Adversarial Machine Learning

Homework Assignment 1

The assignment is due by the end of the day on Tuesday, February 14.

Objective:

- Implement common white-box evasion attacks against deep learning-based classification models.

Note that the assignments and solutions from the offering of the AML course in Fall 2021 can be found at this link. These resources can be helpful for solving the assignments in this course. Also, if you need to refresh your knowledge about training neural networks, the materials from the Python Programming for Data Science course can be found here.

Dataset: We will use the Imagenette dataset. It is a subset of the well-known Imagenet dataset, and contains images of 10 objects (see examples in Figure 1 below). The dataset is split into training and test sets. The training set has 9,469 images, or about 1,000 images per each of the 10 classes. The test set has 3,925 images. There are three different versions of the dataset with different image resolutions. For this assignment we will use the version with low-resolution images of 160 pixels.

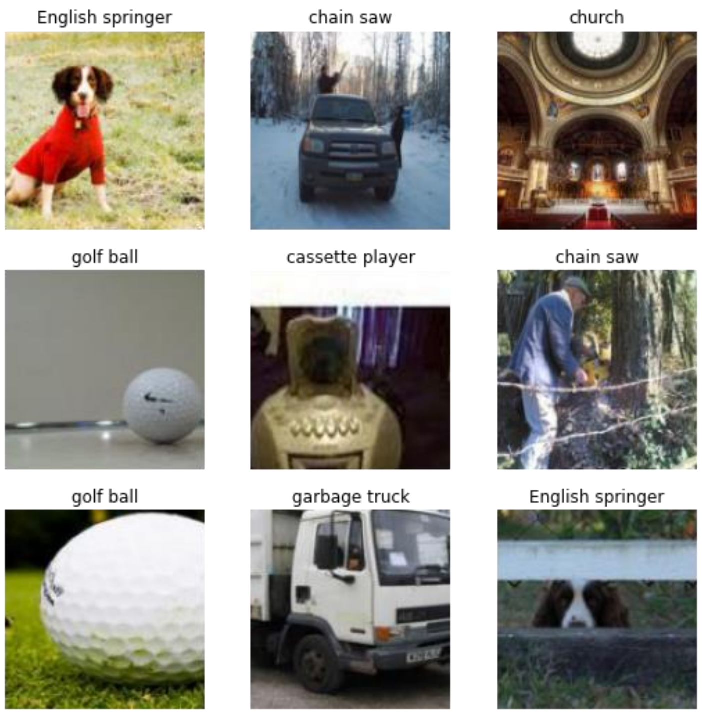

Figure 1. Example images from the Imagenette dataset.

You can download the dataset as a compressed file 'imagenette2-160.tgz' (95 MB) from this Shared folder on OneDrive. If you need more information about the dataset, please check this link. Also, Imagenette is available in the Tensorflow Datasets, so it can be downloaded from there as well.

A starter code for loading the dataset is also available in the Shared folder. If you wish, you can write your own code for loading the dataset, and don't necessarily need to use the provided file.

Task 1: Train a deep-learning model for classification of the Imagenette dataset.

The recommended libraries for implementation are: Keras, TensorFlow, or PyTorch.

For GPU access, Google Colab Pro is recommended ($10 per month). It is also possible to use the free GPU access by Google Colab, although it has limitations. Or, if you have access to GPU from other sources, that is also fine.

An easy and fast way to complete this task is to adopt a pretrained VGG-16 or ResNet (e.g., ResNet50) model, and just add a classifier head on top of the pretrained backbone.

Perform hyper-parameters tuning of your models to obtain an accuracy on the test dataset above 90%.

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%. You don't need to report other performance metrics, since the focus is on adversarial attacks. (b) For each model, plot the training and validation loss and accuracy curves (similar to the example in Figure 2). (c) If applicable, provide any other observations regarding the model or the dataset.

Table 1. Classification accuracy.

Model	Train Set	Validation Set	Test Set

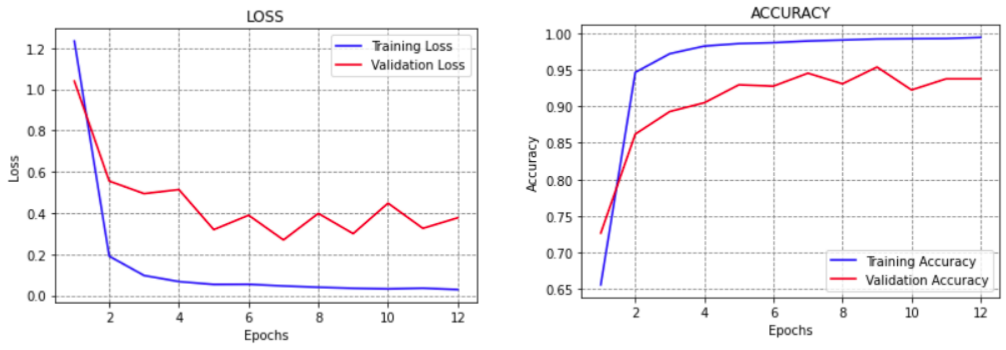

Figure 2. Loss and accuracy plots for a DL model. This is an example, and not the actual expected plot.

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

The attacks to be implemented are Fast Gradient Sign Method (FGSM), and Projected Gradient Descent (PGD).

Codes for the attacks can be found in several libraries: Adversarial Robustness Toolbox, cleverhans, or scratchai. Adversarial Robustness Toolbox is currently the most comprehensive and reliable library, and I recommend to use it for the assignment, but using any other libraries of your choice is also acceptable. For example, this notebook explains how to apply adversarial attacks in ART using the Keras library. Similarly, there are many other notebooks providing explanations on how to use the various attacks and defenses in the ART library.

Apply FSGM and 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 two models versus the perturbation size ϵ (e.g., the expected plot should look like Figure 3, note that ϵ=80/255≈0.3). 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 (the figure should look similar to Figure 4 below). When you run the codes, you will understand better why FSGM is called a fast method.

Estimated time: between 10 and 20 minutes.

Report (30 marks): (a) Fill in Table 2 with the values for the classification accuracy for the clean images and perturbed images, with perturbations levels of ϵ=1/255, 5/255, and 8/255. (b) Plot the accuracy versus perturbation ϵ for FSGM and PGD adversarial attacks (similar to the example in Figure 3). (c) Plot figures with added adversarial perturbation and the labels for ϵ= [3/255, 8/255, 20/255, 50/255, 80/255], similar to the examples in Figure 4. (d) Provide an analysis of the results. (e) Explain the meaning of the perturbation magnitude ϵ=8/255 (we know that this is added perturbation noise, so the expected answer should be more specific than that, e.g., does it relate to a distance metric, norm, how it is applied to the pixels in images, etc.).

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				

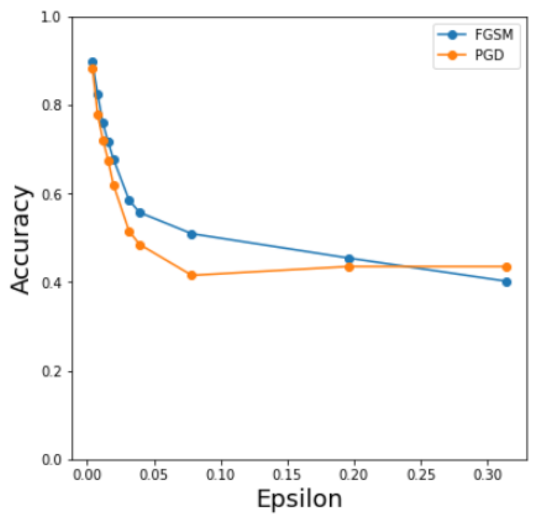

Figure 3. Pots of accuracy versus perturbation. This is an example figure, and not the plot for this task.

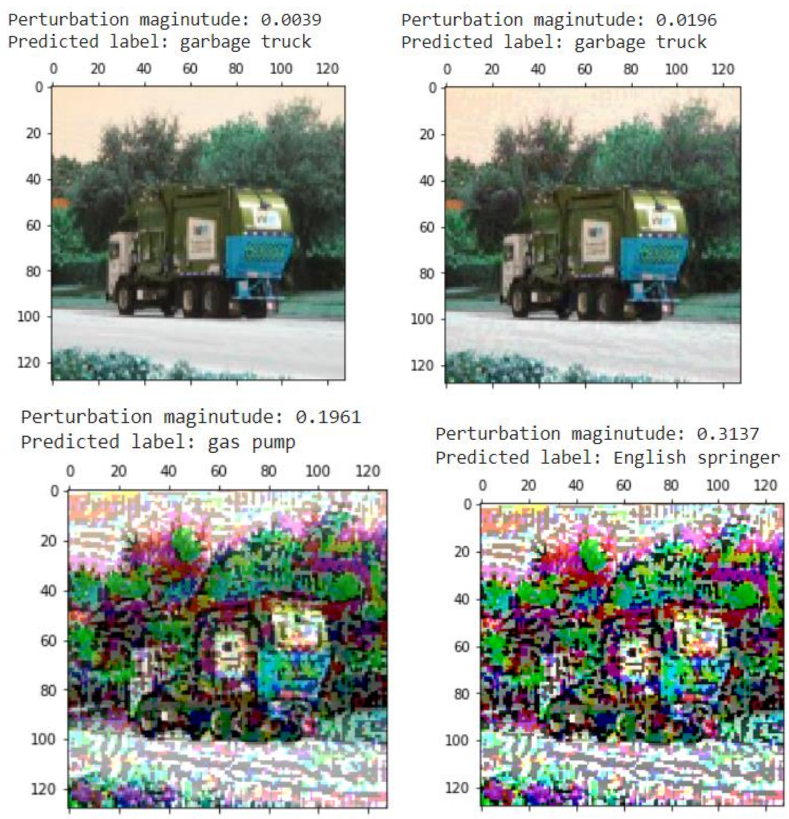
Perturbation maginutude: 0.0039
Predicted label: garbage truck
Perturbation maginutude: 0.0196
Predicted label: garbage truck
Perturbation maginutude: 0.1961
Predicted label: gas pump
Perturbation maginutude: 0.3137
Predicted label: English springer

Figure 4. Examples of adversarial images.

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

Step 1: For these attacks, we will use the 'church' images (class label 4) from the test dataset. Therefore, first extract all test images that have the label class 4. There are 249 such images.

Next, implement FGSM attack on the 'church' images to misclassify them as 'gas pump' images (class label 7). Vary the perturbation magnitude ϵ= [1/255, 3/255, 5/255, 8/255, 20/255, 50/255, 80/255], and report the percentage of adversarially manipulated images that are assigned by the classifier the label 'church', and the percentage of images that are assigned the label 'gas pump'.

Step 2: Repeat the attack with Projected Gradient Descent (PGD), and discuss the performance in comparison to FGSM.

Estimated time: between 15 and 30 minutes.

Report (30 marks): (a) Fill in Table 3 with the values of the classification accuracy for source and target class labels with FGSM and PGD attacks. Discuss what perturbation sizes achieve the best performance, and discuss the differences in the success rate by FGSM and PGD attacks. (b) Plot 5 examples of the original church images and the corresponding adversarial images, as in Figure 5.

Table 3. Classification accuracy of the model on adversarially manipulate images.

Perturbation Level	FGSM attack – church label	FGSM attack – gas pump label	PGD attack – church label	PGD attack – gas pump label
ϵ=1/255				
ϵ=3/255				
ϵ=8/255				
ϵ=20/255				
ϵ=50/255				
ϵ=80/255				

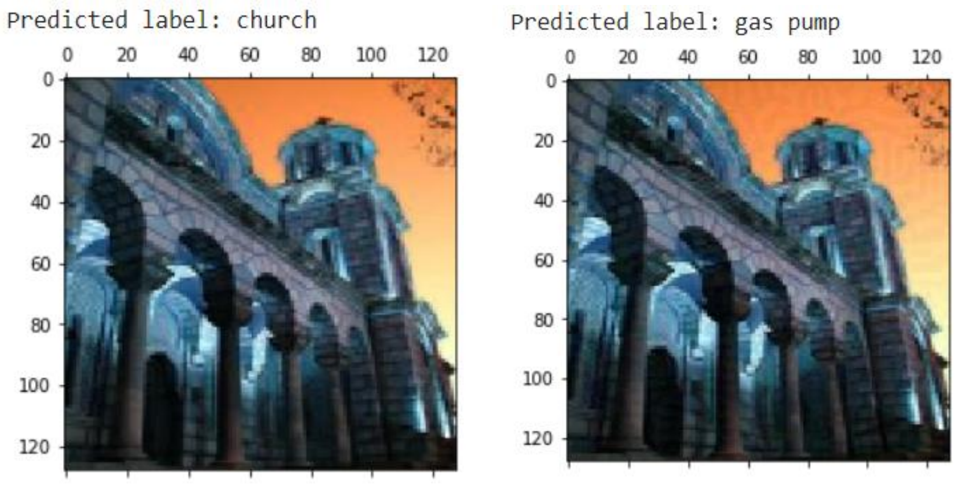

Figure 5. Original and adversarial images.

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. You don't need to submit the dataset. For instance, for this assignment you can submit just one Jypyter Notebook with all three tasks. Or, you can submit separate notebooks for each task, or do it your way.
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.