Research project

Adversarial attacks

November 17, 2020

Charles De Trogoff
Hugo Da Costa

Supervisor:

Geovani Rizk

M2 Intelligence Artificielle, Systèmes, Données
Mathématiques et Informatique de la Décision et des Organisations
University of Paris Dauphine - PSL

Contents

1 Introduction

An adversarial attack is an input that has been intentionally crafted to fool a Machine Learning System. Specifically, we take an image classification system that has been trained to classify images, and an adversarial attack is an input to this network that has been changed maliciously so that it is indistinguishable to a person but that a neural network system gets incorrect.

The existence of these errors raises a variety of questions about out-of-sample generalization and whether bad actors might use such examples to abuse deployed systems. When this phenomenon was first discovered, many researchers began to work on how to defend against these attacks.

These defenses can take different forms, for example neural networks that perform better on such attacks, or that try to figure out how to distinguish between an attack and a normal input. Nevertheless, it seems that all the work around adversarial attacks falls into the same cycle. Indeed, somebody presents a defense or a detection mechanism and claims robustness to adversarial attacks (meaning we either can classify them as adversarial or we classify them correctly, publish a paper about it). And then a couple months later, someone comes along and makes a new adversarial attack that breaks this defense. It seems that this cycle has been going on for many years now. Researchers seem to propose new defense methods only to have it be broken later. This is probably frustrating and it has not been solved yet over the several years that it has been studied.

Current research tries to get ahead of the cycle by creating provable robustness to adversarial attacks. Before, the papers were written about adversarial attacks robustness they would do their own experimental tests against known attack strategies (a new way of finding the noise). Many ways of finding that noise exist in the literature, the simplest is probably to follow the gradient of the model with respect to increasing the loss, see Madry *et al.* (2017). People would test out their model on as many attack strategies as they could find, and they would show robustness on those attack strategies.

In this document, we will present a very humble work which is still in this cycle of trying to propose new attacks. We will present our study of Adversarial attacks, starting by the L_∞ - PGD attack, and the impact on a classic Convolutional neural network trained on CIFAR-10 dataset. It will then be followed by a research for robustness against this attack. Then, we share our work concerning eventual new aspects of L_∞ - PGD attack, possible improvements, and finally the new models of attacks that we have been able to work on. This last part mainly deals with the local aspect of an Adversarial attack.

2 Training of a Convolutional Neural Network over Cifar10 dataset

2.1 Quick description of the CNN's structure

Our work begins with the definition of a convolutional neural network (CNN - as we call it in the future), with the following architecture :

- 3 convolutional layers, with Max Pooling (size 2)
- Between them, we placed a Dropout of 0.2 to prevent our CNN from overfiting
- 4 fully connected layers at the end
- Each of these layers is using ReLU as activation function

2.2 Training and evaluation of quality

To train our CNN over Cifar10 data set, we define an optimizer based on stochastic gradient descent (SGD) with a learning rate equals to 10^{-3}. The loss we use is the Cross-entropy loss. The training has been done over 10 epochs, using batches of size 4. Moreover, we keep 10% our train set to make a validation set and observe the possible overfiting of our CNN (which didn't happen).

At the end of the training, the CNN's loss was approximately at 0.8 and its accuracy on the test set reached almost 70%. The following figures illustrate our training.

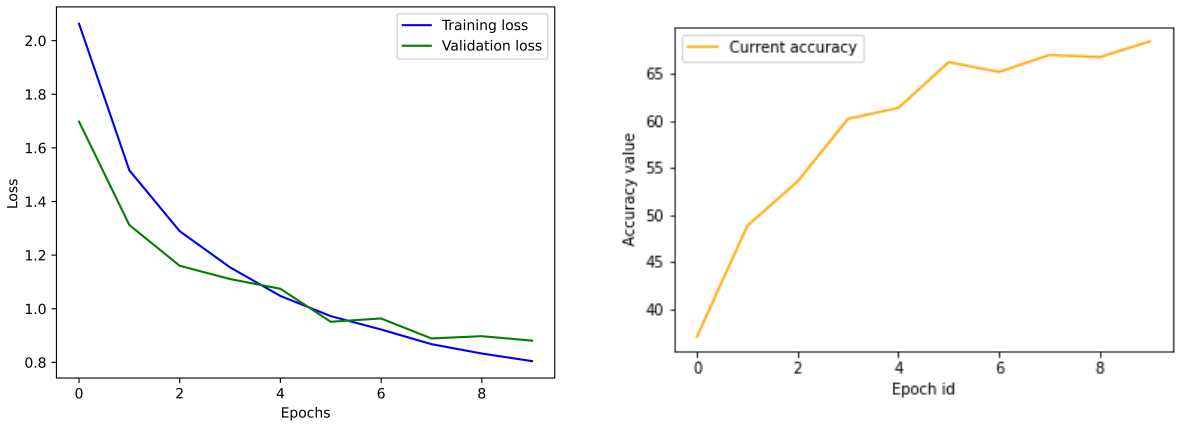


Fig. 1. On the left, evolution of the training loss and validation loss during training on the CIFAR-10 dataset. Dropout has been adjusted to prevent over-fitting. On the right, evolution of the accuracy of the network during training.

Table. 1. Accuracy of the classic CNN on each of the CIFAR-10 category.

Category	plane	car	bird	cat	deer	dog	frog	horse	ship	truck
Accuracy [%]	80.0	72.2	53.0	47.4	69.8	56.8	75.9	74.2	80.6	79.3

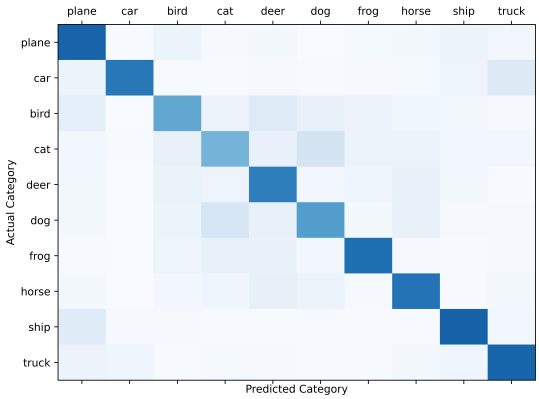


Fig. 2. Confusion matrix of the Classic CNN on the CIFAR-10 test dataset. The accuracy is computed on each of the following labels: {'plane', 'car', 'bird', 'cat','deer', 'dog', 'frog', 'horse', 'ship', 'truck'}.

3 Diving into L_∞ - PGD attack and Adversarial robustness

3.1 Reminder on white box attacks

In the domain of adversarial attacks, we can distinguish white box attack from the black box one. A white box attack has total access to the model and to its parameters. Thanks to these accesses, the attack takes information from the model (the loss for example) and can try to maximize the chance to fool the Classifier, by giving the right parameters values to the inputs.

This kind of attacks is considered much more powerful than the Black box kind (which doesn't have these accesses), and White box attacks are usually crafted to find the weakness of the Classifier (here the CNN).

3.2 Presentation of L_∞ - PGD

Projected Descent Gradient (known as PGD) comes from Madry *et al.* (2017). It is a white box attack which is define by : finding the perturbation that maximises the loss of a model on a particular input while keeping the size of the perturbation smaller than a specified amount referred to as ϵ. Mathematically speaking, this means that PGD is looking to find a perturbation δ (a vector of the same shape than the input images) which will maximizes the loss of the CNN, while in the mean time, δ is bounded in the $B(x, \epsilon)$ (where x is the input image). Different norms can be used, here we choose to study the L_∞ one. Here is the pseudo code of our version of this attack.

Following, an observation of the L_∞-PGD attack on a set of 8 images from CIFAR-10, where we can observe the small perturbation δ, and the misclassification of the CNN.

3.3 Distorsion metric : measure a perturbation δ

As we can see on the Figure (Fig. 3), we add a metric named *Distorsion* over our plot of the attack, so that we can observe "what is really doing" our attack/perturbation.

Algorithm 1 Projected Gradient Descent (PGD)

Inputs : network (CNN), X (clean images), y (labels), $\eta = 2/255$, $\epsilon = 8/255$, $n_{iteration} = 7$
Initialize perturbation δ *(randomly or at zero) and* $X_{adv} = X$
for $i = 1, ..., n_{\text{iteration}}$ **do**
  Actualize δ *value :* $\delta = \delta + \eta sign(\nabla_\delta(L(f(X_{\text{adv.}}, y))))$
  Force $X + \delta$ *to be in the* $B(X, \epsilon)$
  Actualize X_{adv} *value :* $X_{\text{adv}} = X_{\text{adv}} + \delta$
end for
Return X_{adv}*, as attacked inputs*

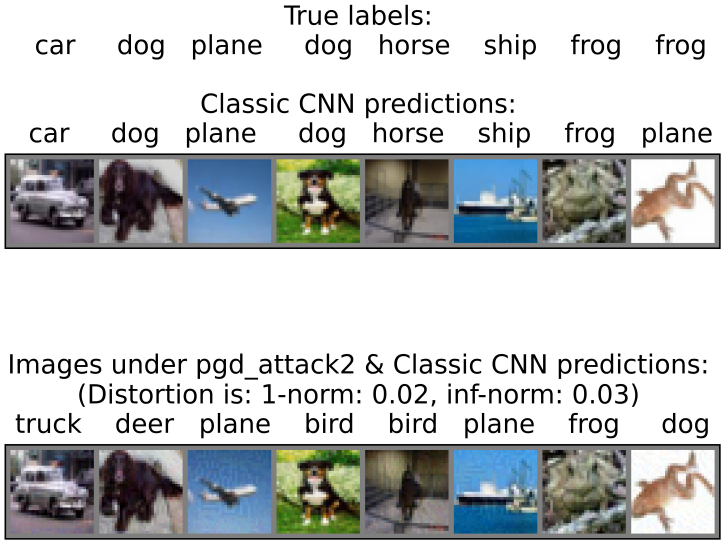


Fig. 3. Raw images and their true labels from CIFAR-10 dataset, predictions from Classic CNN on both these raws images and on their corresponding images attacked by the PGD attack.

We decided to use this metric (which gives us an overview of the movement of δ to assure ourselves that our attack respect conveniently the constraint of being in the bawl of radius ϵ. Following the norm L_2 or L_∞, in each plot or visualization of this document, you will have the possibility to measure the norm of our attaque in the inputs-space, and verify that it is indeed in $B(X, \epsilon)$.

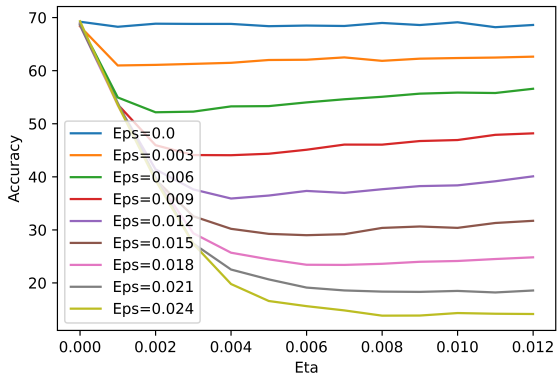
3.4 Study of the hyper-parameters

After defining and implementing our version of L_∞-PGD attack, we remarked that the impact of this attack (visible by the accuracy of the CNN over attacked images) is very sensible to the given hyperparameters. Indeed, instinctively, the major parameters to change would be the radius of the Perturbation's bawl, ϵ by default equals to $8/255$, and also the "learning-rate" or "step" of the attack, η by default equals to $2/255$.

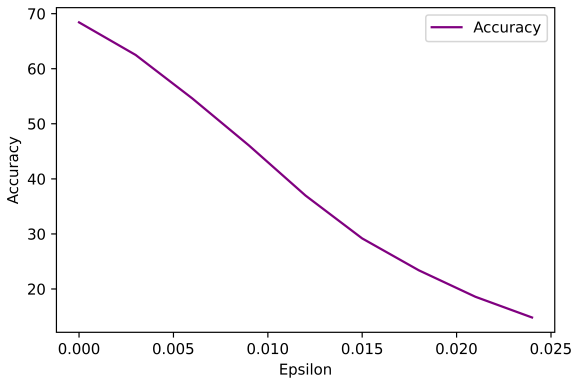
In the nexts plots, we present you different variations of the hyperparameters to better understand their respective importances. First of all, radius of the bawl ϵ, illustrates its impact on the accuracy on Fig.4 (a) and (b). In Fig.4 (a), we can observe that for different values of η, the perturbation named here δ always reaches the Limit of the bawl, visible by the stop of the accuracy decrease. This property is true for each value of ϵ. In a second time, we can observe the power of ϵ growing, that put the CNN classification to an accuracy equals almost to 0%.

In Fig.4 (b), the curve is also very expressive about its meaning. The increase of ϵ, for a given value of η equals to 2/255 as in the Referenced papers, makes significantly the CNN accuracy goes to 0%.

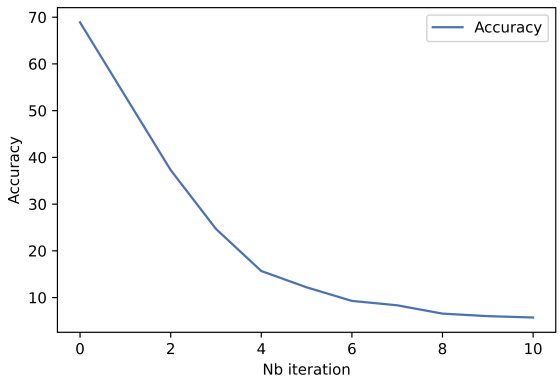
Finally, in Fig.4 (c), we choose to study the variation of the CNN accuracy when the L_∞-PGD attack is defined by different values of iterations. Indeed, the evolution of the pertubation is also very related to the number of steps the attack is allowed to do. As we can see here, increase this number of iterations leads the accuracy of the CNN to 0%.



(a) Accuracy according to η, for differents values of ϵ



(b) Accuracy according to ϵ (radius of the infinite ball of the perturbation). $n_{\text{iterations}} = 7$, $\eta = 2/255$.



(c) Accuracy according to the number of iterations of L_∞-PGD. Here, $\epsilon = 8/255$, $\eta = 2/255$.

Fig. 4. Accuracy of the classic CNN on the PGD-attacked test CIFAR-10 images according to the three main hyper-parameters. As a reminder, ϵ is the infinite norm of the perturbation, η is the step of the PGD, and $n_{\text{iterations}}$ is the number of iterations.

3.5 Adversarial training

After looking in detail at the L_∞-PGD attack, understanding its hyperparameters and their impact on the attack, the second step was to study the Robustness that we can obtain by training a Convolutional Neural Network with attacked image.
Indeed, in the context of Adversarial attacks, two paths are available. The first one, study the power of an attack, looking for weakness in the Classifier and its properties, has been done with our approach of L_∞-PGD attack. The second one is to understand how can the "Adversarial training".

To be able to compare the networks, Classic-CNN and (future) adversarial trained CNN, we decide to train an "empty" new object CNN of our class described in the first part of this document. This second CNN has been trained with :

- layers, dropout, and batches size are the same
- 10 epochs
- inputs are Attacked images by L_∞-PGD attack, with these parameters : $\epsilon = 8/255$, $\eta = 2/255$, 7 iterations

We followed the evolution of the accuracy of this network over Clean images (ie : classic images of the test set of the CIFAR-10 data set) and also over attacked images.

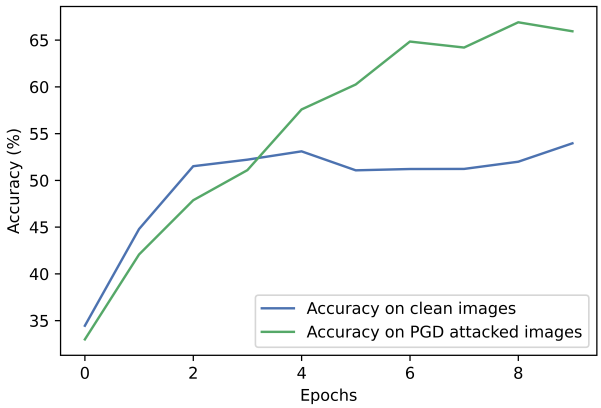


Fig. 5. Accuracies of the CNN during the Adversarial training, done with PGD-attacked images, with $\eta = 2/255$, $\epsilon = 8/255$, and 7 iterations

Take a look at Fig.5, the Adversarial trained CNN's evolution of the two accuracies. When the Adversarial trained CNN has its accuracy over clean images that can't be improved, we observe that its accuracy over attacked images is still increasing. We consider this new Adverarial trained CNN robust over L_∞-PGD attack, and also quite accurate on the Clean images of CIFAR-10 dataset (comparing to the classic CNN).

The next step of our study of Adversarial attacks is to dive into new concept of Attacks, and to compare (if it has sense) the new attacks' power on the Adversarial trained CNN, and on the classic-CNN.

4 Targeted attacks, inspired by L_∞ - PGD

In this section of our project, we decide to study the "targeted" aspect of L_∞ - PGD. In this case the only difference is that instead of maximizing the loss of the true label, we maximize the loss of the true label and also minimize the loss for the alternative label.

The main question and motivation of this section is to search and understand : how can we define a good way to chose the "alternative label"

4.0.1 Random shifted labels

The first step and try of our work is to chose this alternative label, randomly, and apply a gradient descent on it, to learn a bad classification to the model.

Algorithm 2 Random shifted labels

Inputs : network (CNN), X (clean images), y (labels), $\eta = 2/255$, $\epsilon = 8/255$, $n_{iteration} = 7$
Initialize perturbation δ (randomly or at zero)
Create a batch of shifted labels by taking a random fake label : $y_{shifted}$
for $i = 1, ..., n_{\text{iteration}}$ **do**
  Actualize δ value : $\delta = \delta - \eta sign(\nabla_\delta(L(f(X+\delta), y_{shifted}))$
  Impose that $X + \delta$ is still in the $B(X, \epsilon)$.
end for
Return $X + \delta$, as attacked inputs

4.0.2 Nearest neuron shifted labels

While studying the literature, we made the following observation. A small change in the input is going to have a small change in the output. In the literature, small means distance-wise small (it measures the arithmetic distance between two vectors). We make the assumption that a small change in the distance metric is going to result in visual imperceptibility. But we know that euclidean distance is a bad approximation for visual similarity. We can see in Fig. 6 taken from Wang & Bovik (2009) that all of the faces shown are equivalently far away from the reference image situated in the middle. If we measure the distance between any one of these images, it would be the same number. And yet, these look completely different. And many of these are obviously visually dissimilar to this one. The reason for this is that our eyes are measuring relative properties of an image. The main idea that we took from this is that we can try to misclassify with an image which is close to our original image, but far semantically. Or in other words not probable. In the following we will play with this: keep a small perturbation but find a label that is the least probable. For this we will use the neurons of the last layer of the CNN.

So, concretely we decided to suggest a better "bad classification" to the CNN. We knew that L_∞ - PGD already does the best perturbation in the input space, so we decided to look for a perturbation in the semantic / probabilistic aspect. Indeed, the less probable label of an input image, isn't always the furthest class in the input-space.

Here we chose to define this alternative label, by taking the 2nd most probable label (to make the perturbation small enough) and apply a gradient descent on it, to learn a bad classification to the model.

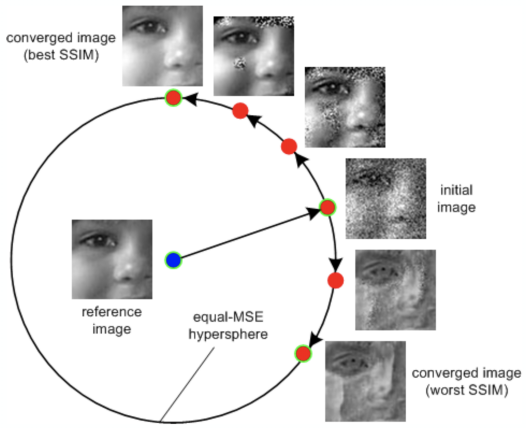


Fig. 6. Images equally far away from a reference image in the L_2 sense can be dramatically different in perceived distance. Figure due to Wang & Bovik (2009).

Algorithm 3 Nearest neuron shifted labels

Inputs : network (CNN), X (clean images), y (labels), $\eta = 2/255$, $\epsilon = 8/255$, $n_{iteration} = 7$
Initialize perturbation δ *(randomly or at zero)*
Create a batch of shifted labels by taking the 2nd most probable prediction for the images X *:* $y_{shifted}$
for $i = 1, ..., n_{\text{iteration}}$ **do**
  Actualize δ *value :* $\delta = \delta - \eta * sign(\nabla_\delta(L(f(X+\delta), y_{\text{shifted}}))$
  Force $X + \delta$ *to be in the* $B(X, \epsilon)$.
end for
Return $X + \delta$*, as attacked inputs*

4.0.3 Furthest neuron shifted labels

Here we chose to define this alternative label, by taking the less probable label (to make the perturbation small enough) and apply a gradient descent on it, to learn a bad classification to the model.

Algorithm 4 Furthest neuron shifted labels

Inputs : network (CNN), X (clean images), y (labels), $\eta = 2/255$, $\epsilon = 8/255$, $n_{iteration} = 7$
Initialize perturbation δ *(randomly or at zero)*
Create a batch of shifted labels by taking the less probable prediction for the images X *:* $y_{shifted}$
for $i = 1, ..., n_{\text{iteration}}$ **do**
  Actualize δ *value :* $\delta = \delta - \eta * sign(\nabla_\delta(L(f(X+\delta), y_{\text{shifted}}))$
  Force $X + \delta$ *to be in the* $B(X, \epsilon)$.
end for
Return $X + \delta$*, as attacked inputs*

4.1 Visualization of the white-box targeted attacks

cf Fig.7

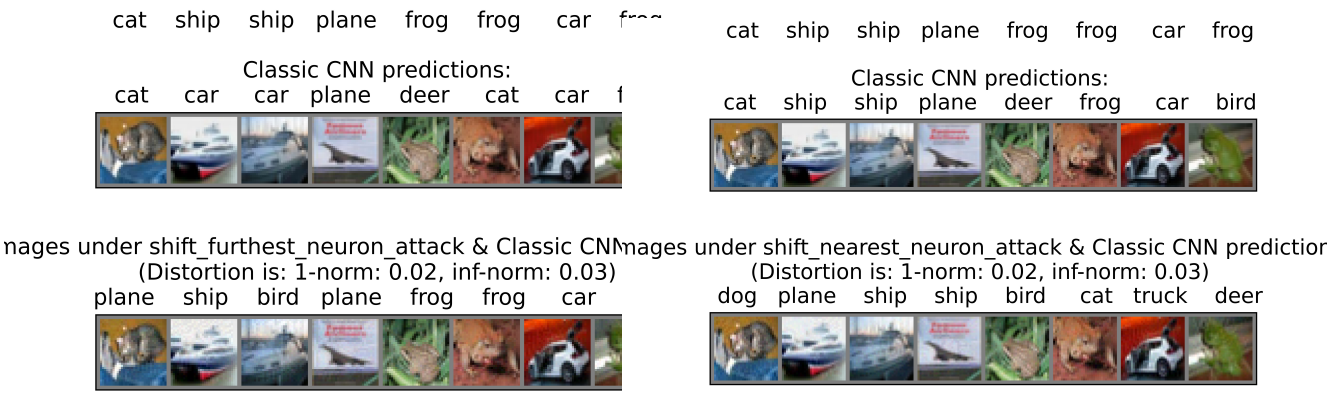


Fig. 7. Visualization of Furthest neuron shifted attack (on the left) and the Nearest shifted neuron, with $\eta = 2/255$, $\epsilon = 8/255$, and 7 iterations.

4.2 Results of our new attacks

cf Table 2.

Table. 2. Accuracies of the classic-CNN and Adv-Trained-CNN against each attack.

attack :	*Acc of classic-CNN [%] against*	*Acc of AdvTrained [%]* against	
		Attack on classic-CNN	*Attack on itlsef*
PGD	7.73	66.04	0.91
Shift random	27.02	60.55	9.75
Shift nearest neuron	27.99	57.84	21.6
Shift furthest neuron	36.35	58.37	17.45

4.3 Conclusion about this kind of attacks

After implementing these attacks, we thought that we should compare them to the L_∞ - PGD results over different accuracy. Indeed, even if these attacks are all white-box attacks for the Classic-CNN, they can also appears into two ways for the Adversarial trained CNN. We decided to study the white-box way for the Adversarial trained CNN, but also the Black box one.

The conclusion that we can make about these attacks, is that the "targeted" caracter, that we thought would be a intelligent "suggestion" of perturbation for an attack like L_∞ - PGD, was finally like a constraint to the attack, that was forced to learn a misclassification to the CNN, where L_∞ - PGD is free to perturbe it as it wants.

We wanted to study the "probabilistic" approach of the classification made by the CNN, forgetting the space dimension decision that it's making. These results tell us that the L_∞ - PGD approach is at this point the best one.

5 An attack inspired by the optimization problem from Carlini & Wagner (2017)

We introduce here a new attack that was inspired from Carlini & Wagner (2017), while taking a lot of freedom. The ideas taken from the attack of L2-Carlini Wagner are mainly:

- We keep the idea of trying to minimize the 2-norm of the δ perturbation in the objective function (reflected in our code by an L_2 regularizer of delta in the loss of the classifier)
- With a second traditional part in the loss (cross-entropy), we keep trying to mis-classify the labels.
- We do not use an ϵ constraint as an argument of this attack since there is already the regularizer in the second part of the loss.

Basically, the new main ideas are:

- We try to mis-classify the network on the most unlikely label by using the smallest amplitude in the last layer of the convolutional network.
- We compute a simple sum of the cross-entropy and the 2-norm of the δ perturbation for the final loss.

A version of the pseudo-code for this new attack is presented in the following algorithm **??**. This version is slightly simplified since it doesn't take in account the decaying rate that was added in the code, but which only has a very small impact on the final accuracy. In this attack, the δ perturbation is updated according to

$$\delta = \delta - \alpha \nabla_\delta [\zeta L_\delta(f(X+\delta), y_{\text{shifted}}) + \|\delta\|_2^2] \tag{1}$$
$$= \delta - \alpha \nabla_\delta [\zeta L_\delta(f(X+\delta), y_{\text{shifted}})] - 2\alpha\delta \tag{2}$$

The attack can be visualized in Fig. 9. In this figure, one can see the predictions on the attacked images with $\zeta = 1.5$ on top, and with $\zeta = 3$ in the bottom. This ζ constant regulates the relative importance of the cross-entropy loss compared to the regularizer part of the loss. It means that a big ζ will diminish the influence of the regularizer, and as a consequence, will produce a stronger attack. On the opposite, putting this constant to zero will produce a very weak attack. This statement is visually confirmed by Fig. 9 where we clearly see that the norm of the perturbation is smaller for smaller ζ. The best trade-off we found was $\zeta = 1.5$ since it allows a not too strong attack. The confusion matrix on the attacked test images can be seen in Fig. 8. Globally, no label is preferred in the attack, not like in the next attack that will be presented later on.

Algorithm 5 Furthest neuron 'Carlini-inspired'

Inputs : network (CNN), X (clean images), y (labels), $\alpha = 1$, $\zeta = 1.5$, $n_{iteration} = 8$
Initialize perturbation δ (randomly or at zero)
Create a batch of shifted labels by taking the less probable prediction for the images X : $y_{shifted}$
for $i = 1, ..., n_{\text{iteration}}$ **do**
  Actualize δ value : $\delta = \delta - \alpha \nabla_\delta [\zeta L_\delta(f(X+\delta), y_{\text{shifted}})] - 2\alpha\delta$
end for
Return $X + \delta$, as attacked inputs
??

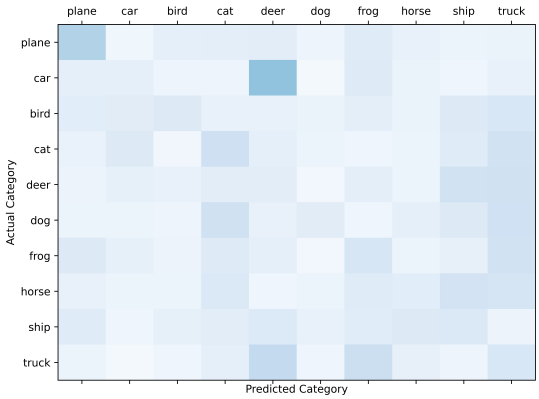


Fig. 8. Confusion matrix of the Classic CNN on the CIFAR-10 test dataset, attacked with the furthest-neuron-carlini attack. The accuracy is computed on each of the following labels: {'plane', 'car', 'bird', 'cat','deer', 'dog', 'frog', 'horse', 'ship', 'truck'}.

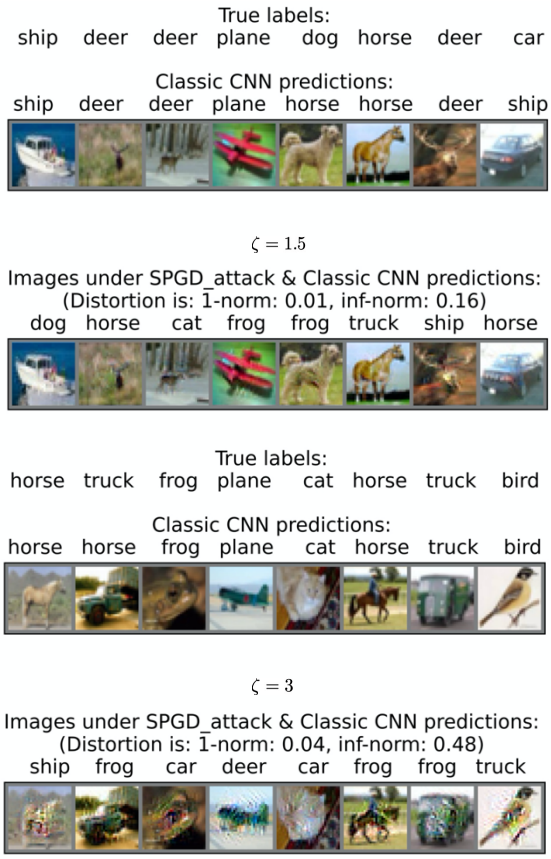


Fig. 9. Raw images and their true labels from CIFAR-10 dataset, predictions from Classic CNN on both these raws images and on their corresponding images attacked by the furthest-neuron-carlini attack, according to the ζ parameter ($\zeta = 1.5$ on top image, and $\zeta = 3$ on bottom image).

6 Locally unconstrained attack

6.1 Main ideas

So far, we have focused on perturbations applied to the entire image. Thus, these previous attacks imposed a small norm constraint so that the final image would be visually almost identical. For this first type of task, it seems that Projected Gradient Descent (PGD) is the most efficient of them all.

If the ultimate constraint is to guarantee visual recognition of the image, another possibility may be to modify only a small part of the image, so that the rest of the image is not modified. Therefore, we propose to determine the optimal combination of pixels forming a square of given width, placed in the bottom right corner of the initial image. The main interest of this type of attack, is to be able to get rid of the constraint on the perturbation norm. Several questions arise: Can a modification of only a few pixels be enough to fool the classifier efficiently? How far can we reduce the width of the square while guaranteeing an efficient attack? Is the effectiveness of the attack not simply proportional to the infinite norm of the perturbation? (In other words, is an attack consisting of single black pixels more effective than any other combination of pixels?).

6.2 Implementation of a "corner" attack

Choice of optimization Concerning the implementation of this attack that we called "corner" attack, we had to make the choice of the optimization technique to use. Many experiences using the native SGD functionality of the Pytorch optimizer were disappointing, compared to the technique finally chosen. Indeed, we finally took inspiration from the gradient rise contained in PGD, i.e. looking for the maximum loss in less than 10 iterations, using the sign of the gradient. Of course, the attack on this part of the image is not constrained in an ϵ-radius as we have already explained. Moreover, the η-step for each iteration is around 10 compared to 8×10^{-3} for previous PGD attacks which much bigger.

Pytorch Autograd Concerning details about the implementation in Pytorch, the implementation of this attack was also an opportunity for us to understand in depth Pytorch's Autograd tool (management of leaves and gradients, parameters and constants), which until then seemed to us like a somewhat magical feature.

- First initialize a zero corner tensor with track-able gradient called C, of shape the bottom right part of the initial input X_{batch} tensor.
- Initialize a "constant" zero tensor (with no track-able gradient) on the left of this corner tensor which is supposed to remain a simple constant which won't be updated during training.
- Repeat the procedure with a "constant" tensor (with no track-able gradient) above the concatenation of the two previous tensors

The concatenation of all of these tensors reproduces a partially upadate-able perturbation δ that will be added to the initial batch tensor X_{batch} throughout the training loop as we can see in Algo. 6, following the following update rule:

$$C = C + \eta \operatorname{sign}(\nabla_C L_C(f(X_{\text{adv.}} + \delta(C)), y) \tag{3}$$

Visualization of attacks In Fig. 13, one can see the comparison between the raw test CIFAR-10 images ans their corresponding attacked versions with first a corner of width 5 pixels, and then 10 pixels. One can note the interesting pattern formed by the pixels in the corner. The infinite norm is around 35 out of 255 which is enough to compute an efficient attack as we said earlier. Of course, since the 1-norm is computed on δ which is mainly composed of zeros, the 1-norm of δ seems quite small compared to the inifinite norm.

Algorithm 6 Corner Attack

Inputs : network, X (batch of clean images), y (corresponding labels), $\eta = 10$, $n_{\text{iterations}} = 7$

Initialize C: the bottom right corner sub-perturbation of δ at zero as: $\delta = \left[\begin{array}{c|c} 0 & 0 \\ \hline 0 & C \end{array}\right]$

for $i = 1, ..., n_{\text{iterations}}$ **do**

Compute loss of net, by comparing its prediction of X_{adv}*' with* y

Actualize C *value :* $C = C + \eta\,\text{sign}(\nabla_C L_C(f(X_{\text{adv.}} + \delta(C)), y)$

end for

Compute X_{adv} *value :* $X_{\text{adv}} = X + \left[\begin{array}{c|c} 0 & 0 \\ \hline 0 & C \end{array}\right]$

Return X_{adv} *value, as attacked inputs*

6.3 Investigations

Width of the corner Fig. 11 shows the accuracy of the model as a function of the width of the perturbation. It should be noted that a very small number of pixels is enough to be considered as a very efficient attack since it is enough to have only one pixel attacked to obtain 18% of accuracy. One can also notice that from about 10 to the total coverage of the image, the width of the attacked corner doesn't impact the accuracy which remains around 15%. As a consequence, if we want to get an attack as offensive as PGD, we can choose a width of 7 for example. *Nota Bene:* we can easily imagine that an attack with a corner of size 32 pixels, but constrained inside an ϵ-infinite ball is exactly the PGD attack.

Number of iterations Since this corner attack is unbounded, the magnitude of the attack (and also the darkness of the pixel) will grow with respect to the number of iterations. Hence, will the attack continue to be more and more efficient with the number of iterations ? Fig. 10 tries to answer this question by showing the accuracy of the Classic CNN on the attacked test set according to the number of iterations $n_{\text{iterations}}$. What can be seen is that the accuracy tends to stagnates around 10% (see paragraph about confusion matrix and accuracy for some explanations). This means that the efficiency of the attack is not due to the size of the perturbation, since the pixels quite quickly (after only 5-7 iterations) settle in an optimal combination to fool the network.

Efficiency of Adv-Training on the Corner Attack One can notice on Fig. 11 that the network made robust against PGD thanks to the Adversarial Training, is not robust at all when it faces images attacked by the corner attack. Indeed, the accuracy is almost as bad as the accuracy of the Classic CNN on the same attacked images. This is understand-able since PGD is a very different attack which deals with a small perturbation on the whole image. In a further study, it

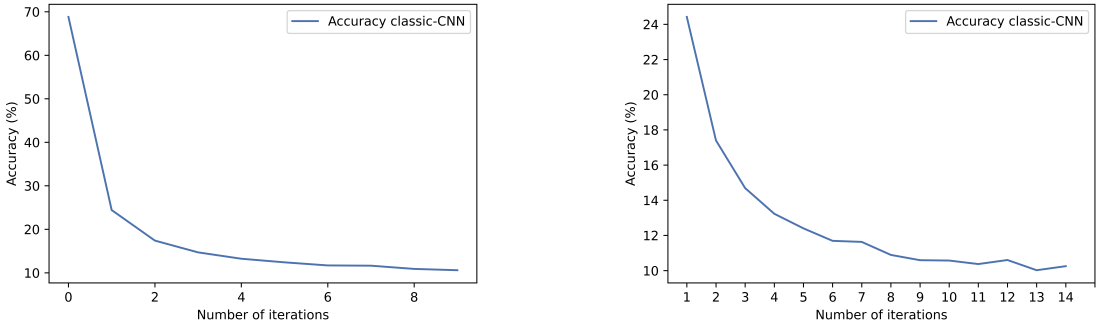


Fig. 10. Accuracy of the classic CNN, on the images attacked by the corner attack, according to the number of iterations. Fixed hyper-parameters are $\eta = 10$, $w_{\text{corner}} = 10$. The difference between left and right is that we just add 0 in the x-axis on the left.

could be interesting to see if it is efficient to use Adversarial training in order to become robust against the corner attack.

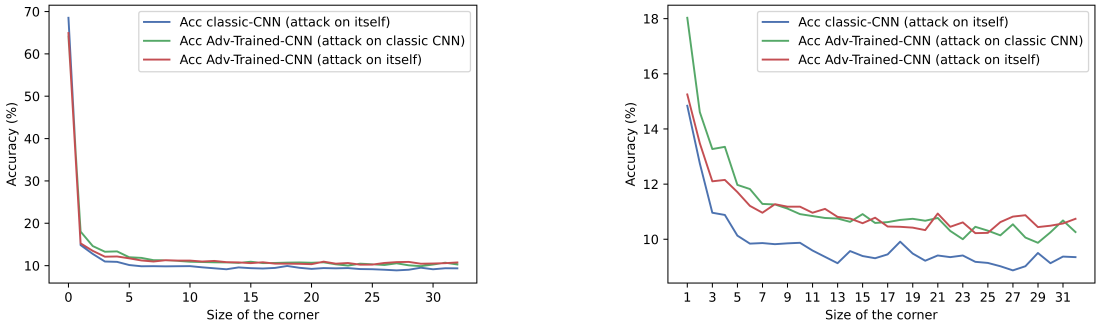


Fig. 11. Accuracy of the {classical CNN, Adv-Trained-CNN (attack on classical CNN), Adv-Trained-CNN (attack on itself)} on the images attacked by the corner attack, according to the corner's width. Fixed hyper-parameters are $\eta = 10$, $n_{\text{iterations}} = 7$. The difference between left and right is that we just add 0 in the x-axis on the left.

Confusion matrix and accuracy for each class A very interesting phenomenon is highlighted by both the confusion matrix of the Classic CNN on the attacked image in Fig. 12, and the Table. 3. We clearly see that almost every attacked test image is classified as a car (94,4%), which brings this strong vertical blue column in the confusion matrix. If we compare to the individual accuracies of the first line of Table. 3, one can note that the Classic CNN was already classifying the cars quite well (72.2%). Nevertheless, this doesn't explain why this label was selected among the others. Investigating on this choice might be the subject of further interesting investigations, unfortunately we didn't have enough time for this. Finally, since all the other accuracies are very close to 0, it means that the average accuracy doesn't have a lot of sense, and it also explains why it is impossible to go below 10% since $(9 \times 0 + 1 \times 100)/10 = 10$.

Table. 3. Accuracy of the classic CNN on raw versus attacked test CIFAR-10 images.

Category	plane	car	bird	cat	deer	dog	frog	horse	ship	truck
Accuracy (raw) [%]	80.0	72.2	53.0	47.4	69.8	56.8	75.9	74.2	80.6	79.3
Accuracy (attacked) [%]	7.1	94.4	1.1	0.2	0.3	1.4	0.6	1.0	7.5	11.3

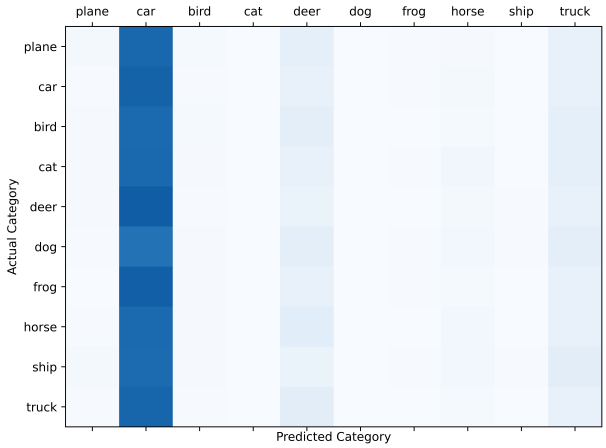


Fig. 12. Confusion matrix of the Classic CNN on the CIFAR-10 test dataset, attacked with the "corner" attack. The accuracy is computed on each of the following labels: {'plane', 'car', 'bird', 'cat','deer', 'dog', 'frog', 'horse', 'ship', 'truck'}.

7 Final comparison of our attacks and conclusion

Finally, we summarized in Table. 4 the results of PGD and the results of our main attacks, only as a reminder since it is not very fair to compare what is not really comparable. Indeed, the constraints on the different attacks are not the same, and it is difficult to find an invariant for these 3 attacks. Again, it has to be noted that the order of magnitude of 10% in the case of the corner attack is just an average between almost 100% on cars and almost 0% on the other labels. We would have liked to investigate more on this last attack.

Table. 4. Accuracies of the classic-CNN and Adv-Trained-CNN against each attack.

attack :	*Acc of classic-CNN [%] against*	*Acc of AdvTrained [%] against*	
		Attack on classic-CNN	*Attack on itself*
PGD	7.73	66.04	0.91
"Carlini"-neuron	12.24	34.21	10.07
Corner	9.66	11.49	10.12

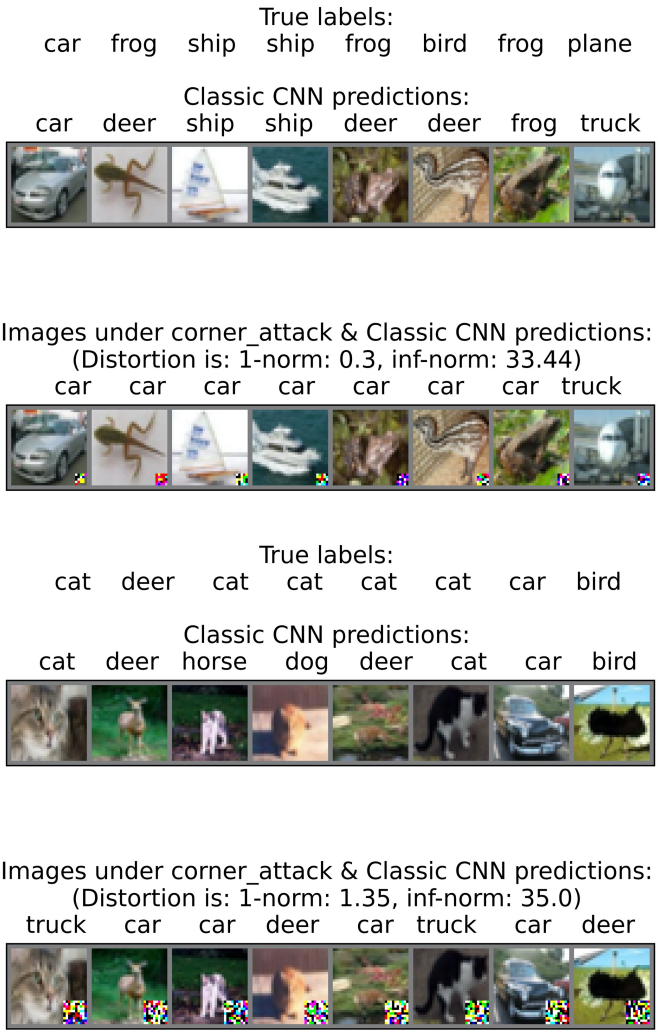


Fig. 13. Raw images and their true labels from CIFAR-10 dataset, predictions from Classic CNN on both these raws images and on their corresponding images attacked by the "corner" attack, according to the corner width: 5 (top) and 10 (bottom).

Bibliography

Carlini, Nicholas, & Wagner, David. 2017. Towards evaluating the robustness of neural networks. *Pages 39–57 of: 2017 ieee symposium on security and privacy (sp)*. IEEE.

Madry, Aleksander, Makelov, Aleksandar, Schmidt, Ludwig, Tsipras, Dimitris, & Vladu, Adrian. 2017. Towards deep learning models resistant to adversarial attacks. *arxiv preprint arxiv:1706.06083*.

Wang, Zhou, & Bovik, Alan C. 2009. Mean squared error: Love it or leave it? a new look at signal fidelity measures. *Ieee signal processing magazine*, **26**(1), 98–117.