ADVANCE Labs - Adversarial Attack 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 unsafe content on images can be manipulated to output the wrong prediction. This phenomenon is known as an adversarial attack. The objective of this lab is for students to gain practical insights into adversarial attacks in online harassment. 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, you will be given a starter code. The task is to follow the instructions in the Jupyter notebook, complete simple coding challenges, use two adversarial methods to attack a NSFW image detection model, and deploy the model by testing it on some samples. When completing the lab report, you will be required to show your completed code snippet and significant output.

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. We suggest you use Google Colab, since it has nearly all software packages pre-installed, 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 Warming Up: Getting Familiar with Jupyter Notebook

The main objective of this lab is to learn how adversarial attacks work for AI/ML models. 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. The lab has three areas: one text area and two code areas. Follow the instructions for the three areas, fill the three areas with the instructed content, and **add a screenshot to your report**.

3.2 Open Adversarial Attack Lab

In this lab, you will attack a cyberharassment detection model already trained on not-safe-for-work (NSFW) images in this lab. This detector is based on a Vision Transformer (ViT) model. Your task is to make this model predict NSFW images as safe for work, i.e., make the model predict that the image is normal. 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.

Read the lab instructions and execute code to download `transormers` first and import neccessary packages. Next, you will complete some simple coding challenges and tasks in this lab. After execution, you are expected to add a screenshot of your output in your report. In the notebook provided, several tasks are to be completed, indicated by "Task #" where # is the task number.

3.3 Datasets Preparing

In this lab, we provide an NSFW dataset. This dataset has two categories - normal and nsfw. The normal categorie include images that are safe for work. The nsfw are images not safe for work. These images are collected following the NSFW data scrapper.

3.3.1 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 block by replacing "None" with your solution. Please show your completed cell block and record your result in your report.

```
# Start code here #
total_number_of_images = None
# End code here #

print(f"The number of images in the test dataset is: {total_number_of_images}")
```

3.3.2 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:

```
# Transform the images to 224x224
image_transform = 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.

3.4 Task3: Evaluate the Pre-trianed Model

Now it is time to deploy the pre-trained model and evaluate it with our test dataset. We have already prepared an evaluation function that you can be utilized to get the accuracy output.

Run the code cells until Task 3, and you are supposed to write your own code to evaluate the model and print out the accuracy. Present your completed cell block and add a screenshot of the output to your report.

Execute the next code cell to demonstrate the prediction with one sample image. Include a screenshot of the output to your report.

3.5 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. And a small perturbation may damage the performance of the orignal model significantly. We will generate such perturbations in this lab using the fast gradient sign method (FGSM) [GSS15] and projected gradient descent (PGD) [Mad+19].

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.

3.5.1 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 $epsilon$ 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.

3.5.2 Task 4-1: Implement FGSM formula

Run the code cells until Task 4-1, then you are supposed to implement equation 2 in the designated block in Task 4-1 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. Then, show your code in your report.

3.5.3 Task 4-2: Pass perturbed images through the model to perform FGSM attack

To perform the FGSM, you need to pass the dataset through the model to execute the attack. Complete the designated blocks in the Task 4-2 code cell to perturb images using FGSM and pass the perturbed images through the model. Run the cell after completing this task and show your code snippet in the report.

3.5.4 Epsilon

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

3.5.5 Task 4-3: Execute the FGSM attack using different epsilon values

Read and run code cells until Task 4-3. We have prepared 3 code cells to help you visulize the changes in accuracy that occur with different ϵ as well as the adversarial images after the FGSM attacks. Run the code cells in Task 4-3, save the outputs, and present them in your report. You will also need to briefly describe your observation in your report.

3.5.6 Projected Gradient Descent (PGD) Attack

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 $L_\infty\epsilon - neighbourhood$ of the original image x [KGB17]. The number of iterations, α and ϵ, are some hyperparameters that can be adjusted in PDG.

3.5.7 Task 5-1: Implement FGSM formula

Understand FGD attack and the code we offered. Implement equation 4 in the designated block in Task 5-1 code cell and run the cell. Include your code snippet in the report.

3.5.8 Task 5-2: Pass perturbed images through the model to perform PGD attack

To perform the PGD, you need to pass the dataset through the model to execute the attack. Complete the designated blocks in the Task 5-2 code cell to perturb images using PGD and pass the perturbed images through the model. Run the cell after completing this task and show your code snippet in the report.

3.5.9 Task 5-3: Execute the FGSM attack using different epsilon values

Similar as FGSM attack, read and run code cells until Task 5-3. Now, complete and execute the code cells under Task 5-3. Please present your code pieces and the outputs in your report. You will also need to briefly describe your observation in your report.

3.6 Discussion

Compare the predictions of the original pre-trained model with the results after two attacks. Describe your observations and discuss the pros and cons of such white-box adversarial attacks. Please share your thoughts in your report.

4 Submission Instructions

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

References

[GSS15] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. *Explaining and Harnessing Adversarial Examples*. 2015. arXiv: 1412.6572 [stat.ML].

[KGB17] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. *Adversarial examples in the physical world*. 2017. arXiv: 1607.02533 [cs.CV].

[Mad+19] Aleksander Madry et al. *Towards Deep Learning Models Resistant to Adversarial Attacks*. 2019. arXiv: 1706.06083 [stat.ML].