CS 725 PROJECT REPORT

TITLE : Adversarial Attacks and Defense using Fast Sign Gradient Method and Black Box Technique

Team Members:

1. Varad Bhatnagar (19305R007)

2. Prince Kumar (193050070)

INDEX

MOTIVATION

Neural Networks are being increasingly used to automate detection and recognition tasks in a variety of fields such as Supply Chain Management, Law Enforcement, Camera Software and Biometrics. Excellent results have been obtained in applications such as handwriting detection, traffic light detection and object detection using various types of Neural Nets.

Neural Nets are the core ‘intelligence’ behind Self Driving Cars which are being tested worldwide in artificial environments before they can be brought on road. In such a situation, the failure of this network will have far reaching consequences and may lead to loss of life and property.

In recent years, it has come to light that well trained and highly accurate Neural Networks are failing in some specific situations. Not only failing, they are often found taking a wrong decision with high confidence. The data points on which this phenomenon occurs are called ‘Adversarial Examples’.

These Adversarial Examples are mostly formed by introducing very small perturbations to a training sample. Most often such perturbations are so small that they can’t be detected by the naked eye i.e. human’s can’t distinguish between a normal sample and closely related Adversarial Sample but Neural Networks often commit errors. Such a failure can not be tolerated in some sensitive fields such as Law Enforcement and Self Driving Cars.

A lot of research is going in this area. Many ideas such as Adversarial Training of Neural Networks have been proposed and many methods to generate such adversarial examples have been proposed. All of this has lead to new architectures such as Generative Adversarial Networks (GANs) which have application in detecting Fake Images and Videos also called Deep Fake Videos.

GOAL

The goal of this project is to demonstrate :

(i) A well trained and highly accurate neural network fails on adversarial samples.

(ii) A neural network which has been trained on adversarial data, performs well on new adversarial samples.

(iii) How to generate adversarial samples :
    (a) When attacker knows the network architecture.
    (b) When the attacked doesn't know anything about the network architecture.

(iv) Working of Fast Sign Gradient Method and Black Box technique.

LITERATURE SURVEY

We studied the following papers before implementing the project:

1. Explaining and Harnessing Adversarial Examples by Ian J. Goodfellow, Jonathon Shlens, Christian Szegedy

    Key Ideas :

    Existence of Adversarial Examples can be explained by Linear behaviour in high dimensional spaces and is not because of extreme non-linearity of Deep Neural Nets.

    Existence of Adversarial Examples is not due to problems in data but due to problems in model.

    The theory and mathematics behind the Fast Sign Gradient Method and how it is a cheap and efficient way of producing Adversarial Examples

2. Practical Black-Box Attacks against Machine Learning by Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z. Berkay Celik, Ananthram Swami

    Key Ideas:

    Accessing labels O* produced by the Deep Neural Net O is the only capability assumed in the threat model.

    The attacker queries the oracle(target model) O with synthetic inputs selected by Jacobian-based heuristic to build a model F approximating the oracle O's decision boundaries.

    The attacker uses substitute network F to craft adversarial samples , which are misclassified by the oracle O due to the transferability of the adversarial samples.

APPROACH

1. Fully connected NN based approach

Our first goal was to train a network on the EMNIST dataset with good accuracy. So, we wrote a wrote and trained a fully connected neural network from scratch on the EMNIST dataset and got an accuracy just over 90% on the test set, limited to our system specifications.

We thought about performing the same task using a CNN as it is better suited for image processing tasks due to its properties of translational invariance and locality.
Also, we were facing compatibility issues in using this together with the Fast Sign Gradient Method in the Cleverhans library because there were several arguments required that were instances of Cleverhans internal classes. As the FSGM depends on the model architecture, the model object is a required parameter in the initialization of this method.

We faced problems in converting our custom model to the class instance and so we abandoned this approach and went ahead with CNN in the Model Zoo of Cleverhans.

2. CNN Based Approach

2.1 When model architecture is known to attacker:

The aim was to demonstrate adversarial attacks and defence on a Deep Neural Network. We broke down the task to be performed into 7 independent subtasks:

1. Train a model using normal training dataset. We will call this the ‘Clean Model’.

2. Test the ‘Clean Model’ on a test dataset to ascertain accuracy. Repeat task 1 with adjusted hyperparameters if the accuracy is not found to be sufficient.

3. Test the ‘Clean Model’ on adversarial examples generated by Fast Sign Gradient Method.

4. Train a model using a mixture of normal and adversarial training dataset. We will call this the ‘Adversarial Model’.

5. Test the ‘Adversarial Model’ on a test dataset to ascertain accuracy.

6. Test the ‘Adversarial Model’ on adversarial examples generated by Fast Sign Gradient Method.

7. Report all the results to draw a conclusion.

2.2 When model architecture is not known to attacker:

The aim was to demonstrate adversarial attacks and defence on a Deep Neural Network. We broke down the task to be performed into 5 independent subtasks:

Initial Collection (1) : Collect a very small set S_0 of inputs representative of the input domain.

Architecture Selection (2) : Select an architecture to be trained as the substitute F. Again, this can be done using high-level knowledge of the classification task performed by the oracle.

Substitute Training : Iteratively train more accurate substitute DNNs F_ρ by repeating the following for $\rho \in 0..\rho_{max}$

- Labeling (3) : By querying for the labels $\tilde{O}(\tilde{x})$ output by oracle O, label each sample $\tilde{x} \in S_\rho$ in its initial substitute training set S_ρ.
- Training (4) : Train the architecture chosen at step (2) using substitute training set S_ρ in conjunction with classical training techniques.
- Augmentation (5): Apply the augmentation technique on the initial substitute training set S_ρ to produce a larger substitute training set $S_{\rho+1}$ with more synthetic training points. Repeat steps(3) and (4) with the augmented set $S_{\rho+1}$.

Report all the results to draw a conclusion.

DATASET(s) and MODEL ARCHITECTURE

We used the following datasets :

1. MNIST – This is a well known dataset of images of handwritten digits from 0 to 9, number of output classes being 10.All images are of size 28*28 pixels and grayscale i.e. pixels can take a value between 0 and 255. There are 70,000 samples in this dataset.

2. EMNIST – The extended version of MNIST. This dataset contains images of handwritten digits and characters from a-z, A-Z, 0-9. The number of output classes are 47 as some classes for some characters which have similar lowercase and uppercase representations (j, i, o, z, etc) have been combined. All images are of size 28*28 pixels and grayscale i.e. pixels can take a value between 0 and 255. There are more than 240,000 samples in this dataset.

We have used a Convolutional Neural Network from the ModelBasicCNN class Cleverhans Model Zoo.

TECHNICAL SPECIFICATIONS

Initially we were using our own laptops to train the model but it was taking a lot of time as the computations were taking a lot of time due to RAM limitations. Then we shifted to Google Colab on which we used a free instance with 12 GB of RAM and a GPU. Training speed grew manifold after this.

Language : Python

Platform : Google Colab

Libraries : Numpy, Tensorflow, Cleverhans, matplotlib
functools, logging

DIVISION OF WORK

We divided the tasks into two parts based on the base papers. Both team members went through both the papers, then discussed the ideas and brainstorming various implementation techniques.

Then we went our own ways. Varad worked on the ‘Explaining and Harnessing Adversarial Examples’ paper and Prince worked on the ‘Practical Black Box attacks against Machine Learning’ paper.

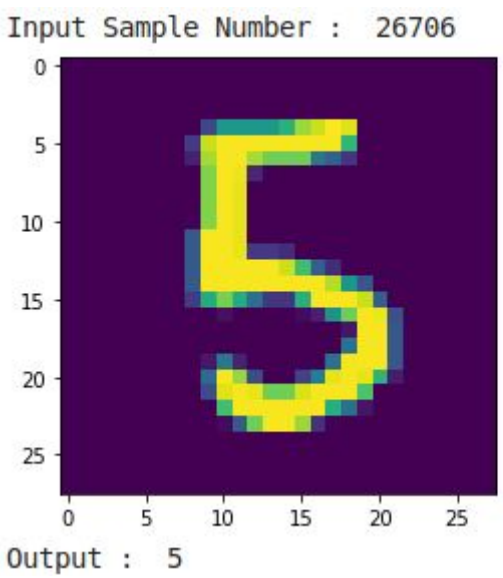
We implemented both the papers and then went through each other’s code and logic to make sure that there are no errors.

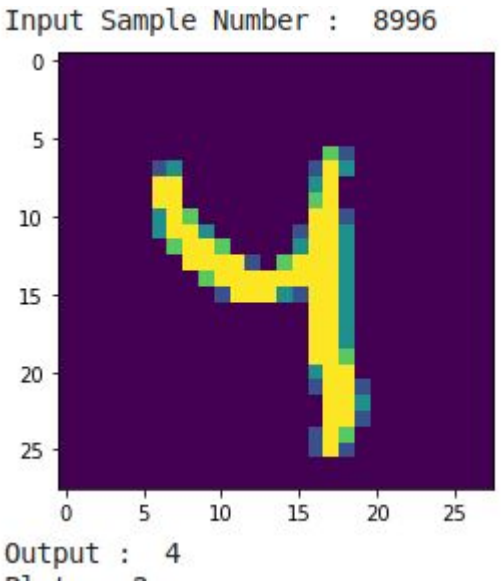
After that, we created the visualization using matplotlib and reports and the presentation together.

From the literature survey till the final demo, the project was completed in about 3 weeks.

RESULTS

Model Results on Test Examples





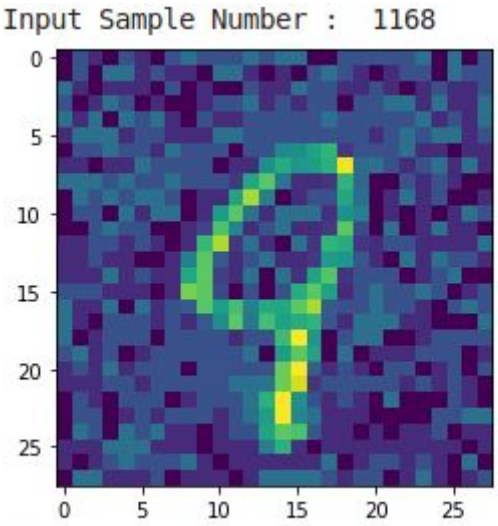
Accuracy in Black Box Method :

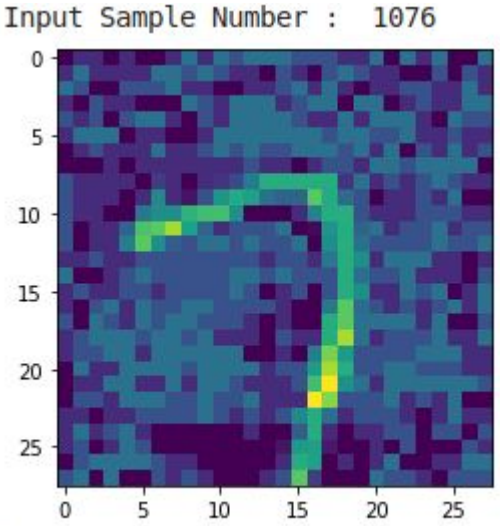
```
{'black_box': 0.992994923857868, 'sub_model': 0.7586802030456853, 'bbox_on_sub_adv_ex': 0.7130964467005076}
```

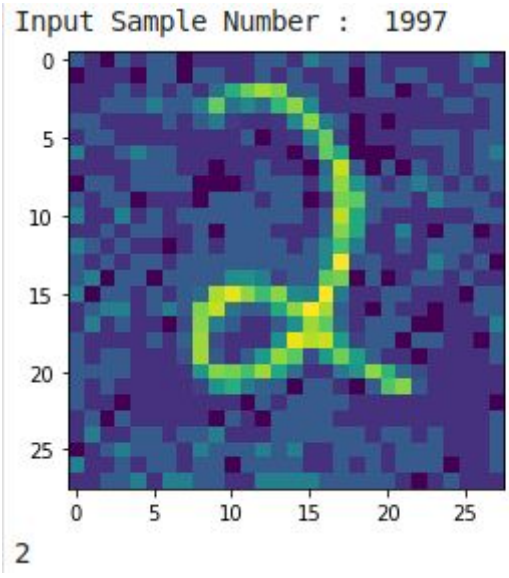
Accuracy in FSGM Method:

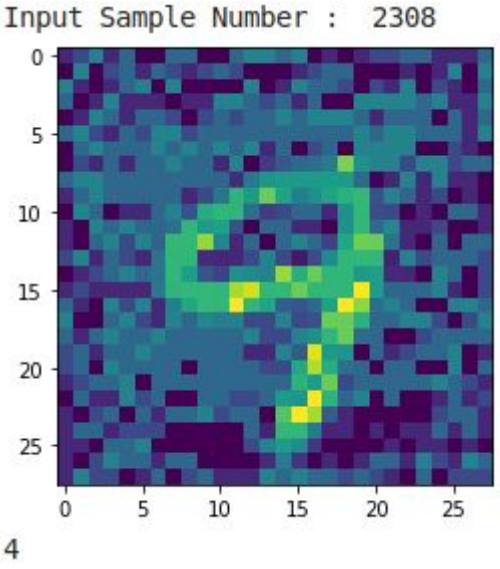
```
[INFO 2019-11-24 14:46:56,165 cleverhans] Epoch 0 took 7.0896759033203125 seconds
Test accuracy on legitimate examples: 0.9883
[INFO 2019-11-24 14:47:02,503 cleverhans] Epoch 1 took 5.879048109054565 seconds
Test accuracy on legitimate examples: 0.9915
[INFO 2019-11-24 14:47:08,898 cleverhans] Epoch 2 took 6.0036187171936035 seconds
Test accuracy on legitimate examples: 0.9922
[INFO 2019-11-24 14:47:15,285 cleverhans] Epoch 3 took 5.978717565536499 seconds
Test accuracy on legitimate examples: 0.9928
[INFO 2019-11-24 14:47:21,602 cleverhans] Epoch 4 took 5.914616823196411 seconds
Test accuracy on legitimate examples: 0.9922
[INFO 2019-11-24 14:47:27,903 cleverhans] Epoch 5 took 5.903960943222046 seconds
Test accuracy on legitimate examples: 0.9929
Test accuracy on adversarial examples: 0.1157
Repeating the process, using adversarial training
Repeating the process, using adversarial training
num_devices:  1
[INFO 2019-11-24 14:47:54,710 cleverhans] Epoch 0 took 15.146755695343018 seconds
Test accuracy on legitimate examples: 0.9833
Test accuracy on adversarial examples: 0.8465
[INFO 2019-11-24 14:48:11,289 cleverhans] Epoch 1 took 14.89499807357788 seconds
Test accuracy on legitimate examples: 0.9833
Test accuracy on adversarial examples: 0.8585
[INFO 2019-11-24 14:48:27,644 cleverhans] Epoch 2 took 14.86587905883789 seconds
Test accuracy on legitimate examples: 0.9889
Test accuracy on adversarial examples: 0.8882
[INFO 2019-11-24 14:48:43,943 cleverhans] Epoch 3 took 14.786616325378418 seconds
Test accuracy on legitimate examples: 0.9884
Test accuracy on adversarial examples: 0.8897
[INFO 2019-11-24 14:49:00,248 cleverhans] Epoch 4 took 14.81812572479248 seconds
Test accuracy on legitimate examples: 0.9911
Test accuracy on adversarial examples: 0.9189
[INFO 2019-11-24 14:49:16,489 cleverhans] Epoch 5 took 14.749626159667969 seconds
Test accuracy on legitimate examples: 0.9902
Test accuracy on adversarial examples: 0.9326
```

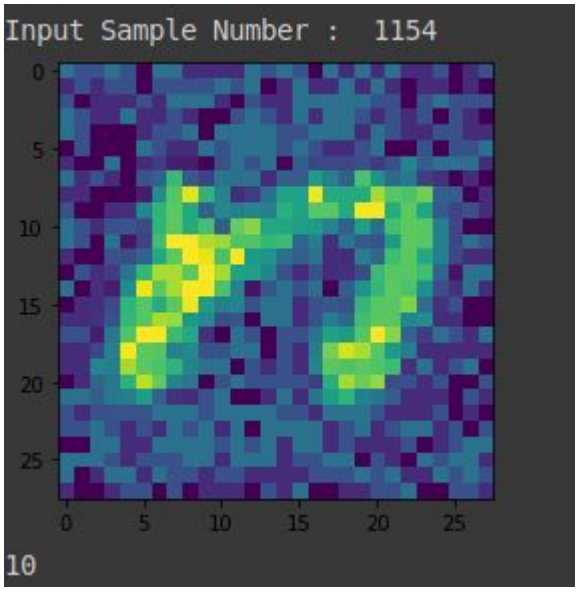
Model Results on Adversarial Examples

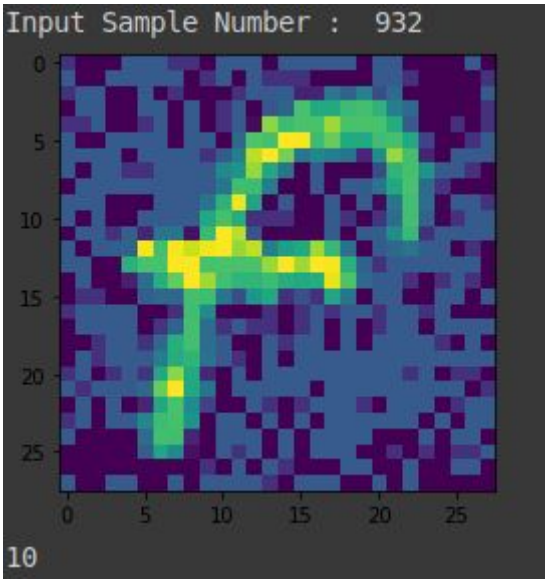


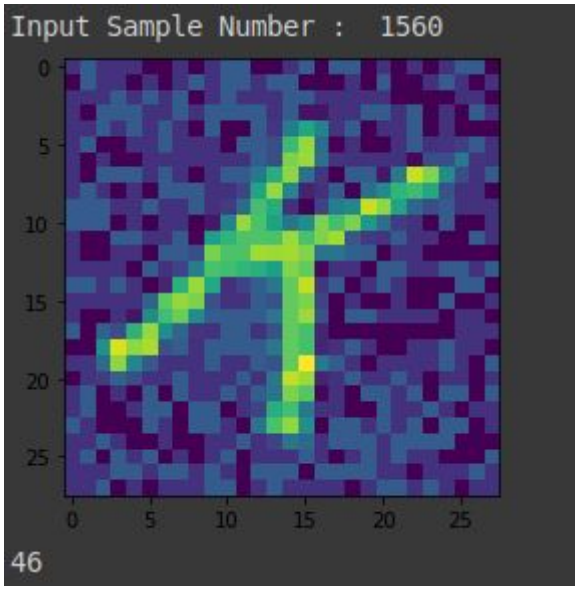


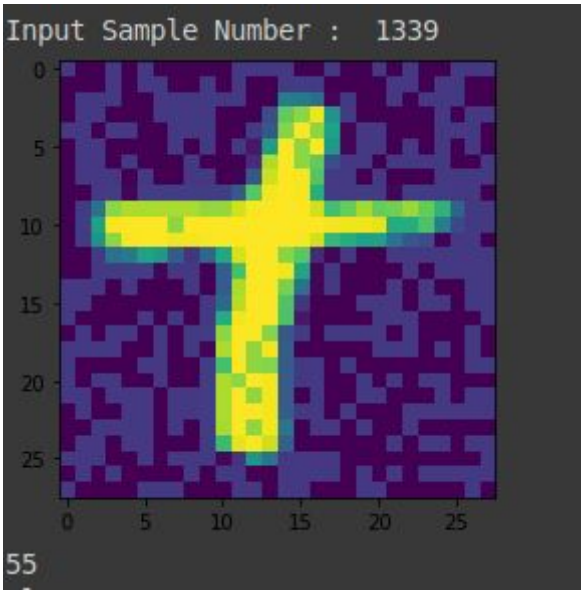





Input Sample Number : 1154
0
5
10
15
20
25
0
5
10
15
20
25
10


Input Sample Number : 932
0
5
10
15
20
25
0
5
10
15
20
25
10


Input Sample Number : 1560
0
5
10
15
20
25
0
5
10
15
20
25
46


Input Sample Number : 1339
0
5
10
15
20
25
0
5
10
15
20
25
55

CODE

Final code can be viewed at :
https://github.com/varadhbhatnagar/Adversarial-CNN-using-FSGM-and-BlackBox

There are four files in total:
(i) bb-mnist.py – This file contains the implementation of the Black Box technique on MNIST.
(ii) bb-emnist.py – This file contains the implementation of the Black Box technique on EMNIST.
(iii) fsgm-mnist.py – This file contains the implementation of the FSGM Technique on MNIST.
(iv) fsgm-emnist.py – This file contains the implementation of the FSGM Technique on EMNIST

Google Colab was used to train the CNN. Link to colab environment :
https://drive.google.com/drive/folders/1OhasLF2kWzlCVOSJBkOsyYW-qiGTk99g
As we were using python libraries such as Numpy and Cleverhans, many functions were already implemented and we just had to use the API calls.

Total lines of Submitted Code ~ 500.
Total lines of Code (Including Unsuccessful Approach) ~ 800

CONCLUSION

We explored the theory behind existence of adversarial examples and their generation using the Fast Sign Gradient Method and Black Box Techniques successfully. These two methods correspond to the two scenarios when :

(i) The attacker knows about the underlying architecture of the Neural Net.
(ii) The attacker doesn't know about the underlying architecture of the Neural Net.

We also demonstrated the impact of such adversarial examples on well trained and highly accurate Neural Network and how easily they get fooled. Not only do they predict the wrong class, they do so with a high degree of confidence.

Through the two methods mentioned above, we demonstrated how to defend against such attacks using Adversarial Training i.e. the training set should include a mixture of Normal and Adversarial Samples.

Thus, we were able to successfully demonstrate the behaviour of a Convolutional Neural Network trained on the MNIST and then EMNIST dataset in presence of Adversarial Examples and also demonstrate a method to defend them using Adversarial Training using the FSGM and Black Box methods.

REFERENCES

Blogs:
https://www.kaggle.com/allunia/how-to-attack-a-machine-learning-model
https://www.kaggle.com/benhamner/adversarial-learning-challenges-getting-started

Datasets:
https://www.nist.gov/node/1298471/emnist-dataset
http://yann.lecun.com/exdb/mnist/

Libraries:
https://github.com/tensorflow/cleverhans
https://github.com/bethgelab/foolbox
https://adversarial-robustness-toolbox.readthedocs.io/en/latest/

Papers:

'Explaining and Harnessing Adversarial Examples' by Ian J. Goodfellow, Jonathon Shlens, Christian Szegedy (https://arxiv.org/abs/1511.07528)

'Practical Black-Box Attacks against Machine Learning' by Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z. Berkay Celik, Ananthram Swami (https://arxiv.org/abs/1412.6572)

Code Base:
http://www.robots.ox.ac.uk/~aarnab/adversarial_robustness.html
https://github.com/hmph/adversarial-attacks
https://github.com/asranand7/Adverserial_Attack

https://github.com/tensorflow/cleverhans/blob/master/cleverhans_tutorials/mnist_tutorial_tf.py

https://github.com/tensorflow/cleverhans/blob/master/cleverhans_tutorials/mnist_blackbox.py