METHODS TO DEFEND ADVERSARIAL IMAGES

Yifeng Xiong
Department of Computer Science
University of California, Irvine
Irvine, CA 92697, USA
yifengx4@uci.edu

ABSTRACT

Deep neural networks, such as CNN, ResNet, are useful tools for image identification. However, with some little change, the adversarial images could fool the neural network. In this project, we try to build a based on Autoencoder. The dataset I use is CIFAR-10. I first train an Autoencoder with clean images, and I use the Autoencoder to preprocess the adversarial image.I get 55% correct on both clean and perturbed images. There are still many improvements we are considering to do.

1 INTRODUCTION

Deep neural networks are the most common techniques that used for image identification. Residual Neural network, He et al. (2016), has test error 8.75% with 20 layers. However, these strong neural networks are not robust for some attack method, such as Fast Gradient Sign Method (Goodfellow et al., 2014). Projected Gradient Descent (Madry et al., 2017), etc. Adversarial attacks can fool the neural networks with only small perturbations to images but cannot fool human beings. The main purpose for this project is to find some ways to enhance the robustness of neural networks. I generate perturbed images with three powerful attacks, their attack success rate are all over 99%. After preprocessing with my autoencoder, the neural network still get 55% correct on both clean and perturbed images.

2 RELATED WORK

2.1 PCA DETECTOR

Principal Component Analysis (PCA) projects a set of n-dimensional data points into a lower dimensional space where the data points exhibit the most variance and hence contain the most information. Hendrycks and Gimpel apply PCA to pairs of normal and adversarial images and find that the PCA coefficients for later principal components of adversarial images have larger variance compared to those of normal images (Hendrycks & Gimpel, 2016). The detector relies solely on the coefficient variance to tell if an image has been attacked.

2.2 DISTRIBUTION DETECTOR

Grosse et al. (2017) compare the underlying distribution of normal images with the distribution of adversarial images and use a kernel-based two-sample test with Maximum Mean Discrepancy (MMD) and Energy Distance (ED) as statistical metric to flag adversarial behavior in a batch of images. However, the major drawback of this statistical approach is that it cannot detect adversarial examples on a per-input basis; even worse, the confidence of the test depends heavily on the number of samples in the image batch.

2.3 SEPARATE NEURAL NETWORK

Gong et al. (2017) propose a simple binary classifier to separate adversarial examples from normal images trained on a mixture of normal and adversarial examples. The model successfully defends

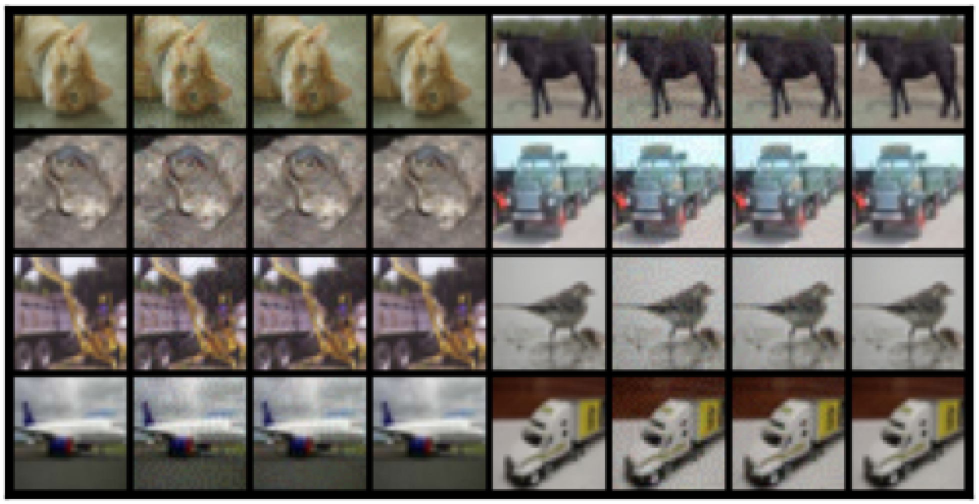

Figure 1: Clean images and perturbed images. The images are list in the order: clean, FGSM, PGD, Deepfool.

against FGSM and JSMA attacks with near 100% accuracy, but only achieve 70% detection rate on CIFAR dataset against CW attack even at the cost of 40% false positive rate.

3 Approach

The first method I tried is to do some experiments with the latent space. I first train an Autoencoder with clean images, and I use the Autoencoder to find the latent space of clean images and perturbed images. Finally, I use a simple multi-layer fully connected neural network to identify whether the image is perturbed. However, the result is terrible. My detector always classified the input as a clean image. I think there are several possible reasons for the result. But I'll firstly introduce the best result I get, and I'll discuss those reasons later.

3.1 Residual Neural Network

The model I use to classify the images, to attack and defend is the Residual Neural Network introduced by He et al. (2016). I mainly follow the structure in the paper, and build a ResNet-20. I train the model with 200 epoches. I get 88.29% correct on test data.

3.2 Attack Method

The goal for the attack method is to find a perturbed image that looks similar to the origin one, where human beings believe it's the same, but for neural networks, it is completely different. For an input image x, neural network F, we want to generate the perturbed image x^*, where $F(x^*) \neq F(x)$, and $Distance(x^*, x)$ is small. We usually choose $L2$ distance. Figure 1 shows some examples of these attack method. In the following, we introduce three attack methods we use.

3.2.1 Fast Gradient Sign Method

Goodfellow et al. (2014) introduces Fast Gradient Sign Method to generate adversarial images by using the derivative of the loss function of the model. It perturbs the image in the direction of derivative by magnitude ϵ.

$$x^* = x + \epsilon \cdot sign(\nabla \boldsymbol{x} L(\theta, x, y)) \tag{1}$$

Where x is the input, x^* is the perturbed image, ϵ is the perturb rate, and L is the loss function.

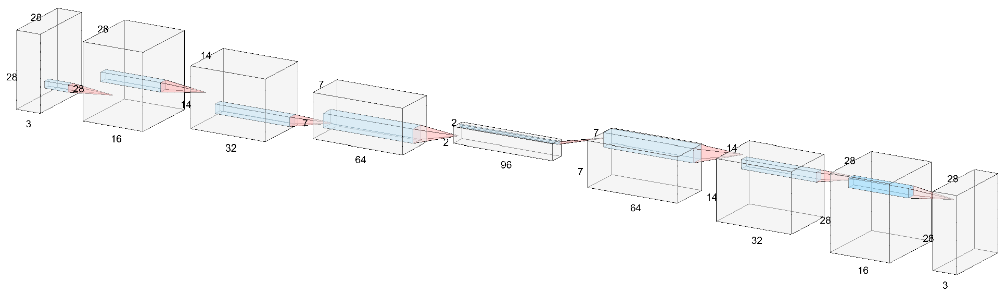

Figure 2: Structure of the Convolutional Autoencoder

3.2.2 Projected Gradient Descent

Madry et al. (2017) introduces Projected Gradient Descent, which is an iterative version of FGSM. Instead of perturbing the image only once with magnitude ϵ, it changes the image many times with a smaller magnitude α. Different from the Basic Iterative Method (Kurakin et al., 2016), PGD firstly adds some noise to the image.

$$x^{t+1} = \Pi_{x+S}(x^t + \alpha \cdot sign(\nabla x L(\theta, x, y))) \tag{2}$$

Where Π is the projection function, which maintains the distance between perturbed image and origin image.

3.2.3 Deepfool Method

Moosavi-Dezfooli et al. (2016) introduces Deepfool Method. It views the multi-class classifier as several binary classifiers. To perturb the image, it first finds the distance from the origin image to the separating affine hyperplane.

$$\hat{l}(x_0) = \underset{k \neq \hat{k}(x_0)}{\arg\min} \frac{|f_k(x_0) - f_{\hat{k}(x_0)}(x_0)|}{\|w_k - w_{\hat{k}(x_0)}\|_2} \tag{3}$$

Then we can get the minimum perturbation:

$$r_*(x_0) = \frac{|f_{\hat{l}(x_0)}(x_0) - f_{\hat{k}(x_0)}(x_0)|}{\|w_{\hat{l}(x_0)} - w_{\hat{k}(x_0)}\|_2^2}(w_{\hat{l}(x_0) - w_{\hat{k}(x_0)}}) \tag{4}$$

Where $f(x) = \mathbf{W}^\top x + b$ is the affine classifier, and w_k is the k^{th} column of $\mathbf{W}$.

3.3 Convolutional Autoencoder

I firstly train an autoencoder with only clean image. Instead of a traditional fully connected autoencoder, I use convolutional layers to decrease the dimension. It has four convolutional layers. Figure 2 is the structure of my autoencoder.

3.4 Defend by autoencoder

My current best model to defend is the autoencoder itself. I input both clean and perturbed images into the model, and then input the output from autoencoder to ResNet. ResNet has 88.29% correct rate with the clean image. Without the data preprocessing, ResNet still has 88% correct on clean image, but it has less than 1% correct on perturbed image. After preprocessing the data with the trained autoencoder, ResNet has 55% correct on both clean and perturbed image, which I believe is somehow useful, since random guess should have only 10% correct.

4 Current Progress

The first idea I have is to do some experiments with the latent space. I guess in the latent space, the perturbed image is a little strange. However, after I generate all the latent space data, and design a multi-layer fully connected neural network to identify whether the image is perturbed, I get very bad result. There are plenty of reasons I believe.

4.1 Latent Space

The dimension of latent space is not large enough. Maybe my latent space is too small, which means during the encoding, I lose so much information, such that the perturbed image is the same as the clean image. That's how my current best model comes from. Since the autoencoder views the perturbed image as a clean image, I can use it to preprocess the data. The result is much better, since I get 55% correct after preprocess the data, which also proves that my autoencoder has the ability to perturb the image back to the origin one. And it can also explain why my first experiment doesn't work well.

4.2 Dataset Choose

The dataset I use currently is CIFAR-10. I think with a more complicated dataset, I can generate a more complicated autoencoder, maybe in this case, it will be sensitive to the perturbed image.

5 Future plan

In the rest several weeks, I plan to do more experiments in the following fields.

5.1 Different Dataset

I'm planning to use CIFAR-100 for the next few experiments. Also, I need to build a more complicated ResNet.

5.2 Normalizing flow

Instead of autoencoder, normalizing flow may be a good choice. Since normalizing flow map the image to a vector that has the same dimension, same size. Maybe less information losses in the pipeline.

References

Zhitao Gong, Wenlu Wang, and Wei-Shinn Ku. Adversarial and clean data are not twins. *arXiv preprint arXiv:1704.04960*, 2017.

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

Kathrin Grosse, Praveen Manoharan, Nicolas Papernot, Michael Backes, and Patrick McDaniel. On the (statistical) detection of adversarial examples. *arXiv preprint arXiv:1702.06280*, 2017.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. June 2016.

Dan Hendrycks and Kevin Gimpel. Early methods for detecting adversarial images. *arXiv preprint arXiv:1608.00530*, 2016.

Alexey Kurakin, Ian Goodfellow, Samy Bengio, et al. Adversarial examples in the physical world, 2016.

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

Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: a simple and accurate method to fool deep neural networks. pp. 2574–2582, 2016.