Report on Adversarial Attacks and Robust Networks

Team ROBUSTNET

BENREKIA Mohamed Ali, TADJER Amina, ZAGHRINI Eloïse

December 20, 2021

ABSTRACT

In recent years, a lot of weaknesses have been demonstrated in classification neural networks. Even in extremely precise and deep models, simple and undetectable by humans attacks on the input data can make the accuracy drop significantly. To remedy this problem, a lot of defense methods have been proposed to improve the robustness of any model. In this project, we will study 3 different attacks : FGSM, PGD and C&W, and their efficiency. We will then test out and compare different defense methods against those attacks : Adversarial Training, Defensive Distillation, Randomized Networks, and Model Ensemble.In this project, we will explore different parameters for each method and visualize the results.

Keywords: FGSM, PGD, C&W, Defensive distillation, Randomized Networks, Network Ensemble

1. INTRODUCTION

To test out the different attacks and defense mechanisms, we will use the CIFAR-10 database throughout the entire paper. This database consists of 10 different classes of animals and transportation methods. To compare different attacks and defense, we will also always use the same deep neural network. The very basic model used in this paper is made out of 2 convolutional layers of kernel size 5, of 6 filters and then 16 filters. It then flattens to predict the 10 different classes. We trained this vanilla model for 40 epochs and were able to reach an accuracy of around 60%.

2. ATTACKS

Depending on the attacker's knowledge of the architecture of the neural network, adversarial attacks are classified into **white-box attacks** where the adversary can access weights of the model, and **black-box attacks**, in which, the attack has a lower control over the model and can only access input and output of the model. Both categories aim at sabotaging the neural network and degrading its performance by fooling it into making wrong predictions. In this report, we focus only on some popular Whitebox attacks such as : FGSM, PGD and C&W. We will also experiment and explore different parameters for every method

2.1 FGSM

Presented by Goodfellow et al. in [3], **FGSM** (Fast Gradient Sign Method) is amongst the first and most popular adversarial attacks to date. It is a simple yet effective method to generate adversarial images by leveraging the way neural networks learn gradients. In a nutshell, FGSM adjusts the input image pixels in the direction of the gradients to maximize the loss value. Formally we can express the attack as follow:

$$perturbedimage = image + epsilon * sign\left(data_{grad}\right)$$

$$x' = x + \varepsilon * sign\left(\nabla_x J(\theta, \mathbf{x}, y)\right)$$

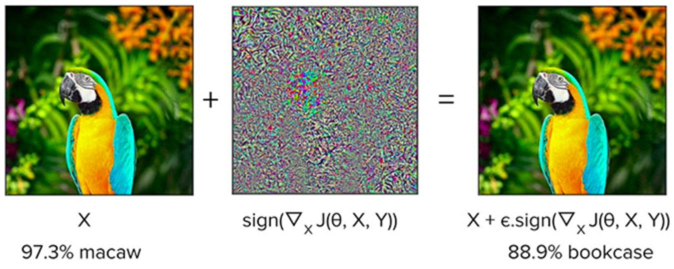

Fig. 1. The Fast Gradient Sign Method (FGSM) for adversarial image generation

As illustrated in the figure, X is the original image correctly classified as a Y = "macaw" and represents the model parameters, and $J(\theta, X, Y)$ is the loss. The attack backpropagates the gradient back to the input data to calculate $\nabla_x J(\theta, X, Y)$. Then it alters the original by ε epsilon in the direction that will maximize the loss (i.e. $sign(\nabla_x J(\theta, X, Y))$). As a consequent, we get an adversarial image X' misclassified by the model as a "bookcase" when it is still look like a "macaw".

It is important to mention that the result is an output image that, according to the human eye, looks identical to the original, but makes the neural network make an incorrect prediction, so we should make sure that the value of ε epsilon is not so high to make the output image "clearly" modified for an observer. In our case we test with different values of epsilon starting from 0 to 0.3.

2.2 PGD

Projected Gradient Descent is a more powerful iterative version of the FGSM attack. It was presented in [5], and proves to be even more efficient than a classic FGSM attack, which is equivalent to a PGD with only one step. Similarly outputs images that have gradient-based noise added to them. Those images look the same to the naked eye, but successfully deceives the model. PGD attacks exist in l2 and l∞ bounded format, but we have focused on the l∞ norm as it gave better results. We conducted all our tests with $N = 40$ steps and $\eta = 1/255$. Formally, we can express the attack as follow. For $k \in [1,N]$ an integer:

$$x_{k+1} = \Pi_{0+\varepsilon}(x_k + \eta * sign\,(\nabla_x J(\theta, \mathbf{x}, y)))$$

$originalimage = x_0$ and $perturbedimage = x_N$

2.3 C&W

Carlini and Wagner in [2] propose a set of optimization-based adversarial attacks (**CW attacks**) that can generate L_0, L_2 and L_∞ norm measured adversarial samples, namely CW_0, CW_2 and CW_∞.It formulates the optimization objective as follows:

$$\min_{\delta} D(x, x+\delta) + c * f(x+\delta) \;\; subject \;\; to \;\; x+\delta \in [0,1]$$

where δ denotes the adversarial perturbation, $D(.,.)$ denotes the L_0, L_2 and L_∞ distance metric, and $f(x+\delta)$ denotes a customized adversarial loss that satisfies $f(x+\delta) \leqslant 0$ if and only if the DNN's prediction is the attack target. To ensure $(x+\delta)$ yields a valid image (i.e., $x+\delta \in [0,1]$), it introduces a new variable k to substitute δ as follows:

$$\delta = \frac{1}{2}[tanh(k) + 1] - x$$

CW attacks achieve 100% attack success rate on naturally trained DNNs for MNIST, CIFAR-10, and ImageNet. They also compromise defensive distilled models, on which L-BFGS and DeepFool fail to find the adversarial samples.

2.4 Results on vanilla model

We evaluate the effectiveness of the aforementioned attacks on our CIFAR-10 classifier, by calculating the accuracy of the model against a test-set of 1024 sample , where we apply the attack only on the correctly classified samples so we ensure that the accuracy variation is due to the attack.

For **FGSM** and **PGD** attacks we compute the accuracy of the model in function of different values of ε epsilon, and for **C&W** we applied it for a single value of learning rate. Results are summarized and illustrated in figure2 and table 1:

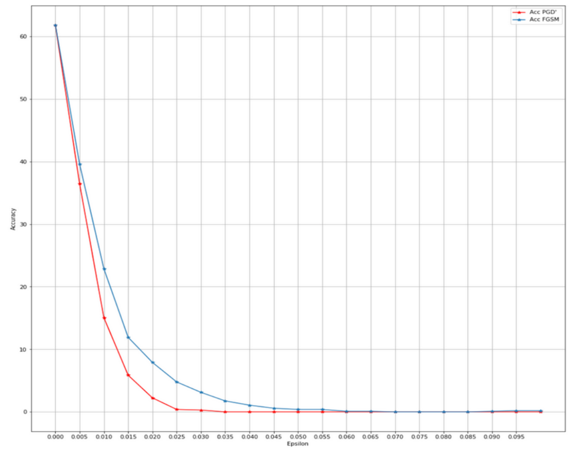

Fig. 2. Vanilla model accuracy when applying FGSM and PGD attacks

We observe that the accuracy of the model drops drastically to 0% for both attacks and since PGD is a stronger attack, we can notice that it has a strong impact on the model even with smaller values of epsilon. It is important to mention that these values of epsilon are chosen so that an observer can not easily notice the difference between original and modified images. We can say that our attacks to the proposed network was successful and it reduced 100% of test accuracy.

	Parameters	Accuracy
No Attack	/	61%
FGSM	$\varepsilon = 0.03$	3.125%
PGD	$\varepsilon = 0.03$ step = 40 $\eta = 1/255$	0.292%
C&W	$Norm = L_2$ $lr = 5e-3$ $max_{iterations} = 1000$	0%

Table. 1. Summary of attacks parameters and model accuracy when applying these attacks

3. DEFENSES

In this section, we highlight the mainstream defense methods such as Adversarial training, Defensive distillation, Randomized Smoothness and Model Ensemble. These mechanisms aim to improve the robustness of neural networks against adversarial attacks. In [1], authors classify these defenses into **Attack-dependent defenses** and **Attack-independent defenses**. In the next section, we perform an empirical robustness analysis to evaluate these defenses.

3.1 Adversarial Training

In adversarial training [4], the training data **are augmented by "adversarial" samples generated using an attack algorithm**. Intuitively, if the model sees adversarial examples during training, its performance at prediction time will be better for adversarial examples generated in the same way. Ideally, we would like to employ any known attack method to generate adversarial examples during training. In practice, we can only afford to use a fast method like FGS or iterative FGS can be employed. Adversarial training uses a modified loss function that is a weighted sum of the usual loss function on clean examples and a loss function from adversarial examples:

$$Loss = \frac{1}{(m-k)}(\sum_{i \in CLEAN} L(X_i|y_i + \lambda \sum_{i \in ADV} L(X * adv_i|y_i)$$

During training, for every batch of m clean images we generate k adversarial images using the current state of the network. We forward propagate the network both for clean and adversarial examples and compute the loss with the formula above.

3.2 Defensive Distillation

Introduced by Papernot et al. in [6] as a defense mechanism to increase the robustness of deep neural networks, **Defensive Distillation** involves the training of two neural networks, the first one trains on hard class labels (0 or 1), while the distillation network instead trains on the class probabilities generated by the first network (also called soft labels), because the full probability distribution contains more information than a single label. According to the authors, the intuition behind is that knowledge extracted in the form of probability vectors can be beneficial to improving generalization capabilities of DNNs outside of their training dataset and as consequence enhances their resilience against perturbations.

Defensive distillation adjusts the training of a classifier by re-configuring its *softmax* layer, which calculates the probabilities $f_i(x)$ for each class i as follow:

$$f_i(x) = \frac{e^{z_i(x)/T}}{\sum_{l=0}^{n-1} e^{z_l(x)/T}}$$

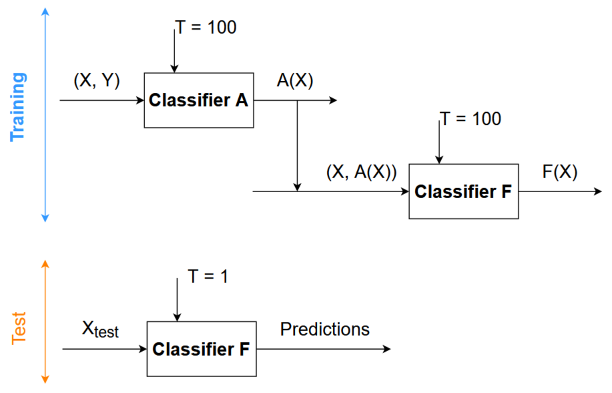

Fig. 3. The general schema for Defensive Distillation

Where the z are the activations of the last layer before the softmax, and T is the temperature of the softmax. High values of T results in probability distribution closer to uniform (close to $\frac{1}{n}$ where n is the number of classes) and lower temperatures has the effect of resulting in discrete probability vectors with a large probability (close to 1) given to the most likely class. At test time the temperature is set back to 1.

With the help of the illustration 3, we summarize the defensive distillation steps as follow:

1. Train a network A on the given training set (X,Y) by setting the temperature of the softmax to T.
2. Train another network F using softmax at temperature T on the dataset with soft labels $(X,A(X))$. We refer the model F as the distilled model.
3. Use the distilled network F with softmax at temperature 1 during prediction on test data X_{test}(or adversarial examples).

Taken Temperature as 100 for training the NetA and NetF.

3.3 Randomized Networks

One strategy when trying to defend a model against white box attacks is to find ways to take away some of the deterministic nature of the model. Because those attacks rely on predicting exactly the way the model works to fool it, by adding randomness in our data and our predictions, we can reduce the effectiveness of the adversarial attacks seen in part 2. We added a Gaussian noise to our training and testing data to try to find the best performing model against FGSM attacks. We experimented adding the noise only at testing time, and both at training and testing for variances ranging from 0.005 to 1.

3.4 Model Ensemble

Similarly to Randomized Networks, we also tested an intuitive idea to add some non deterministic features to protect our model. We trained 10 different models with the exact same architecture, and different initialized weights to give them a different results after training. Then, we attacked this ensemble of model in different ways : We tried taking the average best prediction of all the models and computing the average gradient, but also picking a random model at each prediction or gradient computing which adds more non deterministic behavior to our model. We tested those methods against FGSM attacks.

4. OVERVIEW OF RESULTS

The Table 2 in Appendix recaps our different results for epsion = 0.03.

4.1 Adversarial Training with FGSM and PGD

In adversarial training, we augmented our training dataset with data generated by applying FGSM and PGD attacks. Let's start with FGSM adversarial training results. Initially, we set epsilon to 0.03 and generate adversarial data, here are the results we got against FGSM and PGD attacks (we varied epsilon in the attacks from 0 to 0.075).

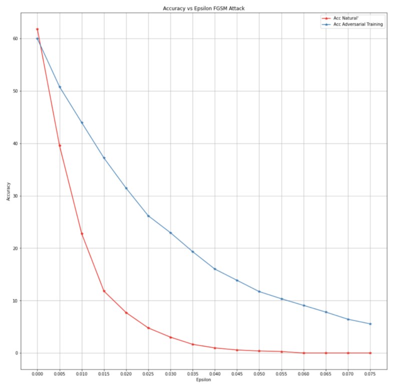

Fig. 4. Results of Adversarial Training Vs FGSM

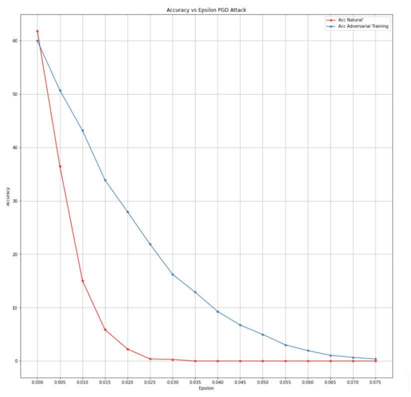

Fig. 5. Results of Adversarial Training Vs PGD

As we can see in Figure4 and Figure5, the adversarial training improves the accuracy of our model against attacks, but this accuracy decreases with big epsilon value, this is why we tried to train our model by using different epsilon values and explore the relation between epsilon values in the defense and epsilon values in the attack. The results are shown in the Figure [10] in the Appendix.

We can notice that :

1. With higher epsilon value in the defense, the model achieves better accuracy.
2. The accuracy reaches it maximum when $Epsilon_{attack}$ coincides with $Epsilon_{defense}$ or are very close.

We also tried to combine two values of epsilon to generate adversarial model, but the results were not improved. Training our model using adversarial data generated with PGD took obviously more time, so we tried less epsilon value and the results are shown in the Figure [7] in the Appendix.

4.2 Distilled Model

For the evaluation of the defensive distillation, we trained a distilled model according to the training described in 3.2 and then we tested it against adversarial data generated using FGSM and PGD attack with different values of ε, we illustrate the results in figure [6]:

Results show that the defensive distillation improved the robustness of the model since that FGSM and PGD attack reduces test accuracy from 60,15% to 49% and from 60,15% to 48,43% respectively. Moreover, the distilled model keeps a good level of accuracy while increasing the value of epsilon and that is due to the fact that the distilled model generalizes well for unseen data since it was trained with a high-temperature softmax. We can say that defensive

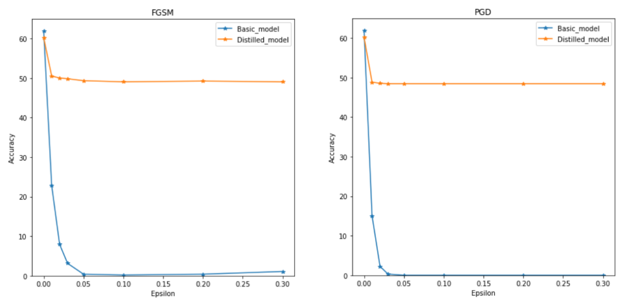

Fig. 6. (left) Distilled model against FGSM, (right) distilled model against PGD.

distillation for the proposed network with temp of 100 was successful and it only reduced 10% of test accuracy.

4.3 Randomized Networks

To try and find the best noise to add to our data, we tried training models with added Gaussian noises with variances ranging from 0.005 to 1, on 40 epochs each. We found that the natural accuracy did decrease with the added noise, to about 45% with the variance of 1. Only adding noise at testing time also did not give impressive results, with the attack being seemingly just as effective than without the noise. In the end, we saw some real results for training noises ranging from 0.5 to 1, and a testing noise of 0.2, with a gain of about 16% of accuracy at epsilon = 0.03. However, we observed that as epsilon increases, this defense becomes virtually nonexistent and the accuracy drops to zero (see figure [8] in Appendix)

4.4 Network Ensemble

This method gave promising results, especially for small epsilons values. We tested 2 different methods for prediction, and 2 different methods for gradient computing in the FGSM attack : either by computing the average based on the results of all the models, or by picking one randomly and using it to compute what we wanted. Here are the results of those experiments for epsilon = 0.03:

	Average gradient	Random gradient
Average prediction	22.6%	43.4%
Random prediction	26.5%	38.7%

With only one model, the accuracy is around 3%. This method then allows to add some robustness to a model. As expected, prediction by adding all the probabilities computed by the model and taking the highest one gives the best natural accuracy, of about 70% which is higher than just one model. However taking a random model each time a gradient needs to be computed gives the most robust way to defend the model against gradient-based attacks such as FGSM. Even though the best result is achieved by using the average of all predictions to predict and taking a random model to compute the gradient, it might be difficult to separate those tasks in real attacks. In that case, sticking to picking out a random network each time gives the most robust defense. It might increasingly better results if we add more models to our ensemble, but it requires much more computing time and training time. Similarly to the Randomized Networks method, this defense is mainly efficient for small epsilons, otherwise the accuracy tends to zero also (see Figure [9] in Appendix).

5. CONCLUSION

Throughout this project, we had the opportunity to work on the robustness of neural network, in which we implemented some state of the art attacks such as FGSM, PGD and C&W and we evaluated their efficiency against an image classifier on the CIFAR-10 dataset, then we improved the robustness of the classifier with some popular defenses like adversarial training, defensive distillation , Randomized smoothness and Ensemble model. These defenses showed different results in term of resilience against the attacks and complexity of execution. We conclude that, training a robust neural networks is not trivial as new attacks are established and there is no defense that fits all.

BIBLIOGRAPHY

[1] N. Carlini. *Evaluation and design of robust neural network defenses*. University of California, Berkeley, 2018.

[2] N. Carlini and D. A. Wagner. Towards evaluating the robustness of neural networks. *2017 IEEE Symposium on Security and Privacy (SP)*, pages 39–57, 2017.

[3] I. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*, 2015.

[4] I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. *CoRR*, abs/1412.6572, 2015.

[5] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu. Towards deep learning models resistant to adversarial attacks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

[6] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami. Distillation as a defense to adversarial perturbations against deep neural networks. In *2016 IEEE symposium on security and privacy (SP)*, pages 582–597. IEEE, 2016.

6. APPENDIX

	Natural	FGSM attack	PGD attack
Vanilla Model	61.8%	3.12%	0.29%
FGSM Adversarial	61.8%	41%	29.8%
PGD Adversarial	61.8%	31%	30.2%
Defensive Distillation	60.1%	**49.8%**	**48.4%**
Randomized Networks	51.9%	19.3%	-
Model Ensemble	**71.9%**	43.4%	-

Table. 2. Summary of different defense accuracies with FGSM and PGD attacks for eps=0.03

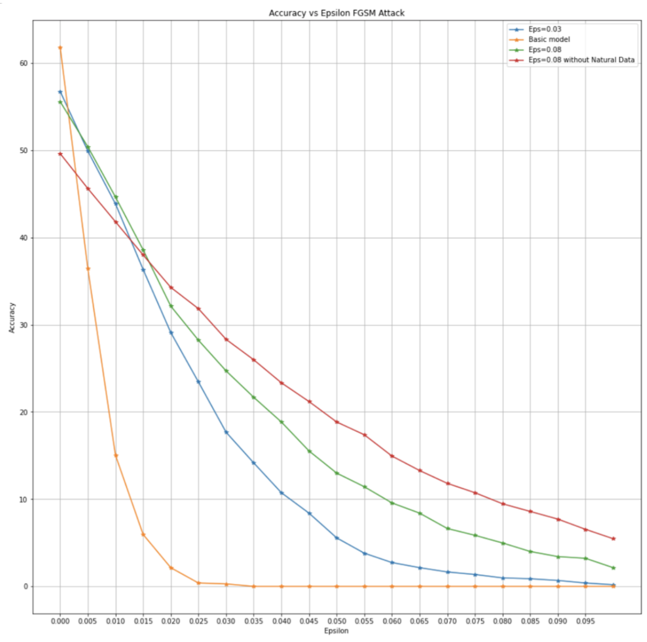

Fig. 7. Results of Adversarial Training with different epsilons

The red graph shows the accuracy of a model trained only with adversarial data (Without natural data). The natural accuracy was decreased, but it gives better results against attacks.

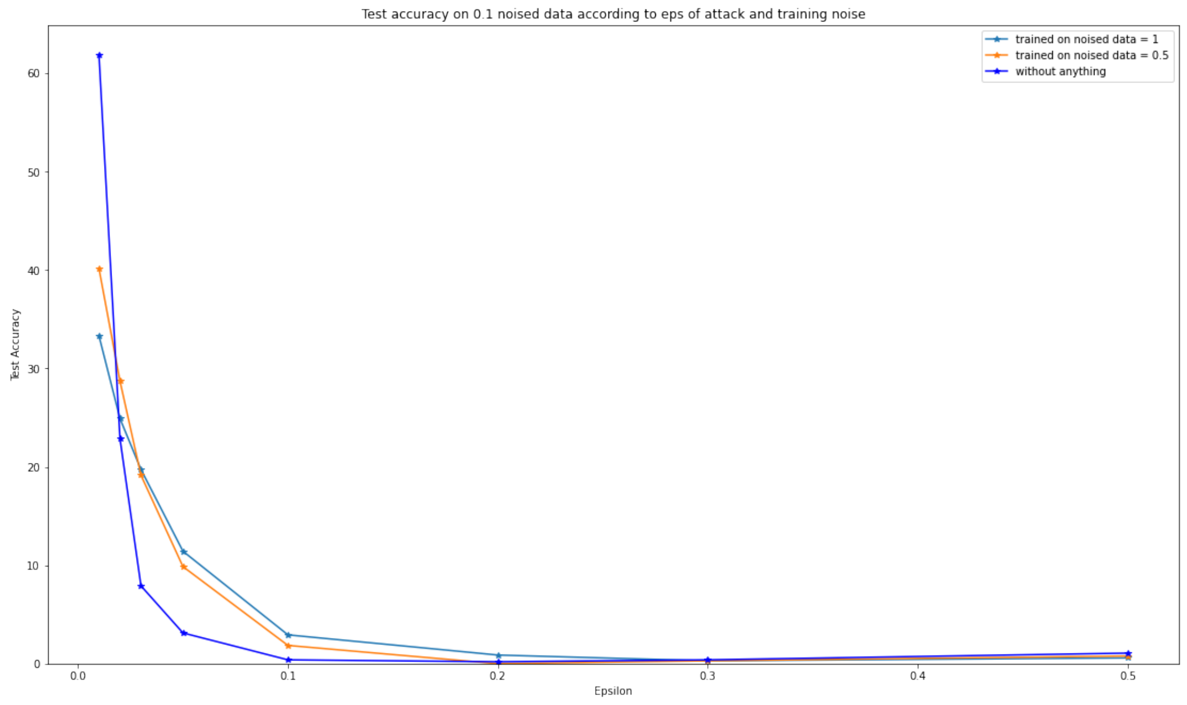

Fig. 8. Results of Randomized Networks with different epsilons for variance 0.5 and 1

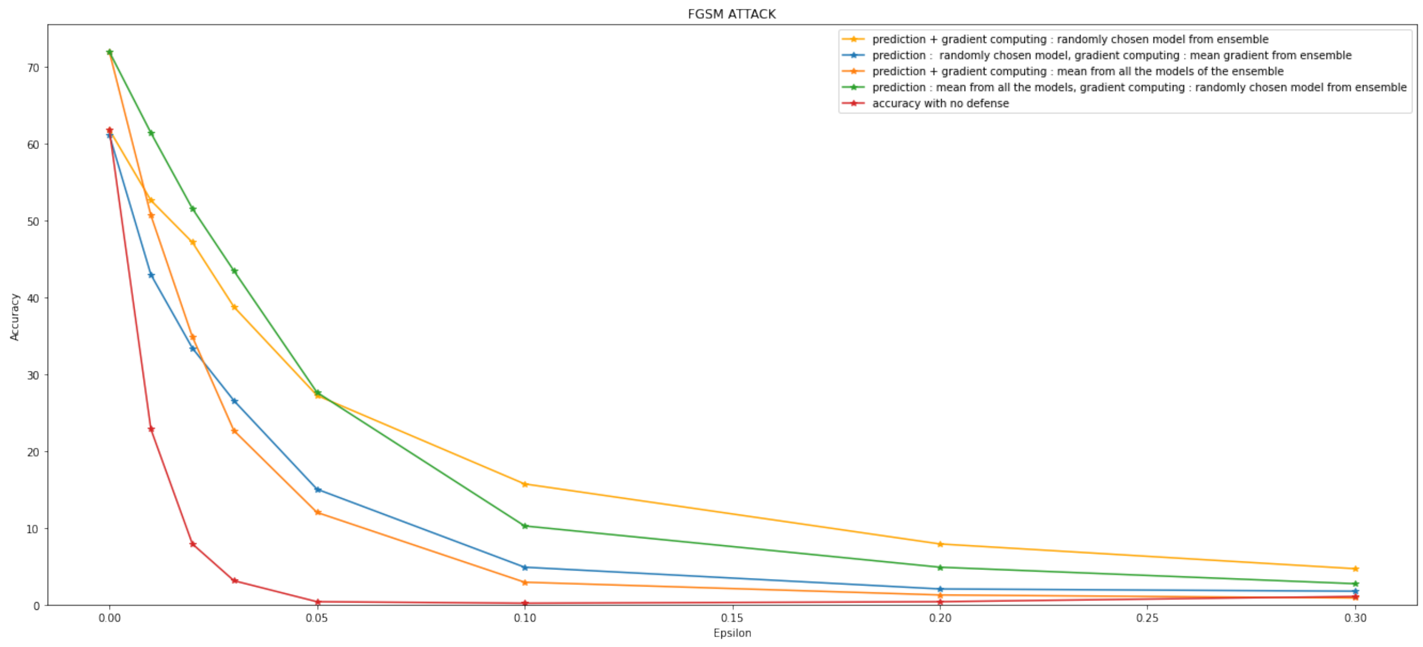

Fig. 9. Results of Network Ensemble with different epsilons for different computing methods

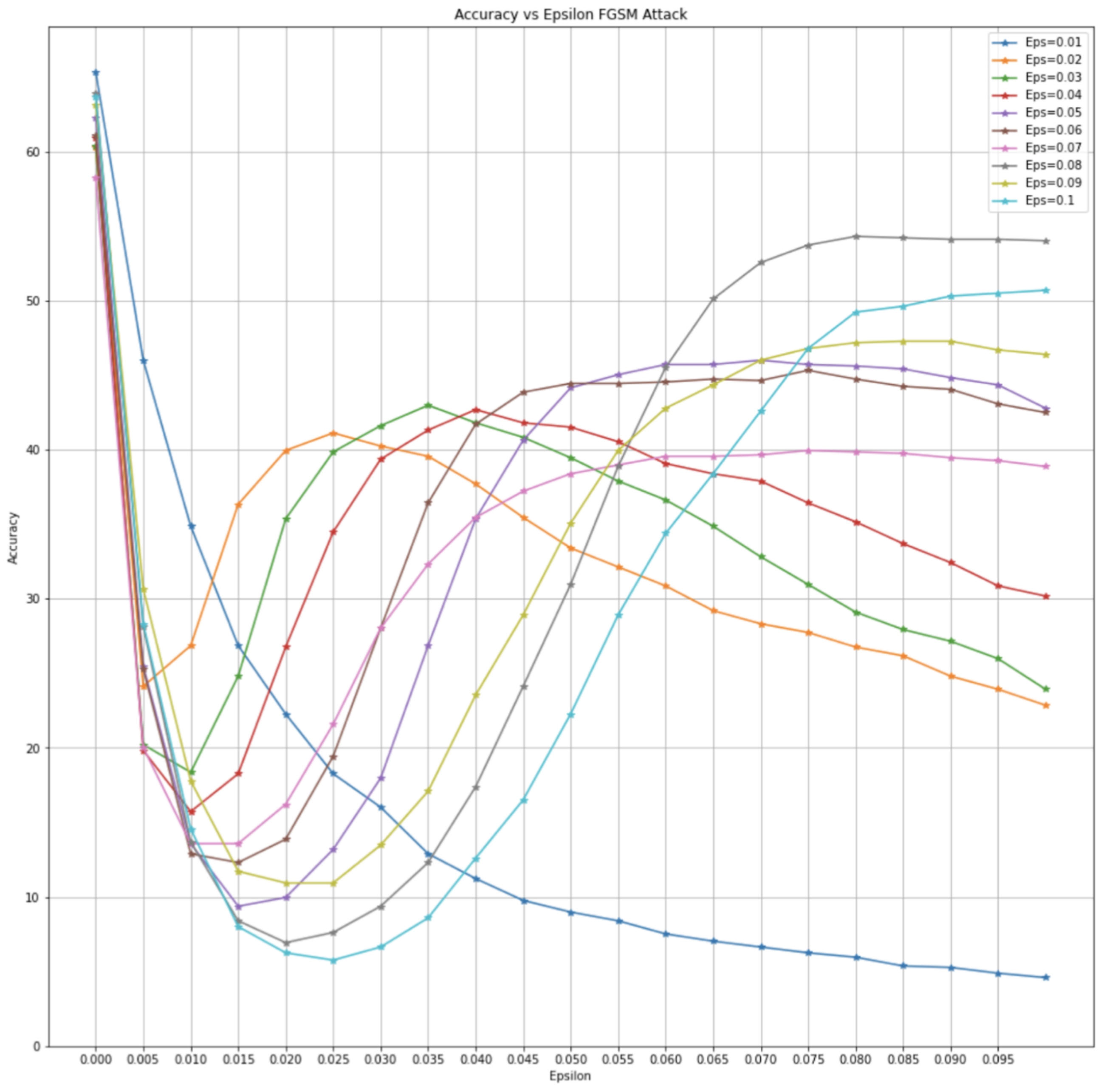

Fig. 10. Results of Adversarial Training with different epsilons