# Training Robust Neural Networks

Clément Preaut*, Iwan Quemada†, Nicolas Durat‡

[1] *Quick-Attacks, Université Dauphine-PSL, 75016 Paris, France*

January 9, 2023

**Abstract**

Adversarial machine learning is a method that aims to trick machine learning models by providing deceptive input. It includes generation and detection of adversarial examples, which are inputs created to deceive classifiers. Such attacks have been explored in many areas. We studied the white-box and black-box adversarial attacks methods on the CIFAR-10 dataset. Then, we implemented adversarial defenses to protect against the different attacks. By incorporating this during training, we greatly improve the models accuracies.

**Keywords:** Adversarial Attacks, FGSM, PGD, NES, Adversarial Defenses

## 1 Introduction

We explore the aversarial attacks techniques. An adversarial attack is a method to generate adversarial examples. An adversarial example is an model input that is designed to cause mistake predictions. There are two types of attacks: white-box and black-box. In the former, the attacker has access to the target model. In the latter, the attacker does not have access to the model and can only observe the targeted model outputs. The strategy for training an adversarially robust model is the adversarial training and the randomized networks. The idea is to create and incorporate adversarial examples (noise) into the training process.

## 2 Naive Classifier

For the project, we train a classifier on the CIFAR-10 dataset (54% of accuracy):

```
==========================================================================================
Layer (type:depth-idx)                   Output Shape              Param #
==========================================================================================
Net                                      [32, 10]                  --
├─Conv2d: 1-1                            [32, 6, 28, 28]           456
├─MaxPool2d: 1-2                         [32, 6, 14, 14]           --
├─Conv2d: 1-3                            [32, 16, 10, 10]          2,416
├─MaxPool2d: 1-4                         [32, 16, 5, 5]            --
├─Linear: 1-5                            [32, 120]                 48,120
├─Linear: 1-6                            [32, 84]                  10,164
├─Linear: 1-7                            [32, 10]                  850
==========================================================================================
Total params: 62,006
```

Figure 1: Naive Classifier Summary

The SGD optimization (learning_rate = 0.001, momentum = 0.9, valid_size = 1024, batch_size = 32, epoch = 10) is performed on the NLLLoss (LogSoftmax activation function).

## 3 Adversarial Attacks

### 3.1 White-box Attacks

A fast gradient-based method (FGBM - Goodfellow et al. (2015)) is used to generate adversarial examples to minimize the maximum perturbation amount added to image pixel to cause misclassification. The process uses the loss gradients

*E-mail: clement.preaut@dauphine.eu
†E-mail: iwan.quemada@dauphine.eu
‡E-mail: nicolas.durat@dauphine.eu

relative to the input image to create a new image that maximizes the loss. This new image is a contradictory image.

$$x^{adv} = x + \epsilon sign(\nabla_x L(\theta, x, y)) \tag{1}$$

The gradients are taken relative to the input image. We determine how much each pixel in the image contributes to the loss value and add a perturbation accordingly. This works quickly because it is easy to find how each pixel contributes to the loss by finding the required gradients. The model parameters remain constant and the goal is to fool an already trained model.

A more powerful technique is the FGSM multi-step variant which is a method for large-scale constrained optimization called projected gradient descent (PGD - Madry et al. (2019)):

$$x^{t+1} = \Pi_{B(x_0,\epsilon)}(x^t + \alpha sign(\nabla_x L(\theta, x, y))) \quad with \quad x_0 = x \tag{2}$$

- $\Pi$: Projection operator
- $B(x_0, \epsilon)$ : Hyperball ($x_0$ centered with radius $\epsilon$)

The model is simple, fast and efficient. It can be adapted to a constraint $l_1$ or $l_2$. We have more choices to make when specifying the attack, such as the step size $\alpha$ and the iteration number.

### 3.2 Black-box Attack

Natural Evolutionary Strategies (NES - Ilyas et al. (2018)) is a method for derivative-free optimization based on the idea of a search distribution $\pi(\theta|x)$. NES maximizes the expected value of the NLLLoss function under the search distribution:

$$\mathbb{E}_{\pi(\theta|x)}[F(\theta)] = \int F(\theta)\pi(\theta|x)d\theta \tag{3}$$

$$\nabla_x \mathbb{E}_{\pi(\theta|x)}[F(\theta)] = \nabla_x \int F(\theta)\pi(\theta|x)d\theta = \mathbb{E}_{\pi(\theta|x)}[F(\theta)\nabla_x log(\pi(\theta|x))] \tag{4}$$

Generally, the search distribution $\pi(\theta|x)$ is a Gaussian random noise around the input image $x$ such that evaluating the gradient with a sampled population $\delta_n$ along this scheme gives :

$$\mathbb{E}[F(\theta)] = \frac{1}{\sigma n}\sum_{i=1}^{n} \delta_i F(\theta + \sigma\delta_i) \tag{5}$$

From this gradient estimate, a PGD with momentum is applied. This NES method is considered as a finite-differences estimate on a random gaussian basis. The algorithm can be written as shown figure 3a.

## 4 Adversarial Defenses

### 4.1 Adversarial Training

Adversarial Training (Goodfellow et al., 2015) is a technique that provides a defense against a particular set of attacks. The goal is to minimize the worst-case error when data is perturbed by an adversary. It is a form of active learning where the model is able to request labels on new points. The algorithm will create and incorporate adversarial examples into the training process. The inner maximization problem is approximated with an FGSM or PGD attack.

$$\hat{L}(\theta, x, y) = \alpha L(\theta, x, y) + (1 - \alpha)L(\theta, x + \epsilon sign(\nabla_x L(\theta, x, y))) \tag{6}$$

The quality of the algorithm relies on the realization of the inner maximization. Indeed, the key aspect of adversarial training is to integrate a strong attack in the inner maximization procedure. Although this process approximately optimizes the robust loss, it is usual to include some loss as this also tends to slightly improve the error performance. It is also usual to randomize the starting positions, otherwise there may be problems with the learning loss surface of the process, so that the gradients at exactly the same points point in a shallow direction.

### 4.2 Mixed Adversarial Training

Mixed Adversarial Training (Araujo et al., 2019) is a method derived from Adversarial Training based on increasing the $l_{infty}$ and $l_2$ training examples. There are two main strategies to use it. The first one is to randomly select a

contradictory example among the most harmful norms.

$$\min_{\theta} \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}} \left[ \mathop{\mathbb{E}}_{p\sim\mathcal{U}(\{2,\infty\})} \left[ \max_{\|\tau\|_p \leq \epsilon} \mathcal{L}(f_\theta(x+\tau), y) \right] \right] \tag{7}$$

The second is to train with the most harmful adversarial example regardless of its norm.

$$\min_{\theta} \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}} \left[ \max_{p\in\{2,\infty\}} \max_{\|\tau\|_p \leq \epsilon} \mathcal{L}(f_\theta(x+\tau), y) \right] \tag{8}$$

Mixed Adversarial Training is a good technique against mixed loss maximization attacks. However, it is poor against perturbation minimization attacks. Therefore, it is little practical use.

### 4.3 Randomized Networks

Random networks (Pinot et al., 2019) are randomization methods that involve injecting noise from the exponential family distribution (Gaussian, Laplace, etc.) into the network during training and inference. In deterministic neural networks, Lipschitz constant becomes larger as the number of layers increases. By injecting noise at $i^{th}$ layer, robustness depends on the first $i$ layers sensitivity and not on the following ones. Randomization allows a continuity control of the the neural network i.e output distribution stability when injecting noise. Thanks to this robustness, we can limit the accuracy loss of a random neural network and provide a trade-off between accuracy and noise intensity. Indeed, random defenses rely on the injected noise amount. When the network has low noise, it causes no accuracy loss with low robustness, while when it has high noise, it causes greater robustness at the expense of accuracy loss. Thus, the noise must be calibrated to preserve both the accuracy and the robustness to attacks. In theory, this defense is more effective than the adversarial training whatever the attack.

## 5 Experimental setting

### 5.1 Evaluation

The CIFAR-10 dataset consists of 60000 32x32 images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images. It is divided into five training batches and one test batch, each with 10000 images. The test batch contains exactly 1000 images from each class. The training batches contain the remaining random images. Attacks were evaluated with the architecture described previously. We use the accuracy as evaluation metric and models are all $l_\infty$ norm attacks. Our objective is to evaluate the accuracy w.r.t the influence of the different parameters of the models ($\epsilon$, $\alpha$, $\sigma$ etc.) and the number of iterations to understand the models' functioning. Then, we check out and compare the scores of the different models with and w/o adversarial defenses. An exception is made for test of randomized network where $l_2$ norm is chosen.

### 5.2 Randomized networks

We have injected randomness in two different parts of the model training. First we have injected Gaussian noise with different standard deviation to the initial input of the model, that is to say the training pictures. Another way of injecting randomness was to add the same Gaussian noise, but at several steps. We added it to the input of the first convolutional layers.The standard deviation takes the value from 0, so no noise, 0.1, 0.3 and 0.5. We trained the models on 10 epochs. We also trained a Mixed Adversarial Trained model with epsilon 0.03 on 10 epochs to compare performances with this previous defense method.

## 6 Results

### 6.1 Adversarial Attacks

#### 6.1.1 FGSM and PGD

We can see how PGD attack weaken dramatically the accuracy of a naive model. For $\alpha = \epsilon$, with one iteration, FGSM is the same as PGD, which is theorically coherent. We do not need to choose too many iteration for PGD as 10 iterations is give pretty same results as with 50 iterations. But we can see that we have to choose a high enough number of iterations, at least to give $\alpha \times iterations > \epsilon$ if we want to modify enough the pictures and do not reach the limit of the attack too early as we see in the green curve of figure 2.

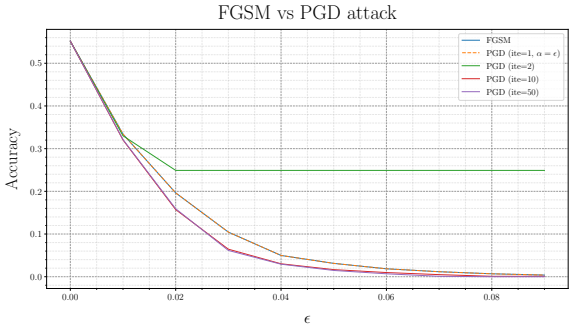



Figure 2: Influence of $\epsilon$ on Training Accuracy

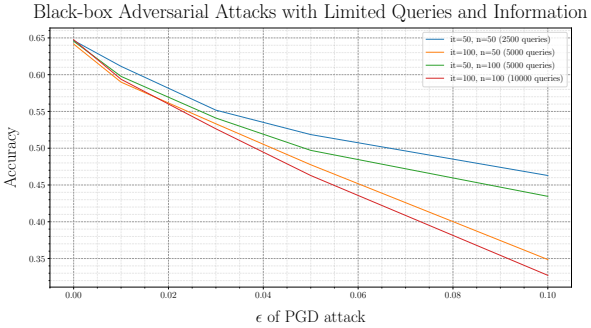
### 6.1.2 NES

After the FGSM and PGD, we tried to implement the Natural Evolutionary Strategies (NES), which is a black box attack. We implemented a version where we only use the best label prediction of the model (there exist lot of different version of black box which depend to the different access of the information of the model). The basic mechanism is to estimate the gradient with the output of the model : we add a gaussian noise to an image and figure out the gradient with the new classification of the noisy image. After that we compute a PGD just like before but with the estimate gradient. We can see some result figure 3b. We only present here our result in $l_{infty}$ norm attack because of the poor impact on accuracy in $l_2$. 'it' represent the number of iteration use in PGD and 'n' the number of sample use for estimate the gradient.

**Algorithm 1** NES Gradient Estimate

**Input:** Classifier $P(y|x)$ for class $y$, image $x$
**Output:** Estimate of $\nabla P(y|x)$
**Parameters:** Search variance $\sigma$, number of samples $n$, image dimensionality $N$
$g \leftarrow \mathbf{0}_n$
**for** $i = 1$ **to** $n$ **do**
  $u_i \leftarrow \mathcal{N}(\mathbf{0}_N, \boldsymbol{I}_{N \cdot N})$
  $g \leftarrow g + P(y|x + \sigma \cdot u_i) \cdot u_i$
  $g \leftarrow g - P(y|x - \sigma \cdot u_i) \cdot u_i$
**end for**
**return** $\frac{1}{2n\sigma} g$

(a) NES Algorithm




(b) Influence of hyper-parameters on Accuracy.

Figure 3: Natural Evolutionary Strategies

## 6.2 Adversarial Defenses

### 6.2.1 Adversarial training

Figure 4 shows us how the higher is $\epsilon$ during adversarial training, the lower is the initial accuracy without attack, but the more resistant the model is to attacks.

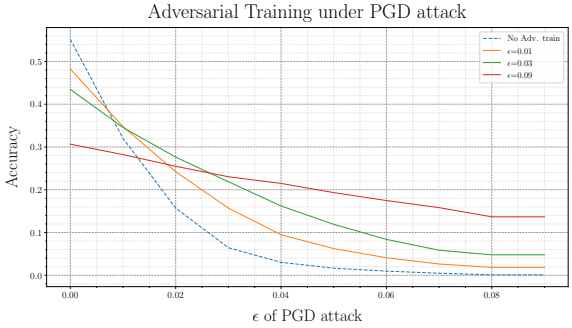



Figure 4: Impact of the $\epsilon$ in Adversarial Training ($\alpha = 0.5$) on Accuracy, under PGD attacks

### 6.2.2 Mixed Adversarial training

This defence mechanism, the Mixed Adversarial training, is a variation of the precedent adversarial training, but during the training it learn with both $l_2$ and $l_{infty}$ norm attack. As we can see figure 5, this method offer a good compromise between defence against $l_2$ and $l_{infty}$ norm attack. The 'adv l2' (green), respect. 'adv linf' (orange), is a model trained with $l_2$, respect. $l_{infty}$, norm attack. The basic model in blue and the mixed train in red.

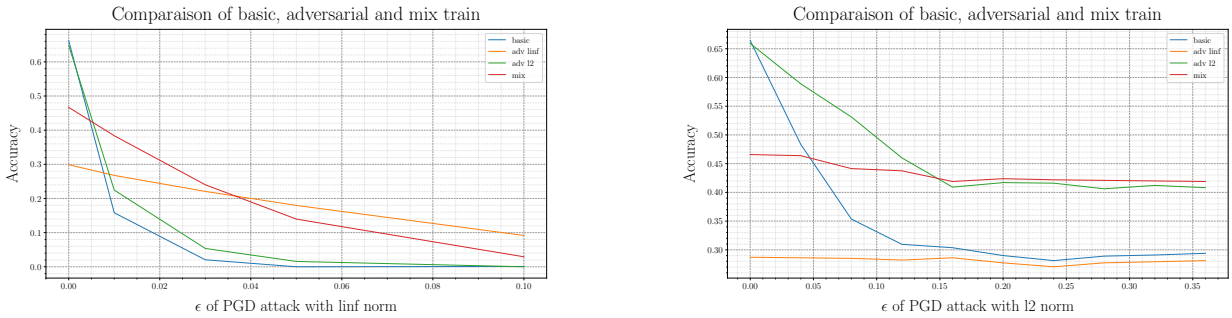



Figure 5: Adversarial training and mixed defence comparison.

### 6.2.3 Randomized network

We can see on the top left of figure 6 that the best standard deviation for a randomized input training is 0.1. It offers the best trade-off between stability and accuracy, here equivalent in the case of no attack to the non noised model. The top right graph shows that for a small number of epochs, we can have much better performances than mixed adversarial training. The bottom left figure shows that again the best standard deviation for a randomized input training is 0.1. Finally, the bottom right graphs illustrate that the random input and network training, in the case of only a few layers are randomized, offers equivalent performance. In further experiments, we could add noise on more layers to see the results.

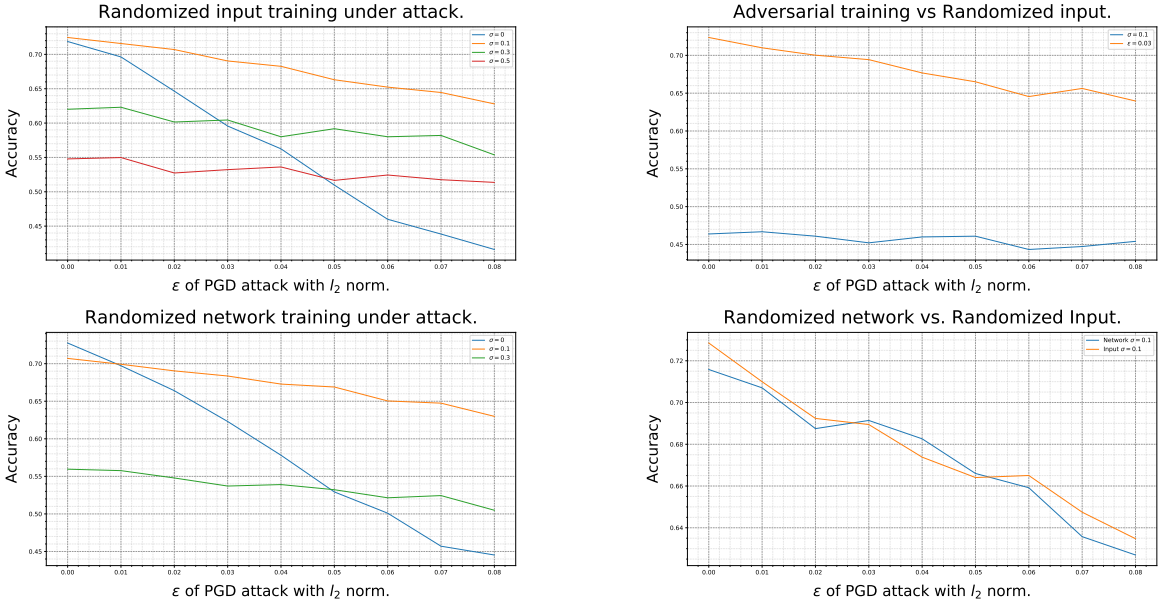



Figure 6: Adversarial Defenses models comparison

# 7 Conclusion

To sum up, we have found that attacks are very harmful to accuracy of models not prepared against that. The attacks are quite fast to execute concerning white-box attacks. About the defense concern, we can prepare our model by using the attack method during the training. The drawback is that the attack method used during the training would be significantly resistant to its specific associated attack's norm. So by using adversarial mixed training, we can generalize the defense but it costs much more training time. A solution which was quite efficient to get good defense relatively fast was randomized training.

# References


Alexandre Araujo, Laurent Meunier, Rafael Pinot, and Benjamin Negrevergne. 2019. Robust neural networks using randomized adversarial training.

Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and harnessing adversarial examples.

Andrew Ilyas, Logan Engstrom, Anish Athalye, and Jessy Lin. 2018. Black-box adversarial attacks with limited queries and information.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2019. Towards deep learning models resistant to adversarial attacks.

Rafael Pinot, Laurent Meunier, Alexandre Araujo, Hisashi Kashima, Florian Yger, Cédric Gouy-Pailler, and Jamal Atif. 2019. Theoretical evidence for adversarial robustness through randomization.