PSL Research University

Master 2 IASD - Data Science Project

Robust Neural Networks

Authors :
Mehdi Bentaleb
Yassine Abou Hadid
Mohamed Benmaiza

Supervisors :
Benjamin NEGREVERGNE
Alexandre Vérine

January 2023

Contents

1 Introduction

While artificial intelligence (AI) is progressing considerably both in terms of application fields and performance, it still needs to progress in terms of robustness. Indeed, it can still be extremely weakened by tiny modifications that are imperceptible to humans, such as the modification of a single pixel of an image, but with disastrous consequences. This is the objective of the adversarial attacks. These attacks constitute a real constraint in terms of security of AI systems and appear to be an effective weapon for cyber criminals.

In this project, we will perform several adversarial attacks on a deep learning model named VGG-19. Then, we will implement different defense mechanisms and study their effects on our original Cifar-10 dataset.

2 White Box Attacks

2.1 Base Model : VGG-19

VGG19 is a very deep convolutional neural network, with 19 weight layers, which makes it capable of learning very complex features from the input data. This can make it a good choice for tasks where a high level of performance is desire. It is known to have the best performance on classifying images.
Its architectures is as follows :

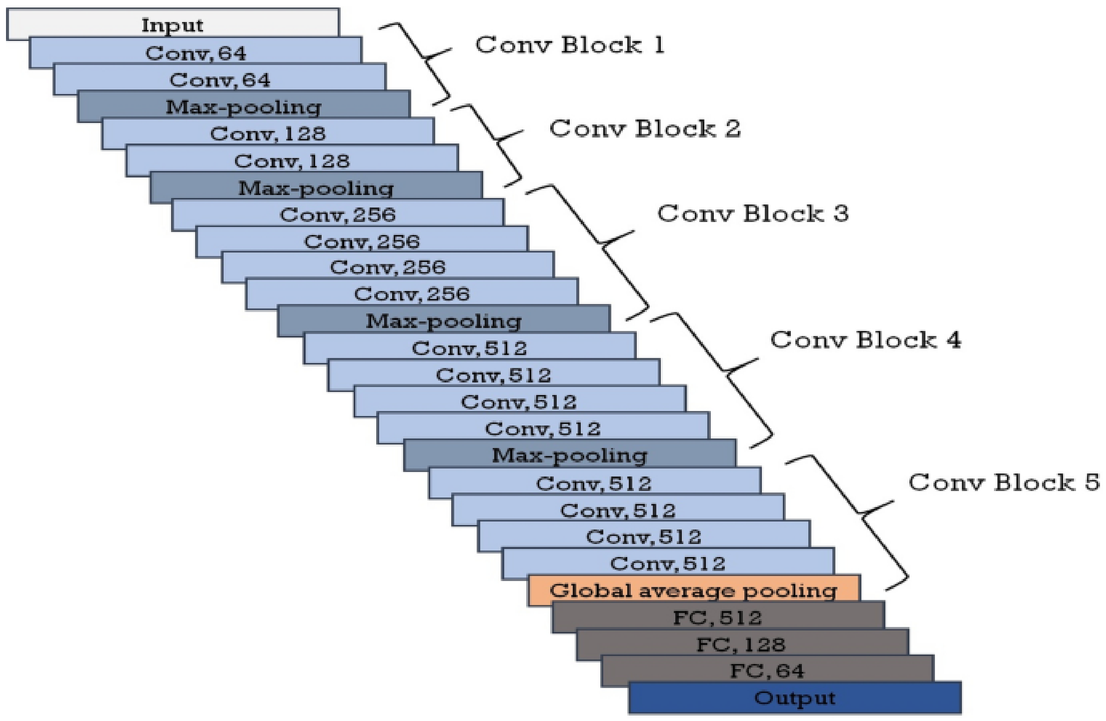


FIGURE 1 – VGG-19 Architecture

2.2 FGSM

The FGSM (Fast Gradient Sign Method) attack consists in taking a sample input x and a target model f, and perturb x in such a way that the perturbed input x' is highly likely to be misclassified by the model. This is done by computing the gradient of the loss function with respect to the input x, and then adding a small multiple of this gradient to x to create the perturbed input $x' : x = x + \epsilon sign(\Delta_x l(x, y))$.
The intuition behind this is that the gradient of the loss with respect to the input gives us the direction in which the input should be changed in order to maximize the loss. By adding a small multiple of this gradient to the input, we can create an input that is highly likely to be misclassified by the model.

2.3 PGD

PGD (Projected Gradient Descent) is a variant of FGSM that uses an iterative optimization process to find the perturbed input that is most likely to be misclassified by the model. The basic idea is similar to FGSM, but instead of adding a single multiple of the gradient to the input, PGD performs a series of gradient descent steps to find the optimal perturbation. This can be more effective than FGSM in some cases, as it allows for a more fine-grained search for the perturbation that is most likely to cause the model to make an error. It consists more precisely in finding an adversarial perturbation inside of an lp-norm ball of radius ϵ, taking a step of size η at each iteration.
We generate perturbations with the following :

$$\begin{cases} x_0 = x \\ x_{t+1} = proj_{B(x_0,\epsilon)}(x_t + \eta.sign(\nabla_x l_f(x, y))) \end{cases}$$

We performed the PGD method for generating adversarial examples, with two different versions : L2 and L_∞ (distance metric that is used to measure the difference between the original input and the adversarial example.) In general, the L2 distance is more sensitive to small changes in the input, while theL_∞ distance is more sensitive to large changes in the input. As a result, the L2 version of PGD may be more effective at generating small perturbations that are difficult for the model to detect, while the L_∞ version may be more effective at generating large perturbations that are more easily detectable.

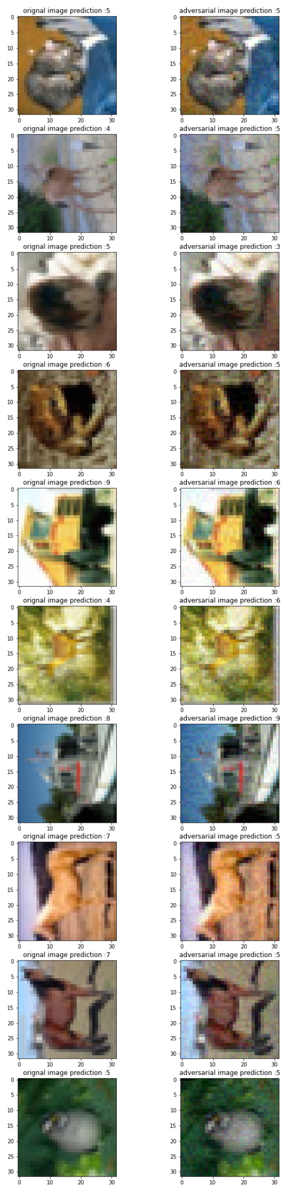

orignal image prediction :5
adversarial image prediction :5
orignal image prediction :4
adversarial image prediction :5
orignal image prediction :5
adversarial image prediction :3
orignal image prediction :6
adversarial image prediction :5
orignal image prediction :9
adversarial image prediction :6
orignal image prediction :4
adversarial image prediction :6
orignal image prediction :8
adversarial image prediction :9
orignal image prediction :7
adversarial image prediction :5
orignal image prediction :7
adversarial image prediction :5
orignal image prediction :5
adversarial image prediction :5

Figure 2 – FGSM attack performed on 10 images

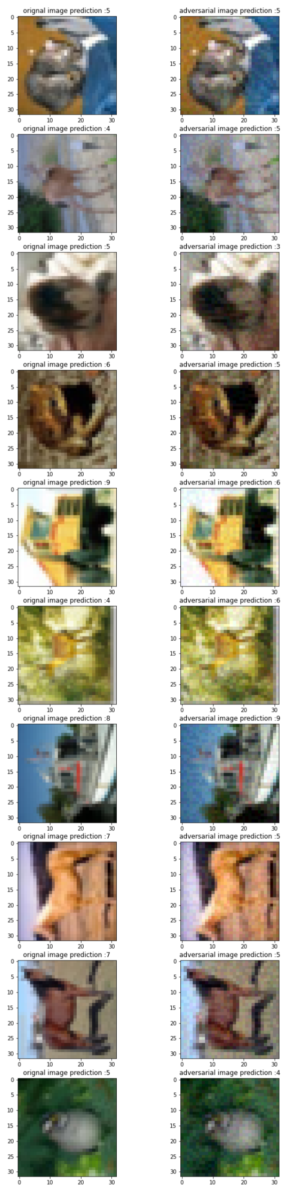

orignal image prediction :5
adversarial image prediction :5
orignal image prediction :4
adversarial image prediction :5
orignal image prediction :5
adversarial image prediction :3
orignal image prediction :6
adversarial image prediction :5
orignal image prediction :9
adversarial image prediction :6
orignal image prediction :4
adversarial image prediction :6
orignal image prediction :8
adversarial image prediction :9
orignal image prediction :7
adversarial image prediction :5
orignal image prediction :7
adversarial image prediction :5
orignal image prediction :5
adversarial image prediction :4

FIGURE 3 – PGD L_∞ attack performed on 10 images

2.4 Adversarial training

Adversarial training is one of the most well known methods to train neural networks to be more robust against adversarial attacks. It consists on training models with a dataset containing generated adversarial examples in addition to the normal inputs. The intuition behind this idea is to "push" the decision boundary of the right class around adversarial examples in the hope that most of the inputs in the lp-norm ball around natural examples are correctly classified.

To train the network, we use the following adversarial risk : $\min E_{(x,y)}\left(\max_{\|\xi\|\leq\epsilon} l_{f_\theta}(x+\xi,y)\right)$

We trained three models with a dataset half perturbed by changing the nature of attacks (FGSM, PGD L2 and PGD L_∞). The robustness of the resulting model is compared with the robustness of a classical model in fig.4 using FGSM (parameter ϵ) and PGD (parameter ϵ , stepsize $\epsilon/10$ and 10 iterations).

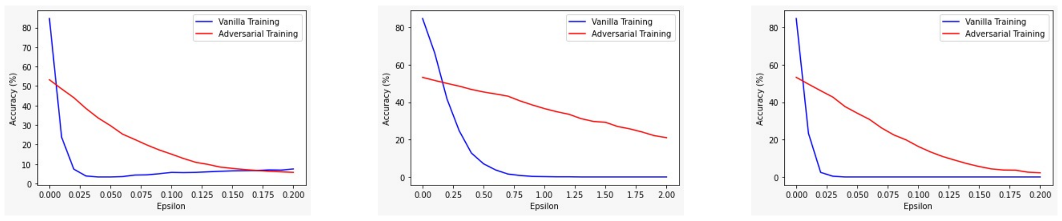


FIGURE 4 – Robustness of the models with respect to the parameter of the FGSM (left), PGD- L2 (center) and PGD L_∞ (right) attacks

In the table below, we have summarized our best results (Adversarial Model trained with PGD L_∞).

Model	Data Tested	Accuracy
Base Model	Original Test Set	82.12
Base Model	FGSM-perturbed Test Set	8.11
Base Model	PDG-L2 perturbed Test Set	6.64
Base Model	PDG-L_∞ perturbed Test Set	2.59
Adversarial Model	Original Test Set	65.14
Adversarial Model	FGSM perturbed Test Set	50.68
Adversarial Model	PGD L2 perturbed Test Set	58.3
Adversarial Model	PGD L_∞ perturbed Test Set	49.52

TABLE 1 – Accuracy of the Base Model and the Adversarial Model (with $\epsilon = 0.06$) on different Test Sets

We can remark that the adversarial Model performs worse than the base model for original dataset.

3 Black Box Attacks

The attacks that we have explored so far are white-box attacks, i.e. we have access to the model parameters to make adversarial examples based on the gradient. In practical cases, it is more likely to only have access to the model predictions. Thus, we can't use backpropagation to compute the gradient : this is the black-box setting.

3.1 NES

‘

The NES (Noise-based Evolutionary Strategy) method is a type of black-box attack. Thus, the attacker only has access to the model's input and output, and must try to craft adversarial examples based on this limited information. The basic idea behind the NES method is to use an evolutionary algorithm to search for perturbations to the input that will cause the model to make an error.

Here is a high-level outline of the algorithm :

— 1.Initialize a population of random perturbations
— 2.Evaluate the fitness of each perturbation using a fitness function based on the output of the model.
— 3.Select the fittest individuals from the population.
— 4.Apply genetic operations (such as crossover and mutation) to create new offspring.
— 5.Evaluate the fitness of the new offspring.
— 6.Repeat steps 3-5 until a satisfactory adversarial example is found, or until some other termination condition is met.

NES helps us to estimate the gradient and then minimize P(y | x) where y is the label and x the image, without using the model parameters.

The NES attack is particularly powerful since that by testing it on our model, we got an accuracy of 4.0 on the attacked test set.
To defend against the NES method, we can do adversarial training, or use input preprocessing techniques, such as adding random noise to the input to make it more difficult for an attacker to craft effective adversarial example.

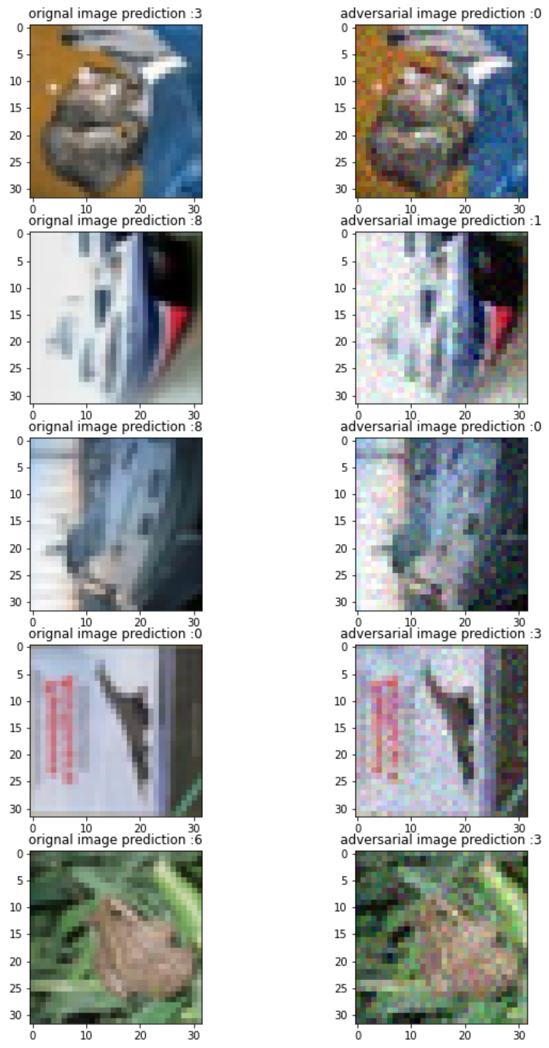

orignal image prediction :3
adversarial image prediction :0
orignal image prediction :8
adversarial image prediction :1
orignal image prediction :8
adversarial image prediction :0
orignal image prediction :0
adversarial image prediction :3
orignal image prediction :6
adversarial image prediction :3

FIGURE 5 – NES attack performed on 4 images

3.2 Injection Noise Attack

We created a random disturbance generator that will try to trick the network. The principle is to generate a very weak random disturbance and to look at what the network predicts.
We calculate the accuracy of this method on a perturbed test set with gaussian noise with $\sigma = 0.1$ by the intuitive formula : 1 - Number of failures/ Total Number of tests. We got an accuracy of 42.57 on the Vanilla model and 39.23 on the Adversarial Model.
This shows that the adversarial Model trained with PGD L_∞ doesn't add any robustness to the model and doesn't perform well against this type of attacks.

4 Other Defense Mechanisms

4.1 Comparing VGG models with adversarial training

Comparing the robustness of different VGG models using adversarial training can be useful, as it can help determine which model is more resistant to adversarial examples. This can be especially useful in situations where the model will be deployed in a real-world setting, where it may be subject to various types of perturbations or attacks.

We took different versions of VGG (with different number of convolutional layers) and trained them with adversarial training and evaluated their performance on a test set of Adversarial examples. We found that VGG-13 is the most robust as shown by the results below :

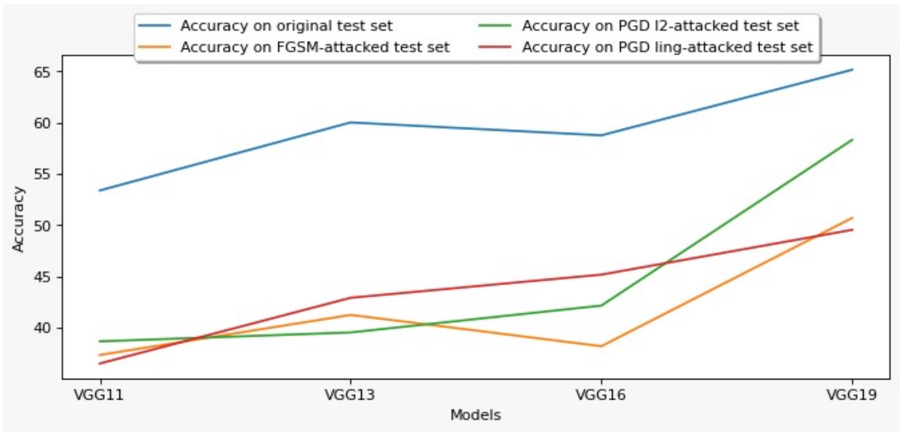


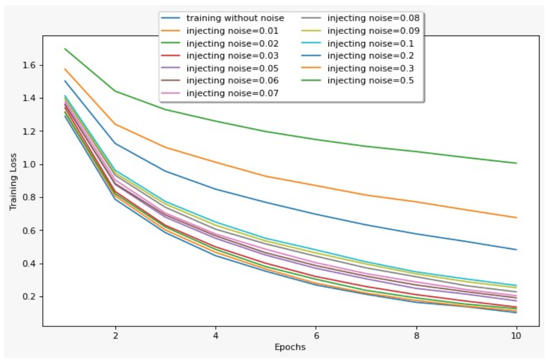
FIGURE 6 – Different VGG models accuracies

4.2 Randomized Network

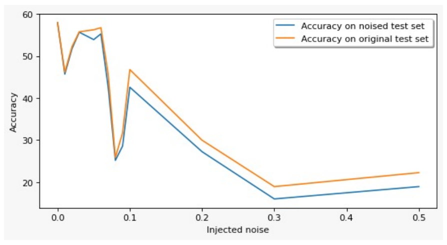
Randomized networks with Gaussian noise are a type of training procedure that can be used to train more robust neural networks. The idea is to inject random noise into the inputs or weights of the network at each training step, in order to make the network more resistant to perturbations in its inputs. We implemented it by adding Gaussian noise to the inputs of the network during training. This can be done by sampling random noise from a Gaussian distribution with mean 0 and standard deviation sigma, and adding it to the inputs before they are passed through the network. By injecting noise into the network in this way, we can train a more robust model that is less sensitive to perturbations in its inputs or weights. This can be particularly useful in situations where the test data may be different from the training data in ways that we cannot fully anticipate.

We can see in fig (a) that more the noise injected is important the more the training loss decreases rapidly. We took $\sigma = 0.06$ for our Randomized Network because it is where we have the best accuracy on test set and noised test set as shown in fig (b).

In the Table 2, we can see that the Randomized Network is very robust against noise injected attacks. It can also be used to protect againt FGSM and PGD attacks since that the accuracy has greatly increased by comparing it to that of the vanilla model.



(a) Variation of Training Loss with different noises



(b) Variation of Accuracies with different noises

Data Tested	Accuracy
Original Test Set	56.73
noise-injected Test Set	55.27
FGSM-perturbed Test Set	21.13
PDG-L2 perturbed Test Set	33.21
PDG-Linf perturbed Test Set	35.72

Table 2 – Accuracy of Randomized Network (with $\sigma = 0.06$) on different Test Sets

5 Conclusion

Our results prove that deep neural networks can be made more or less resistant to adversarial attacks but not without negative effect on the performance for the original data. We conclude also that adversarial trained networks perform well against white box attacks and there are some technique such as Randomized network that gives more robustness against black box attacks.

One improvement could be to train in a better way our model with data augmentation and try to do adversarial training by testing every attacks on augmented data.At the end, we can ask the question of the dependence on the database chosen and whether we could hope to obtain better results with data that does not are not as blurred as those of CIFAR10.
As a side note, we used Google Colab GPUs to perform (heavy) computations.

References

[1] Ian J. Goodfellow, Jonathon Shlens, Christian Szegedy (2015), *Explaining and Harnessing adversarial examples*

[2] Andrew Ilyas, Logan Engstrom, Anish Athalye, Jessy Lin (2018), *Black-box Adversarial Attacks with Limited Queries and Information*

[3] Jeremy M Cohen, Elan Rosenfeld, J. Zico Kolter (2019), *Certified Adversarial Robustness via Randomized Smoothing*

[4] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, Adrian Vladu (2019), *Towards Deep Learning Models Resistant to Adversarial Attacks*

[5] Harry 24K, https ://github.com/Harry24k, *PGD and FGSM Implementation*