Adversarial Machine Learning

Homework Assignment 2

The assignment is due by the end of the day on Thursday, October 14.

Objectives:

- Implement adversarial defenses for white-box evasions attacks against deep learning-based classification models.

Part 1: Binary Input Detection Defense **50 marks**

Binary Input Detector trains a neural network with original clean images having labels 0, and adversarially perturbed images having labels 1. The trained NN then predicts whether new images are clean or adversarially manipulated images. Only clean images will be sent to the main classifier model to predict their class; the detector will refuse to send adversarial images to the main classifier.

The easiest way to apply this defense is by using the Adversarial Robustness Toolbox, which has an implemented function BinaryInputDetector. The ART library also provides a Jupyter notebook example showing the detection of images from the CIFAR10 dataset.

Dataset: We will use a dataset of celebrity faces provided at https://github.com/telecombcn-dl/2018-dlai-team4, which is a subset of a larger dataset called LFW (Labeled Faces in the Wild). The images are collected from the web, and 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, and are resized to 100×100 pixels. You can download the dataset (12MB) and a simple starter code for loading the dataset from the Shared folder on OneDrive.

Figure 1. Sample images from the celebrity faces dataset.

Task 1: Train a VGG16 deep-learning classification model on the celebrity faces dataset. It is expected to achieve a classification accuracy on the test set greater than 90%. Use a pretrained model, and you may need to do some finetuning, but you shouldn't expect major difficulties.

Report (5 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 a test set accuracy above 90%. (b) Plot the training and validation loss and accuracy.

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

Use the following three adversarial attacks: Fast Gradient Sign Method, Projected Gradient Descent, and DeepFool.

Use the set of 100 images called 'adv_test_imgs' in the Data_Loader notebook for creating the attack.

Step 1: Create adversarial samples using FGSM attack on the 'adv_test_set' set of 100 images, against the trained VGG16 classifier. Find a perturbation size ϵ for FGSM attack that achieves less than 40% accuracy on the adversarial images.

Step 2: Create adversarial samples using PGD attack on the 'adv_test_set' set of 100 images, against the trained VGG16 classifier. Find a perturbation size ϵ for PGD attack that achieves less than 20% accuracy on the adversarial images.

Step 3: Create adversarial samples using DeepFool attack on the 'adv_test_set' set of 100 images, against the trained VGG16 classifier. Find a perturbation size ϵ for DeepFool attack that achieves less than 30% accuracy on the adversarial images.

It is preferred to use $\epsilon \leq \frac{10}{255} \approx 0.039$, if possible. For most attacks perturbation of about $\epsilon = 5/255$ should achieve the goal. If needed, apply finetuning to some of the parameters for the attacks (e.g., the perturbation step size for PGD, number of iterations for DeepFool, etc.) to reach the required accuracy for the attacks.

Report (15 marks): (a) Fill in Table 1 with the values for the classification accuracy on the clean 'adv_test_set' of 100 images and perturbed 'adv_test_set' of 100 'adv_test_set' images. For full marks, the classification accuracy on the adversarial images should be less than 40% for FGSM, less than 20% for PGD, and less than 30% for DeepFool. (b) For each attack, plot a random set of 9 adversarial images, and display the true label and the model prediction, similar to the shown image in Figure 2. (c) Provide an analysis of the results.

Table 1. Classification accuracy on clean and adversarial test images.

Attack	Clean images ('adv_test_set')	Adversarial images ('adv_test_set')	Perturbation ϵ
VGG16 - FGSM			
VGG16 - PGD			
VGG16 - DeepFool			

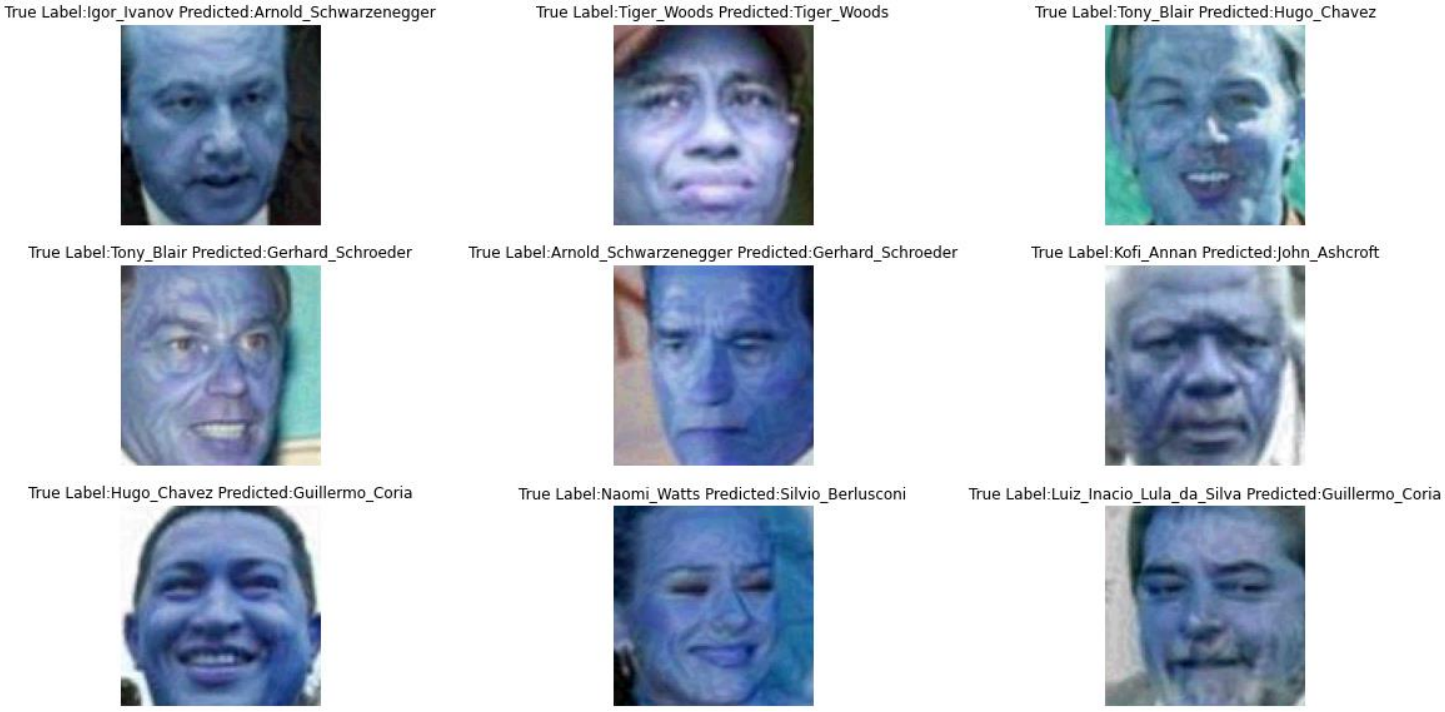


Figure 2. Sample of adversarial images created by FGSM.

Task 3: Implement binary input detection defense to identify clean and adversarial images.

Step 1: Now use the settings for the attacks from Task 2 and create adversarial images for the train set of images (3,043 images) with the FGSM attack (it is also named 'adv_train_imgs_fgsm' in the Data_Loader notebook_. Ensure that the classification accuracy on the adversarial train images (3,043 images) is again less than 40% (e.g., if needed, adjust the perturbation level).

Step 2: Train a Binary Input Detector model on the dataset consisting of clean train set of images and adversarial train set of images. Use the same (or similar) VGG16 architecture as the classifier model, but change the output to two classes (clean and attacked images). Train for 10-30 epochs until you obtain high accuracy (e.g., 95% or closer to 100%), but also don't overfit it (e.g., don't let it train for additional 20 epochs after the train accuracy reached 100%). For the trained model, report the success rate of the detector (i.e., ratio of detected vs total images) on the original clean 3,043 images and on the adversarially manipulated 3,043 images.

Step 3: Report the success rate in detecting adversarially manipulated images for the subset of adversarial images 'adv_test_imgs' from Part 2 for all three attacks (i.e., report the success rate on the test images attacked with FGSM, PGD, and DeepFoool).

Step 4: Evaluate the success rate for detecting adversarial FGSM images from the subset of 100 'adv_test_imgs' images using the following levels of perturbation $\epsilon = \{1/255,\ 3/255,\ 5/255,\ 10/255,\ 20/255\}$. Display a plot of the success rate versus the level of perturbation, similar to the one presented in Figure 3.

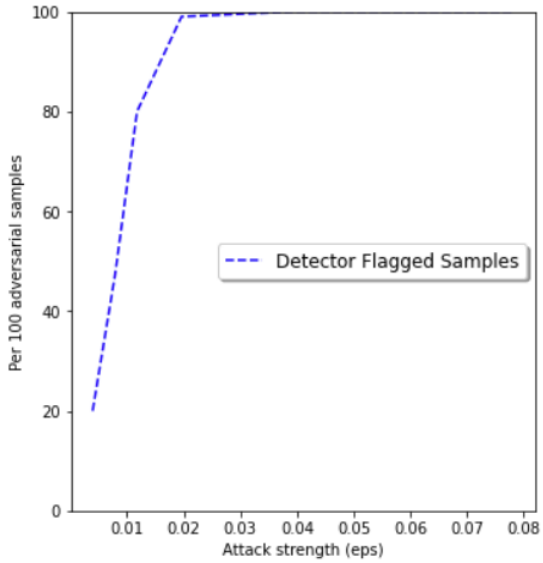


Figure 3. Number of correctly detected FGSM attacked images by the FGSM detector.

Step 5: Repeat steps 1-4 to train Binary Input Detector models for PGD attack, by using the adv_train_imgs_pgd set of 1,000 images from the Data_Loader notebook. Ensure that the classification accuracy on the adversarial train images is less than 20% for PGD.

Step 6: Repeat steps 1-3 to train Binary Input Detector models for DeepFool attack, by using the adv_train_imgs_df set of 100 images from the Data_Loader notebook. Ensure that the classification accuracy on the adversarial train images is less than 30% for DeepFool. Step 4 is not required for DeepFool attack.

Overall, train three individual Binary Input Detector models by using clean train set images and adversarial train set images created by FGSM, PGD, and DeepFool.

Report (30 marks): (a) Fill in Table 2 with the values of the success rate in detecting clean and adversarial images for all attacks. For full marks, it is expected to report a success rate (classification accuracy) by the classifiers on the train set of adversarial images (column 2 in Table 2) less than 40% for FGSM, less than 20% for PGD, and less than 30% for DeepFool. Also, the success rate in detecting adversarial images by the detector (column 4 in Table 2) is expected to be over 70% on the 'adv_test_imgs' subset of 100 images for that attack (i.e., the detector trained on PGD adversarial images needs to able to detect at least 70% of the test set of PGD adversarial images, but this does not pertain to adversarial images generated by FGSM or DepFool). (b) For the detectors against FGSM and PGD attacks (and not DeepFool, since it will take a long time), plot the success rate on the 'adv_test_imgs' subset of 100 images versus perturbation levels of $\epsilon = \{1/255,\ 3/255,\ 5/255,\ 10/255,\ 20/255\}$, similar to Figure 3. (c) Discuss the performance of the binary input detection defense approach, and provide any other observations related to this defense approach and the individual attack methods.

Table 2. Detection **success rate** by the Binary Input Detector.

	Clean train images (adv_train_imgs)	Attacked train images (attacked adv_train_imgs)	Clean subset of test images (adv_test_imgs)	Attacked subset of test images (attacked adv_test_imgs)	Subset of test images (adv_test_imgs) attacked by	Subset of test images (adv_test_imgs) attacked by
FGSM					PGD:	DF:
PGD					FGSM:	DF:
DeepFoool					FGSM:	PGD:

Part 2: Adversarial Training Defense **50 marks**

Implement an **Adversarial Training defense** for improving the robustness of a classifier to adversarial attacks. Adversarial Training approach trains a classifier using both clean images and adversarially perturbed images, but unlike the detector model in Part 1 that is trained using 2 labels – clean and adversarial samples, the classifier in Adversarial Training is trained by using all class labels in the dataset – i.e., 62 class labels for the celebrity faces dataset.

The Adversarial Robustness Toolbox offers a function AdversarialTrainer, which can be used for this task. The following Jupyter notebook example demonstrates how to implement the defense on the MNIST dataset.

Task 1: Implement adversarial training defense to harden a classifier against adversarial examples.

Step 1: Use the VGG classifier from Part 1 trained on the celebrity faces dataset. Either reuse the adversarial images for the 'adv_test_set' of 100 images created with FGSM, PGD, and DeepFool attacks from Part 1, or create new adversarial images on 'adv_test_set' for the FGSM, PGD, and DeepFool attacks. Report the classification accuracy by the VGG classifier on the clean and attacked images in the first row in Table 3.

Step 2: Implement an Adversarial Trainer on the VGG model using Projected Gradient Descent attack with a perturbation size of $\epsilon = 5/255$ on the train set of 3,043 images. The ART function will do everything for you, and it will first create adversarial images for the train set of images, and afterward it will train the model. (If you have GPUs issues, like out of memory messages, it is okay to use a smaller subset of the train set (e.g., 1,000 images) for training the Adversarial Trainer).

Step 3: Evaluate the accuracy by the robust classifier on the clean train images, on the subset of clean 'adv_tst_imgs,' and the corresponding adversarial test images attacked by FGSM, PGD, and DeepFool. Report the classification accuracies in the second row in Table 3. It is required to achieve a classification accuracy of at least 80% on the adversarially manipulated images with PGD.

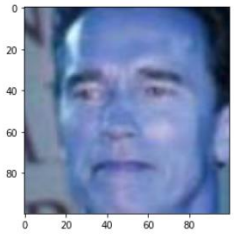
Report (40 marks): (a) Fill in Table 3 with the classification accuracy on the clean train and test images, and attacked images with FGSM, PGD, and DeepFool. For full marks, the classification accuracy on the PGD attacked 'adv_test_imgs' should be at least 80 %. (b) Write a brief analysis of the results, and compare the performance of adversarial trainer and the binary input detector.

Table 3. Correctly classified image.

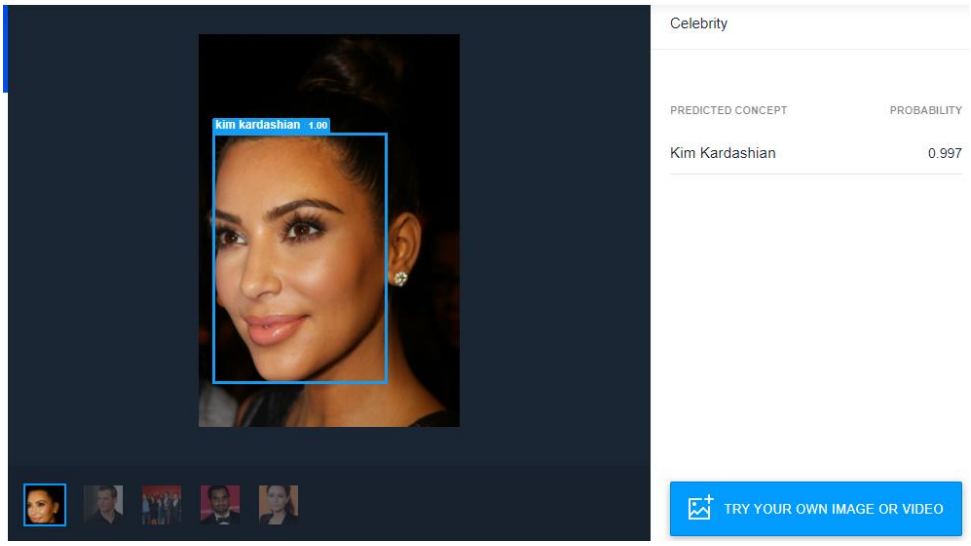
	Clean train images (3,043)	Clean subset of test images (adv_test_imgs)	FGSM attacked subset of test images (adv_test_imgs)	PGD attacked subset of test images (adv_test_imgs)	DF attacked subset of test images (adv_test_imgs)
VGG classifier					
Adversarially trained classifier					

Task 2: Try a simple attack on clarifai's web ML model for celebrity's recognition.

Use the 'clarifai_img' from the Data_Loader notebook. It is a single image of Arnold Schwarzenegger, shown below.



Select 'Try Your Own Image', and check if the clean image is correctly classified as Arnold Schwarzenegger.



Next, apply the PGD attack to the image, and find the minimum perturbation level for the image to be misclassified by the clarifai's model. E.g., you can save the attacked image by using `imageio.imwrite(path +'img1.jpg', adv_img.astype(np.uint8))` and upload it to the website.

Report (10 marks): Plot the final image that is misclassified, and state the applied level of perturbation. Provide a brief discussion.

Submission documents:

Submit all codes as Jupyter Notebooks, and a brief report with tables, figures, results (either as an MS Word/PDF file, or commented Jupyter notebooks). Provide a short analysis of the results, and your opinion on the performance of these two defense approaches.