ADVANCE Labs - Adversarial Attack on Cyberbully Detection Models Lab

1 Lab Overview

In this lab, you will learn about how AI/ML models that have been trained to detect cyberbullying on images can be manipulated to output the wrong prediction. This is known as 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 as it pertains to online harassment such as cyberbullying, and to learn how to use existing adversarial attack methods to fool AI/ML models and how to develop AI/ML solutions that are robust against such attacks.

In this lab, students will be given a starter-code. Their task is to follow the instructions provided 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 own samples.

Content Warning: This lab contains some inappropriate images. We minimized showing such content samples in this lab. They do not represent the views of the authors.

2 Lab Environment

This ADVANCE lab has been designed as a Jupyter notebook. ADVANCE labs have been tested on the Google Colab platform. We suggest you to 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 own 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 attack on models trained to detect cyberbullying. Before proceeding to that, let us get familiar 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. Try accessing the lab environment for this task here.

In this lab, you will be completing some simple coding challenge. You will be asked to complete and execute sections of a code. After execution, you are expected to add screenshot of your output in your report. You know when you are to complete a coding section when there is a text in the text area that says **Todo**, hints have been provided to help you with completing this task. You must complete the coding section/cell following the text area. In the coding cell, you are only required to fill in the missing variable(s) or formulas in the designated areas. The code block you will have the form:

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

You are to replace all the *None* statements in within the designated block with the correct statement unless stated otherwise. In the notebook provided, there are 8 task to be completed indicated by ”Task #”. You have to complete these 8 tasks plus the discussion at the end of this document. In your report, attach screen shot of your code and outputs of task.

3.2 Adversarial Attack on Cyberbullying Detection Model

In this lab, you will attack a cyberbullying detection model already trained on not safe for work (NSFW) images. 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 are non-NSFW. You will use a small NSFW dataset we collected to test your attack. You need to 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 NSFW dataset, this dataset have 5 categories - drawings (images that are safe for work), neutral (images of things that are safe for work), hentai (inappropriate images not safe for work), porn (pornography images not safe for work) and sexy (explicit contents that are not pornography but not sage for work). You can learn more about these labels in the official NSFW data scrapper. You can download the dataset by running the following cell under lab overview in jupyter notebook.

```
!wget https://clemson.box.com/shared/static/x72ytb2tdp2ygnyc61enz4bsl1pj6kj0.zip
```

Run all the code in this section (Lab Overview) of the notebook. You will need to report the size of the testing dataset, the shape of an image after transformation and its corresponding ground truth (label/class/-category). Note: we use label, class and category interchangeably in this lab.

3.2.2 Preprocessing Data

Follow the instructions in the text areas and run the subsequent codes to preprocess your data. We need to make each image of the same size, convert images to tensors, crop and normalize. This makes sure each image is in the form the model is expecting 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])
                                        ])
```

After applying the transforming you need to report the shape/size of each image, label size and actual label (drawings, hentai, neutral, porn or sexy).

3.2.3 Deploy and Run Custom Samples

Download the samples file and use your model to detect the cyberbullying samples in this file. Report the samples that were detected as cyberbullying. Do you think your model is good enough? In this lab, you will later learn how to use hyperparameters to improve the performance of your model.

3.3 Adversarial Attack

Deep neural networks are vulnerable to adversarial attacks despite their high accuracy in various tasks. This vulnerability is usually in the form of a little change to the input that causes the network to predict incorrect output. This small change is often imperceptible to the human vision. Figure 1 show 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. In this lab, we will be generating such perturbations using the fast gradient sign method (FGSM) and projected gradient descent (PGD). These methods are briefly described below.

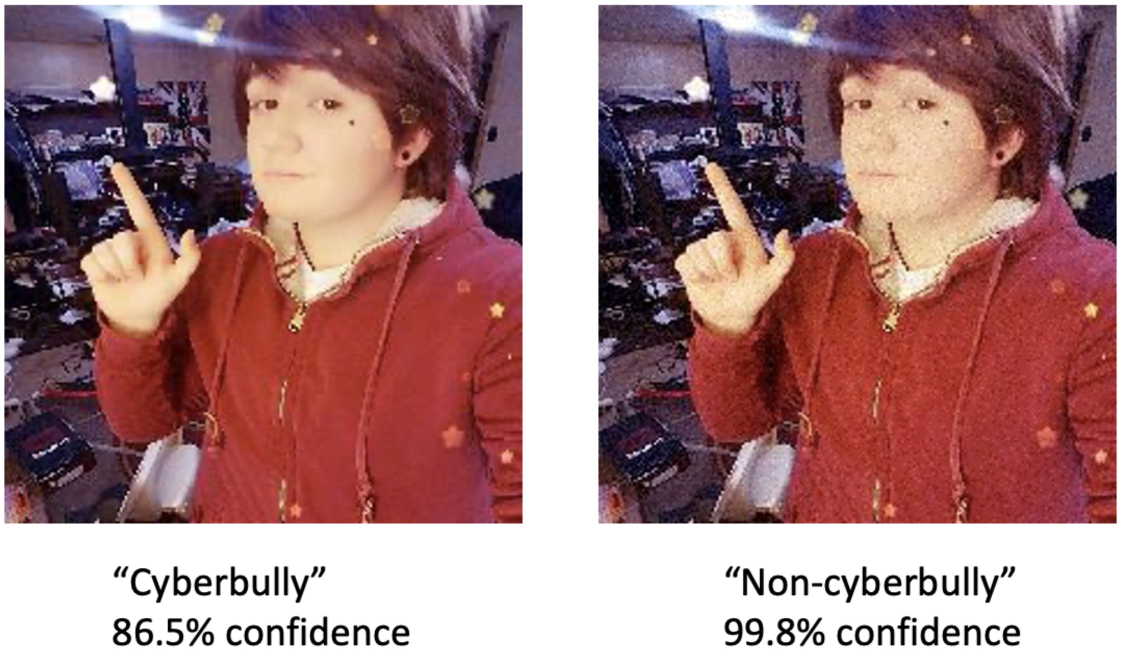

Figure 1: Example of perturbed input used for adversarial attack.

Adversarial attack can be categorized into two types - white-box and black-box attack. In a white-box attack, information about the target model in known by the adversary. The information include model parameter, architecture and training data. The adversary does not have information about the target model in a black-box attack. In this lab, we will be performing a white-box attack since we know the model architecture, its parameters and training data. Adversarial attacks can be target or untargeted. In a targeted attack, the model is fooled to not predict a specific label while in the untargeted attack we do not care about a specific label as long as the prediction is not correct.

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:

$$x' = x + \delta$$

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:

$$\delta = \epsilon * sign(\nabla_x J(\theta, x, l))$$

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 adjust the magnitude of the perturbation. Informally, δ is a vector of the same dimension as x where the elements in this vector is the sign of the gradient of the cost function with respect to x.

Task 3 in the notebook: Implement this equation in code. You have been provided an incomplete function that implements FGSM, you are to understand the function and complete it by implementing the equation above.

3.3.2 Projected Gradient Descent (PGD)

One drawback of FGSM is that is it 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, 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:

$$x'_0 = x$$

Repeat:

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

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]. There are hyperparemeters that can be adjusted in PDG - number of iterations, α and ϵ.

Task 4 in the notebook: Implement this equation in code. You have been provided an incomplete function as in the FGSM task.

3.3.3 Epsilon

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

3.3.4 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 what you have observed. You also need to provide explanation to the observations that are interesting or surprising. Please also list important code snippets followed by explanation. Simply attaching 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.