Exploiting Vulnerabilities in Neural Networks

One Pixel Attack on Image Classifying NN's

Kalyan Garigapati - 140001011

Sai Teja - 140001012

KeyWords

Adversarial Perturbations	Changes we do on an Image to decrease the confidence of a Neural Network over classification
Differential Evolution	This is an optimization technique we use in this project later on
DNN	Deep Neural Network
Additive Perturbation	Adversarial perturbations which is added on the input
Black Box Attack on NN	Attacking an NN without knowing its Structure or gradient of its cost function.

Introduction

A Deep neural network tries to classify the image by initially extracting the features and then on assigning a class based on the extracted details.

A human can misclassify images if the noise over an image is in a very particular pattern, so that the extracted features are incorrect. A human brain has billions of neurons dedicated for vision purposes, So it is very hard to fool humans. But to fool a neural network we just need very little carefully crafted noise.

For this project we have tried to fool/attack a common Deep Neural Networks by adding very limited perturbations (precisely one pixel) over the input image.

Objective

Given a trained image classifying Neural Network we have to add one pixel of noise in each and every input image of the test data set so that its total prediction accuracy falls down. The total process will be carried out by treating the neural network as a black box.

So, we do not have knowledge of the structure of the neural network, nor its gradient is available. [An attacker can calculate the gradient of the neural network only if its internal structure of each layer is available]

Methodology

Generating adversarial images can be formalized as an optimization problem with constraints. We will see the input image (width = height = n) as a very long vector **X** (of size $n^2 = N$). Let **F** be the target image classification neural network which takes image as input and returns the probability of the image belonging the the target class.

$X = (x_1, x_2, x_3, \ldots, x_N)$

$F_t(X)$: probability of X belonging to the class t

Let $\mathbf{e(X) = (e_1, e_2, \ldots, e_N)}$ be the function which returns the additive perturbation over an image X, so that $F_t(X + e(X))$ is very low.

By attack, what we mean is: We will find **e(X)** for an X, so that

$$F_t(X + e(X)) < F_i(X + e(X))$$

for some $i \in$ Classes(X) and i ~= t

But for us, as the name of the project implies, we can only find e(X) with the constraint

$$e_i \mathrel{!}= 0 \quad : \qquad \text{for some } i = j \in (1,N)$$

$$e_i = 0 \quad : \qquad \forall\ i \in (1,N) \text{ and } i \sim= j$$

This essentially becomes an optimization problem with the

Function to be minimised as :

$$\text{minimize}_{\,e(x)}\, F_t(X + e(x))$$

$$\text{Subject to } ||e(x)||_0 = 1 \quad [\text{ No. of active dimensions of } e(x) = 1]$$

The constraint corresponds to the condition that the attack should be made with only one pixel change.

There is one more type of attack that we won't be dealing in our project. This is called targeted attack. Here we try to find the adversarial perturbations such that the neural network will classify it as the target class desired by the attacker.

The optimization problem for this attack is:

$$\text{maximize}_{\,e(x)}\, F_{targetted\_class}(X + e(x))$$

$$\text{Subject to } ||e(x)||_0 = 1 \quad [\text{ No. of active dimensions of } e(x) = 1]$$

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 of the input vector. In fact, one-pixel perturbation allows the modification of an image towards a chosen direction out of n possible directions with arbitrary strength. The Figure 1 in the next page illustrates the freedom of perturbations where N=3.

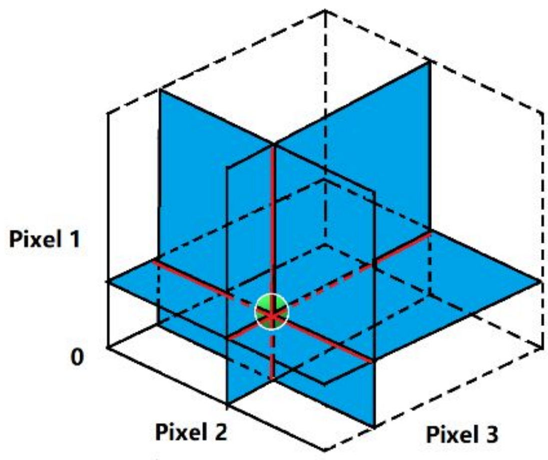


Figure 1

The green point with some pixels values (p1,p2,p3) denotes a natural image. In the case of one-pixel perturbation, the search space is the three perpendicular lines denoted by red and black stripes

Differential Evolution - DE

DE is a population based optimization algorithm which belongs to the general class of evolutionary algorithms. it has mechanisms in the population selection phase that keep the diversity such that in practice it is expected to efficiently find higher quality solutions than gradient-based solutions. During each operation a set of children solutions is generated according to the current father population. The children are compared to their fathers, and the children better than their fathers will survive for the next iteration. Since we are not comparing among children, diversity will be maintained in every iteration, thus leading to better global optimal positions.

In our test images (32x32, i,e, 1024 dimensions), initially we will take 400 randomized uniform candidate solutions. Each candidate solution is a 5 dimensional vector [x,y,R,G,B] containing position of the pixel and RGB values. In the subsequent iterations the next 400 candidate solutions will be produced as follows:

$$x_i (g + 1) = x_{r1} (g) + F (x_{r2} (g) + x_{r3} (g))$$

$$r1 \neq r2 \neq r3$$

x_i is an element of the candidate solution

F is the scale parameter set to be 0.5

g is the current index of generation

In our project we repeat this process for 100 iterations or until the NN's confidence on the actual class decreases to less than 5%.

After the final iteration, The best of the 100 candidates will be proposed as the optimal solution.

Results

We have trained a neural network to classify images from the CIFAR-10 dataset. The CIFAR-10 dataset consists of 60000 32x32 colour images in 10 classes, with 6000 images per class.

The NN is a 5 layer convolutional neural network with the structure as Follows:

[Convolution - Pool 1, Convolution - Pool 2, Dense-1, Dense-2, Dense-3]

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 28, 28, 6)	456
max_pooling2d_1 (MaxPooling2	(None, 14, 14, 6)	0
conv2d_2 (Conv2D)	(None, 10, 10, 16)	2416
max_pooling2d_2 (MaxPooling2	(None, 5, 5, 16)	0
flatten_1 (Flatten)	(None, 400)	0
dense_1 (Dense)	(None, 120)	48120
dense_2 (Dense)	(None, 84)	10164
dense_3 (Dense)	(None, 10)	850

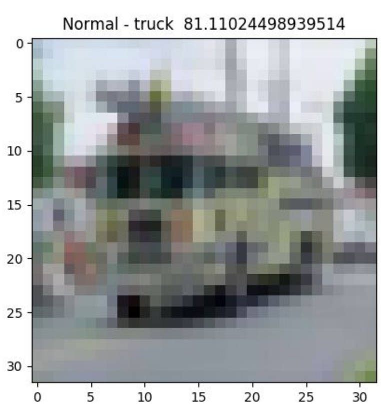
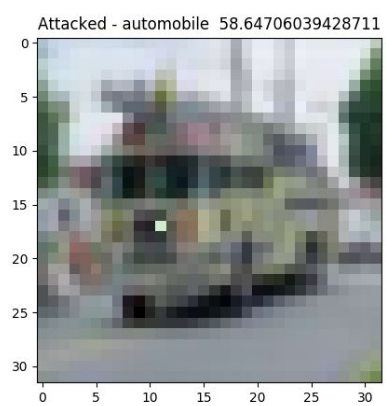
The classification network after training has achieved nearly 80% accuracy.

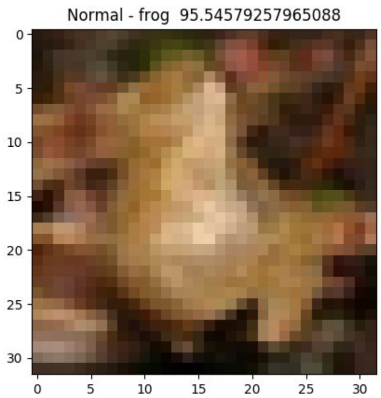
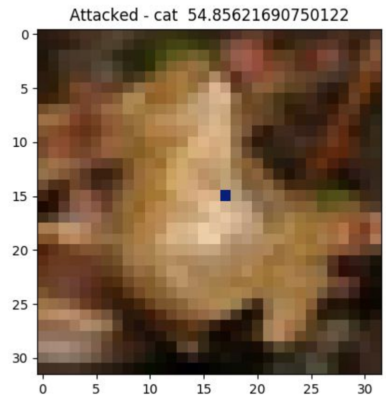
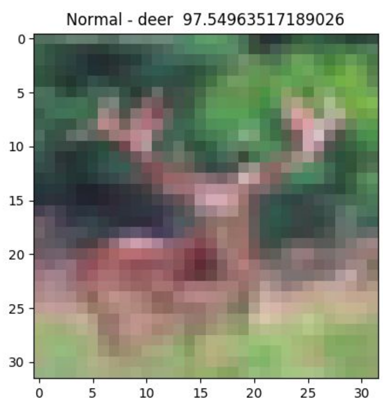
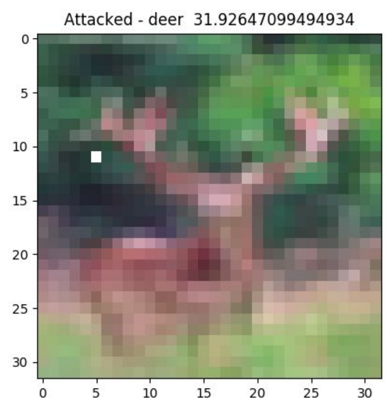
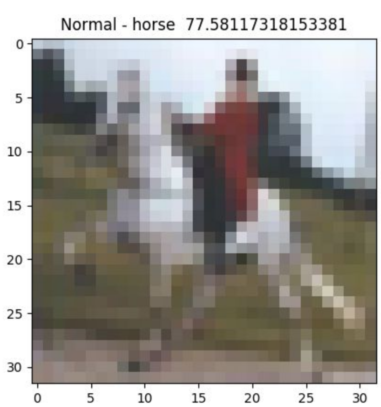
After attempting to attack 100 images in the dataset, 71 attacks have been successful.

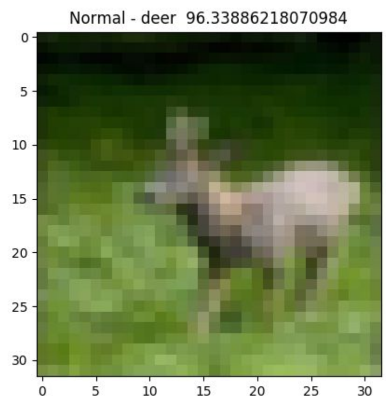
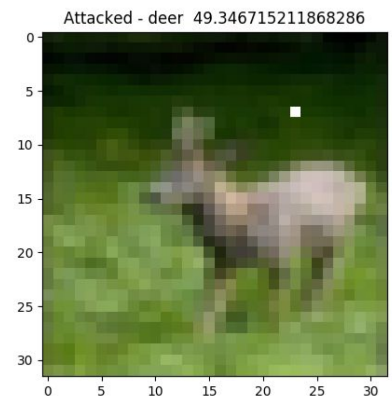
Some sample attacks on the images are as follows:

	perturbation	ImageNo.	Actual	After attack	Attack success
0	1	5067	truck	automobile	True
1	1	1575	bird	frog	True
2	1	6218	truck	bird	True
3	1	7874	frog	frog	False
4	1	4470	airplane	truck	True
5	1	2392	dog	dog	False
6	1	9423	bird	horse	True
7	1	5599	bird	deer	True
8	1	6755	truck	horse	True
9	1	7301	frog	frog	False
10	1	6018	dog	deer	True
11	1	5280	airplane	ship	True
12	1	9184	horse	dog	True
13	1	6278	dog	frog	True
14	1	1697	truck	truck	False

Images:

Without Perturbations	Prediction on the perturbed Image
	
Truck- 81%	Automobile- 58%

	
Frog- 95%	Cat -54%
	
Deer - 97%	Deer - 31%
	
Horse - 77%	Dog - 85%

	
Deer - 96%	Deer- 49%

Conclusions:

Small sized neural networks are not robust to well crafted human-imperceptible noises. A single pixel change can hugely alter the confidence of these networks.