Denoising with Networks: A Black-Box Defense to Adversarial Attacks

John Gilbert
Graduate Student in Computer Science
National Taiwan University

Abstract

This paper presents a method of pre-processing with a denoising autoencoder that enhances model robustness to adversarial attacks, with a slight compromise in regular non-adversarial accuracy. Additionally, this method was shown to work well with training on adversarial images to further improve adversarial robustness. Overall, the addition of the autoencoder for pre-processing adversarial images helps to reduce adversarial noise and increases the complexity of performing a successful adversarial attack.

Code is available at: `https://github.com/johngilbert2000/ai_security_defense`

1 Introduction

Neural networks can be fooled by images perturbed slightly such that the perturbation targets the network's computation of gradients. Such adversarial images often look unchanged to the human eye, while effectively devastating the network's ability to perform classification.

The Fast Gradient Sign Method (FGSM) is a standard adversarial attack that effectively performs gradient ascent by modifying input images [1]. Formally, FGSM is performed as follows:

$$\hat{x} = x + \epsilon \cdot \text{sign}(\nabla_x L(\theta, x, y)$$

where $\hat{x}$ corresponds to adversarial sample images, x input images, and perturbations being the sign of the gradient with respect to network parameters, θ, the input images, x, and the true labels, y.

Projected Gradient Descent (PGD) is an iterative approach to adversarial generation that extends FGSM by performing FGSM in a loop, where the adversarial output is used as input for the next iteration in the following fashion:

$$x_{t+1} = \Pi_{x+S}(x_t + \alpha \cdot \text{sign}(\nabla_x L(\theta, x, y)))$$

where α is the learning rate, functioning similar to ϵ in FGSM. PGD is known to produce more powerful adversarial images, although it takes longer to produce such images [2].

Approaches to defending against such attacks typically either aim to reduce the noise in the images with some form of pre-processing, or attempt to make the model in question more robust to that attack with a form of adversarial training [2,3]. Training a model on adversarial images can help increase its robustness, although this has been shown to at times lead to a slight reduction in accuracy against non-adversarial images [4].

This experiment expands on the idea from Cohen et al. that a form of smoothing on adversarial images can help with model robustness. However, rather than performing Gaussian smoothing on images, an autoencoder neural network was trained to pre-process adversarial images instead. Autoencoders work by compressing images to a significantly smaller latent space, prior to reconstructing the images.

As a result, autoencoders naturally tend to produce a form of smoothing on the images due to a loss of information in compression, and furthermore they can be trained to remove noise [5]. Additionally, fine-tuning on adversarial images was performed to enhance model robustness.

2 Methods

Dataset All models were trained on the CIFAR-10 training set or an adversarial equivalent, containing 50,000 images. Models were assessed using either the CIFAR-10 test set or an adversarial equivalent, containing 10,000 images. The dataset contains 10 classes of 32x32 color images.

Adversarial Attacks Both the Fast Gradient Sign Method (FGSM) and Projected Gradient Descent (PGD) attacks were initially performed against the classification model of ResNet-20, and later against SePreResNet-56. An epsilon of 8 on a 0-255 scale was used for all attacks, and all attacks were untargeted. PGD was performed for 10 steps with a learning rate of 0.01, with a perturbation of random noise bounded by epsilon used initially in the first step with each image. Adversarial training samples were generated by performing FGSM and PGD on all 50,000 CIFAR-10 training images, and an adversarial test set was generated by performing FGSM and PGD on 10,000 CIFAR-10 test images.

Preprocessing A generic denoising convolutional autoencoder with 3x3 kernels was trained to pre-process adversarial images. The encoder consisted of two downsampling convolutional layers, followed by a decoder with upsampling convolutional transpose layers and a final convolutional layer. The layer input shapes are summarized in Table 1, with batch size not shown:

Table 1: Autoencoder Architecture

Layer	Shape
Input	(32,32,3)
Conv2D	(16,16,16)
Conv2D	(8,8,8)
Conv2D Transpose	(16, 16, 8)
Conv2D Transpose	(32, 32, 16)
Conv2D	(32, 32, 3)

The autoencoder was trained by using adversarial attack images as inputs and the original non-adversarial image counterparts as targets, with mean squared error (MSE) as the loss function. The learning rate was initialized to 10^{-3} with the Adam optimizer, and gradually decayed over 100 epochs with a minimum learning rate of 10^{-4}.

The above autoencoder architecture was selected after varying the number of layers and layer sizes, followed by testing the accuracy of a pre-trained ResNet-20 on the reconstructed autoencoder outputs. It was found that too small of a latent space significantly reduced the accuracy due to reconstructed images being too blurry, and too large of a latent space did not result in effectively pre-processing adversarial images. The above architecture was experimentally found to be a good compromise between adversarial accuracy and non-adversarial accuracy.

After being cleaned by the autoencoder, images were normalized according to examples in the tf2cv repository using the mean and standard deviation of RGB values of the entire CIFAR-10 dataset.

Classification Models Image classification was initially performed with a pre-trained ResNet-20 due to its relatively lightweight size, and then afterward other models were assessed to select for a more effective architecture. Pretrained models were taken from the tf2cv package[1] for the purpose of generating adversarial samples and fine-tuning against such adversarial samples. Given memory and time constraints, as well as a GPU issue that arose, the SePreResNet-56 model was chosen due to a higher fine-tune performance against adversarial samples compared to ResNet-20, while still being lightweight and allowing for a relatively fast training and inference time.

[1]Pretrained Model Source: `https://github.com/osmr/imgclsmob/`

Additionally, EfficientNet was fine-tuned on CIFAR-10 and its adversarial samples, given that it is considered state-of-the-art on CIFAR-10. However, due to time and memory constraints, as well as a lack of access to GPUs, this was not pursued to completion.

Fine-Tuning Both ResNet-20 and later SePreResNet-56 were fine-tuned on a combination of adversarial images and autoencoder outputs. Both FGSM and PGD adversarial images were produced from the entire CIFAR-10 training set of 50,000 images for the purpose of fine-tuning. Autoencoder outputs included the reconstructed images produced by the autoencoder from regular non-adversarial images, as well as from using PGD and FGSM images. Additionally, images from the original CIFAR-10 training set were included in fine-tuning to help prevent the pre-trained model from unlearning its capacity to classify regular CIFAR-10 images.

A large batch size of 256 was selected after some hyperparameter tuning, as this seemed to help to prevent divergence with regard to model accuracy on regular CIFAR-10 images. It is possible that the large batch size was beneficial as it allowed a single batch to include many types of images (FGSM, PGD, autoencoder outputs, and regular CIFAR-10 images), thereby allowing each batch to be more balanced during training. Additionally, training was done with the Adam optimizer, with an initial learning rate of 10^{-4} that was eventually decreased to 10^{-5}.

3 Results

This section contains results from assessing classification accuracy with ResNet-20 and SePreResNet-56. Regular input refers to the original CIFAR-10 test set of 10,000 images, and FGSM and PGD inputs refer to the corresponding adversarial images produced by directly attacking the model being assessed.

Table 2: ResNet-20 Accuracies

	Initial		Fine-Tuned	
Input	No AE	With AE	No AE	With AE
Regular	93.84%	79.09%	88.98%	82.56%
FGSM	24.37%	66.78%	59.47%	75.38%
PGD	0.13%	72.84%	61.50%	79.36%

Table 2 shows the performance of the original pre-trained ResNet-20, with and without autoencoder (AE) preprocessing, for both adversarial and regular images. Initial pertains to the assessment prior to fine-tuning the ResNet on any adversarial images, and Fine-Tuned refers to accuracies after fine-tuning on both adversarial samples and autoencoder outputs for 3 epochs with a learning rate of 10^{-4} and a batch size of 256. It was found that a large batch size helped prevent the network from diverging during training and unlearning its previously learned information. Additionally, a low learning rate was used as the network was already pre-trained on CIFAR-10.

Table 3: SePreResNet-56 Accuracies

	Initial		Fine-Tuned	
Input	No AE	With AE	No AE	With AE
Regular	95.32%	81.21%	92.13%	88.16%
FGSM	46.52%	73.67%	77.19%	84.30%
PGD	4.50%	76.14%	76.89%	85.95%

Table 3 shows the initial accuracies of the original pre-trained SePreResNet-56 model on regular images, FGSM attack images, and PGD images with and without autoencoder (AE) preprocessing. Note that FGSM and PGD attacks were directly targeted at the SePreResNet-56 model, and fine-tuning was performed initially on both the attack images and autoencoder outputs for several epochs, prior to fine-tuning just on autoencoder outputs. The batch size was kept at 256, and the learning rate was initially set at 10^{-4} and then decreased to 10^{-5}. Experimenting with higher learning rates and lower batch sizes appeared to result in the model being more prone to diverge with regard to regular image accuracy.

4 Discussion

In all experiments, the use of a denoising autoencoder helped reduce the effectiveness of adversarial attacks. There are multiple factors that likely contribute the autoencoder's effectiveness. Firstly, autoencoders have a natural tendency to blur images upon reconstruction. While too much blur can be counterproductive to classification, this does effectively act as a form of smoothing that reduces the sharpness of adversarial noise, and hence hinders the effectiveness of an adversarial attack. Secondly, the act of compressing an image to a small latent space can naturally cause some information to be lost, and given that the autoencoder is trained to reconstruct original images from adversarial images, the loss of adversarial information upon compression to the latent space is encouraged. Thirdly, adversarial attacks target the gradients of a model architecture, and the autoencoder adds complexity to this process. The inclusion of an additional neural network in pre-processing makes it so that the adversarial attacks have to both fool the autoencoder as well as the classification model. This significantly increases the complexity of performing a successful adversarial attack.

It can be observed from the Results section that in all cases, the inclusion of the autoencoder caused a reduction in accuracy with regular non-adversarial images, while causing an increase in accuracy for adversarial images. This is likely due to slight blurring of the reconstructed images. Given that the images are only 32x32 in size, it is reasonable to expect that adding a blur to the images could cause the content of some of the images to become more unrecognizable, thereby reducing regular image accuracy. It is possible that the use of a more advanced autoencoder, such as one similar to NVAE or VQ-VAE, could produce better results [6,7].

The use of fine-tuning on adversarial examples as well as autoencoder outputs also resulted in a large boost in accuracy. It is possible that the accuracy could be improved further by a larger, more sophisticated model with more fine-tuning. However, it is also possible that a PGD attack could target such a model, and run for more iterations. Hence, the use of the autoencoder for pre-processing enhances the safety of an adversarial fine-tuning defense.

5 Conclusion

In this experiment, a pre-trained model was shown to be made more robust to adversarial attacks. The use of an autoencoder for image pre-processing proved to help in defense against adversarial samples, as it effectively performed smoothing, caused a loss of adversarial information, and increased the complexity needed for an attack to succeed. The combination of fine-tuning against adversarial samples and autoencoder pre-processing showed to be a reasonable defense, although some accuracy was compromised with regard to non-adversarial images.

References

[1] Ian J. Goodfellow, Jonathon Shlens & Christian Szegedy (2015) Explaining and Harnessing Adversarial Examples. *In International Conference on Learning Representations (ICLR)*.

[2] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, Adrian Vladu. (2019) Towards Deep Learning Models Resistant to Adversarial Attacks. `https://arxiv.org/abs/1706.06083`.

[3] Jeremy Cohen, Elan Rosenfeld, J. Zico Kolter. (2019) Certified Adversarial Robustness via Randomized Smoothing. `https://arxiv.org/pdf/1902.02918.pdf`

[4] Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, Aleksander Madry. (2019) Robustness May Be at Odds with Accuracy. `https://arxiv.org/pdf/1805.12152.pdf`

[5] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, Pierre-Antoine Manzagol. (2008) Extracting and Composing Robust Features with Denoising Autoencoders. `https://www.cs.toronto.edu/~larocheh/publications/icml-2008-denoising-autoencoders.pdf`

[6] Arash Vahdat, Jan Kautz. (2020) NVAE: A Deep Hierarchical Variational Autoencoder. `https://arxiv.org/pdf/2007.03898.pdf`

[7] Ali Razavi, Aaron van den Oord, Oriol Vinyals. (2019) Generating Diverse High-Fidelity Images with VQ-VAE-2. `https://arxiv.org/pdf/1906.00446v1.pdf`