up a CNN model that can eliminate the random noise of each image in the denoising part, the next goal is to recognize the object on each image. Convolution kernels of convolution neural networks can detect the vertical edge images relative to the entire image as a vector of the connection layer. It ignores the image itself of the two-dimensional space characteristics. The convolution operation is very good at handling this kind of local feature. It can extract more useful information in images (as in the edge of the image).

Therefore, we build up the LeNet, let the FGSM attack be performed on LeNet, and obtain the adversarial samples. Then, apply the denoising CNN model trained by random noise onto the adversarial examples. Next, modify the adversarial attacks' intensity and obtain various adversarial examples with adversarial noise of different intensities. Then, apply the denoising CNN models on those different adversarial examples, respectively. Finally, analyze the classification accuracy of those adversarial examples after being denoised by the denoising CNN model.

The results show that even though the denoising CNN model is trained with random noise, it can still eliminate some of the noise of the adversarial examples, which indicates that there are some common patterns between random noise and adversarial attacks. Section 3 shows the experimental results, including the classification accuracy of the adversarial examples after being denoised by the denoising CNN model under different adversarial attack intensities. The results show that there are similarities between adversarial attacks and random noise. Therefore, it can be promising to develop methods to resolve adversarial attacks based on resolving random noise.

2. Method

2.1. Denoising CNN Model

The denoising CNN model contains nine layers. The first layer is a convolutional layer. It converts the input image with 28*28*1 into an image of 28*28*32. The second layer is a max-pooling layer, which converts each image into a size of 14*14*32. The third layer is also a convolutional layer, through which each image size remains the same. The fourth layer is a max-pooling layer, which shrinks each image to size 7*7*32. The fifth layer is a convolution layer, which does not change the size of each image. The sixth layer is a 2D nearest neighbor upsampling layer, which converts each image size into 14*14*32. The seventh layer is a convolutional layer, which does not change the size of each image. The eighth layer is a 2D nearest neighbor upsampling layer, which converts each image size into 28*28*32. The ninth layer is a convolutional layer, which converts each image into a size of 28*28*1. Thus, after passing through all nine layers, the size of each image remains unchanged. The deep learning image convolution output size calculation formula is:

\[ N = (W - F + 2P) \cdot S^{-1} + 1 \]  \hspace{1cm} (1)

where the input image size is \( W \times W \), the filter size is \( F \times F \), the stride is \( S \), the number of pixels for padding is \( P \). The structure of this CNN model is shown in Figure 1.
Figure 1. The structure of the denoising CNN model

We trained the denoising model for 50 epochs with an initial noise factor 0.8. The random noise is added to each image by randomly picking pixel points from each image and randomly adding those pixel points to each image. An example of the original image is shown in Figure 2. An example of the image after adding noise is shown in Figure 3. After denoising using our trained denoising CNN model, the image looks like Figure 4.
2.2. **Build the LeNet**

The original form of LeNet was first introduced in 1989 by Yann LeCun et al [7]. Later in 1998, after trying several methods to recognize handwritten characters and comparing them with standard handwritten digit recognition benchmarks, Yann LeCun et al [8-10] found that the convolutional neural network outperforms all other methods. The LeNet used in this paper contains four layers. The first layer is a convolutional layer with five convolution kernels, one input channel, and ten output channels. The second layer is a convolutional layer with five convolution kernels, ten input channels, and ten output channels. The third layer is a fully connected convolutional layer with each input sample size 320, and each output sample size 50. The fourth layer is a fully connected convolutional layer with each input sample size 50, and each output sample size 10. The structure of the LeNet is shown in Figure 5.

![Figure 5. The structure of LeNet](image)

2.3. **Obtain the Adversarial Examples on LeNet**

Next, we trained the adversarial attacks on this LeNet model. Adversarial attacks generate adversarial examples. The adversarial examples are purposely designed images with noise that can fool machine learning models but are not detectable to human eyes. In other words, the images being attacked by adversarial attacks look similar to the original images. However, they may be misclassified by machine learning models. There are many methods to generate adversarial attacks, for example, the Limited-memory BFGS (L-BFGS), FastGradient Sign method (FGSM), Jacobian-based Saliency Map Attack (JSMA), Deepfool Attack, Carlini & Wagner Attack (C&W), Generative Adversarial Networks (GAN), and Zeroth-order optimization attack (ZOO) [11]. In this paper, we used the FGSM.

After having the adversarial attacks attack the LeNet model, we collected the attacked figures as the adversarial examples. We expect to see that the attacked figures look similar to the original figures.

2.4. **Denoise the Adversarial Examples by the Denoising CNN Model**

After obtaining the adversarial examples, we can then try to resolve the adversarial attacks on those adversarial examples with our denoising CNN model, which is trained with random noise to see whether this model can eliminate adversarial attacks or not.

Let the initial intensity of the adversarial examples, i.e., epsilon, be 0.1. The denoising CNN model is then applied to those adversarial examples. Next, increase the epsilon to 0.2, and apply the denoising CNN model on those adversarial examples again. Keep increasing the epsilon to 0.3, 0.4, 0.5, and apply the denoising CNN model to adversarial examples. Observe how the classification accuracy of the adversarial examples reacts when they are denoised with the denoising CNN model trained with the random noise. If the classification accuracy is not 0, it demonstrates that although the denoising CNN model is trained with random noise, it can still restrain some adversarial attacks.
3. Experiment results and analysis

3.1. Denoising
This paper uses MNIST (Modified National Institute of Standards and Technology dataset) as the dataset. MNIST is the handwritten digit recognition dataset that contains 60,000 small square 28×28 pixel grayscale images of handwritten single digits between 0 and 9. The MNIST dataset is from the National Institute of Standards and Technology (NIST). The training set consists of handwritten numbers from 250 different people, 50 percent high school students, and 50 percent people who work at the Census Bureau. The test set is the same proportion of handwritten numeric data. The dataset is available on Yann LeCun’s homepage [12]. It contains 60,000 28x28 grayscale images of the ten digits, along with a test set of 10,000 images.

We first load the MNIST dataset. Then we add noise to each image by randomly picking pixel points from each image and randomly adding those pixel points to each image.

The denoising model is trained for 50 epochs with a random noise factor of 0.8. The higher the noise factor is, the “noisier” the images become. Here, a noise factor of 0.8 is picked, which is a suitable amount of noise, so we can get a denoising model that performs well. An example of the original MNIST images is shown in Figure 3. An example of the MNIST images after adding random noise is shown in Figure 4. After denoising using our trained denoising CNN model, the images look like Figure 5. After applying our denoising CNN model on the noisy MNIST images, we can see that the resulting images are still legible. Therefore, our denoising CNN model performs well with dealing with MNIST with random noise.

3.2. Adversarial Attacks
We made further testings by changing the intensity (epsilon) of the adversarial attacks. As epsilon increases, the adversarial attack becomes more intense, making the MNIST images more challenging for the classifier to recognize. Therefore, we monitored the classification accuracy under different epsilons to see how well our denoising CNN model, which is trained on random noise, performs when dealing with adversarial attacks. Each time we changed the epsilon, we recorded the classification accuracy that the adversarial examples get when applying the CNN denoising model trained with random noise. In other words, the accuracy infers how much noise the denoising model can eliminate for the adversarial examples.

Table 1. Classification accuracies under different epsilons

| Intensity of the Adversarial Attack (epsilon) | 0.1   | 0.2   | 0.3   | 0.4   | 0.5   |
|---------------------------------------------|-------|-------|-------|-------|-------|
| Classification Accuracy                      | 0.1838| 0.2111| 0.1859| 0.1061| 0.0465|

First, we set epsilon to be 0.1, which is very low, indicating that the adversarial attack, in this case, is not intense. The accuracy is 0.851. Next, we gradually increase the intensity of the adversarial attacks and see how our denoising CNN model performs on those adversarial examples. When epsilon is creased to 0.2, the accuracy becomes 0.4301. Then, further, increase epsilon to 0.3. The accuracy becomes 0.0869. Then increase epsilon to 0.4. The accuracy decreases to 0.0167. Finally, increase epsilon to 0.5. The accuracy becomes 0.0063. The results are shown in Table 1. It is evident that when epsilon is more significant than 0.2, as the intensity of the adversarial attack increases, the accuracy decreases. Furthermore, the classification accuracies are all above 0, which denotes that even though the denoising CNN model is trained on random noise, it can still reduce the noise of adversarial attacks.

4. Discussion
By applying the denoising CNN model trained on the random noise onto the adversarial examples, we found that it successfully eliminates some noise. However, a limitation of this paper is that we can only make a reasonable assumption that there exist some standard features between random noise and adversarial attacks. However, we cannot precisely determine the standard features or what patterns those
common features have. Therefore, exploring further how and why random noise and adversarial attacks share some standard features in future studies would be promising. Then, with more information about those standard features, we can find more effective ways to resolve adversarial attacks.

5. Conclusion
Using the denoising CNN model trained by random noise to denoise the adversarial examples increases the classifying accuracy of the adversarial examples. The result shows that when adversarial attack intensity is 0.2, the classifying accuracy is the highest. As the epsilon increases from 0.2 to 0.5, the accuracy gradually decreases. Thus, as the attack intensity increases, the number of adversarial samples that can be processed by the denoising CNN model is decreasing, and the accuracy is getting lower and lower. This shows that the intersection of the samples generated by the confrontational attack and the random noise becomes smaller and smaller when the confrontation intensity becomes greater.

Therefore, it indicates that the denoising CNN model, trained by random noise, eliminates some of the noise of the adversarial examples. Moreover, as the adversarial attack gets more intense, the denoising model becomes less valuable. Thus, it implies that there are patterns in common between random noise and adversarial attacks.

The denoising model trained by random noise can also reduce the noise of adversarial examples, inspiring us to use a random noise denoising model to work as the pre-process method when dealing with adversarial examples. Specifically, when looking for methods to deal with adversarial attacks, we can find methods similar to the random noise denoising model.

References
[1] G. Dong, G. Liao, H. Liu and G. Kuang, "A review of the autoencoder and its variants: a comparative perspective from target recognition in synthetic-aperture radar images," in IEEE Geoscience and Remote Sensing Magazine, vol. 6, no. 3, pp. 44-68, Sept. 2018.
[2] L. Liu, "Deception, robustness and trust in big data fueled deep learning systems," in 2019 IEEE International Conference on Big Data (Big Data), Los Angeles, CA, USA, 2019 pp. 3-3. doi: 10.1109/BigData47090.2019.9005597
[3] D. Yinpeng, L. Fangzhou, P. Tianyu, S. Hang, Z. Jun, H. Xiaolin, and L. Jianguo, “Boosting adversarial attacks with momentum,” Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018, pp. 9185-9193.
[4] H. Sandy, P. Nicolas, G. Ian, D. Yan, and A. Pieter, “Adversarial attacks on neural network policies,” arXiv:1702.02284 [cs.LG], 2017.
[5] N. Akhtar and A. Mian, “Threat of adversarial attacks on deep learning in Computer Vision: A Survey,” in IEEE Access, vol. 6, pp. 14410-14430, 2018.
[6] Thakur, Rini Smita, Ram Narayan Yadav and Lalita Gupta, “State-of-art analysis of image denoising methods using convolutional neural networks,” IET Image Process, vol. 13, pp. 2367-2380, 2019.
[7] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, “Backpropagation applied to handwritten zip code recognition,” Neural Computation, vol. 1, pp. 541-551, 1989.
[8] Y. LeCun, “Generalization and network design strategies,” Technical Report CRG-TR-89-4, Department of Computer Science, University of Toronto, 1989.
[9] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, “Handwritten digit recognition with a back-propagation network,” Advances in Neural Information Processing Systems 2 (NIPS*89), 1990.
[10] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” Proceedings of the IEEE, vol. 86, pp. 2278 - 2324, 1998.
[11] Gaudenz Boesch, “What Is Adversarial Machine Learning? Attack Methods in 2021,” No-code AI vision, 2021.
[12] Y. LeCun, THE MNIST DATABASE of handwritten digits, http://yann.lecun.com/exdb/mnist/.