Building Robust Neural network models
for Image classification

Dakini MALLAM GARBA, Killian SUSINI, Christian KAYO

Context

Image classification is one of the major topics in Computer Vision and can be applied in different domains.
For the model to be reliable, it has to be robust. It appears in [1] that small perturbations, non visible to the human eye, applied to an image can change the class predicted by the model.

The aim of this project is to make an image classification model more robust to small perturbations. To achieve this, different techniques have been explored :

- Attacks mechanisms to produce adversial images.
- Defense mechanisms to train the model against these images.

1 Attacks Mechanisms

In general, to get an adversarial example we use an example and change it in such a way that it is not easily visible to the human eye. An idea to generate an adversarial example is to find the example within a fixed radius (hypersphere S) that maximizes the loss J of the model defined by θ:

$$\hat{x} = \max_{\delta \in S} J(\theta, x + \delta, y).$$

1.1 FGSM

Because each pixel can be changed independently, using an l_∞-ball with a small size ($\epsilon > 0$) allows to maximize the loss while keeping the change to every pixel limited. To do so, one can use general constrained optimization techniques, but they can be expensive to compute. An approximation, the Fast Gradient Sign Method (FGSM), was proposed in [1]. It works by taking a single step in the general direction of the gradient ($\nabla_x J(\theta, x, y)$) of the image. If the gradient is directly used, compute the point that is the farthest in the hypersphere in the direction of the gradient. A better heuristic is too instead use the sign of the gradient, which essentially lead us to compute the corner of the hypersphere the closest to the gradient direction.

$$\hat{x} = x + \epsilon sign(\nabla_x J(\theta, x, y)).$$

This single step is already sufficient to cause a commonly trained model to fail on most inputs.

1.2 PGD

An extension of the above method, prosposed in [2], can be obtained using Projected Gradient Descent (PGD). It is essentially multiple step variant of the above method, where each step is taken in a smaller

l_∞-ball of size $\alpha < \epsilon$ and projecting back into the l_∞-ball of size ϵ if the input is outside of it (hence Projected Gradient Descent).

$$x^{(t+1)} = \prod_{x+S} \left[x^{(t)} + \delta sign(\nabla_x J(\theta, x, y)) \right].$$

Even more generally, one can instead use a l_2-ball instead of an l_∞-ball (So the Euclidean norm instead of the max norm, as mentioned in [3]) which cause the attack to be slightly different.

With only a few steps (10), the adversarial examples generated causes regular model to fail in even more cases than for FGSM.

2 Defense Mecanisms

2.1 Adversarial Training

Adversarial Training, first proposed in [1], aims to protect the model against adversarial examples by training against them during the learning. During training, given a data sample, the model learns to predict the class of the natural sample (as usual), and also to correctly predict the class of the attacked sample, based on the current state of the model. This part is modelled by a regularisation term, as such,

$$\tilde{J}(\theta, x, y) = \alpha J(\theta, x, y) + (1 - \alpha) J(\theta, \hat{x}, y),$$

where $\alpha = 0.5$ usually (which is what we use), and $\hat{x}$ is the adversarial image of x. This last one is obtained through one of the attack mechanism presented in 1 (FGSM, PGD_∞, PGD_2). The choice of attack (and there hyper-parameters) affect the robustness of the obtained model against certain attacks (see Results 3). Indeed, we expect a model trained with l-∞ attacks to not be as resistant against l-2 attacks, and vice-versa. The natural evolution, Mixed Adversarial Training, uses both l-∞ and l-2 attacks during training (In our case, the model is trained on an equal proportion of natural, PGD_2 and PGD_∞ samples).

2.2 Randomized Smoothing Network

Randomized Smoothing Network was proposed in [4]. The principle is the following :

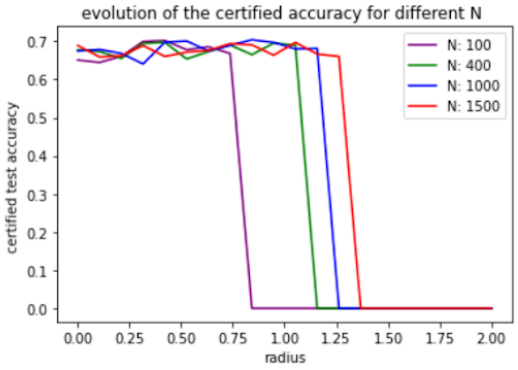
1. Assume that we have a classifier named f trained with gaussian data augmentation.
2. We have an image x. We make n copies of this image, to which we add random gaussian noise.
3. f assigns labels to the n images. This way, we have the probability of x belonging to each class.
4. The smooth classifier abstains if the probability of the 2 more popular classes are close, otherwise it returns the first one.

The methods also allows for certification, which is certifiying the robustness of the smooth classifier around a radius r.

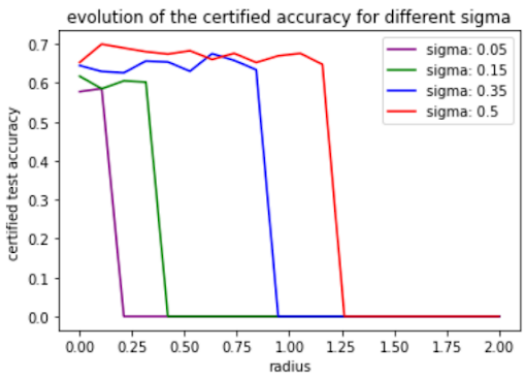
In Figure 1, we have the evolution of the approximated certified accuracy. We see that the accuracy decreases with large radii until reaching a limit where it drops to zero.

Figure 1a gives us the evolution when trying different values for N, the number of image copies generated. As we can see, a higher value of N permits certifying around a biggest radius.

in Figure 1b, we compare the evolution the accuracy when trying different values for σ , the standard deviation used to generate the noise added to the copies of the image. The value of σ works as a



(a) with different N values



(b) with different sigma values

Figure 1: Evolution of the approximated certified accuracy

trade-off between the robustness and the accuracy. A low σ enables to certify small radii with high accuracy but not large radii. When σ is high, we can certify large radii but the small radii have a low accuracy.

3 Results

We conducted several experiments to determine the robustness of different models against various attacks. The results are summarized in table 1. It describes the accuracy (percents of correctly classified inputs) of various defense methods against diverse attack methods.

The images used for training are standardized (all the values are set between 0 and 1). As mentioned, the adversarial methods mentioned require to define some hyperparameters. For PGD, we test both l_∞-ball and l_2-ball attacks (PGD_∞ and PGD_2 respectively). In each case, we use 10 steps. We have also to define the size of the hypersphere. For the l_∞-ball (FGSM or PGD_∞), we use 0.03, which was a value that made the "basic classifier" fail nearly all the classification on PGD_∞. This is larger than what would be required to make the adversarial example unnoticeable to the human eye (usually, each colors are encoded on 8 bits, so 0.007 would be within one pixel of displacement [1]). This larger value makes the problem harder on the model however. Using the same ball size for the l_2-ball (used by PGD_2) gives an easier problem, so we adjusted it to 0.045, which makes the "basic classifier" perform similarly on PGD_∞ and PGD_2. Finally, we used two-third of the ϵ for the δ (inner step, which is 0.02 and 0.03 for PGD_∞ and PGD_2 respectively).

We should note that in [3], they use a similar setup (with many more methods), and notably they use 0.031 and 0.83 for PGD_∞ and PGD_2 respectively. They picked 0.83 because then this would make the two balls have a similar volume. We tried to do that, but in our case the PGD_2 examples were very hard for every models, and we preferred to pick the values that made comparisons easier.

All the models use the same architecture, and are trained for 30 steps. In total, 5 models are trained:

1. *Basic classifier*: trained on the images without any modification (beside standardisation);
2. *Basic classifier with noisy training*: Gaussian noise is added to each image during the training;
3. *Basic classifier with PGD_2 adversarial training*: As described in 2, with the same hyperparameters used for the attack ($\epsilon = 0.045, \delta = 0.03, t = 10$);

4. *Basic classifier with PGD_∞ adversarial training*: As described in 2, with the same hyperparameters used for the attack ($\epsilon = 0.03, \delta = 0.02, t = 10$);

5. *Basic classifier with mixed adversarial training*: As described in 2, same hyperparameters. The loss was split into equal part natural images, PGD_∞ and PGD_2 adversarial images.

The last row of the table 1, *Randomized Smoothing*, is the method described in 2 using the *Basic classifier with noisy training.*

As expected, the model without any "defense" folds against each of the three methods, going from 58.2% on the natural dataset to 3.81% when the simplest attack (FGSM) is used. As we can see, just training the model with gaussian data augmentation makes it already pretty robust against all the attacks, despite the nature of the l_∞-ball based attacks (which are quite different to gaussian noise). However, it does struggle more against PGD_∞ than PGD_2 adversarial examples. It's behavior is slightly better but similar to the PGD_2-AT, with a noticeable boost in accuracy on natural images.

We can see that PGD_∞-AT and PGD_2-AT have opposite accuracy on $\text{PGD}_2/\text{PGD}_\infty$ examples, which makes sense considering the examples they saw during their training. And precisely for the same reasons, Mixed-AT perform well against every type of attack, combining the advantages of both methods. We can however note a loss of performance for each of the AT methods, especially Mix-AT (51.76%), on natural images.

Finally, we notice that using randomized smoothing significantly improves the results, and works much better than adversarial training on all the attacks we implemented, reaching nearly 40% for adversarial examples. Given the starting point (0.29% for PGD), we can say that the defense mechanism training and method was fructuous.

4 Conclusion

The goal of this project was to implement state-of-the art methods to make a neural network more robust, in the context of image classification. Generating adversial examples with standard methods like FGSM and PGM, and implementing defense mechanisms, enabled us to improve the performance of our model on such task. Further improvements may be obtained with other attack and defense mechanisms.

Accurracy	Basic classifier	Basic classifier + noisy train	Adversial training (PGD-l2)
Natural	58.2	62.16	54.78
FGSM	3.81	26.17	25.00
PGD l-inf	0.29	21.77	20.99
PGD l-2	0.29	25.16	23.82
Accurracy	Adversial training (PGD- linf)	Mix adversial training	Randomized Smoothing
Natural	54.41	51.76	59.71
FGSM	28.32	28.41	38.1
PGD l-inf	25.58	26.46	37.68
PGD l-2	19.33	27.15	39.8

Table 1: Results.

References

[1] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining ans harnessing adversial examples," *ICLR*, 2015.

[2] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," 2017. [Online]. Available: https://arxiv.org/abs/1706.06083

[3] A. Araujo, L. Meunier, R. Pinot, and B. Negrevergne, "Robust neural networks using randomized adversarial training," 2019. [Online]. Available: https://arxiv.org/abs/1903.10219

[4] J. M. Cohen, E. Rosenfeld, and J. Z. Kolter, "Certified adversarial robustness via randomized smoothing," 2019. [Online]. Available: https://arxiv.org/abs/1902.02918