Adversarial Machine Learning

Homework Assignment 2

The assignment is due by the end of the day on Thursday, February 13.

An explanation regarding the assignment was provided during the class meeting on Tuesday, February 4. Please watch the recording of the lecture, as it will help with the assignment, especially if you haven't worked with PyTorch before.

Please note that the submitted solutions are expected to follow the examples provided on Canvas in Modules/Codes/PyTorch_White_Box_Evasion_Attacks. If your code does not follow the provided examples, marks will be deducted. Also, marks will not be assigned if the codes are generated by coding assistants. It is okay if you write the code and afterward use help from coding assistants to debug the code: it is not okay to ask a coding assistant to generate the code and then copy-paste it into the notebook.

And to remind you again, the assignments and solutions from the offering of the AML course in the previous semesters can be found at these links: Spring 2023 (link), and Fall 2021 (link). These resources can be helpful for solving the assignments. Also, if you need to refresh your knowledge about training neural networks, the materials from the Applied Data Science with Python course can be found here.

Objective:

- Implement white-box evasion attacks against deep learning-based classification models using the PyTorch library.
- Implement transferable black-box evasion attack against deel learning classification models.

Part 1: White-box Evasion Attacks with PyTorch (70 marks)

Dataset: For this assignment, we will use a dataset consisting of 8,676 images with 101 categories of various objects. The dataset is provided with the other assignment files.

Task 1: Train a deep-learning model for classification of the objects dataset, using the PyTorch library.

For loading the dataset please check the file Loading_Custom_Dataset_PyTorch in the shared folder in Canvas titled PyTorch_White_Box_Evasion_Attacks. The dataset is organized so that it has 101 folders, where each folder contains the images for one category and the titles of the folders correspond to the categories in the dataset.

You will need to use GPU for this assignment. Google Colab Pro is the recommended option for GPU access.

Use a pre-trained VGG-16 model, and finetune it to the images of objects in the dataset.

Perform hyperparameter tuning to obtain an accuracy on the test dataset above 90%.

If needed, please review the lecture on training Neural Networks with PyTorch by following this link.

Estimated running time: between 10 and 30 minutes using a GPU.

Report (40 marks): (a) Fill in Table 1 with the values for the classification accuracy for the train set, validation set, and test set of images. For full marks, it is expected to report a test accuracy above 90%. (b) For each model, plot the training and validation loss and accuracy curves. (c) If applicable, provide any other observations regarding the model or the dataset.

Table 1. Classification accuracy.

Model	Train Set	Validation Set	Test Set

Task 2: Implement non-targeted white-box FGSM and PGD evasion attacks against the deep learning model from Task 1.

Use the Cleverhans library for implementing the attacks for this part. Check the file Evasion_Attacks_PyTorch in the shared folder in Canvas titled PyTorch_White_Box_Evasion_Attacks for an example of implementing these attacks with the FGSM and PGD methods.

Apply the Fast Gradient Sign Method (FGSM) and Projected Gradient Descent (PGD) attacks to create non-targeted adversarial examples using the first 200 images of the test set. Apply the following perturbation magnitudes: ϵ= [1/255, 3/255, 5/255, 8/255, 20/255, 50/255, 80/255]. Plot the overall accuracy of the model versus the perturbation size ϵ. For the FGSM attack, plot the first clean test image and the adversarial images with added adversarial perturbation of ϵ= [3/255, 8/255, 20/255, 50/255, 80/255], and display the predicted label.

Estimated time: between 10 and 20 minutes.

Report (20 marks): (a) Fill in Table 2 with the values for the classification accuracy for the clean images and perturbed images, with perturbation levels of ϵ=1/255, 5/255, and 8/255. (b) Plot the accuracy versus perturbation ϵ for FGSM and PGD adversarial attacks. (c) Plot figures with added adversarial perturbation and the labels for ϵ= [3/255, 8/255, 20/255, 50/255, 80/255]. (d) Provide an analysis of the results.

Table 2. Classification accuracy on clean and adversarial images.

Model	Clean images	Adversarial images ϵ=1/255	Adversarial images ϵ=5/255	Adversarial images ϵ=8/255
FGSM attack				
PGD attack				

Task 3: Implement non-targeted white-box CW evasion attack against the deep learning model from Task 1.

Use the Cleverhans library for implementing the attack, and apply the Carlini & Wagner L2 attack. An explanation of this attack and the used input parameters can be found in this link. Apply the attack using the first 10 images of the test set. Print the accuracy and the average perturbation by the attack for the set of 10 images.

Report (10 marks): (a) Print the accuracy and perturbation of the attack for the set of 10 images. (b) Write between 5 and 8 sentences to explain how the Carlini & Wagner L2 attack works and explain the difference of the attack in comparison to FGSM and PGD attacks.

Part 2: Transferability Attack (30 marks)

Create a substitute model for celebrity recognition, and transfer adversarial samples to a corresponding model hosted by Clarifai. This is a black-box attack, because we don't have access to the model hosted by Clarifai.

Task 1: Train a deep-learning Vision Transformer model for classification of images of celebrities.

Dataset: We will use a dataset of celebrity faces, which is a subset of a larger dataset called LFW (Labeled Faces in the Wild). The images are collected from the web, and are labeled with the name of the person. The dataset for this assignment consists of 5,113 images of 62 celebrities. The images have only the face of the person cropped out from the original images. Sample images are shown in Figure 1.

Use the provided Data Loader Part 2 file to load the dataset.

For the Vision Transformer, use the provided code named ViT_PyTorch as a guidance for training the model. The code employs a Vision Transformer (ViT) with a Linformer backbone. You should be able to use the same model with just a minor modification for the number of classes.

Perform hyperparameter tuning (learning rate and number of epochs) to obtain accuracy on the test dataset above 80%.

Estimated running time: between 5 and 20 minutes.

Report (20 marks): (a) Report the classification accuracy for the train set, validation set, and test set of images. For full marks, it is expected to report test accuracy above 80%. (b) Plot the training and validation loss and accuracy curves. (c) If applicable, provide any other observations regarding the training of the model.

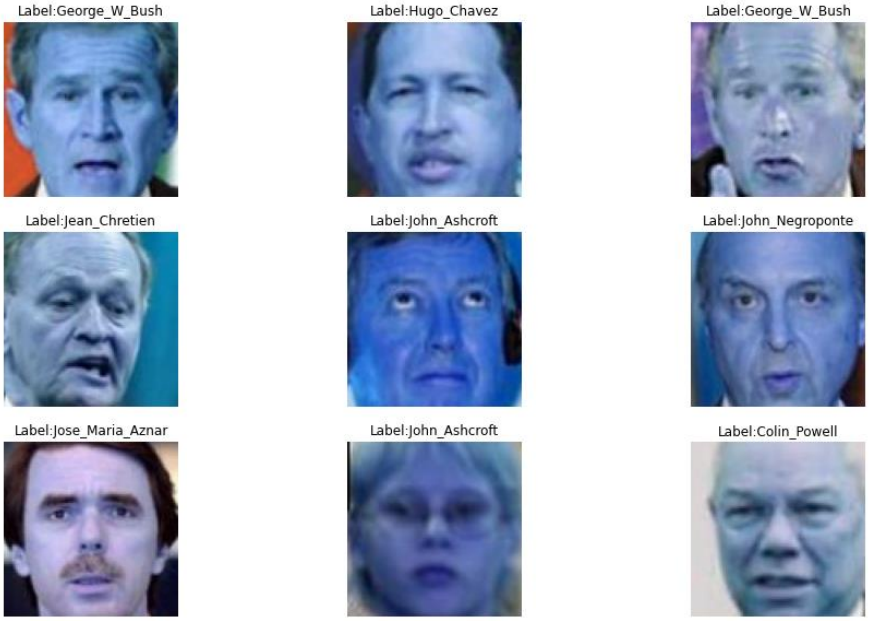

Figure 1. Sample images from the celebrity faces dataset.

Task 2: Create adversarial samples against Clarifai's web ML model for celebrity recognition.

Step 1: Select the image of Jennifer Lopez with index 343 from the test dataset for creating adversarial samples. Plot the image with the ground truth label and the predicted label by the DL model.

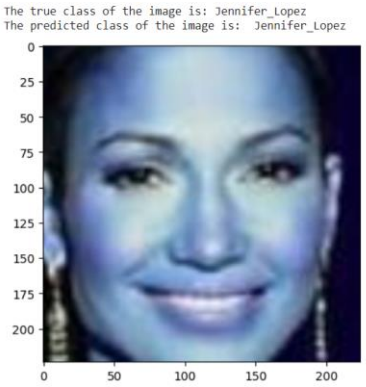

Figure 2. Selected image for the attack.

Step 2: Visit the Clarifai website: https://www.clarifai.com/models/celebrity-face-recognition

You will be prompted to sign in, and I believe that the easiest way is to sign in with an existing Gmail account or GitHub account.

The API is shown in Figure 7. Click on the "**+**" button and select "**Try your own image or video**" to upload the selected image in Step 1.

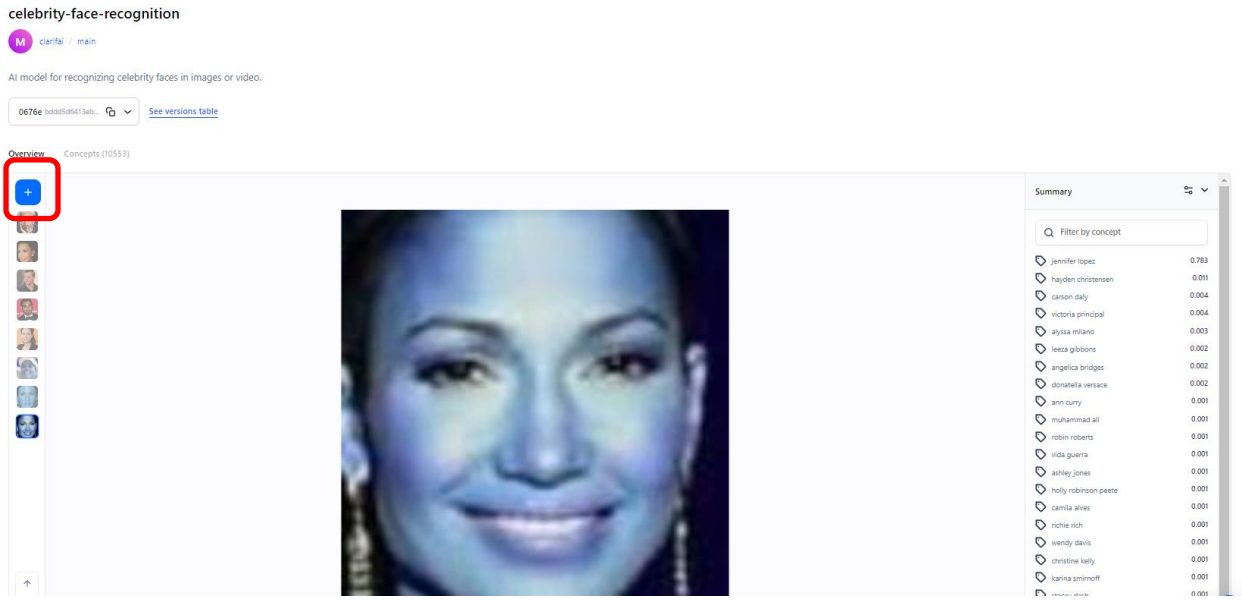

Figure 3. Clarifai API for celebrity recognition.

Note in Figure 3 that the model correctly classified the image of Jennifer Lopez with a high confidence of 0.783.

Step 3: Apply the PGD attack to create non-targeted adversarial sample for the image from Step 1.

In PyTorch, you can create a batch from one image using:

```
sub_dataset = Subset(test_dataset, [343])
subsest_dataloader = DataLoader(sub_dataset, batch_size=1)
```

You can save the adversarial sample with the following code:

```
import torchvision.transforms.functional as TF
from torchvision.utils import save_image
TF.to_pil_image(adversarial_image[0].cpu()).save('im1.jpg')
```

Upload the adversarial sample to Clarifai's website to check if it is misclassified.

Find the minimum perturbation level for the image to be misclassified by the Clarifai's model.

Plot the adversarial image with the lowest perturbation.

Step 4: Repeat the same steps for the image of David Beckham with index 442 from the test dataset. Find the minimum perturbation level for the image to be misclassified by the Clarifai's model.

Step 5: Repeat the same steps for the image of Winona Ryder with index 53 from the test dataset. Find the minimum perturbation level for the image to be misclassified by the Clarifai's model.

Estimated running time: between 5 and 10 minutes.

Report (10 marks): Plot the original images and the ground-truth label, the adversarial images, the predictions by the Clarifai's model, and state the applied level of perturbation. (b) Provide a brief discussion of your opinion of the target model, and whether you found it easy or difficult to perform the attacks.

Submission documents:

The assignment documents are submitted on Canvas. Note that it is possible to submit multiple files at the same time, just drag-and-drop the files or attach all the files while the submit window is open.

1. Submit all your codes as Jupyter Notebooks. Please try to comment your codes extensively, introduce the names of all used variables, avoid one-letter variables, etc., to improve the readability of the codes. For instance, for this assignment you can submit one single Jupyter Notebook with the codes. You don't need to submit the dataset on Canvas.
2. Prepare a brief report with tables, graphs, and results: the report can be prepared either as a separate MS Word/PDF file, or it can be integrated into your Jupyter Notebooks by typing your analysis directly into text cells in the Jupyter Notebooks.