Data Science Project A3 - IASD 2023
An Adversarial Approach to Image Classification

Hippolyte Gisserot, Guillaume Kunsch, Benjamin Sykes
Group: *insidious_gbh_attack_epsilon_bounded*

1 Introduction

In the past few years, deep learning models have become more and more efficient due to a huge increase in global computational power. However, despite their high level of precision, techniques have been developed to intentionally fool these models, the idea being to take advantage of their structure to feed them with slightly transformed input data and generate wrong predictions. Such attacks on machine learning models are known as adversarial attacks.

In this work, we train a basic classifier on the CIFAR-10 data set and analyse how it reacts to several data perturbation types. Then, still working on this baseline model, our goal is to train a robust version of it, i.e., as resilient as possible to adversarial attacks.

2 Baseline classification model

The main purpose of this work is not to train the best possible classifier, but rather to assess the evolution of its resilience to a bunch of attack types depending on the way it is trained, i.e., the data it is fed with.

The architecture we use for the model is therefore fairly simple, as follows:

1. 1 Conv2D layer with 6 (5x5) kernels,
2. 1 MaxPooling2D layer with stride 2,
3. 1 Conv2D layer with 16 (5x5) kernels,
4. 3 consecutive FC layers of size 120, 84 and 10 respectively.

Training this model on natural images for 20 epochs and with binary cross-entropy loss yields a test accuracy of 68.5%. Now, the goal is to assess how this performance evolves when the model is fed with attacked samples.

3 Attack mechanisms

3.1 White-box attacks

White-box attacks are a family of perturbations that require a full access to the model (weights, loss function, gradients...) in order to be implemented. Implementing these kind of attacks is therefore a strong hypothesis, as it means the attacker has full knowledge of the architecture and weights.

3.1.1 Fast gradient sign method (FGSM)

The idea behind FGSM [1] is to perturb images in an ϵ-bounded direction that maximizes the loss function. In other terms, the goal is to perturb samples enough to potentially generate a wrong prediction from the classifier, but without making it possible to notice any difference with the naked eye (see Figure 1).

For a given sample x belonging to class y and when ϵ is small, this amounts to solving the following optimization problem:

$$\max_{\|\delta\| \leq \epsilon} Loss(x + \delta, y) \approx \max_{\|\delta\| \leq \epsilon} \delta^T \nabla_x Loss(x, y) \tag{1}$$

When $\|\cdot\| = \|\cdot\|_\infty$, the above simplied problem 1 has a closed form solution and the resulting image is:

$$x_{perturbed} = x + \delta^* = x + \epsilon sign(\nabla_x Loss(x, y)) \tag{2}$$

NB: this solution is not exact since it is derived from a first-order approximation.

Table 1 shows the performances of the baseline model on FGSM-perturbed images, for different values of ϵ. As expected, the larger the value of ϵ, the more important the perturbation and therefore the lower the accuracy.

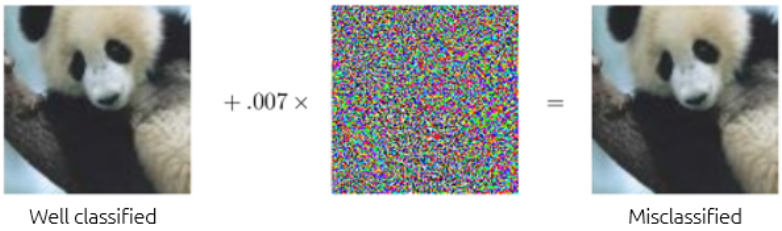

Figure 1: FGSM attack illustration [1]

ϵ	0	0.001	0.01	0.1
Test accuracy	68.5%	56.4%	20.1%	0.1%

Table 1: Performance of the baseline model on FGSM-attacked images

3.1.2 Projected gradient descent (PGD)

The PGD method [2] can be thought as an iterative version of FGSM. Intuitively, for an untargeted attack, the idea is to translate the natural image to an increasing-gradient direction with a step of $t \ll 1$ before projecting it on a hyperball with small radius ϵ, and repeating the process several times in a row (N). The goal of such a method is to get a better approximation of the best attack that can be performed in a given range of magnitude. An illustration of the optimal solution found by both PGD and by FGSM if given in Figure 2 for the $\|\cdot\|_\infty$ norm.

If the attacks were to be targeted, the iterations would be very similar, but we would first compute the gradient for the loss of iterate image with the target label and we would go in the decreasing direction of this loss.

The corresponding algorithm for a given sample x with associated label y can be written as follows for an untargeted attack:

- $x_0 = x$
- For $k = 1, ..., N$: $x_{k+1} = \Pi_{B(x,\epsilon)}(x_k + t sign(\nabla_x Loss(x, y))$, where:
  - $t > 0$ is the step size
  - Π is the projection operator
  - $B(x, \epsilon)$ denotes the hyperball for a specified norm with center x and radius ϵ

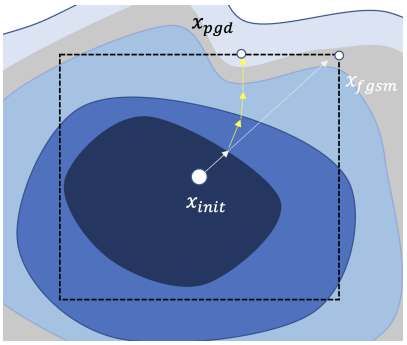

Figure 2: Optimal solution of PGD compared to FGSM

Table 2 shows the performances of the baseline model on PGD-perturbed images, for different values of ϵ and t set equal to $\frac{\epsilon}{4}$ which we found to be optimal. We used $N = 40$. An important caveat for PGD is that in order to find a solution at least as good as the one of FGSM, the parameters should be set such that

$$N \cdot t \leq \epsilon \tag{3}$$

This means that in the worse case, the obtained result is that of FGSM as we let the algorithm go as far as the border of the ball. With very small values of ϵ we found that the accuracy after the PGD and the FGSM attacks

were very similar. This could be explained by the fact that for a small epsilon, the first order approximation given by FGSM is very close to reality and PGD finds the same local optimal solution.

ϵ	0	0.001	0.01	0.1
Test accuracy	68.5%	54.6%	12.6%	0.0%

Table 2: Performance of the baseline model on PGD-attacked images

Once again, the same observation is made: the higher ϵ, the lower the accuracy. But one can also notice that the accuracies are comparatively lower than for FGSM, the PGD iterative process making it possible to get a better estimate of the best bounded perturbation.

However, taking too large values of ϵ is irrelevant as it would highly hinder image quality. $\epsilon = 0.01$ appears to be a good compromise in this respect (significant accuracy deterioration without making a difference to the naked eye) and is used in the rest of this work.

3.2 Black-box attacks

Contrary to white-box attacks, black-box attacks are performed without knowing anything about the underlying model.

3.2.1 Random perturbation

Random perturbations are the most simple way of attacking images. The idea is only to add random noise to a given sample x, within an ϵ-bounded range of directions having $\delta \sim Unif([-1,1]^{32\times32\times3})$:

$$x_{perturbed} = x + \epsilon\delta \tag{4}$$

3.2.2 Difference of means

The difference of means method is more sample-specific and requires an access to the full training data set. It proceeds in several step:

1. For each label, compute the mean of all images belonging to this label $(\mu_0, ..., \mu_9)$
2. For each sample x (with associated label y), compute the euclidian distance between μ_y and $(\mu_i)_{i\neq y}$ and select the minimum value $\delta^* = d(\mu_{i^*}, \mu_y)$
3. Finally, use δ^* as the perturbation direction: $x_{perturbed} = x + \epsilon\delta^*$

In short, this perturbation method is targeted towards the mean-wise closest class.

3.2.3 Gradient approximation by finite difference

Finally a naive approach to attack a model without having access to its weights is to compute the gradient by finite difference. In this case, we assume that we have access to the output of the model for a given input. The idea is to approximate $\nabla Loss(x)$ by the tensor of coefficients $(\frac{f(x+e_i\epsilon)-f(x)}{\epsilon})$ for a given $\epsilon \ll 1$ and where the e_i are the canonical basis vectors of $\mathbb{R}^{32\times32\times3}$.

This assumes the attacker can use the model to run some predictions on data, and that the output is visible to him. This method can be used for example as the basis of a FGSM attack, by perturbing the image by a small step-size times the estimated gradient. It is important to note that this method is trivial to run from the same device where the model is stored and running, but it gets far more complicated when the model is online, as in our simple setting, this method already requires $3 \times 32 \times 32 = 3072$ iterations.

If evaluating the model were costless, then a finer approximation would be given by the second order approximation : $\nabla_i Loss(x) \approx \frac{f(x+e_i\epsilon)-f(x-e_i\epsilon)}{2.\epsilon}$. While the first approximation costs $3 \times 32 \times 32 = 3072$ calls to the model, its error is $\mathcal{O}(\epsilon)$ and the second order approximation requires twice as many accesses but has a error of $\mathcal{O}(\epsilon^3)$. An other property that could be used if the evaluations were costless, would be to iterate the attack, using the PGD framework. Nevertheless, as we just saw, in our example, we are already calling the model some 3072 times, iterating the PGD framework with $n_{\text{iter}=40}$ would mean multiplying by 40 the number of calls to the model compared to the first implementation, which makes this attack merely theoretical. The results we found for an $\epsilon = 10^{-3}$ are that of the FGSM attack.

4 Adversarial training methods

4.1 Training on FGSM or PGD-perturbed data

The key idea of training a model robust to the FGSM and PGD attacks, is to include attacked images in the training process of the network. At each iteration, instead of using only the generic images in the loss, we minimise a linear combination of the loss computed on the generic data and the loss computed on untargeted attacked images. In plain words, we this way optimize the model to fit the normal images, and prevent it from "overreacting" to attacked images. We thus train the model over the new loss function defined in 5 where h is the attack function that maps the input images x to some attacked images $h(x)$.

$$Loss_{\text{robust}}(x, y, \alpha, h) = \alpha Loss(x, y) + (1 - \alpha) Loss(h(x), y) \tag{5}$$

We therefore have a new hyperparameter α that captures how much we want the model to be robust to adversarial FGSM/PGD attacks. The challenge is to find a value of α which will, at the same time keep a good prediction on natural images, and weaken the adversarial attacks. We found the value $\alpha = 0.9$ to be a good trade off.

In Figure 3, we show the evolution of loss and accuracy while training a model using the loss in 5, we also look at the evolution on adversarial images. The training was made on 20 epochs, with $\alpha = 0.9$

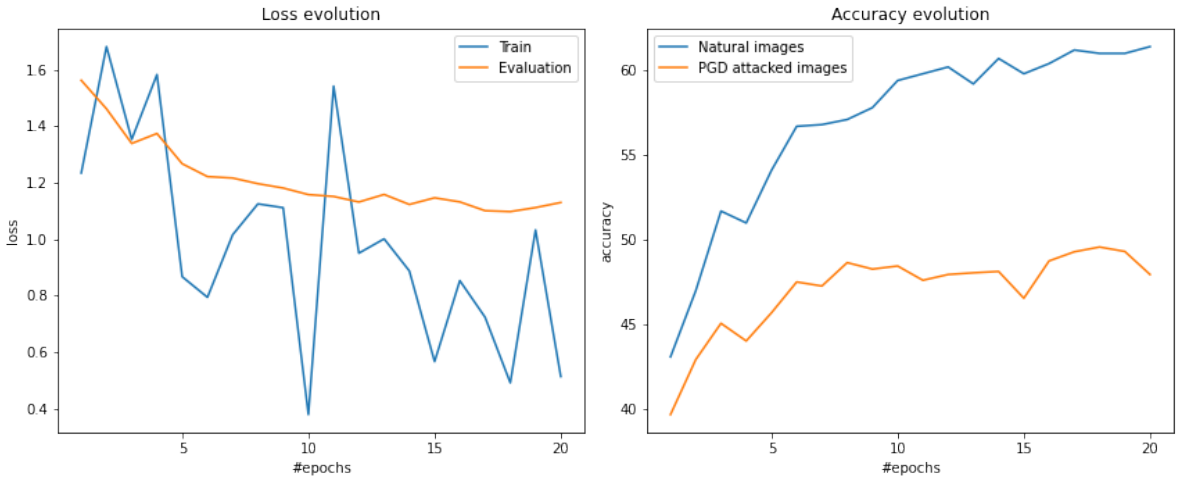

Figure 3: Training of a robust to PGD model

4.2 Training on VAE-generated data

As a recall, a variational autoencoder is a model that takes an image as an input, then maps it to a smaller-dimension latent space, samples from this space before finally trying to reconstruct the input image as reliably as possible. Intuitively, a VAE estimates the distribution from which each data sample is drawn.

In this section, the idea is to build and train a full VAE on the training data and then concatenate the classifier (which has been trained separately) at the end of it. The intuition behind is to create a counterattack mechanism this time not during the training phase but when testing new images. Figure 4 illustrates how this pipeline works.

1. First, the attacker feeds the model with a perturbed image.
2. Then, the image is sent through the VAE, which builds a reconstruction of the input by sampling from its estimated underlying distribution: the goal of this step is to temper the effect of the attack by introducing some randomness, the VAE being supposed to identify from wich more general distribution the perturbed image is drawn.
3. Finally, the reconstructed image is sent through the classifier and makes a class prediction.

To conclude, table 3 is a summary showing the performance of each of the trained models against the different types of attacks.

Observations:

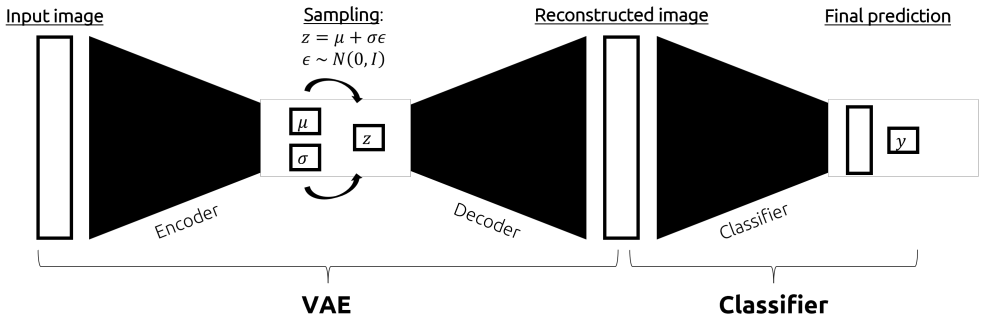

Figure 4: VAE-based testing pipeline

Models / Attacks	Vanilla	Robust FGSM	Robust PGD	VAE-based
None	68.5%	63.8%	61.4%	28.7%
FGSM ($\epsilon = 0.01$)	20.1%	43.8%	45.3%	23.2%
PGD ($\epsilon = 0.01$, $t = \epsilon/4$)	12.6%	42,9%	44.7%	16.0%
Random ($\epsilon = 0.01$)	67.9%	63.1%	61.3%	28.5%
Difference of means ($\epsilon = 0.01$)	66.3%	61.8%	62.0%	27.9%
Finite difference	20.1%	43.8%	45.3%	23.2%

Table 3: Models' performance versus various attacks

- Unsurprisingly, the robust FGSM and PGD models have better performances on FGSM- and PGD-perturbed samples than the Vanilla one, while showing a slight decrease in accuracy when fed with natural images. In short, there is trade-off between the desired performance on attacked and on natural images.
- Fully black-box attacks (random and difference of means), i.e. those which do not have access at all to the model, are not very efficient. They should work better on more overfitting models with more sinuous borders.
- The finite difference method is a very good approximation of FGSM (same observed accuracies).
- The VAE-based model shows a disappointing performance, probably due to a too small number of parameters in the VAE part.

5 Conclusion

In this work we have implemented various attacks and robust training for image classification. There is no attack-proof network and each training method has it's own characteristics that we have tried to emphasize in theses lines (the more complex the method, the harder it is to train). The next step for our work would imply training the VAE with more parameters on more computing power, and trying to find a way to train both the VAE and the classifier in a coordinated fashion.

Besides, a method we would have liked to implement in further works is the enforcement of Lipschitz continuity on the model layers [3]. The idea is to ensure the overall architecture of the network is a Lipschitz function, as a result a small perturbation in the input will automatically result in a small perturbation in the output. In a sense for such a network, it wouldn't be possible to fool the network at the naked eye.

References

[1] Goodfellow et al. (2014) Explaining and harnessing adversarial examples, ICLR 2015

[2] Madry et al. (2017) Towards Deep Learning Models Resistant to Adversarial Attacks

[3] Gouk et al. (2020) Regularisation of neural networks by enforcing Lipschitz continuity