Adversarial Attacks and Defense Techniques for Convolutional Neural Networks

Deepank Dixit
Department of Computer Science and Automation
deepankdixit@iisc.ac.in | SR : 04-03-06-19-52-21-1-20269

Sindhuja Rao
Department of Electrical Communication Engineering
sindhujarao@iisc.ac.in | SR : 04-02-06-19-52-21-1-20283

Abstract

Convolutional neural networks (CNNs) have been widely successful at predicting and processing pixel data for image classifications. However, with the rise in adversarial attacks in more complex models, CNNs are vulnerable to many adversarial attacks. One of the methods is using adversarial examples to be introduced in the training set to create small perturbations which seem to be imperceptible to humans but cause impactful loss to a model's accuracy. The report includes the development of a few such attack methods to generate adversarial examples and the construction of defense techniques to guard against such examples. Attack method covered in this project are Fast-Gradient Signed Method (FGSM), iterative FGSM or Projected Gradient Descent (PGD) and to show the data model is vulnerable, we use One-Pixel Attack. To guard our Deep Neural Network (CNN) against the FGSM attack, we use Adversarial retraining with the same on MNIST dataset. This exposes a vulnerability of the neural network, which, if exploited can cause disruptive results for many real-world use-cases of DNNs. Hence understanding security in such models and developing defense strategies for adversarial attacks is crucial.

1 Problem Statement

Since its inception, Machine Learning has been used to create models to solve real-world problems. Whether working with identifying objects in self-driving autonomous vehicles, performing image analysis to identify cancerous cells, or preserving historical documents. Since humans are more perceptive of visual patterns, CNNs help with processing image data with higher accuracy to solve complex problems. Such use-cases are heavily dependent on human activity, and therefore the misclassification caused by CNNs could have critical consequences. For instance - misclassifying objects in self-driving vehicles can be dangerous, misclassifying a malignant cancer tumor as benign can be life-threatening and security risks in Biometric systems when criminals use identity theft risk data privacy. Hence, preparing and deploying a defense mechanism is essential to make the network impervious to specific attacks.

2 Literature Survey/Background

Adversarial attacks for DNNs have become more impactful in reducing confidence in models through the years. Hence, understanding security for models depends on factors like intent of an adversary and his knowledge about the model [2]. This leads to classifying adversarial attacks based on these factors, for example, White-box and black-box or evasion and poisoning or Single-step or iterative or greedy. We have considered our CNNs as a Victim Model to mimic an adversary's attempt to create perturbations and introduce adversarial examples. And, adversarial examples have been of interest as Goodfellow et al. 2015 [1] deeply discussed how Neural Networks perform under adversarial attacks and how even complex DNNs are vulnerable to the smallest perturbations. One such classification of attacks in [3] is based on which parameters of the model are targeted: Gradient, Score or Decision. Fast Gradient Sign Method is a Gradient-based attack that finds a gradient value for which Loss functions for the model are maximized. This works significantly well to depict confidence level changes in datasets such as MNIST dataset that is used in the project. And defenses include retraining the model with adversarial examples so that model learns from a misclassified dataset and overcomes the attack [8].

Additionally, adversarial attacks on different data sets have caught attention in recent years as the use-cases for DNNs have been evolving. Interesting use-case such as facial recognition was discussed in detail by Yigit Alparslan et al. 2020 [7] which involves using untargeted methods to introduce adversarial examples when the adversary does not target any specific class which the model should misclassify datasets as. Similar to this work, we introduce a study of vulnerability of our model when introduced with an untargeted one-pixel attack [9].

3 Current Method Description

3.1 Fast Gradient Sign Method

Fast Gradient Sign Method (FGSM) is a simple yet effective adversarial attack for machine learning models. Goodfellow et al. (2015) proposed generating adversarial examples with FGSM attack and adding them back to the training data set to improve the robustness of the model [1].

Compared to its iterative variants such as Basic Iterative Method (BIM) and Projected Gradient Descent (PGD) attacks, FGSM is faster in generating adversarial examples, because it only involves calculating one back-propagation step. In one part of this project, we use FGSM to produce 60,000 adversarial images in a small 45 seconds duration.

Since an attacker requires gradient information to perform this attack, this is categorized as a White-box attack. The algorithm for FGSM takes one small step in the direction of the negative gradient of the Loss function L [3]. The step-size δ is small enough so that the $||\delta||_\infty \leq \epsilon$, where ϵ is a small ball around the input x.

Loss function measures the difference between the predicted value and the label given in the training set. Using FGSM, we create an adversarial example by perturbing the input feature by a small quantity δ such that the loss L $(\theta,\ x_0 + \delta,\ y_0)$ value increases, causing the model accuracy to go down.

Our adversarial example is defined as

$$x^{adv} = x_0 + \delta \tag{1}$$

with

$$\delta = \arg\max_{\delta \in S}\ L(\theta, x + \delta, y) \tag{1.1}$$

where, L is the loss function, δ is the adversarial perturbation, and S $\in$ R^d is the set of allowed perturbations to the input. S is usually defined as S = $\{\delta : ||\delta|| \leq \epsilon\}$.

Now we use FGSM to write the attack equation as follows -

$$x^{adv} = x + \alpha \cdot sign(\nabla_x L(\theta, x, y)) \tag{3}$$

Comparing (1) with (3), notice that the perturbation $\delta = \alpha \cdot sign(\nabla_x L(\theta, x, y))$. Due to the constraint $||\delta||_\infty \leq \epsilon$, value of δ can be clipped to lie within the range -ϵ to ϵ, i.e.

$$\delta = clip(\alpha \cdot sign(\nabla_x L(\theta, x, y)), [-\epsilon, \epsilon])$$

Hence, δ can be succinctly written as -

$$\delta = \epsilon \cdot sign(\nabla_x L(\theta, x, y))$$

where x is the original input, x^{adv} is the updated input, ϵ is a small quantity that controls the magnitude of perturbation in each image, and $\nabla_x L(\theta, x, y)$ represents the gradient of classification loss L($\cdot$) w.r.t. input x.

3.2 One Pixel Attack

When running models under random noise to create attacks, it is seen that there is a high rate of misclassification. In DNNs recently, instead of using random noise over image data, making changes to a single pixel is also seen to be reducing confidence of the machines.
Similar to what we have discussed in section 3.1, creating adversarial images can be thought as an optimization problem. The input images can be represented as a collection of scalars of pixels in vector form. For an image classifier f receiving n-dimensional inputs of original images, x = (x1, .., xn) which can classify them as t. We now have another vector, vector e(x) = (e1, .., en), which is the collection of perturbed images after modifying x. The goal is to map the test images to an adversarial class adv instead of t. This is seen as an optimization problem as below:

$$\underbrace{maximize}_{e(x)^*} \quad f_{adv}(x + e\ \ (x))\ ,\ subject\ to\, e(x) \leq D$$

Here, D is the permissible modification. Based on this concept, we take D = 1 to create the one-pixel attack [11].

4 Defense using Adversarial Training

Adversarial training is a technique in which new adversarial examples are continually introduced throughout the training process. In the original development of adversarial training, batch normalization was not used for smaller models; however, it is recommended when scaling the adversarial training to large datasets. Training examples can be grouped into a pre-defined batch size. Each batch contains benign and adversarial examples and is iterated with each training step.

The number and weight of adversarial examples in each batch are controlled independently using a loss function[10].

5 Project Implementation

5.1 Obtain MNIST image dataset, Do the Preprocessing

The project's first step is to fetch and preprocess the MNIST dataset of 1 X 28 X 28 greyscale images of handwritten digits between 0 and 9. We do this by preparing a training dataset of 50,000 and test dataset of 10,000 examples. Once the dataset is collected, we create PyTorch Dataloaders to divide our training examples into batches before beginning to train the model.

5.2 Train CNN for image classification on the MNIST dataset images

We now create the entire model in PyTorch and develop the training algorithm for the same. We Create MnistCNN class with 2 convolutional layers and 2 linear layers. We use nn.Sequential() container through which several modules are added for layering and activation purposes. We define the loss and forward functions, get the predictions using the model to pass input images, compute the loss, and iteratively minimize it by adjusting the hyper-parameters. At this stage, our CNN can predict the label for images with reasonably good accuracy.

5.3 Generate adversarial examples using gradient-based FGSM attack method

We then implement our first attack by generating adversarial examples using FGSM, as discussed in section 3.1. Such adversarial examples cause the model to misclassify our inputs.

5.4 Adversarial Training as a defense against the FGSM attack

We use the adversarial examples to re-train our model. The input for our model now becomes: $x + \delta$, and it is fed back into the network, followed by loss computation and adjusting of the parameters using backpropagation.

```
Algorithm 1: FGSM Attack - Summary
  1. Implement FGSM attack that returns the perturbed images using loss-gradient and a fixed
  epsilon value
  2. Get predictions on the adversarial data loader and test data loader, and compute model
  accuracy
  3. Retrain the model on original dataset along with the generated adversarial dataset
  4. Compute accuracy after adversarial training
```

5.5 Create adversarial examples for one-pixel Attack

```
Algorithm 2: One pixel attack - Summary
  1. Copy the test data and create perturbations by fixing number of pixels = 1
  2. Iterate over 28 x 28 pixel area for each image, set pixel value =1 to create white pixel
  3. Compare each image prediction with original prediction after attack
  4. Calculate final number of vulnerable images and compute accuracy
```

5.6 Network Architecture, Parameters, and Code References

5.6.1 Model

(i) nn.Sequential() : conv1 -> ReLu -> Pool1 -> conv2 -> ReLu -> Pool2 -> linear1 -> linear2 -> LogSoftMax
(ii)*Loss Function:* nn.NLLLoss()
(iii)*Model Hyper-parameters:*
lr= $1e^{-1}$: time taken=38s, Train Accuracy:10.44%, Test Accuracy:10.28%, Loss value very high
lr= $1e^{-2}$: time taken=40s, Train Accuracy:98.59%, Test Accuracy:98.59%, Loss Value low
lr= $1e^{-3}$: time taken=40s, Train Accuracy:96.13%, Test Accuracy:96.27%, Loss Value medium high

5.6.2 FGSM:

(i) ϵ = 0.7 to demonstrate the drastic change in accuracy.
(ii) def fgsm_attack(image, epsilon, loss_gradient) :
https://pytorch.org/tutorials/beginner/fgsm_tutorial.html#fgsm-attack
(iii) def plot_images(data,target,yp,M,N): https://adversarial-ml-tutorial.org/adversarial_examples/

5.6.3 One-Pixel:

Functions referred from: https://bytepawn.com/mnist-pixel-attacks-with-pytorch.html

6 Results and Observations

Using these methods for adversarial attacks, we observed that CNN image classifiers are vulnerable to different attacks and the accuracy is impacted differently.

Changing the value of epsilon while generating adversarial examples for FGSM attack, we see change in the confidence of the model vary. Here we compile the accuracy of model on test dataset for different values of epsilon -

Epsilon	Accuracies (%)			
	Test Data Original	Adv Data	Adv Data post Adv Training	Test Data post Adv Training
0	98.59	98.14	98.18	98.29
0.2	-	97.11	98.18	98.37
0.4	-	95.33	96.78	98.16
0.7	-	57.96	94.47	93.64
0.9	-	23.74	95.59	86.48

Hence it is easy to see that models can be easily disrupted with change in epsilon value. The epsilon should be small enough to fool the model into misclassifying an image that is almost indistinguishable from the non-perturbed image.

An adversarial image generated via the FGSM method can be seen as below:

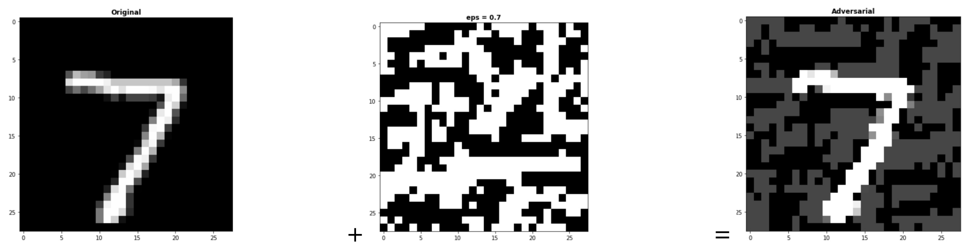


Figure 1: Original Image + Noise (ϵ = 0.7) = Adversarial Image

We can also see some examples from 1st batch of adversarial images before and after adversarial training. It demonstrates a clear improve in accuracy in predicting the adversarial images correctly as earlier pointed out through the tabular data.

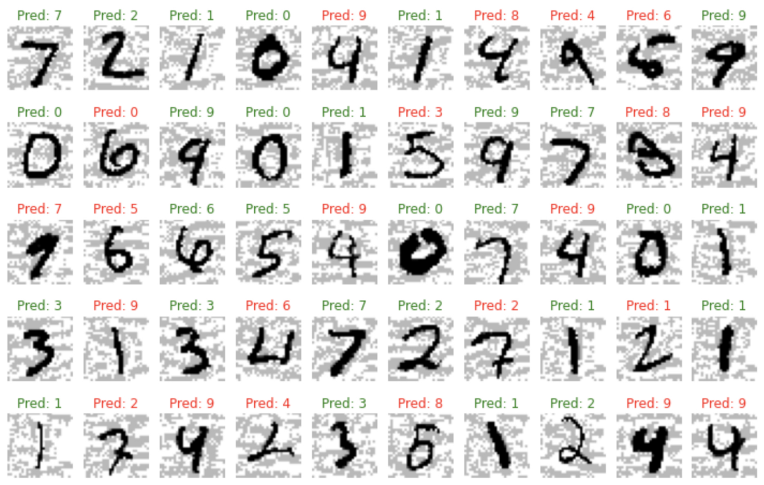

Pred: 7 Pred: 2 Pred: 1 Pred: 0 Pred: 9 Pred: 1 Pred: 8 Pred: 4 Pred: 6 Pred: 9
Pred: 0 Pred: 0 Pred: 9 Pred: 0 Pred: 1 Pred: 3 Pred: 9 Pred: 7 Pred: 8 Pred: 9
Pred: 7 Pred: 5 Pred: 6 Pred: 5 Pred: 9 Pred: 0 Pred: 7 Pred: 9 Pred: 0 Pred: 1
Pred: 3 Pred: 9 Pred: 3 Pred: 6 Pred: 7 Pred: 2 Pred: 2 Pred: 1 Pred: 1 Pred: 1
Pred: 1 Pred: 2 Pred: 9 Pred: 4 Pred: 3 Pred: 8 Pred: 1 Pred: 2 Pred: 9 Pred: 9

Figure 2: Demo : Adversarial Misclassifications after FGSM Attack, before Adversarial Training

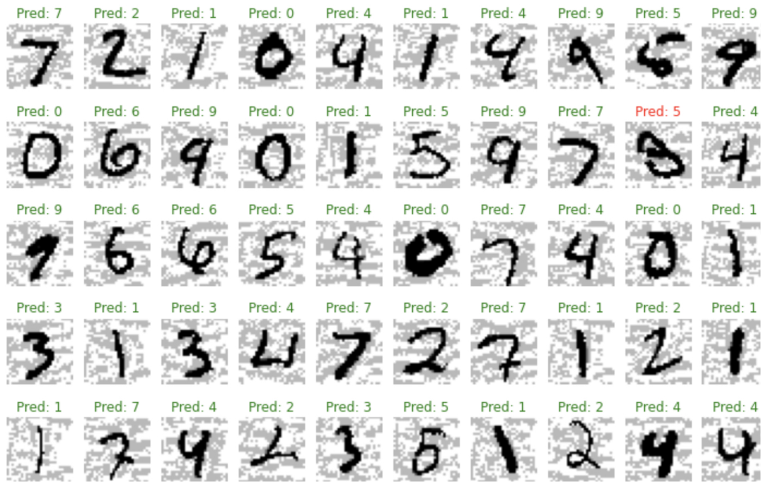
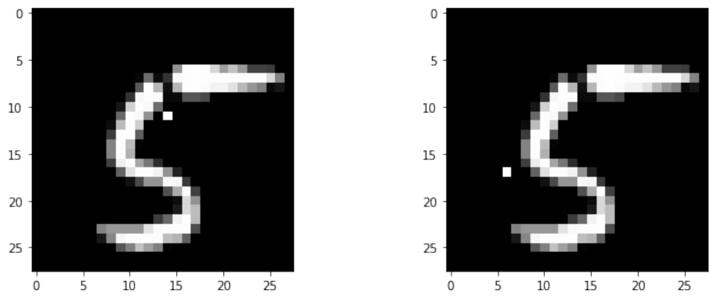


Figure 3: Demo : Adversarial Misclassifications after Adversarial Training

As evident from Figure 3, the accuracy is recovered to a great extent after completing the Adversarial Training.

Moving on to the one pixel attack - there are certain images that we find to be more vulnerable than others. Since this will be a visible difference to human-eye, we fix the number of pixels to be attacked to 1, and recreate the attack.

Here, instead of creating random noise, we iterate over all pixel positions to see how many images are misclassified. This gives us an idea about the model's vulnerability. We have found few images to be more vulnerable than others. For example, the below image of Digit 5 is always misclassified, irrespective of pixel position:



(12, 12) vulnerables:	{6273, 8577, 901, 8582, 8589, 9103, 9871, 9492, 8989, 4001, 1059, 295, 1194, 3500, 4269,
	6700, 1080, 4024, 5946, 1342, 5439, 5312, 7488, 9284, 8395, 3792, 8400, 9042, 3541, 3674, 2778, 474, 4060, 8797, 4324, 5476, 7653, 5355, 4590, 4976, 8434, 8051, 2165, 1912, 6268, 7550}
	–>6273 is 4, classified as 8
	–>8577 is 4, classified as 8
	–>901 is 4, classified as 1
	–>8582 is 4, classified as 8
	–>8589 is 4, classified as 8

For our model with accuracy with 98.59%, we have found total vulnerabilities of 177 images. Hence, We have found around 49 images that are actually vulnerable to one-pixel attack.

And we find the accuracy of the model for MNIST dataset on one-pixel attack is less drastic as compared to the FGSM attack. This is seen to be expected as the MNIST data used are grayscale images hence higher pixels changes or edge cases of pixel attacks will be more visible. We can assume that colored images are more vulnerable to one pixel attacks than grayscale images.

7 Challenges faced during Implementation

(i) Understanding how the PyTorch works, how to store and reuse prediction and image tensors
(ii) Multiple iterations for different values of epsilon during the FGSM attack to see which has the most noticeable adversarial effect.
(iii) Reusing DataLoaders for multiple iterations, and training the model on multiple DataLoaders simultaneously
(iv) Adversarial training kept failing due to data.requires_grad = True for adversarial data. Took long time to debug and fix

References

[1] Ian J. Goodfellow, Jonathon Shlens & Christian Szegedy *Explaining and Harnessing Adversarial Examples* In: arXiv e-prints, arXiv:1412.6572 (Dec. 2014), arXiv: 1412.6572 [stat.ML] https://doi.org/10.48550/arXiv.1909.08072

[2] Han Xu, Yao Ma, Haochen Liu, Debayan Deb, Hui Liu, Jiliang Tang, & Anil K. Jain *Adversarial Attacks and Defenses in Images, Graphs and Text: A Review* In: arXiv e-prints, arXiv:1909.08072 (Sep 2019), arXiv:1909.08072 [cs.LG] https://doi.org/10.48550/arXiv.1909.08072.

[3] Yao Li, Minhao Cheng, Cho-Jui Hsieh & Thomas C. M. Lee *A Review of Adversarial Attack and Defense for Classification Methods* In: arXiv e-prints, arXiv:2111.09961 (Nov 2021), arXiv:2111.09961 [cs.CR], https://doi.org/10.48550/arXiv.2111.09961

[4] Wieland Brendel, Jonas Rauber & Matthias Bethge *Decision-Based Adversarial Attacks: Reliable Attacks Against Black-Box Machine Learning Models* In: arXiv e-prints, arXiv:1712.04248 (Dec 2017), arXiv:1712.04248 [stat.ML], https://doi.org/10.48550/arXiv.1712.04248

[5] Jing Lin, Long Dang, Mohamed Rahouti & Kaiqi Xiong *ML Attack Models: Adversarial Attacks and Data Poisoning Attacks* In: arXiv e-prints, arXiv:2112.02797 (Dec 2021), arXiv:2112.02797 [cs.LG], https://doi.org/10.48550/arXiv.2112.02797

[6] Ali Shafahi, W. Ronny Huang, Mahyar Najibi, Octavian Suciu, Christoph Studer, Tudor Dumitras & Tom Goldstein *Poison Frogs! Targeted Clean-Label Poisoning Attacks on Neural Networks* In: arXiv e-prints, arXiv:1804.00792 (Apr 2018), arXiv:1804.00792 [cs.LG], https://doi.org/10.48550/arXiv.1804.00792

[7] Yigit Alparslan, Ken Alparslan, Jeremy Keim-Shenk, Shweta Khade & Rachel Greenstadt *Adversarial Attacks on Convolutional Neural Networks in Facial Recognition Domain* In: arXiv e-prints, arXiv:2001.11137 (Jan 2020), arXiv:2001.11137 [cs.LG], https://doi.org/10.48550/arXiv.2001.11137

[8] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras & Adrian Vladu *Towards Deep Learning Models Resistant to Adversarial Attacks* In: arXiv e-prints, arXiv:1706.06083 (June 2017), arXiv:1706.06083 [stat.ML], https://doi.org/10.48550/arXiv.1706.06083

[9] Jiawei Su, Danilo Vasconcellos Vargas & Sakurai Kouichi *One pixel attack for fooling deep neural networks* In: arXiv e-prints, arXiv:1710.08864 (Oct 2017), arXiv:1710.08864 [cs.LG], https://doi.org/10.48550/arXiv.1710.08864

[10] Alexey Kurakin, Ian Goodfellow & Samy Bengio *Adversarial Machine Learning at Scale* In: arXiv e-prints, arXiv:1611.01236 (Nov 2016) , arXiv:1611.01236 [cs.CV], https://doi.org/10.48550/arXiv.1611.01236

[11] One-pixel code reference: https://bytepawn.com/mnist-pixel-attacks-with-pytorch.html