AI for Cybersecurity - Speaker recognition

Group 9:
Rocco Michele Capobianco
Eugenio Carpentieri
Fausto Celentano
Valerio Costantino

1 Experimental Setup

Our setup is based on the Adversarial Robustness Toolbox (ART) library, Tensorflow and Keras. The experiments have been performed on a Tesla T4 using Google Colab.

1.1 Dataset

The dataset used for the experiments is a subset of the VoxCeleb [1]dataset. It is composed by 152799 samples from 1251 identities. The number of samples for each identity is somewhat balanced. In fact, the "positive" identity class was chosen by selecting the one with largest number of samples, namely the "id10300" with 304 samples.

Then, to select the data for the "negative" identity class, a balanced number of samples have been taken from the remaining 1250 identities.

The final data used for training is composed by 304 positive and 304 negative samples, for a total of 608. This dataset has been splitted into training, validation and test set with a 15% split for both validation and test.

The validation set has been used to optimize the training of the model while the test set has been used to perform and evaluate the different attacks that have been tested.

1.2 Classifier architecture and training

We designed two classifier. Both of them are based on the "SpeakerID" model that has been used as a feature extractor.

This model is a multi-class classifier that recognized the speaker among the 1251 identities of the original dataset. It is composed by a pre-processing layer that computes the Log-mel spectrogram and a Resnet-18 as a backbone. We discarded the top dense layer to obtain obtain feature vectors of size 512 of our samples.

The first classifier we designed (classifier A) is a binary classifier using a three-layer MLP from the computed feature vectors. The first layer has 32 neurons, the second has 16 neurons and the last has 2 neurons, each of which represents the probability of the two classes in one-hot encoding. We chose not to have a one-neuron layer since, to perform DeepFool and Carlini-Wagner attacks using the ART library, it is required to provide a multi-class classifier instead of a binary classifier. We show the architecture of this model in the figure below.

To train the classifier A, we chose a sparse categorical cross entropy, since it is a multi-class classifier with two classes as stated above, and with Adam optimizer and to assess the performance we computed the accuracy on a validation set that is the 15% of the original data.

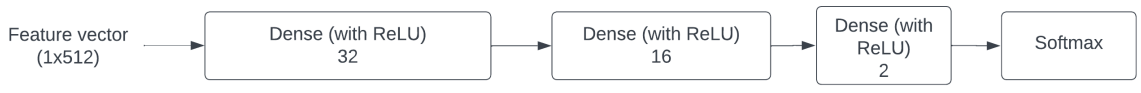


Figure 1: Architecture of the MLP model

Then we proceeded to build a second model (classifier B) as a K-NN binary classifier using the sci-kit learn library. This model takes as input the feature vector to speed up the training and the evaluation since it will be used only to assess the transferability of the attacks performed on classifier A. The nearest neighbour criterion used is the cosine distance, since it is magnitude invariant and is less sensitive to scale changes between vectors than Euclidean distance.

Since the feature representation already has good performance and the problem we are facing is a simple binary classification, we reached 100% accuracy on our test set with both classifiers.

2 Attacks

Every tested attack has been performed on the classifier A model (MLP) using the test set samples to generate adversarial samples, that means that the attacks are white-box. In particular, we decided to apply untargeted attacks, this means that we want to misclassify a sample with any of the classes different from the true one. In our case, this means that we want to misclassify an authorized user as a non-authorized one and vice versa.

We performed four kind of attacks:

- FGSM
- BIM
- PGD
- Carlini-Wagner

The first three aim to compromise the accuracy of the model by using gradient information and they represent an improvement of the previous model (PGD for BIM and BIM for FGSM). Carlini-Wagner is, instead, based on optimization problem.

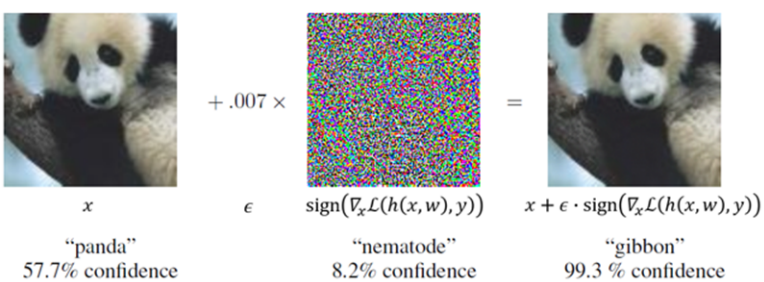
For each of these attacks, we plotted security evaluation curve for each parameter and a general one with perturbation as x-axis that is computed as the $\mathcal{L}_1$ norm of the difference between the clean sample and the adversarial sample. It is important to note that the general perturbation plot depends on several possible combinations of parameters of the specific attack and it is sorted from lowest to highest and thus it represent an average trend of the attack performance.

2.1 FGSM

Fast-Gradient Sign method [2] is a white-box attack, i.e., the attacker has complete knowledge of the neural network to be attacked (or a surrogate for the network to be attacked) The logic is to exploit the knowledge of the loss function by creating a sample that increases the loss for the true class. The algorithm is as follows for each test sample x':

- Take the original sample x of the true class y.
- Make predictions about x using the known model $h(x, w)$ where w are the weights of the model.
- Calculate the loss of the prediction based on the true class label y, i.e., $\mathcal{L}(h(x, w), y)$
- Calculate the gradients of the loss with respect to x, i.e., $\nabla_x \mathcal{L}(h(x, w), y)$
- Calculate the sign of the gradient, i.e. $sign(\nabla_x \mathcal{L}(h(x, w), y))$
- Use the signed gradient to construct the output adversarial sample with a perturbation proportional to the noise magnitude ϵ:

$$x' = x + \epsilon \cdot sign(\nabla_x \mathcal{L}(h(x, w), y)) \tag{1}$$



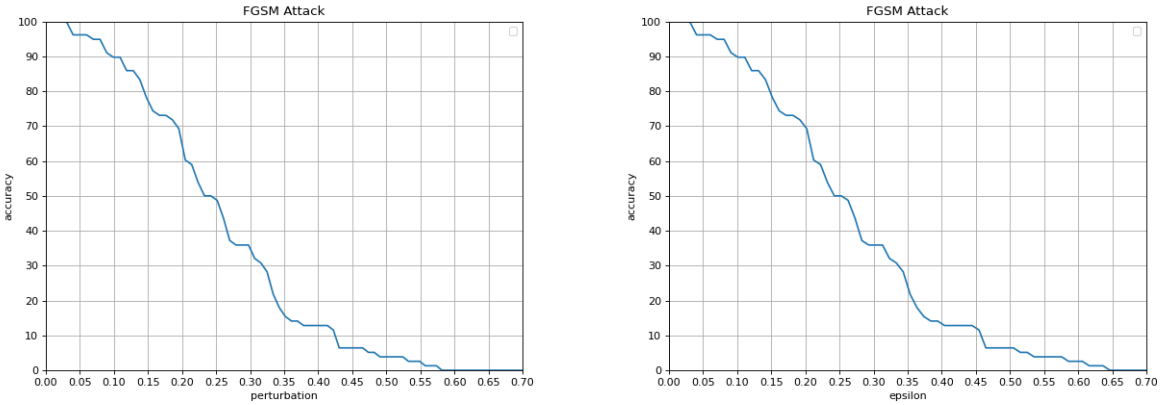


Figure 2: Security evaluation curves on FGSM attack

The only parameter to tune to perform the attack is ϵ. We chose to generate 100 equally spaced points between 0 and 1 and to evaluate the perturbation and the accuracy of the classifier in each of those.

Since there is only one parameter i.e. ϵ, we can see from the plots that there is linear relation between the perturbation and the parameter. After $\epsilon = 0.45$, the accuracy of the classifier is below 10%.

2.2 BIM

The Basic Iterative Method (BIM)[3] is a variant of FGSM: it repeatedly adds noise to the sample x over multiple iterations in order to cause misclassification. With FGSM we are limited because we have only one attempt to make, we have no way to change the approach. With BIM, on the other hand, the same process is iterated. The number of iterations is t and α is the amount of noise that is added at each step:

$$x'_t = x_{t-1} + \alpha \cdot sign(\nabla_x \mathcal{L}(h(x, w), y) \tag{2}$$

Which is very similar to what we had with FGSM, but more steps (with smaller α) of adding noise increases the chances of misclassification of the sample. In this case the parameter α can be smaller in each iteration than ϵ because before with FGSM we had only one attempt and therefore we have to have ϵ which is the largest possible, but if we have more attempts we can use smaller parameters because in this way we can have a larger margin on perturbations.

Using the ART library, we have the following parameters:

- epsilon (that we chose to compute as $eps\_step \cdot max\_iter$)
- eps step (that is the α parameter in the equation 2)
- max iter (that is the t parameter in the equation 2)

We chose to generate 100 equally spaced points between 0.0001 and 1 for the eps step parameter. Instead, we let max iter vary between these values: [3, 5, 7, 10, 20, 30].

As expected and as we can see from the plots, on average this attack performs better i.e. with the same perturbation we get a lower accuracy on the adversarial samples.

On the right figure, we can see that increasing the number of iterations of the algorithm gives us a more powerful attack and thus, we can reduce the step size of ϵ when we increase the number of iterations.

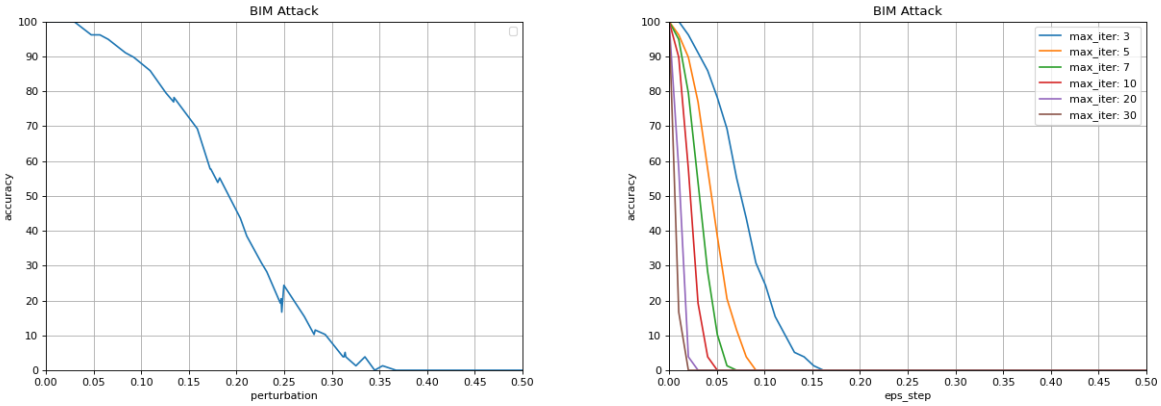


Figure 3: Security evaluation curves on BIM attack

2.3 PGD

Projected Gradient Descent (PGD) [4], with respect to BIM, introduces an element of randomness, i.e., at each iteration it adds a small variation on the sample to be attacked, so ϵ becomes the maximum perturbation and the point where we move to is in a neighborhood given by ϵ using a projection function Π.

$$x'_t = \Pi_\epsilon(x_{t-1} + \alpha \cdot sign(\nabla_x \mathcal{L}(h(x_{t-1}, w), y))) \tag{3}$$

PGD uses a random initialization for x, adding random noise from a uniform distribution with values in the range $(-\epsilon, \epsilon)$. PGD is considered to be the strongest first-order attack i.e. attacks that uses only the gradients of the loss function with respect to the input. The ART implementation has the following parameters:

- epsilon (that we chose to compute as $eps\_step \cdot max\_iter$)
- eps step (that is the α parameter in the equation 3)
- max iter (that is the t parameter in the equation 3)
- num random init (number of random initializations within the epsilon ball, is the Π_ϵ in the equation 3)

This can be shown from the plot that we are showing below. As we can see, the average trend is similar to the one shown on the BIM attack but better than the FGSM one.

Regarding the parameter choice, we can see that increasing max_iter leads to a better performance while changing num_random_init does not show any particular improvement. We can also see that the results are similar to the ones obtained with the BIM attack. We thought that this might be due to the fact that the model is very simple and thus not sensible to the difference between the two attacks.

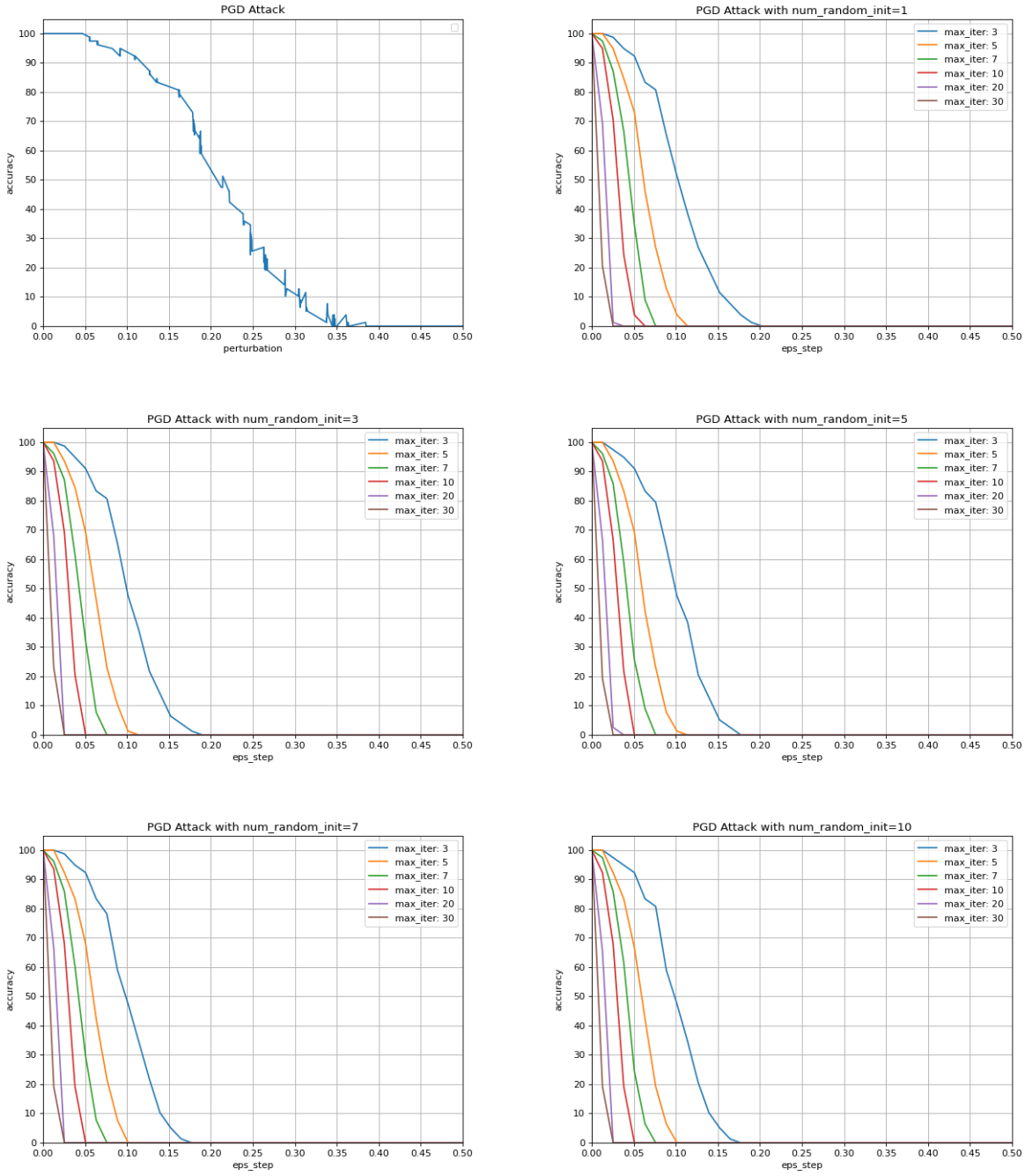


Figure 4: Security evaluation curves on PGD attack

2.4 Carlini-Wagner

The Carlini-Wagner attack [5] is defined as an optimization procedure. The goal of this optimization procedure is to search for the minimum distance D that guarantees us that the new sample has a class T, where the class T is any class other than the right one.

The algorithm proceeds applying noise patterns iteratively with the goal of minimizing the distance that we choose to evaluate that can be one among: $\mathcal{L}_0$, $\mathcal{L}_1$, $\mathcal{L}_2$ and $\mathcal{L}_\infty$ norms. We chose to consider the $\mathcal{L}_2$ norm because this is the one that should be able to give us a measure of the variation on the sample that a human perceives.

Since it is an optimization procedure that proceeds in steps and stops when it reaches the target, we easily understand that this attack will be the slowest one of all. In fact, the biggest obstacle of using Carlini-Wagner attacks is the computational load for generating the attacks.

In the ART library, this method requires a set of parameters:

- confidence (a higher value produces examples that are farther away, from the original input, but classified with higher confidence as the target class)
- learning rate (smaller values produce better results but are slower to converge)
- binary search steps (number of times to adjust constant with binary search
- max iter (the maximum number of iterations)
- initial const (the initial trade-off constant c to use to tune the relative importance of distance and confidence)

Due to computational problems, we had to keep some parameters fixed and other variable. In particular, we fixed confidence with a value of 0 and binary search steps with 1 since, from our tests, they provided some good results when varying the other parameters.

The learning rate has been made to range between 0.0001 and 1 selecting 100 equally spaced points. Moreover, we let max iter and initial const vary between these values: [1, 3, 5, 7, 10, 20, 30].

We show the plots of the security evaluation curve for this attack. From the plots, we can observe that the general trend is a bit more steep but more variable, especially between a perturbation of 0.20 and 0.30, with some unexpected spikes. This could be due to the fact that the plot is averaging between all possible parameter combination, for example two particular combinations might have similar perturbation but different accuracy.

Moreover, we can see that changing the initial const is meaningful since binary search step is fixed with a low value (1), thus increasing initial const lead to stronger attacks. This behavior is present also for the max iter parameter. For the learning rate, very low and very large values provide worse attacks, this is more relevant for a low value of max iter (for example 1, the blue line).

Figure 5: Security evaluation curves on Carlini-Wagner attack

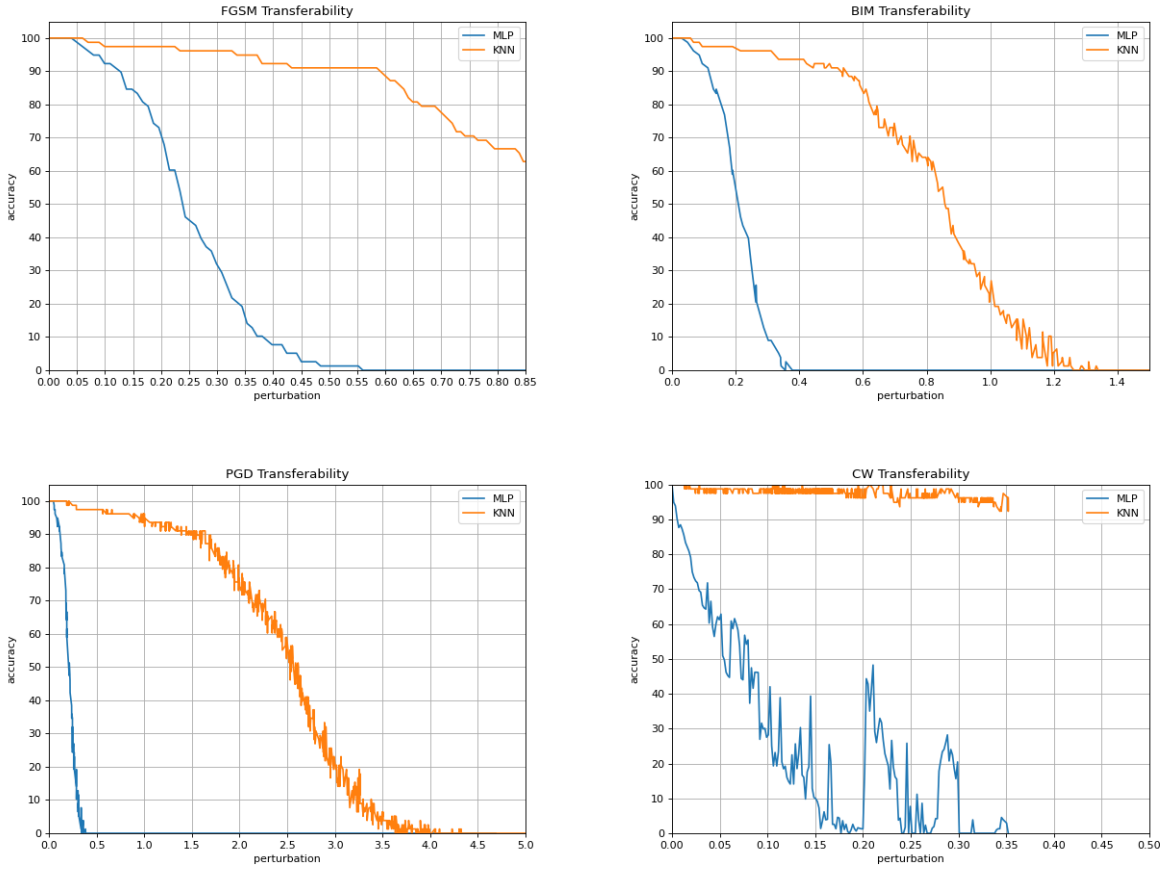


Figure 6: Transferability curves

3 Transferability

Transferability of an attack refers to the ability of an attack performed on a given classifier to be "transferred" to other similar models. In other words, if an attack is successful on one model, it might also be able to be successful on other models using the same type of architecture, or on models trained on similar datasets.

Gradient based attacks such as FGSM or BIM are very effective even on other models because the algorithm that generates the attack is not as driven towards the attacked model. On the other hand, Carlini-Wagner and DeepFool attacks, which work a lot on separation regions and optimization procedures, are very effective on the network being attacked, but show less transferability to other networks.

We evaluated the transferability of the attacks on a K-NN model trained on the same data of the MLP model. The plots show that the attacks on the K-NN model does not have the same effect on the MLP. One reason for this result may be that it was performed with some specificity on the first model, using a detailed knowledge of its internal structure.

4 Defense

4.1 Adversarial Training

Adversarial training is a technique used for defending a model from adversarial attacks and it consists in training the same model using both original and adversarial samples. The goal of adversarial training is to strengthen the network with respect to possible attacks, but without losing accuracy on the original clean samples, and it may be that by adding more and more corrupted samples maybe we generalize better on average on all these variations, but we lose accuracy on the clean samples.

To perform Adversarial Training we used the AdversarialTrainer object of the ART library. The object takes as input the classifier to be defended, one or more attack's configurations and the ratio of adversarial samples in the new training set with the respect to the original ones.

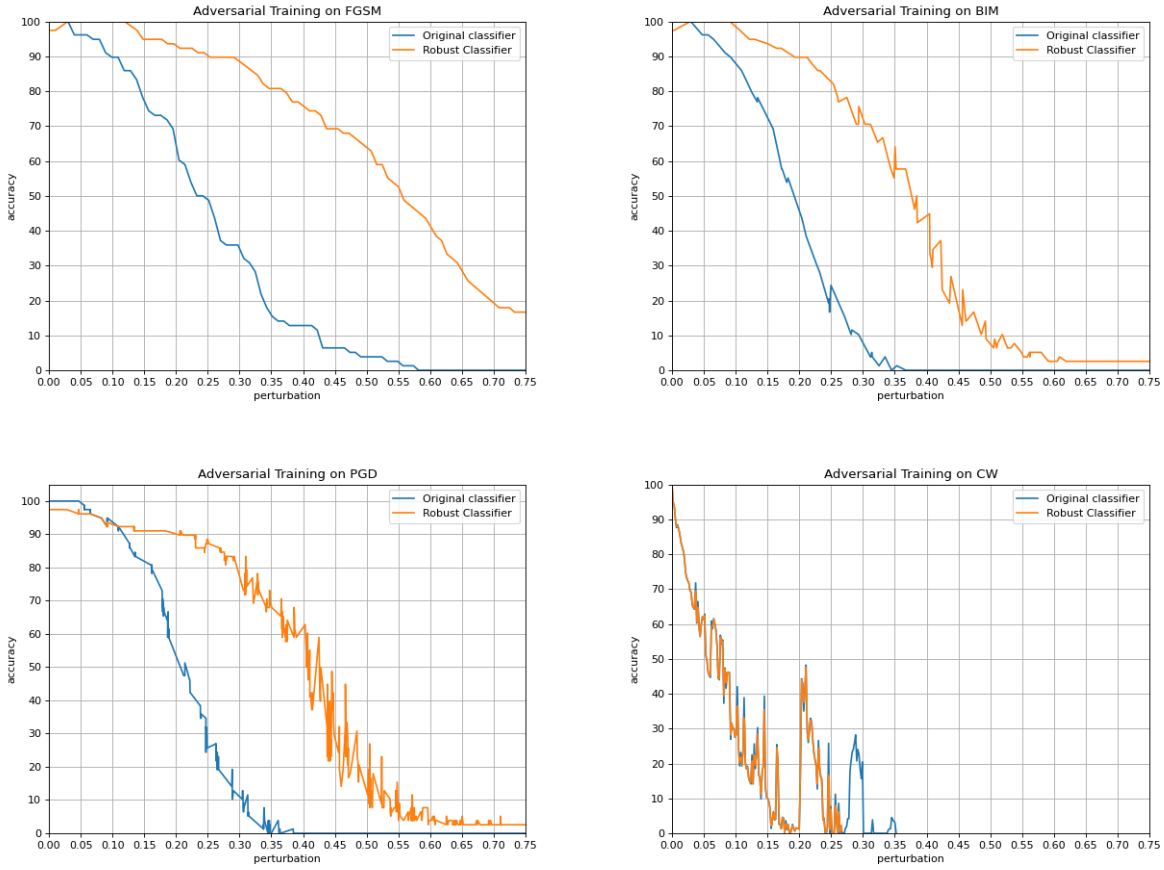


Figure 7: Adversarial training curves

We chose to include all four tested attacks with these configurations:

- FGSM with $\epsilon = 0.4$
- BIM with $eps\_step = 0.05$ and $max\_iter = 10$
- PGD with $eps\_step = 0.05$, $max\_iter = 10$ and $num\_random\_init = 10$
- CW with $binary\_search\_steps = 1$, $confidence = 0$, $max\_iter = 3$, $learning\_rate = 0.3$ and $initial\_const = 10$

This choice has been made because we wanted to have a more robust defense that has not be achieved using only a subset of these four attacks in our tests. The chosen configurations are quite strong since the original classifier with these kind of attacks has an accuracy between 10% and 20%.

The ratio of the adversarial samples has been chosen as 100% to be more robust and because we saw that the accuracy on the clean samples decrease only of 3%.

Using this technique we achieved some quite good results especially on FGSM, BIM and PGD attacks while on Carlini-Wagner this method is not effective since we have basically the same performance as the original classifier.

4.2 Detector

Since the adversarial training approach did not lead to optimal results, we also opted for a detector defense. We designed and trained a MLP model that classifies samples as original or adversarial. If a sample is classified as adversarial, it is discarded and does not go as input of the classifier.

It has the same architecture and training configuration as the MLP classifier used as white-box for the attacks. The training data for this classifier is composed by of all the original samples of the training set and their respective adversarial samples attacked with different attacks each with different intensities.

In particular, the attack configurations provided for generating the adversarial samples that make up the detector training set are as follows:

- FGSM with $\epsilon = 0.3$
- FGSM with $\epsilon = 0.5$
- BIM with $eps\_step = 0.05$ and $max\_iter = 10$
- BIM with $eps\_step = 0.07$ and $max\_iter = 5$
- PGD with $eps\_step = 0.05$, $max\_iter = 20$ and $num\_random\_init = 10$
- PGD with $eps\_step = 0.07$, $max\_iter = 5$ and $num\_random\_init = 3$
- CW with $binary\_search\_steps = 1$, $confidence = 0$, $max\_iter = 3$, $learning\_rate = 0.3$ and $initial\_const = 10$
- CW with $binary\_search\_steps = 1$, $confidence = 0$, $max\_iter = 7$, $learning\_rate = 0.25$ and $initial\_const = 7$

The results are shown below. We plotted, for each attack, the accuracy of the detector on the respective adversarial samples and the accuracy on the complete model composed by the detector and the classifier. This accuracy has been computed as the sum of the accuracy of the detector and the accuracy of the classifier since this one does not get as input the samples classified as adversarial, as explained above.

As we can see, this method is more effective than Adversarial Training since we can obtain a stable performance on almost all attacks. The only one that remains tough to defend is Carlini-Wagner but we can still guarantee an accuracy of at least 75% for all the perturbation range.

We also have to note that the detector's performance on the PGD adversarial samples decrease for a high level of perturbation (beyond the range of the plots below). The accuracy stabilizes at 50% accuracy at a perturbation of around 4. This is due to the fact that the samples are too noisy and the classifier is not able to make a correct decision. However, in a real world scenario, this might not be a problem since a human can easily detect those samples as not authentic because they are very different from what they are used to, and thus the attacker might be afraid to perform attacks at this level of intensity.

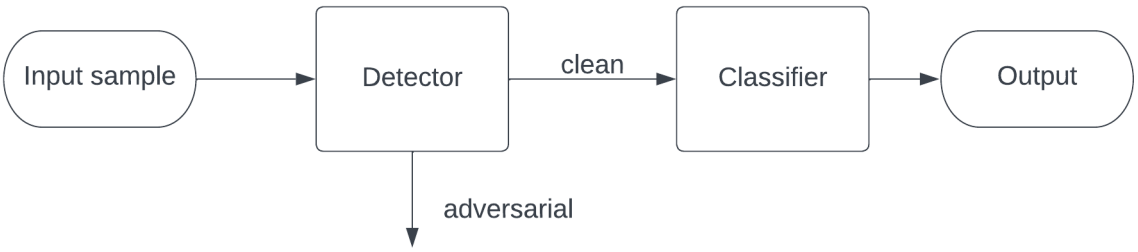


Figure 8: Complete architecture after detector implementation

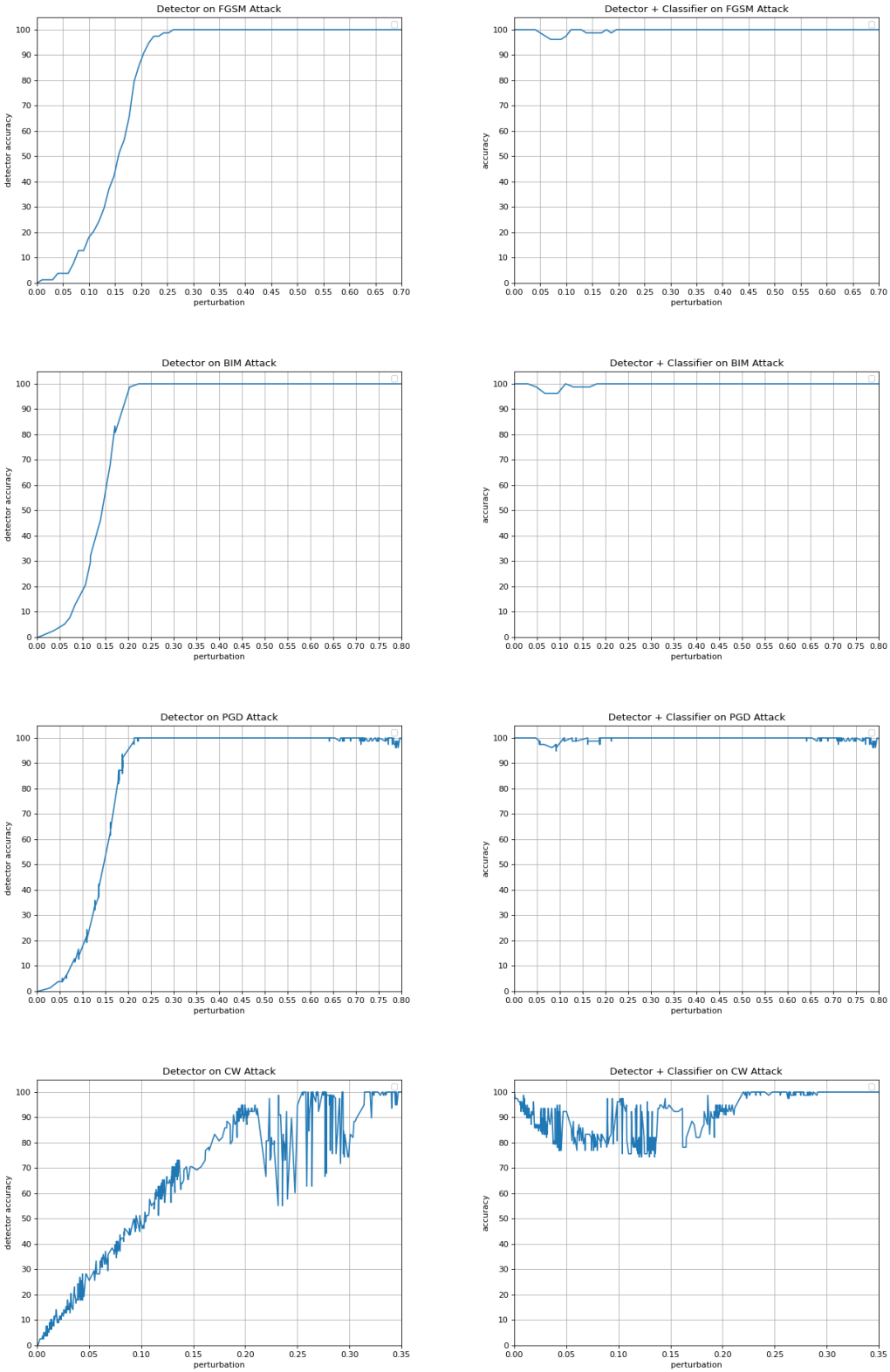


Figure 9: Detector defense curves

List of Figures

References

[1] A. Nagrani, J. S. Chung, and A. Zisserman. "VoxCeleb: a large-scale speaker identification dataset". In: *INTERSPEECH*. 2017.

[2] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. *Explaining and Harnessing Adversarial Examples*. 2014. eprint: `arXiv:1412.6572`.

[3] Reuben Feinman et al. *Detecting Adversarial Samples from Artifacts*. 2017. eprint: `arXiv:1703.00410`.

[4] Aleksander Madry et al. *Towards Deep Learning Models Resistant to Adversarial Attacks*. 2017. eprint: `arXiv:1706.06083`.

[5] Nicholas Carlini and David Wagner. *Towards Evaluating the Robustness of Neural Networks*. 2016. eprint: `arXiv:1608.04644`.