One Pixel Attack

Franco Pinto 1592439 Ciprian Floris Dinu 1593921

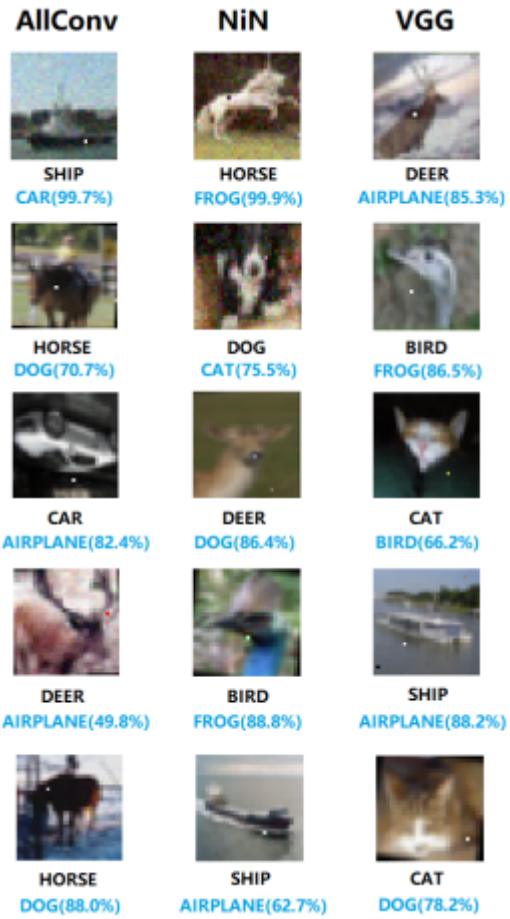
September 20, 2018

Abstract

Deep Neural Networks can be pretty good at identifying images, but they can also be fairly brittle, and a slew of research projects over the last few years have been working on making the networks' image classification less likely to be deliberately fooled.
One particular line of attack involves adding particularly-crafted noise to an image that flips some bits in the deep dark heart of the network, and makes it see something else where no human would notice the difference.

1 Introduction

In this implementation we are considering a very limited scenario in which we can perturb only a pixel of a 32x32 image ; there exists also studies for the 3 or 5 pixels attacks(for further details see the original paper of "One pixel attack for fooling deep neural networks"). Such modification can cause the classifier to label the modified image as a completely different class. This work is related to a semi-black-box attack in the sense that we require only black-box feedback(confidence labels), but no inner information about the neural networks such as gradients and network structures. This topic is strictly connected to cybersecurity aspects, because nowadays image recognition is very important, for example Google's image classification, or self driving cars because they can cause accidents.



2 Methodologies and tools

2.1 Problem Description

Generating adversarial images can be formalized as an optimization problem with constraints. We assume an input image can be represented by a vector in which each scalar element represents one pixel. Let f be the target image classifier which receives n-dimensional inputs, x = (x1, .., xn) be the original natural image correctly classified as class t. The probability of x belonging to the class t is therefore ft(x). The vector e(x) = (e1, .., en) is an additive adversarial perturbation according to x and the limitation of maximum modification L. Note that L is always fixed for one pixel modification. The goal of adversaries in the case of untargeted attacks is to cause a model to misclassify an image. This means that we want to perturb an image as to minimize the confidence probability of the correct classification category and maximize the sum of the probabilities of all other categories.
The problem involves finding two values:
a) which dimensions that need to be perturbed and

b) the corresponding strength of the modification for each dimension.
In our approach, the equation is:

$$max_{\boldsymbol{e(x)^*}} \quad f(x + e(x)) \tag{1}$$

$$\text{subject to } \parallel e(x)_0 \parallel \leq d \tag{2}$$

,where d is a small number. In the case of one-pixel attack d = 1. The one-pixel modification can be seen as perturbing the data point along a direction parallel to the axis of one of the n dimensions. Overall, few-pixel attack conducts perturbations on the lowdimensional slices of input space. In fact, one-pixel perturbation allows the modification of an image towards a chosen direction out of n possible directions with arbitrary strength.

2.2 Differential Evolution

When performing black-box optimizations such as the one pixel attack, it can be very difficult to find an efficient gradient-based optimization that will work for the problem. It would be nice to use an optimization algorithm that can find good solutions without relying on the smoothness of the function. In our case, we have discrete integer positions ranging from 0 to 31 and color intensities from 0 to 255, so the function is expected to be jagged. Differential evolution is a type of evolutionary algorithm where a population of candidate solutions generate offspring which compete with the rest of the population each generation according to their fitness. Each candidate solution is represented by a vector of real numbers which are the inputs to the function we would like to minimize. The lower the output of this function, the better the fitness. The algorithm works by initializing a (usually random) population of vectors, generating new offspring vectors by combining (mutating) individuals in the population, and replacing worse-performing individuals with better candidates.

In the context of the one pixel attack, our input will be a flat vector of pixel values:

$$X = (x_1, y_1, r_1, g_1, b_1, x_2, y_2, r_2, g_2, b_2, ...)$$

These will be encoded as floating-point values, but will be floored back into integers to calculate image perturbations. First we generate a random population of n perturbations

$$\mathbf{P} = (X_1, X_2, \ldots, X_n)$$

Then, on each iteration we calculate n new mutant children using the formula

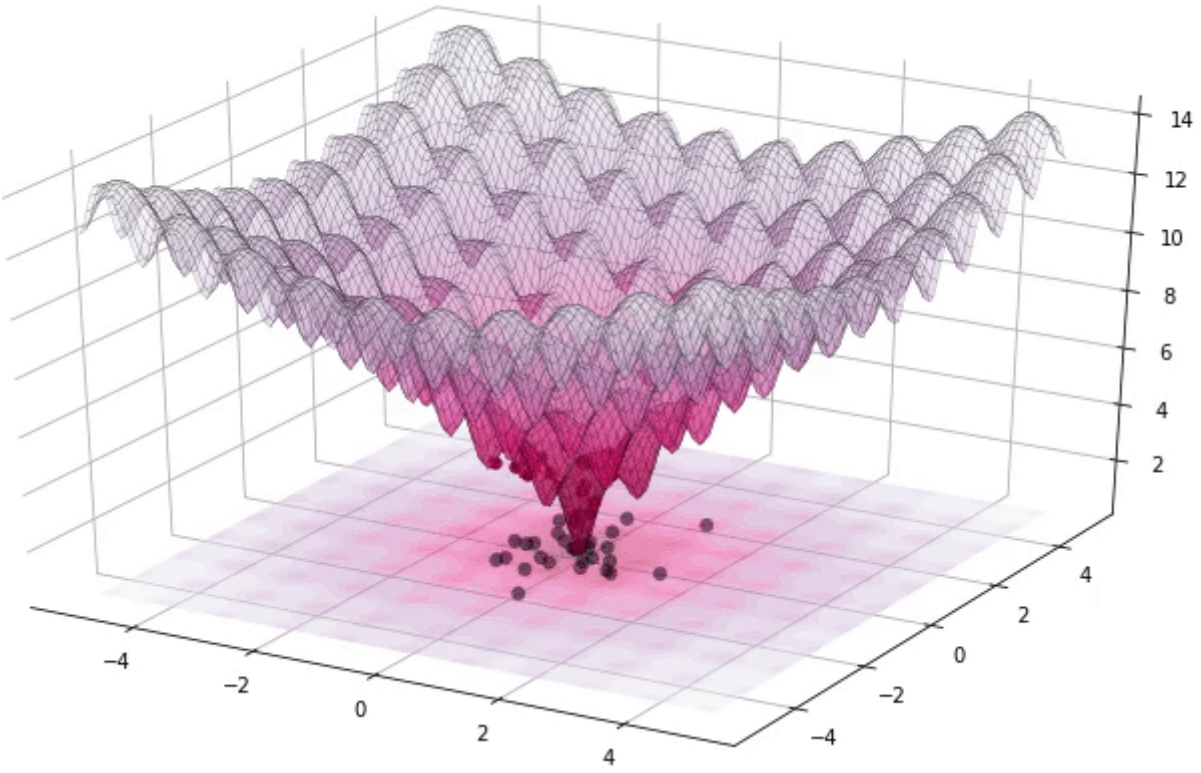
$$X_i = X_{r1} + F(X_{r2} - X_{r3})$$

such that

$$r1 \neq r2 \neq r3$$

where $r1, r2, r3$ are random indices into our population $\mathbf{P}$, and $F = 0.5$ is a mutation parameter. Basically, we pick 3 random individuals from the previous generation and recombine them to make a new candidate solution. If this candidate X_i gives a lower minimum at position i (i.e., the attack is closer to

success), replace the old X_i with this new one. This process repeats for several iterations until our stopping criterion, attack success, which is when we find an image that successfully completes the attack.



2.3 Tools

We have developed the code using Jupyter Notebook in Python. Moreover we have used libraries like scipy, pandas, numpy, matplotlib and Keras. The last one has been used in order to load the dataset which, for our case, is Cifar10 and contains 10000 images divided in 10 categories. We used two pretrained neural networks:

- Lenet, a Convolutional Neural Network(CNN) that is a special kind of Multi-Layer-NN. CNN are designed to recognize visual patterns directly from pixel images with minimal preprocessing, in fact they can recognize with extreme variability and robustness handwritten characters and geometric transformation.
- ResNet, a Residual Neural Network(RNN) that is an artificial neural network (ANN) of a kind that builds on constructs known from pyramidal cells in the cerebral cortex. The intuition on why this works is that the neural network collapses into fewer layers in the initial phase, which makes it easier to learn, and thus gradually expands the layers as it learns more of the feature space. During later learning, when all layers are expanded,

it will stay closer to the manifold and thus learn faster. A neural network without residual parts will explore more of the feature space. This makes it more vulnerable to small perturbations that cause it to leave the manifold altogether, and require extra training data to get back on track.

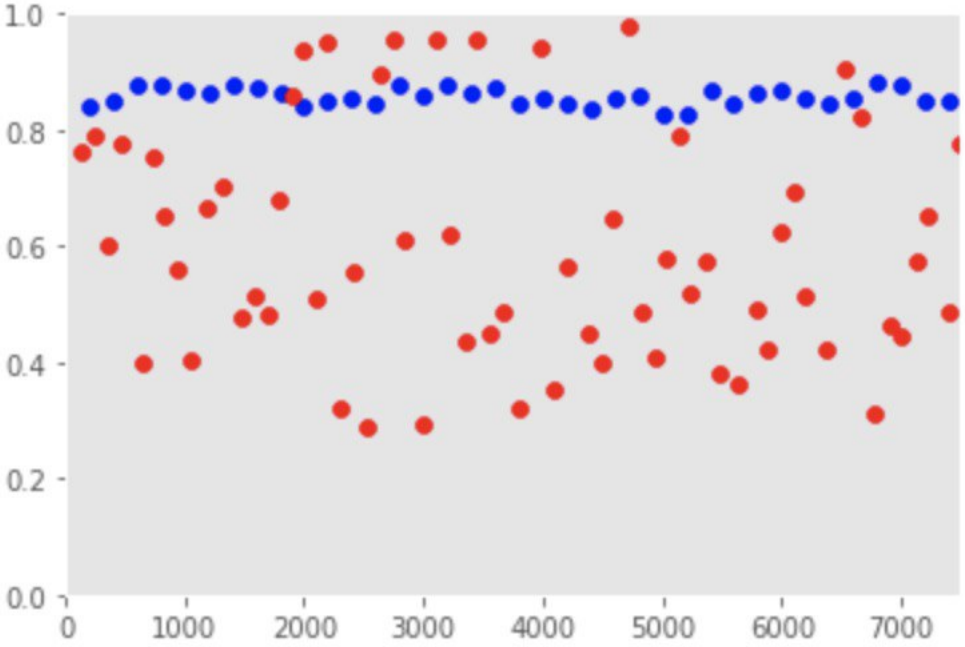
3 Attack

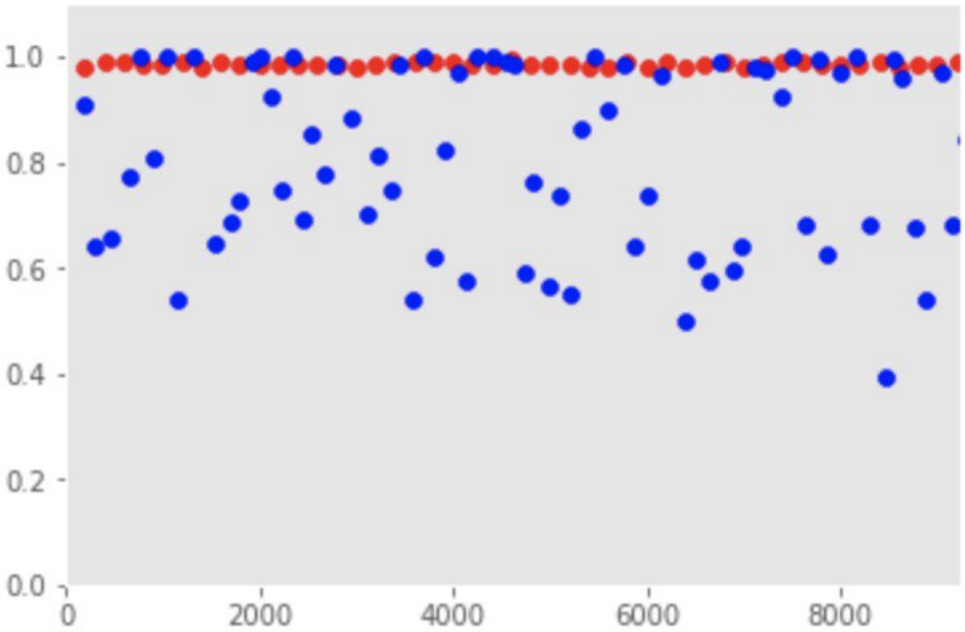
We are considering the Cifar10 model which contains 10 categories which are: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck. We use differential evolution algorithm for finding the best pixel to be perturbed in order for the attack to succeed and we set in the code that the maximum number of iterations is 125.
The early-stop criteria will be triggered when the label of true class is lower than 5% in the case of non targeted attack. Then the label of true class is compares with the highest non-true class to evaluate if the attack succeeded.
The attack that we have devoloped is the untargeted one, which goal is to minimize the confidence of the correct class and to maximize the sum of the wrong classes. Differently from this, the targeted attack consists in making the model to predict as the correct category, the class given in input by the attacker(for futher details see the original paper). For example consider a model that says with 60% of confidence that an image is a frog and the image is indeed a frog, with an untargeted attack we want that the maximum confidence is not the one related to the correct class; instead, for a targeted attack we want that the model returns as a prediction a specific wrong class given by us. In the following figure we observe how the attack works and in particular how the confidence for the ship class decreases and the predicted class, which is different from the correct class, is airplan because it has an higher confidence.

```
Confidence: 0.515298
Confidence: 0.515298
Confidence: 0.515298
Confidence: 0.515298
Confidence: 0.515298
Confidence: 0.515298
Confidence: 0.425688
```

4 Results

Through the evaluate models method in the helper file we find the confidence for all images, so we can distinguish which images were correctly classified, in fact for the Lenet model we find out that 7488 out of 10000 predictions were correct with an average confidence of 86%; instead for the ResNet model we had 9231 correct results with an average confidence of 99%. We can observe the plotted values for correct and wronf predictions, the first one is for the Lenet model and the other is for the ResNet model.





5 Future Works

We think that it could be interesting to see how the models behave with respect to an image that is doubled, because the prediction's confidence with this attack

could increase or decrease and we could compare these results with the already achieved ones. Another challenge may be to contrast the differential evolution based attacks in order to have more resistant models or the other way around to improve the attacks for increasing the success probabilities. In the next picture we show a possibile example of a doubled image that we can use as input to our future work.