Adversarial training for image classfication

BIPARO Team : Nathan Bigaud, Salomé Papereux, Matthieu Rolland

December 2021

Contents

1 Setting and definitions

1.1 Introduction

White box adversarial attacks attempts to exploit models structure to fool its prediction efforts. An adversarial attack consists in subtly modifying an original image such that the changes are almost undetectable to the human eye but will yet lead to mis-classification by a neural network.

The most common constraint on those attacks are to impose an upper bound on how much perturbed examples can deviate from the original example - for instance in a ℓ_∞ and ℓ_2 ball.

We can also differentiate two main types of attacks which are non-targeted attacks and targeted attacks. One enforces the model to only misclassify the adversarial image, while the other aims to get the adversarial image misclassified as a specific target class.

1.2 Adversarial Attacks

1.2.1 Fast Gradient Sign Method (FGSM)

The fast gradient sign method attack consists of adding noise calculated by multiplying the sign of the gradient with respect to the image we want to perturb by a small constant ε. As ε increases, the model is more likely to be fooled, but the perturbations become easier to identify as well.

Shown below is the equation to generate an adversarial example within an ℓ_∞ ball from any original image x where ε is a very small number, ∇x is the gradient of the loss function w.r.t x, J is the loss function, h our classifier model, θ the model weights, and y the true label.

$$Adv(x) = x + \varepsilon * sign(\nabla x J(h_\theta(x), y))$$

1.2.2 Projected Gradient Descent Attack (PGD)

The PGD attack can be described as an iterative variant of FGSM. The algorithm is the following:

- Start from a random perturbation in the ball around a sample
- Take a gradient step in the direction of greatest loss
- Project perturbation back into the ball
- Repeat k times to converge

The algorithm can be written as follows:

$$x_0 = x, \quad x_{t+1} = \Pi_{B(0,\varepsilon)}(x_t + \alpha sign(\nabla x J(h_\theta(x), y)))$$

Note that the hyperparameters number of iteration t and the step size α may play a role in the efficiency of the attack, so they should be carefully chosen.

2 Adversarial Training

2.1 Some results using a Basic Network

Adversarial training is the process of training our model using compromised images. This is can be viewed as a preprocessing step where we create specific perturbations that best fool our model.

During such a training, we measure at each epoch the ongoing training accuracy on attacked example for training, the test accuracy on natural examples and the test accuracy for attacked example to evaluate the robustness of the network to a chosen type of adversarial attacks.

We tested several Basic neural network models on the CIFAR10 dataset, training and attacking with various attacks. We tested these on the PGD attack with ℓ_∞ and ℓ_2 norm, and the FGSM attack with ℓ_∞.

In table 1, we trained the Basic model for 40 epochs with a learning rate change at epochs 20 and 30 in order to compare the response of certain attacks to various defenses. With such a simple model we established that this method of training was enough to guarantee correct results. Those preliminary results allowed us to see that PGD training yields good robustness against all first-order adversaries (attacks that rely only on first-order information like the gradient) as stated in Madry et al. 2017.

	ℓ_∞ PGD training	ℓ_2 PGD training	ℓ_∞ FGSM training
Natural examples	40,7	**47,7**	43,2
ℓ_∞ PGD attack	**27,46**	27,22	24,15
ℓ_2 PGD attack	28,86	**32,7**	25,4
ℓ_∞ FGSM attack	27,96	28,41	**29,34**

Table 1: Test accuracies of different training under different attacks of a Basic Net

2.2 Training hyperparameter ε

In this section we wanted to define the impact of our adversarial training hyper-parameter ε on the robustness of our model knowing that an attack could be of any ε. We chose the PGD ℓ_∞ attack with $\varepsilon = 0.03$ as a reference attacker and trained our Basic model with different ε with ℓ_∞ PGD attack to evaluate the optimal ε to train with.

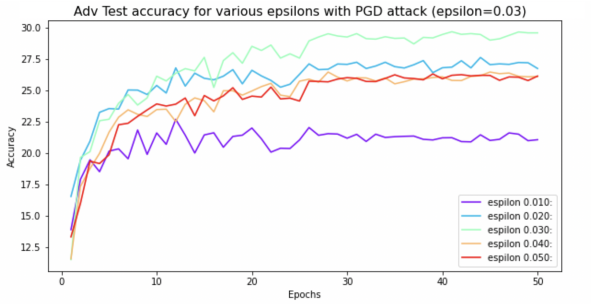


Figure 1: Adversarial PGD ℓ_∞ training test accuracy for various ε against PGD ℓ_∞ attack ($\varepsilon = 0.03$)

The graph of figure 1 allowed us to conclude that the most efficient ε to train our model with is the one we could be attacked with. We also see that 'over-training' is less damaging than 'under-training'. Given that in practice it is impossible to guess which ε an attacker could use, when in doubt practitioners should pick the largest realistic ϵ for training.

2.3 Mixed Adversarial Training

Mixed Adversarial Training is a variation of adversarial training that uses both ℓ_∞ and ℓ_2 bounded adversarial examples as training examples. We decided to use the *MAT-Rand* strategy that consists in randomly selecting one adversarial example among the two most damaging ℓ_∞ and ℓ_2, and to use it as a training example, as described in Araujo et al. 2020. Using a Wide Resnet18 we could obtain the following table.

Such results allowed us to get better performances than classic adversarial training against ℓ_2 **attacks** while keeping good overall results for other attacks.

	PGD ℓ_∞ training	PGD ℓ_∞/ℓ_2 Random training
Natural examples	**79.63**	76.94
ℓ_∞ norm PGD attack	**49.44**	46.52
ℓ_2 norm PGD attack	41.48	**52.08**
ℓ_∞ norm FGSM attack	**64.06**	61.44

Table 2: Test accuracies of different training mix of a Wide Resnet18 (in percentages)

2.4 Learning rate and convergence

A usual rule of thumb on complex datasets is that the longer a neural network is trained, the better and more accurate it is. Adversarial training has proved to be different, as emphasized by Rice, Wong, and Kolter 2020. After a certain point, further training will continually decrease training loss while increasing test loss. We tested this for ourselves using a BasicNet (fig. 2) the change of performance is linked with the first drop in the learning rate decay.

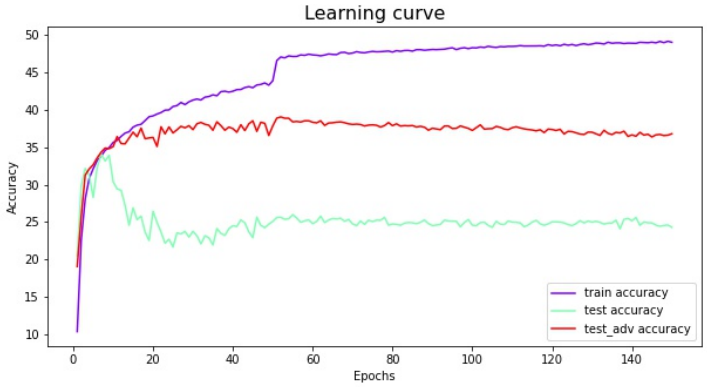


Figure 2: Over-fitting in robust adversarial training

Rice, Wong, and Kolter 2020 proposed two main solutions to face over-fitting: complexify the architecture of the model in terms of size (which still improves test set performance despite over-fitting), or proceed to early stopping. Due to limited resources, we chose the latter approach.

3 Network architecture and optimization

3.1 Network capacity

Here we aim to test at a small scale a simple claim in Madry et al. 2017 : model capacity by itself increases accuracy and robustness.

	Standard Resnet20	Wide Resnet18	VGG11
Natural examples	70.01	**79.94**	75.16
ℓ_∞ norm PGD attack	39.76	**43.52**	43.48
ℓ_2 norm PGD attack	48.10	**54.08**	52.54
ℓ_∞ norm FGSM attack	53.82	**61.44**	57.24

Table 3: Accuracies of different model architectures (in percentages)

We train three models over 50 iterations on a random mix of ℓ_∞ and ℓ_2 PGD: a baseline Resnet20 with about 250 k parameters, a wide ResNet18 with around 11 millions parameters, close to the one used in the paper, and a 'deeper' VGG11 model with about 11 million parameters (table 6). The

hyperparameters used are `learning_rate`= 0.01, `epsilon` = 0.03 and `alpha` = 0.0075, and a learning schedule dividing the `learning_rate` by 10 at 30 and 40 iterations.

Unsurprisingly, more capacity drastically improve robustness and in our case natural accuracy. The intuition put forward by Madry et al. 2017 being that robust classification requires a much more complicated decision boundary, as it needs to handle the presence of possible adversarial examples.

We also find that for a given network capacity, wider rather than deeper network perform better in terms of robustness - a stylized fact known since Madry et al. 2017, and exploited in most adversarial studies (see Wu et al. 2021 for instance). The theoretical explanation for this difference remains an open topic. A recent systematic comparison of wide and deep networks (Nguyen, Raghu, and Kornblith 2020) found that increase in any of the two reveals the emergence of a characteristic block structure that reflects the similarity of a dominant first principal component, they tend to make different errors at a class or example level.

3.2 Randomness and robustness

We explore two commmon ways of introducing randomness in the networks and improve generalization : noise injection at predict and NoiseNets.

3.2.1 Noise Injection

Noise injection is the injection of random noise at inference, taking x at input the model will predict $f_\theta(x+\eta)$. As stated by Araujo et al. 2020, it is in practice less effective than adversarial training but has a solid theoretical basis.

To isolate the effect of noise injection, we compared ℓ_∞ norm PGD adversarial training of two BasicNets and two ResNets, introducing Gaussian Noise injection at prediction in one of them, and compare their results. For the BasicNet, we observe a clear gain in test accuracies accross the board. The results for the ResNet are more surprising, as noise injection seem to increase adversarial robustness but decrease the natural accuracy - a result that would need confirmation trough systematic testing.

	BasicNet	BasicNet with Noise	Wide ResNet	Wide Resnet with noise
Natural examples	24.02	**25.53**	**79.94**	78.13
ℓ_∞ norm PGD attack	19.71	**20.78**	43.52	**44.04**
ℓ_2 norm PGD attack	18.56	18.58	54.08	54.11
ℓ_∞ norm FGSM attack	19.76	**20.32**	61.44	61.15

Table 4: Adversarial test accuracies of a BasicNet and a wide Resnet with and without noise injection at predict (in percentages)

3.2.2 Network randomisation

We implemented the approach of Rakin, He, and Fan 2018, introducing trainable Gaussian noise injection at each layer of the network. In their paper, the noise addition improves both clean- and perturbed-data accuracy in comparison to the state-of-the-art defense methods.

We tested this using similarly-sized ResNets, one with noisy layers one without, following the same procedure as previously.

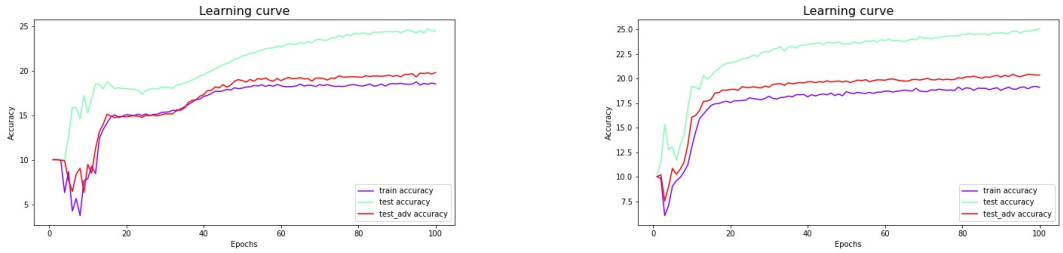
While we do observe a significant increase in adversarial accuracy, this comes at an important cost in terms of natural accuracy - a very different result from the paper results.

One possible explanation would be the need for a different training schedule for a noisy version of the ResNet to fully exploit the architecture's potential. For lack of time and computing resources,

	NoiseNet20	ResNet20
Natural examples	64.62	**70.01**
ℓ_∞ norm PGD attack	**41.15**	39.76
ℓ_2 norm PGD attack	**49.01**	48.10
ℓ_∞ norm FGSM attack	49.51	53.82

Table 5: Adversarial test accuracies of a Resnet20 and a NoisyResNet20

we have not engaged in a full retraining of the ResNet, but reproducing the experiment at smaller scale with a BasicNet, we observe a fairly different behavior of the Noise Net, with a slowly increasing plateau arising much earlier in training and lasting longer, suggesting longer training time could have lead to better results.



(a) BasicNet (b) NoisyBasicNet

Figure 3: Learning curves over 100 iterations of a BasicNet and a NoisyBasicNet

Another promising avenue we did not have the time to explore was Bayesian networks, in particular very recent work by Panousis, Chatzis, and Theodoridis 2021, which achieves impressive performances using a stochastic competition-based activation (Stochastic Local Winner-Takes-All).

3.3 Putting it all together : network optimization strategy

Using an iterative exploration process, we combined the teachings of several papers on network capacity, randomness, training mix, and learning rate.

- **Network capacity**: our final submission is a 17 million parameters wide ResNet, increasing width of our original 11 million wide network, while remaining under the constraint of GitHub file size limit.
- **Randomness**: we add noise injection at predict, building on Araujo et al. 2020, allowing using the same value for allowed perturbation amplitude and the epsilon used in our adversarial attacks.
- **Training mix**: we build on Wu et al. 2021 and use a mix of 95% adversarial examples and 5% of natural examples to train our network. The difference with the paper's approach is that rather than integrating the mix in the optimization problem, we randomly introduce natural batches at training, to minimize computations.
- **Learning schedule**: we use a shorter version of the training schedule in Wu et al. 2021, over only 50 epochs and implementing early stop as suggested by Rice, Wong, and Kolter 2020.

	Wide Resnet18	Wider Resnet18 manual mix
Natural examples	79.63	**80.37**
ℓ_∞ norm PGD attack	49.44	**52.34**
ℓ_2 norm PGD attack	52.48	**56.02**
ℓ_∞ norm FGSM attack	64.08	**65.35**

Table 6: Accuracies of different Resnet Designs (in percentages)

The result is a wide ranging improvement of the model. On the platforms, absolute numbers are different, but comparative accuracies do correspond. Most notably, our network outperforms most other networks in natural accuracy, while retaining good adversarial performances - however the final score on the platform does not take into account natural accuracy.

4 Conclusion

In this work we have explored the trade off stated in Su et al. 2018 between robustness and accuracy, with our best model only reaching around 80 % of accuracy for a fairly network in comparison to the dataset.

The two most promising avenues for further exploration, given more time and resources, would be looking at

- **Bayesian networks**: as described in Panousis, Chatzis, and Theodoridis 2021 seem to be the most promising avenue to achieve greater performance with similar resources.
- **NASNets**: an automated and theoretically informed network architecture optimization procedure, (see Guo et al. n.d. and Liu and Jin 2021.
- **Attack forensic**: as described in Bonnet, Furon, and Bas 2020. Given the difficulty of designing an all encompassing defense to adversarial attacks, being able to probe images at predict and determine which type of attack they would most likely be the result of could greatly improve the defense of a network in real world applications.

References

Araujo, Alexandre et al. (2020). "Advocating for Multiple Defense Strategies against Adversarial Examples". In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, pp. 165–177.

Bonnet, Benoıt, Teddy Furon, and Patrick Bas (2020). "Adversarial images through stega glasses". In: *arXiv preprint arXiv:2010.07542*.

Guo, Minghao et al. (n.d.). "When NAS Meets Robustness: In Search of Robust Architectures against Adversarial Attacks SUPPLEMENTARY MATERIAL". In: *training* 100.150 (), pp. 30–60.

Liu, Jia and Yaochu Jin (2021). "Multi-objective search of robust neural architectures against multiple types of adversarial attacks". In: *Neurocomputing* 453, pp. 73–84.

Madry, Aleksander et al. (2017). "Towards deep learning models resistant to adversarial attacks". In: *arXiv preprint arXiv:1706.06083*.

Nguyen, Thao, Maithra Raghu, and Simon Kornblith (2020). "Do wide and deep networks learn the same things? uncovering how neural network representations vary with width and depth". In: *arXiv preprint arXiv:2010.15327*.

Panousis, Konstantinos P, Sotirios Chatzis, and Sergios Theodoridis (2021). "Stochastic Local Winner-Takes-All Networks Enable Profound Adversarial Robustness". In: *arXiv preprint arXiv:2112.02671*.

Rakin, Adnan Siraj, Zhezhi He, and Deliang Fan (2018). "Parametric noise injection: Trainable randomness to improve deep neural network robustness against adversarial attack". In: *arXiv preprint arXiv:1811.09310*.

Rice, Leslie, Eric Wong, and Zico Kolter (2020). "Overfitting in adversarially robust deep learning". In: *International Conference on Machine Learning*. PMLR, pp. 8093–8104.

Su, Dong et al. (2018). "Is Robustness the Cost of Accuracy?–A Comprehensive Study on the Robustness of 18 Deep Image Classification Models". In: *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 631–648.

Wu, Boxi et al. (2021). "Do Wider Neural Networks Really Help Adversarial Robustness?" In: *Thirty-Fifth Conference on Neural Information Processing Systems*.