Defence Mechanisms Against One-Pixel Attack

Nikitha Ravi
Department of Computing Science
Simon Fraser University
Burnaby, BC V5A 1S6
nravi@sfu.ca

Venkata Sai Pavan Kumar Kosaraju
Department of Computing Science
Simon Fraser University
Burnaby, BC V5A 1S6
vkosaraj@sfu.ca

Rohith Sooram
Department of Computing Science
Simon Fraser University
Burnaby, BC V5A 1S6
rsooram@sfu.ca

Syed Ikram
Department of Computing Science
Simon Fraser University
Burnaby, BC V5A 1S6
sikram@sfu.ca

Abstract

With the advent of AI on the rising, security is a critical aspect in this rapidly evolving field. Recent studies have shown how the Deep Neural Networks can be fooled easily to give the incorrect predictions by designing the input in a specific way which are called as adversarial attacks. A simple adversarial attack, one-pixel attack where changing one pixel of the image fools the Deep Neural Networks. To combat this, we introduce two methodologies one, random data augmentation and second, adding denoising layers to the DNN model.

1 Introduction

Adversarial images are the images created by adding a fine-tuned perturbation which makes a Deep Neural Network misclassify an image. These perturbations are so subtle that they are not visible to human eye, but still it leads to the image being misclassified.

There are different methods in which an adversarial image is being generated, with basic iterative method being commonly used. These methods exploit much finer perturbations, which do not alter the image enormously. They can be implemented with no knowledge on the underlying Neural Network and this particular method is known as Black Box Attack. One – Pixel attack is one such black box attack which fools a Deep Neural Networks (DNN) by adding just one-pixel adversarial perturbations to an image. It is proven as an effective method for creating adversarial images.

One-Pixel attack

One-Pixel attack is a method for generating one-pixel adversarial perturbations based on Differential Evolution. It requires less adversarial information and fools more types of networks. It generates adversarial images by modifying one pixel of an image to a certain colour aiming to minimize the confidence of the neural network.

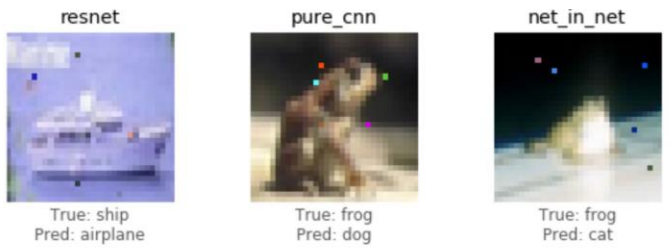


Figure 1: One-Pixel attack on different networks with actual and predicted labels.

The Differential Evolution algorithm iteratively generates adversarial images that modifies a random pixel and runs the images through the neural network. Then it combines the previous pixel positions and colours together and generates multiple adversarial images from them and runs the new images through the neural network. If any pixel lowers the confidence of the network, then they are considered to be the current best solutions. The Differential Evolution algorithm repeats the above steps for multiple iterations and generates an adversarial image that has the least confidence for the neural network.

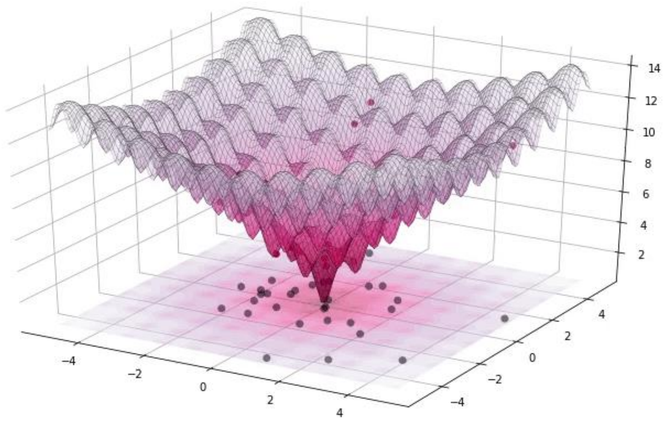


Figure 2: Example of Differential Evolution iteratively optimizing the 2D function.

The perturbations are encoded into an array where each element in the array is a tuple holding five elements: x-y coordinates and RGB value of the perturbation. The DE generates new adversarial images using the below DE formula,

$$x_i(g+1) = x_{r1}(g) + F(x_{r2}(g) + x_{r3}(g)),$$
$$r1 \neq r2 \neq r3,$$

where xi is an element of the array, r1, r2, r3 are random numbers, F is the scale parameter set to be 0.5, g is the current index of generation.

These adversarial images pose threat to the security of the DNNs and this should be taken care of, to make the DNNs more robust. We have come up with defence approaches to prevent the pixel attack.

2 Approach

By understanding the One-Pixel attack and how it affects the neural network, it is clear that the location of the pixel plays a significant role for the image to be perturbed. Performing changes on the pixel colour and the position can help the neural network to classify the image accurately. Based on these, we have come up with approaches to make our network less sensitive to these kinds of attacks.

2.1 Random Data Augmentation

Data augmentation is an optimization mechanism which is used to improve generalization properties. In this approach, we present simple yet powerful augmentation techniques to alter the pixel positions. This can be done either by a Flip or a Rotation of the adversarial image.
With a normal Flip or Rotation to the adversarial image, the neural network does not attain robustness because the One-Pixel attack will still be able to find that particular pixel to attack. By adding randomness to a prediction function, it transforms the image randomly, ie; random flipping of the image or rotation by a random degree before feeding it to the Deep Neural Network. This causes One-Pixel attack to either generate adversarial images in a more significant time or not generate any adversarial image at all.

2.2 Image Denoising: Adding Autoencoding Layers to The Deep Neural Network

In the Random data augmentation approach the noise in the adversarial image still persists even though the DNN classifies the image accurately. This noise can be eliminated by adding denoising autoencoding layers before the first layer of the network, to make sure it receives denoised image instead of a perturbed image. Although one-pixel attack is a black-box attack, adding encoding and decoding layers to the model eliminates the noise in the pixels, thereby giving a clean image to the network.

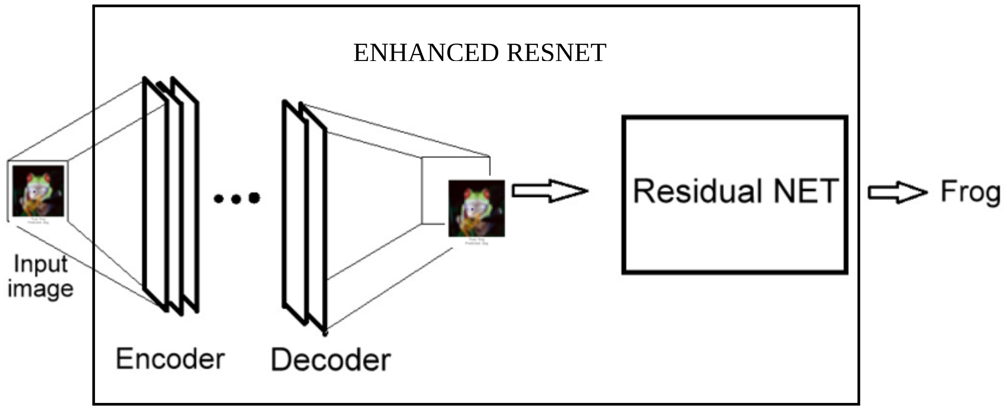


Figure 3: Enhanced ResNet architecture – ResNet with autoencoding layers.

Denoising autoencoders take partially corrupted images whilst training to recover the original undistorted images. To train an autoencoder to denoise data, it is necessary to perform preliminary stochastic noise insertion, in order to corrupt the image and use the corrupted image as an input to the autoencoder, with the only exception that the loss should still be computed for the original image against the corrupted image.

3 Experiments and Results

We considered our baseline model as ResNet using CIFAR-10 dataset and experimented with our defence approaches. We have added a module to capture real-time images and classify them. We found that physical images of the adversarial perturbations can fool the Deep Neural Network.

Realtime demo of how the model labels the image printed on paper

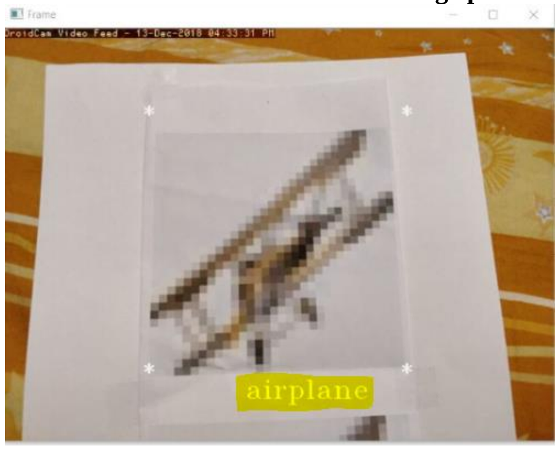


Figure 4: Airplane image classified accurately

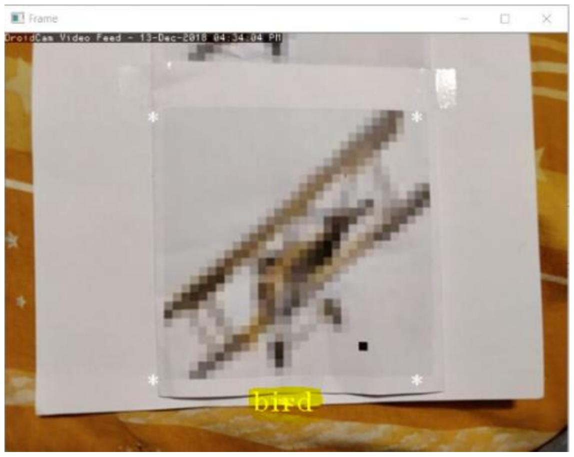


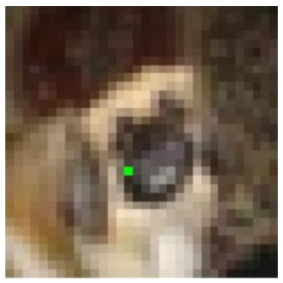
Figure 5: Adversarial image generated by one-pixel attack, misclassified as bird.

3.1 Random Data Augmentation

In this approach we defined our own prediction function on which we experimented different types of augmentation techniques. If no augmentation was applied to the image the time taken for adversarial images to be generated by the attacker is the fastest. When a flip or a rotation augmentation is done to the image the attacker generated adversarial images in a significantly faster time which is almost similar to no augmentation. But when a random data augmentation is done to the image the time taken by the attacker to generate adversarial images is slower and this technique decreases the probability of the image from being attacked. The below table describes in detail about the experiments done and the results obtained.

Table 1: Time taken for one-pixel attack to generate adversarial image w.r.t each augmentation.

AUGMENTATION TECHNIQUE	TIME IN SECONDS FOR ONE-PIXEL ATTACK TO GENERATE ADVERSARIAL IMAGE
None	6.327
Image Flip	5.335
Image Rotation	7.315
Random Data Augmentation	29.185


Adversarial Image Classsified as Frog
Confidence – 0.590027

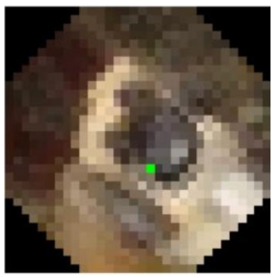


Image Rotation – 45 Degrees
Confidence – 0.894399

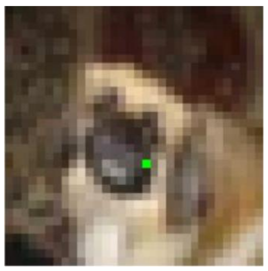

Horizontal Flip
Classified as Dog
Confidence – 0.739278

Figure 6: Illustration of different augmentations resulting in different confidence levels.

In the above images, the first image is the adversarial image which was classified as Frog instead of Dog and performing random data augmentation on the image the results are as in the above images. The confidence levels vary according to the type of augmentation technique which is applied randomly.

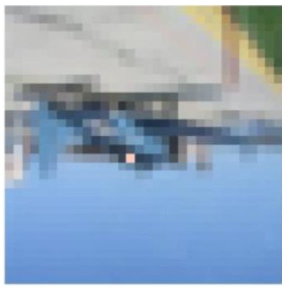


Figure 7: Vertical flip applied on adversarial image

Random data augmentation also helps in increasing the confidence level of the adversarial image, approximately closer to the original image before attack.

Figure 7 is an adversarial image which is classified as airplane with confidence of 0.53698. Although the one-pixel attack on the image was not successful, it was able to bring down the confidence level from 0.999116. Thus when random data augmentation was applied (Vertical Flip, as seen in the image), the confidence level for airplane is gained back to 0.997410.

3.2 Image Denoising

We have added Convolutional Autoencoding layers to our baseline model ResNet, called the Enhanced ResNet (Kindly refer to Figure 3). We have trained these layers in a way, such that they generate denoised and clean images, free from adversarial perturbations. Random perturbations were added to the input images, calculating the loss functions against the original images, in order to train these autoencoding layers accordingly.

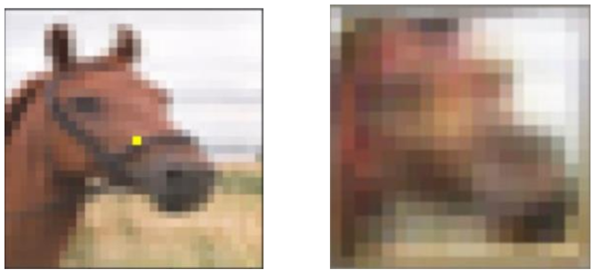
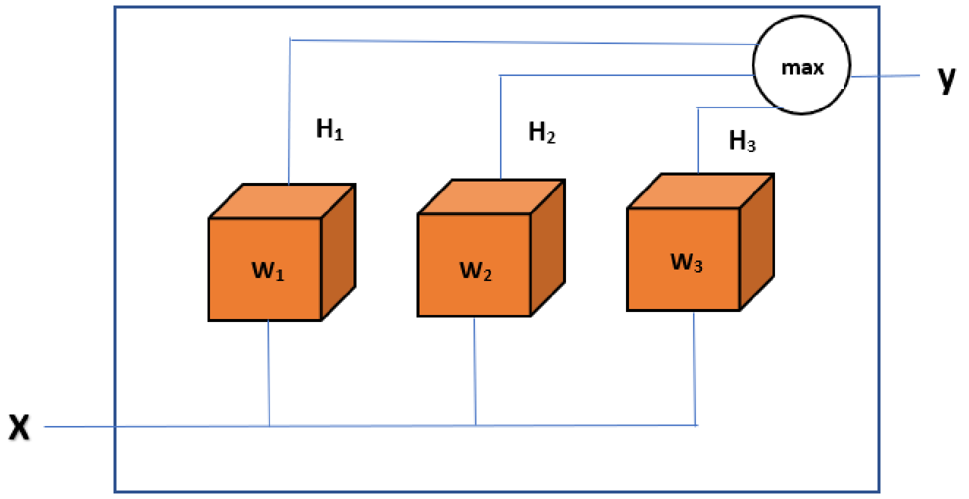


Figure 8: Intermediate output of the Enhanced ResNet designed.

The above Figure 8 shows the intermediate outputs of the newly designed network, which eliminates the one-pixel perturbations and generates clean images. This image is then on fed to the ResNet network which classifies the images accurately. In this method, there will be information loss when denoising is performed, which leads to loss in confidence level of the image or misclassification of the image. This depends on how well the autoencoding layers are being trained.

4 Future Work

We are proposing a new architectural design for neural networks to completely prevent the adversarial attacks. The design is as shown in Figure 10.



$$H_1 = W_1X,\ H_2 = W_2X,\ H_3 = W_3X \ |\quad y = \max(H_1, H_2, H_3)$$

Figure 9: Architecture proposed to defend against adversarial attacks.

The baseline model which is considered is being trained initially thrice, so that it results in three different weight vectors which predict accurate results for the given inputs. The model is laid as shown in the above figure, such that an input X when given to the model, each of the weights generate the same output with varying probabilities or confidence levels. This happens since the weights are different from each other. A max function is then applied to these output probabilities to get the best possible confidence for the output class.

Since majority of the adversarial images implement basic iterative algorithms, the attacker will generate perturbations with respect to a particular weight and does not consider the remaining weights. Say for example if the model with weight W2 is being attacked, the models with weights W1 and W3 are intact and produce the same confidence levels as before. So, the generated adversarial image will decrease the confidence level of the model with weight W2. Since a max function is being applied, the best confidence is still retrieved from either of the models with weights W1 or W3. In the next iteration the attacker performs perturbations with respect to models with weights W1 or W3, during which W2 will give high confidence for the output since the perturbations are made based on W1 or W3 models. This infinite iteration cycle prevents the attacker from generating an adversarial image. We will be experimenting this proposed approach to find how well the designed model performs.

5 Conclusion

We have presented a number of ways to defend Deep Neural Networks (DNN) against one-pixel attacks. We have demonstrated how the random data augmentation can help the network predict the input image accurately using horizontal flip, vertical flip or rotation of the input image to a certain random degree. This decreases the significance of one particular pixel in the adversarial image. The other mechanism demonstrated was, adding a simple deep autoencoder using pytorch as the initial layers to the network. This reduces the noise added by the adversarial image thereby improving the confidence for the input image. Finally, we have also devised a new architecture to eliminate these adversarial attacks. The above-mentioned methodologies help prevent DNNs to be fooled by adversarial attacks.

5.1 Contributions

Pavan Kosaraju

- Worked on the Image denoising technique by adding autoencoding layers to the deep neural network.
- Completely trained and tested the network for confidence on predictions done.
- Devised a new defence mechanism, which is part of the future work section.
- Contributed to report work.

Syed Ikram

- Researched on defence mechanism ideas for one-pixel attack like augmentation and modifying network layers.
- Created real time application to demonstrate one-pixel attack and verified defence techniques against it.
- Contributed to report work.

Nikitha Ravi

- Worked on normal data augmentation technique performing Flip like Horizontal Flip, Vertical Flip and Rotations to an image by certain degrees.
- Compared the effectiveness of the methods and tested the network to check confidence on predictions done.
- Designed the Poster for the Poster presentation.
- Contributed and created the Report to be submitted for the project

Rohith Sooram

- Researched and implemented how the Differential Algorithm works to select a pixel to be attacked.
- Worked on random data augmentation, compared results with normal data augmentation and found time for one-pixel attack convergence.
- Contributed to report work.

References and Citations

[1] Jiawei Su, Danilo Vasconcellos Vargas and Kouichi Sakurai - One pixel attack for fooling deep neural networks arxiv.org/pdf/1710.08864.pdf

[2] Alexey Kurakin, Ian J. Goodfellow and Samy Bengio - ADVERSARIAL EXAMPLES IN THE PHYSICAL WORLD arxiv.org/pdf/1607.02533.pdf

[3] One-Pixel attack in Keras - https://github.com/Hyperparticle/one-pixel-attack-keras

[4] Kind-PyTorch turotial: Autoencoder model https://github.com/GunhoChoi/Kind-PyTorch-Tutorial/tree/master/06_Autoencoder_Model_Save

[5] Proper ResNet Implementation for CIFAR10/CIFAR100 in pytorch https://github.com/akamaster/pytorch_resnet_cifar10