ADVANCE Labs - Adversarial Attack on Cyberbully Detection Models Lab

The development of this document is partially funded by the National Science Foundation's Security and Trustworthy Cyberspace Education, (SaTC-EDU) program under Award No. 2114920.

1 Lab Overview

In this lab, you will learn how AI/ML models trained to detect cyberbullying on images can be manipulated to output the wrong prediction. This phenomenon is known as an adversarial attack. Cyberbullying is bullying performed via electronic means such as mobile/cell phones or the Internet. The objective of this lab is for students to gain practical insights into adversarial attacks in online harassment, such as cyberbullying. To learn how to use existing adversarial attack methods to fool AI/ML models and develop robust AI/ML solutions against such attacks.

In this lab, students will be given a starter code. Their task is to follow the instructions in the Jupyter notebook, complete simple coding challenges, use two adversarial methods to attack a cyberbully detection model, and deploy the model by testing it on their samples.

Content Warning: This lab contains some inappropriate images. We minimized showing such content samples as they do not represent the authors' views.

2 Lab Environment

This ADVANCE lab has been designed as a Jupyter notebook. ADVANCE labs have been tested on the Google Colab platform. You should use Google Colab since it has nearly all software packages preinstalled, is free to use, and provides free GPUs. You can also download the Jupyter notebook from the lab website and run it on your machine, in which case you will need to install the software packages yourself (you can find the list of packages on the ADVANCE website). However, most of the ADVANCE labs can be conducted on the cloud, and you can follow our instructions to create the lab environment on the cloud.

3 Lab Tasks

3.1 Getting Familiar with Jupyter Notebook

The main objective of this lab is to learn how to perform adversarial attacks on models trained to detect cyberbullying. Before proceeding, let us familiarize ourselves with the Jupyter notebook environment.

Jupyter notebooks have a Text area and a Code area. The Text area is where you'll find instructions and notes about the lab tasks. The Code area is where you'll write and run code. Packages are installed using `pip` and need to be preceded with a `!` symbol.

You will complete some simple coding challenges and tasks in this lab. You will be asked to complete and execute sections of a code. After execution, you are expected to add a screenshot of your output in your report. In the notebook provided, eight tasks are to be completed, indicated by "Task #" where # is the task number. You must complete these eight tasks plus the discussion at the end of this document to complete

this lab successfully. Following each task is a code cell with some part of the code replaced with ***None*** statements. You are to understand the code and replace all the ***None*** statements within the designated block with the correct statement unless stated otherwise. The designated block starts with a comment (Start code here) and ends with a comment (End code here). In the designated block, you are only required to replace the ***None*** statements with the correct statements (variable(s) or formulas). For example, in the designated block in the code cell below, you must replace ***None*** with the correct statement.

```
    # Start code here #
    ...
    total_number_of_examples = None
    ...
    # End code here #
```

Get started. Run the cell under "Required modules" to load the required frameworks for this lab.

3.2 Adversarial Attack on Cyberbullying Detection Model

You will attack a cyberbullying detection model already trained on not-safe-for-work (NSFW) images in this lab. This model is based on the ResNet50 architecture. Your task is to make this model predict NSFW images as safe for work, i.e., make the model predict that the image is non-NSFW. You will use a small NSFW dataset we collected to test your attack. You will observe how different adversarial attacks and parameters affect the model prediction accuracy and document your observation. You can access the lab by clicking here.

3.2.1 Dataset

In this lab, we provide an NSFW dataset. This dataset has five categories - drawings, neutral, hentai, porn, and sexy. The drawings and neutral categories are images that are safe for work. The hentai, porn, and sexy (explicit contents that are not pornography) are images not safe for work. You can learn more about these labels in the official NSFW data page.

3.2.2 Task 1: Number of Images

To get a sense of our dataset, we will view some images and count the number of images in our dataset. Run the cell before Task 1 code cell to download the dataset and view an image from the dataset. Complete the code in the designated block in Task 1 code cell and record your result.

3.2.3 Task 2: Size of image, its label, and class

Run the code cells after Task 1 to preprocess the dataset. We need to make each image of the same size, convert images to tensors, crop, and normalize. This preprocessing step ensures each image is in the form the model expects and can improve performance. Here is a sample of this transformation from the lab:

```
test_image_transforms = transforms.Compose([transforms.Resize(224),
                                            transforms.CenterCrop(224),
                                            transforms.ToTensor(),
                                            transforms.Normalize(
                                                mean=[0.485, 0.456, 0.406],
                                                std=[0.229, 0.224, 0.225])
                                        ])
```

Complete the code in the designated block in Task 2 code cell and record your result. You will need to report the dimension (size/shape) of the test image after transformation, the dimension of the label, and its ground truth (label/class/category). Note: we use label, class, and category interchangeably in this lab.

3.3 Adversarial Attack

Deep neural networks are vulnerable to adversarial attacks despite their high accuracy in various tasks. This vulnerability is a little change to the input that causes the network to predict the incorrect output. This small change is often imperceptible to human vision. Figure 1 shows an example of a perturbed cyberbully image on the right and the original image on the left. Observe how close the perturbed example is to the original and the high prediction probability (86.5%) assigned by the model. We will generate such perturbations in this lab using the fast gradient sign method (FGSM) and projected gradient descent (PGD). These methods are briefly described below.

Figure 1: Example of perturbed input used for adversarial attack.

Adversarial attacks can be categorized into two types - white-box and black-box attacks. In a white-box attack, the adversary knows information about the target model. The information includes model parameters, architecture, and training data. The adversary does not have information about the target model in a black-box attack. We will perform a white-box attack in this lab since we know the model architecture, its parameters, and training data. Adversarial attacks can be targeted or untargeted. The model is fooled in a targeted attack, not to predict a specific label. In contrast, in an untargeted attack, we do not care about a specific label as long as the prediction is incorrect.

3.3.1 Fast Gradient Sign Method (FGSM) Attack

The fast gradient sign method is a fast method for generating adversarial examples. Given a clean image with no perturbation, FGSM efficiently finds an adversarial perturbation that, when added to the clean image,

maximizes the classifier (neural network) loss leading to misclassification. Consider an adversarial example $x^{'}$ defined as:

$$x' = x + \delta \tag{1}$$

where x is a vector of a clean image, δ is a small noise (perturbation) added to the clean image having the same dimension as x. FGSM efficiently computes δ by solving the following equation:

$$\delta = \epsilon * sign(\nabla_x J(\theta, x, l)) \tag{2}$$

where x is the clean input image. θ is the model parameters, l is the true label of x, and $J(\theta, x, l)$ is the cost function used in training the neural network and ϵ is a scalar value that adjusts the magnitude of the perturbation. Informally, δ is a vector of the same dimension as x, where the elements in this vector are the sign of the gradient of the cost function with respect to x.

3.3.2 Task 3: Implement FGSM formula

Run the code cells under *White-box Attack*, implement equation 2 in the designated block in Task 3 code cell, and run the cell. You have been provided an incomplete function that implements FGSM. You must understand and complete the function by implementing equation 2.

3.3.3 Projected Gradient Descent (PGD)

One drawback of FGSM is that it is a one-step attack. We compute the gradient at a point and take one step. Projected gradient descent (PGD) is an extension of FGSM that repeatedly applies FGSM to generate adversarial images. PDG applies FGSM multiple times given a small step size α, and the pixels of the adversarial images generated at each iteration are clipped to be within $\epsilon - neighbourhood$ of the original image. PDG formulation is as follows:

$$x'_0 = x \tag{3}$$

Repeat:

$$x'_{N+1} = Clip_{x,\epsilon}\{x'_N + \alpha * sign(\nabla_x J(\theta, x'_N, l))\} \tag{4}$$

where $Clip_{x,\epsilon}\{x'\}$ is a function that clips the pixel values of x' to be in L_∞ $\epsilon - neighbourhood$ of the original image x [KGB+16]. The number of iterations, α and ϵ, are some hyperparameters that can be adjusted in PDG.

3.3.4 Task 4: Implement PGD formula

Implement equation 4 in the designated block in Task 4 code cell and run the cell.

3.3.5 Task 5: Pass perturbed images through the model to perform FGSM attack

To perform the FGSM, you must pass the dataset through the model to execute the attack. Complete the designated blocks in the Task 5 code cell to perturb images using FGSM and pass the perturbed images through the model. Run the cell after completing this task.

3.3.6 Task 6: Pass perturbed images through the model to perform PGD attack

To perform the PGD, you must pass the dataset through the model to execute the attack. Complete the designated blocks in the Task 6 code cell to perturb images using PGD and pass the perturbed images through the model. Run the cell after completing this task.

3.3.7 Epsilon

Epsilon is a hyperparameter that controls the magnitude of the perturbation. The bigger the epsilon, the more perceptible the perturbation would be.

3.3.8 Task 7: Execute the FGSM attack using different epsilon values

Execute the visualization cells for visualization purposes :

```
    def plot\_accuracy\_vs\_epsilon(...):
        ...
```

```
    def plot_adversarial_samples(...):
        ...
```

Complete the designated block in Task 7 code cell and run the cell. Document your observation on the epsilon values and accuracy from the graph generated.

3.3.9 Task 8: Execute the PGD attack using different epsilon values

Complete the designated block in Task 8 code cell and run the cell. Document your observation on the epsilon values and accuracy from the graph generated.

3.3.10 Discussion

What are your observations about FGSM and PGD?

4 Submission Instructions

You need to submit a detailed lab report, with screenshots, to describe what you have done and observed. You also need to explain the observations that are interesting or surprising. Please also list important code snippets followed by an explanation. Simply attaching a code without any explanation will not receive credits.

References

[KGB+16] Alexey Kurakin, Ian Goodfellow, Samy Bengio, et al. *Adversarial examples in the physical world*. 2016.