|  |  |  |
| --- | --- | --- |
| Fall 2024 | CS53331/4331 Adversarial Machine Learning | Assignment 1 |

|  |  |
| --- | --- |
| **Title** | **The Deep Learning Stuff** |
| **Due date** | Monday, October 14th, before the class. |
| **First Name** | Ucchwas Talukder |
| **Last Name** | Utsha |
| **Student ID** | R11836597 |
| **Marks** | 100 |

Note: Please answer the following questions and submit them through Blackboard. Be sure to submit it to assignment 1. DO NOT write the report by hand and submit a scanned version. Just write the answers in a Word document and submit it. Both Word and PDF submissions are accepted.

# Submission Instruction (3 documents)

You are required to submit three documents:

1. ***Report.*** Just fill out the above report and submit it as a Word or PDF document.
2. ***Ipynb file.*** The code that you have written. Preferably in an ipynb document. You can submit it as a .py file as well.
3. ***Txt file of the code.*** We need your code in the .txt file as well. Use whatever way you prefer. The fastest would be to download the file as a .py file and change the extension to .txt

# Objectives

This assignment has three main objectives:

1. Implement untargeted attacks and check the results (FGSM, PGD, DeepFool, and C&W L2)
2. Implement targeted attacks and check the results (FGSM, PGD, and C&W L2)
3. Analyze defense using adversarial training.

# Get started

Download the assignment files from Blackboard. You will need the report (This file), the .ipynb file where you will put your code, the dataset, and the adversarial trained model.

# Dataset

For this assignment, we will use a traffic sign recognition dataset. The German Traffic Sign Recognition Benchmark (GTSRB) dataset consists of almost 51K images of traffic signs. There are 43 classes, and the size of their images is 32×32 pixels. Some of the images are shown below. Please download the dataset from Blackboard or [here](https://drive.google.com/file/d/1lt6qvYnkEZ-qXREufkzjonocn_1_moPi/view?usp=drive_link). You will need the starter code from Blackboard to preprocess it and build a VGG16 model you will attack. More information about the dataset can be found [here](https://www.kaggle.com/datasets/meowmeowmeowmeowmeow/gtsrb-german-traffic-sign).



As we did with Assignment 0, it is recommended that you upload the dataset into your personal Google Drive to follow the Colab instructions as they are. Of course, if you prefer to use other than Colab, you will need a similar preprocessing.

# Instruction for Colab (repeated from Assignment 0)

To get started with Google Colab, simply go to [Google Colab](https://colab.research.google.com/), sign in with your Google account, and create a new notebook. You can write and execute Python code directly in the notebook. To access your dataset stored in your Google Drive (previous step), first run the following code to mount your Drive:

A black background with white text

Description automatically generated

Follow the authorization steps, and your Drive will be accessible at **/content/drive/My Drive/.** You can then load your dataset into the notebook by providing the correct file path. This part of the code is provided for you in the .ipynb file of Assignment 0. You will need to setup the drive connection and run the code.

To use the free GPU provided by Colab, you can change the runtime to access a GPU by clicking on **"Runtime" > "Change runtime type” and** selecting **"T4 GPU"** from the **Hardware accelerator** dropdown menu. You can always use higher GPU powers at a cost (Colab Pro is $10 per month), but you should be fine with the free version, considering that you start the assignment early enough.

Colab comes with many pre-installed libraries, but if you need to install additional Python packages, you can do so with pip. For example:



Remember to save your work frequently.

After you've completed your work in Google Colab, you can easily download your notebook from Google Colab, go to **"File" > "Download" > "Download.ipynb"**.

# Other than Colab

If you don’t prefer Colab or notebook, you always have the option to run it on your computer (especially if it has a GPU) or access HPCC resources at TTU (needs an account with my permission).

# Additional resources

1. TensorFlow resource <https://www.tensorflow.org/>
2. PyTorch resources <https://pytorch.org/get-started/pytorch-2.0/>
3. Deep learning with Python <https://dl-with-python.readthedocs.io/en/latest/>
4. Get started with Colab <https://colab.research.google.com/>

# Task 1-untargeted attacks (40 pts)

Let’s start with basic **non-target white-box attacks**. First, we will implement some non-target white-box attacks we studied in class. Your downloaded code from Blackboard will build a VGG16 model for you. You will attack that deep learning model throughout the assignment.

This task aims to implement the following attacks: Fast Gradient Sign Method (FGSM), Projected Gradient Descent (PGD), Deep Fool, and C&W with L2 norm. You don’t need to implement these attacks from scratch. Code for them can be in several libraries, including [Adversarial Robustness Toolbox (ART)](https://adversarial-robustness-toolbox.readthedocs.io/en/latest/), [cleverhans](https://github.com/cleverhans-lab/cleverhans), or [scratchai](https://github.com/iArunava/scratchai) (Just traditional libraries. Maybe there are better ones now). Using ART for this assignment is recommended, but using any other libraries of your choice is also acceptable. This [notebook](https://github.com/Trusted-AI/adversarial-robustness-toolbox/blob/main/notebooks/art-for-tensorflow-v2-keras.ipynb) is a good start on how to use ART, and multiple other notebooks are available [here](https://github.com/Trusted-AI/adversarial-robustness-toolbox/blob/main/notebooks/README.md).

Your task is to apply attacks to create non-target adversarial examples using the first 500 images of your test set (check imgs\_adv and labels\_adv in your code). For FSGM, PGD, and DeepFool, apply perturbations magnitudes: 𝜖= [0.05, 0.1, 0.2, 0.25, 0.3]. For C&W, use the L2 norm to perform the attack. You should provide us with the following results.

**Note:** [10 pts] will go to your code for attack implementation. The comparison code evaluation will be included in its respective questions.

1. [4 pts] Plot the clean image vs adversarial image for three images of your choice. This should happen for all perturbation magnitudes (if any) and for all attacks. An example of FGSM is shown below. Do it for FGSM, PGD, DeepFool, and C&W (separate figure for each)

A collage of images of a person's face

Description automatically generated

**Ans:** I plotted **7, 8, 9 indices** images.

**For FGSM:**

A collage of different signs

Description automatically generated

**For PGD:**

A collage of different signs

Description automatically generated

**For DeepFool:**

**A collage of different signs

Description automatically generated**

**For C&W:**

**A collage of several signs

Description automatically generated**

1. [6 pts] For two images of choice, plot the clean image vs. adversarial images of all attacks (if the perturbation is needed, choose 𝜖= [0.1]). You should produce one similar figure but with the four attacks and the original.

**Ans:**

**A collage of different types of traffic signs

Description automatically generated**

1. [5 pts] Fill out Table 1 and Table 2 for accuracy and add the noise of each attack. Adversarial classification accuracy is the accuracy on adversarial samples only.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Table 1: Adversarial classification accuracy of the models | | | |  |
| **Model** | **Clean image** | **Adversarial images with 𝜖=0.05** | **Adversarial images with 𝜖=0.2** | **Adversarial images with 𝜖=0.3** |
| VGG16-FGSM | 93.8% | 27.4% | 17.2% | 14.2% |
| VGG16-PGD | 93.8% | 23.6% | 7.8% | 7.8% |
| VGG16-DeepFool | 93.8% | 69.4% | 28.4% | 25.4% |
|  | **Clean image** | **Adversarial images** | | |
| C&W L2 | 93.8% | **93.8%** | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Table 2: Noise of the models | | | |
| **Model** | **Adversarial images with 𝜖=0.05** | **Adversarial images with 𝜖=0.2** | **Adversarial images with 𝜖=0.3** |
| VGG16-FGSM | 2.72 | 9.99 | 14.23 |
| VGG16-PGD | 2.72 | 8.03 | 11 |
| VGG16-DeepFool | 0.62 | 2.49 | 3.74 |
|  | **Adversarial images** | | |
| C&W L2 | **0.20** | | |

1. [5 pts] Plot accuracy versus perturbation 𝜖 for FSGM and PGD adversarial attacks (similar to the example in the following figure).

A graph of a graph with blue and orange dots

Description automatically generated

**Ans:**  
A graph with red and blue lines

Description automatically generated

1. [5 pts] Briefly provide insights on the model’s performance under attacks and any other observations regarding the results.

**Ans:**

**Impact of Adversarial Attacks:** Across all attacks (FGSM, PGD, DeepFool, and C&W L2), the model's performance significantly deteriorates as the perturbation ϵ increases. This highlights the model's vulnerability to adversarial attacks, where even small perturbations cause noticeable drops in accuracy.

**FGSM vs. PGD:** Both FGSM and PGD showed decreasing accuracy as ϵ increased. However, PGD was more effective in reducing the model's accuracy, particularly at higher ϵ values, where the accuracy stabilized at around 7.8%, compared to FGSM, which remained slightly higher.

**DeepFool Performance:** DeepFool introduced less noise compared to FGSM and PGD, but it still effectively reduced the model’s accuracy, especially at higher ϵ values. This suggests that DeepFool can generate adversarial examples that are more subtle but still successful.

**C&W L2 Attack:** Unlike the other attacks, the C&W L2 attack did not significantly lower the model’s accuracy (93.8%), which suggests that the model is more resistant to this type of attack in this scenario.

**General Observation:** Larger perturbations consistently lead to lower accuracy for FGSM, PGD, and DeepFool, while C&W L2 shows less impact on performance.

# Task 2-targeted attacks (30 pts)

Now, let’s do some **target attacks** with white-box assumptions. Use the images with the Stop sign (label 14) from the overall test set for this task. There are around 270 images of that kind. Implement FGSM attacks on the Stop sign images to misclassify them as speed 30 sign images (label 1). Apply perturbations magnitudes: 𝜖= [0.05, 0.1, 0.2, 0.25, 0.3] for these attacks and report the classification accuracy on the Stop sign images and the Speed Limit 30 sign images.

Apply the same thing using PGD attacks and compare the results.

Apply the same thing using C&W L2 attacks (perturbations magnitudes) and compare the results.

Then, answer the following questions.

**Note:** [10 pts] will go to your code for attack implementation. The comparison code evaluation will be included in its respective questions.

1. [5 pts] Plot the clean image vs adversarial image for a Stop sign image of your choice. This should happen for all perturbation magnitudes (if any) and all attacks.

**Ans:** I have plotted 3 clean vs adversarial stop sign images.

**For FGSM:**

**A collage of images of a stop sign

Description automatically generated**

**For PGD:**

**A collage of images of a stop sign

Description automatically generated**

**For C&W L2:**

**A collage of a stop sign

Description automatically generated**

1. [10 pts] Fill out Table 3 for accuracies.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Table 3: Classification accuracy of the model on adversarial original and target class images. | | | | |
| **𝜖** | **FGSM attack – Stop sign images** | **FGSM attack –** **Speed Limit 30 sign images** | **PGD attack – Stop sign images** | **PGD attack –** **Speed Limit 30 sign images** |
| 0.05 | 0.37% | 22.22% | 0.37% | 55.56% |
| 0.1 | 6.30% | 3.70% | 0.74% | 14.81% |
| 0.2 | 0.37% | 0% | 0% | 3.7% |
| 0.25 | 0% | 0% | 0% | 0% |
| 0.3 | 0% | 0% | 0% | 0% |
|  | **C&W attack – Stop sign images** | | **C&W attack –** **Speed Limit 30 sign images** | |
| - | 0% | | 100% |  |

1. [5 pts] Briefly provide insights on the model’s performance under attacks and any other observations regarding the results.

**Ans:** The model demonstrates significant vulnerability to adversarial attacks, especially as the perturbation magnitude ϵ increases. For **FGSM** and **PGD** attacks, accuracy on Stop signs drops sharply with higher ϵ, reaching 0% at 𝜖 = 0.25 and above, showing the model's inability to classify correctly under stronger attacks. PGD proves more effective than FGSM, reducing accuracy to 0% even at lower ϵ values (0.2).

For the **C&W L2** attack is the most successful for Stop signs, achieving 0% accuracy across all perturbation levels, but it does not impact the classification of Speed Limit 30 signs, where the model maintains 100% accuracy. This suggests that the model is highly susceptible to adversarial perturbations but retains some robustness under specific attack types.

# Task 3- Adversarial Defense (20 pts)

Now, let’s defend against adversarial attacks. This [notebook](https://github.com/Trusted-AI/adversarial-robustness-toolbox/blob/main/notebooks/adversarial_training_mnist.ipynb) is a good start for adversarial training. Due to its complexity and the time taken to build an adversarial-trained model, we decided to give the model to you, and your only task will be to analyze it. The VGG classifier from earlier parts is used to build the adversarial model. New adversarial images were generated and used to build the new model. This model is provided in your assignment, and you will need to analyze it. Your tasks for this part will

* Analyze the code (commented) provided to build the adversarial model and answer question 1.
* Measure the performance of the trained robust model and compare it to the original VGG model used in earlier parts. For the attack part, we will use FGSM and PGD on the samples we reserved for adversarial generation in the earlier parts (use eps=0.1). You will probably need to implement your evaluation for things to work here. Record the classification accuracies that have been attacked.

1. [5 pts] Answer the following questions about the provided model or code for it.
   1. What attacks were used to build the model

Ans: The FGSM attack was used to build the adversarially trained model.

* 1. What perturbation (if any) was used to build the attack samples?

Ans: ε = 0.1.

* 1. What percentage of training samples was used to generate the adversarial samples used in the built model?

Ans: 50% of the training samples.

* 1. How many epochs were used?

Ans: 25.

* 1. What needs to be changed in the code to generate an adversarial-trained model based on C&W attacks? Write the code for it as an answer.

Ans: To generate an adversarial-trained model based on C&W attacks, we need to replace FGSM with the C&W attack in the Adversarial Defence code.

Code:

classifier = KerasClassifier(model=model, clip\_values=(0, 1))

cw\_l2 = CarliniL2Method(classifier=classifier, targeted=False, max\_iter=10)

adv\_trainer\_cw = AdversarialTrainer(classifier, cw\_l2, ratio=0.5)

adv\_trainer\_cw.fit(imgs\_train, labels\_train\_cat, nb\_epochs=25, batch\_size=16)

model.save(MODEL\_DIR + 'adversarial\_cw\_l2\_trained.h5')

1. [10 pts] Fill out Table 4 for accuracies.

|  |  |  |
| --- | --- | --- |
| Table 4: Classification accuracy of the traditional and adversarial trained model | | |
| **Setup** | **VGG16** | **Adversarial trained model** |
| **Clean train images** | 97.05% | 99.73% |
| **Clean subset of test images** | 93.97% | 96.78% |
| **FGSM attacked subset of test images** | 22.60% | 45.8% |
| **PGD attacked subset of test images** | 8.00% | 51.6% |

1. [5 pts] Briefly provide insights on the model’s performance under attacks and any other observations regarding the results.

**Ans:** The adversarial-trained model significantly outperforms the standard VGG16 model when subjected to adversarial attacks. For clean images, both models show high accuracy, but the adversarial-trained model has a slight edge. Under **FGSM** and **PGD** attacks, the **adversarial-trained model** maintains much higher accuracy (45.8% and 51.6%, respectively) compared to the **VGG16** model (22.6% and 8%). This highlights the effectiveness of adversarial training in improving robustness against adversarial attacks. However, despite the improvement, the adversarial-trained model still experiences a notable accuracy drop under stronger attacks which indicates that while adversarial training provides better resilience, it doesn't fully eliminate vulnerability to highly optimized adversarial perturbations (ε = 0.1).

# Submission Instruction (3 documents)

You are required to submit three documents:

1. ***Report.*** Just fill out the above report and submit it as a Word or PDF document.
2. ***Ipynb file.*** The code that you have written. Preferably in an ipynb document. You can submit it as a .py file as well.
3. ***Txt file of the code.*** We need your code in the .txt file as well. Use whatever way you prefer. The fastest would be to download the file as a .py file and change the extension to .txt