Adversarial Reprogramming of Neural Networks
CS663 Project Report

Parth Shettiwar 170070021
Yash Jain 170050055
Devansh Garg 170050029

November 2019

1 Introduction

The aim is to implement an adversarial attack on a neural network so that it produces different outputs which might be favourable to the attacker. Unlike other attacks where the aim is to degrade the performance of the model, here we will reprogram the model to perform a specific task. The main difference between this method and Transfer learning is that the weights are changed(trained) in the latter. Here we will just modify the input to the neural network model (trained on a particular task) with weights unchanged, such that it can be used to perform a different task. For example, Inception V3 model trained on ImageNet dataset, can be reprogrammed to classify MNIST digits or count the number of squares in an image. In practice, there is no constraint that adversarial attacks should adhere to some framework. Thus, it is crucial to proactively anticipate other unexplored adversarial goals in order to make machine learning systems more secure

2 Methods

The overall task is to reprogram the model to perform a task chosen by the attacker, without the attacker needing to compute the specific desired output Suppose the model takes inputs x and produces output $f(x)$ originally. We would like to perform adversarial task for inputs $\tilde{x}$ and producing $g(\tilde{x})$ as outputs. To achieve the reprogramming,we would learn the adversarial reprogramming functions h_f and h_g such that h_f transform the input $\tilde{x}$ to the domain of x which can be fed to $f(x)$. h_g would convert the outputs back to the domain of $g(\tilde{x})$. Effectively

$$h_g(f(h_f(\tilde{x}))) = g(\tilde{x}).$$

where

- $\tilde{x}$ is a small image
- g is a function that processes small images
- h_f is a function such that puts it the small image $\tilde{x}$ at the centre of some larger image x which is to be learned
- f is a function that processes these larger images(in our case it is the Resnet50 model) to classify them as one of ImageNet labels
- h_g is just the hard coded mapping between the produced labels and the adversarial labels

The following example (Figure 1) illustrates this:
The adversarial task is to count squares in an image. For that we must map the ImageNet labels, say first 10, to adversarial task labels. Then we take the adversarial image and put that in a centre of a larger image which has its corresponding centre portion removed out. The larger image is to be learned here. We then pass this new image to the ImageNet classifier(say Resnet or Inception model), which will classify it as one of ImageNet labels. Finally the hardcoded mapping function h_g will classify it as one of the adversarial labels.
We have implemented this with MNIST classification as the adverserial task.

Note: This method is different from Transfer learning where we use weights trained on one dataset and train the top layers to use it for classification of some other dataset. Here we don't change the weights of the Resnet model, rather we are modifying the input to the model.

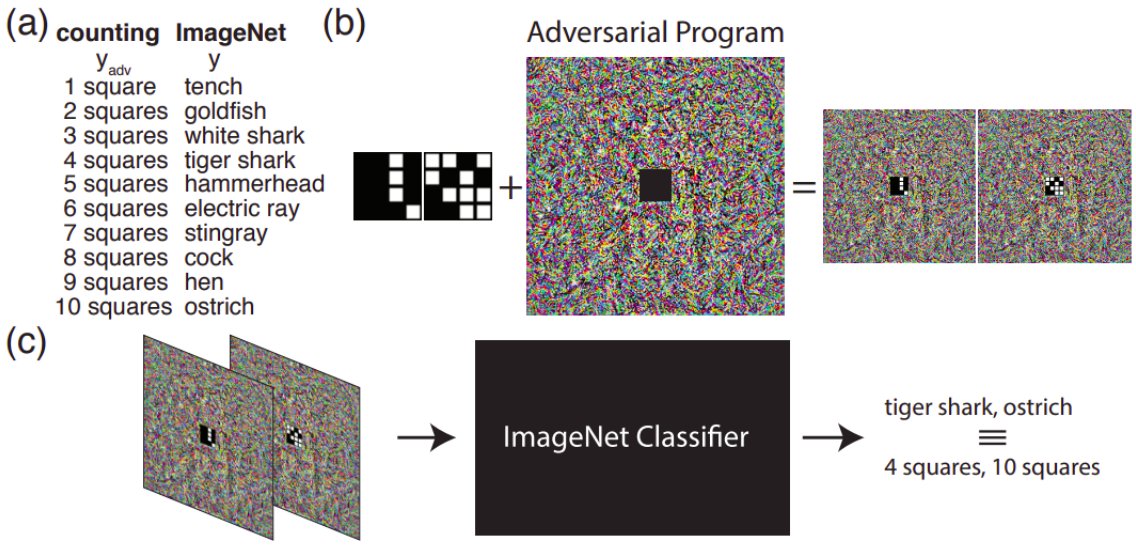

Figure 1: Reprogramming the Neural Network

2.1 Implementation Detail

The adversarial program - say W is to be learned of size $R^{n\times n\times 3}$ where n is the ImageNet image width This program will be same for all images and not specific to a single image. Then we apply a mask so that we can accommodate the adversarial data.

$$P = tanh(W \odot M)$$

The mask M is such that it is all 1 except the central portion where it is 0.Also we use tanh function as it keeps the output between (-1,1), which is required for the ImageNet as input.Also we keep the adversarial data at centre but some other scheme is also possible.Intuitively we should keep it symmetric, hence its kept at centre.Next we add the adversarial image($\bar{X}$), on which we apply the adversarial task, to the computed quantity P.

$$X_{adv} = \bar{X} + P$$

The image $\bar{X}$ will go in the Central portion of P where it is masked off.
Let $P(y|X)$ be the probability that an ImageNet classifier gives to ImageNet label y ϵ 1, . . . , 1000, given an input image X. The adversarial goal is thus to maximize the probability $P(hg(y_{adv})|X_{adv})$.The optimization problem is

$$\hat{W} = \arg\min_{W}(-logP(hg(y_{adv})|X_{adv}) + \lambda||W||_F^2$$

where λ is a regulariser to avoid overfitting and function h_g is a hardcoded mapping between ImageNet labels and adversarial task labels.The cost of this computation is minimal and attacker needs only to store the program and add it to the data, leaving the majority of computation to the target network

2.2 Hyperparameters

- Learning rate = 0.05 with Adam optimizer
- Decay = 0.96
- Steps per epoch = 2
- λ = 1e-8
- epochs = 10

2.3 Observation and Results

Our implementation is on MNIST digits database(acting as adversarial database) acting on a ImageNet classifier. We took 60,000 MNIST training images and 10,000 test images.Further, We have used Resnet 50 v2 as the ImageNet classifier. The first 10 ImageNet labels were assigned to the MNIST digits. The adversarial program looks like this :

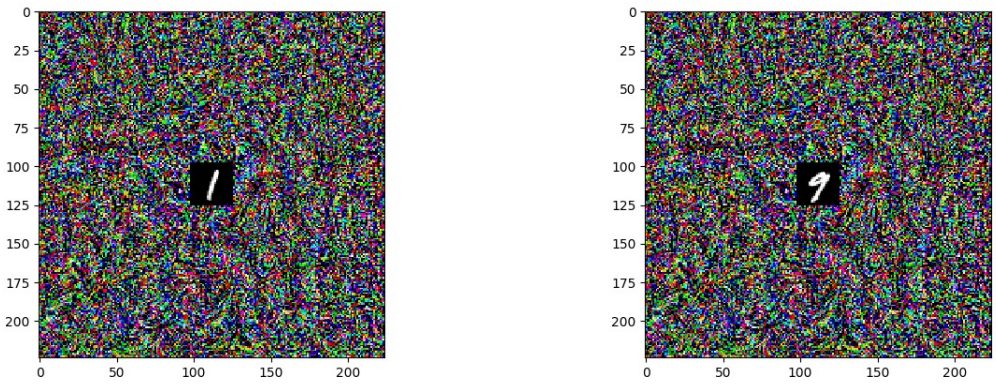

Figure 2: Adversarial programme to classify MNIST digits using ImageNet classifier. Left:MNIST digit 1, Right:MNIST digit 9 embedded in a adverserial larger image

Here the image is of size 224 x 224 x 3 of which 28 x 28 x 3 centre block is occupied by MNIST image.
For digit 1, we accurately classified it, however digit 9 was classfied as 7.
About **79.75**% accuracy with L2-cross entropy loss of **0.78** was achieved on MNIST Database

2.4 Conclusion and Future Use

We saw adversarial reprogramming on classification tasks in the image domain.It was seen that trained networks can be reprogrammed to classify MNIST examples, which do not have any resemblance to images.
A possible future use could be in RNN (Recurrent Neural Networks), where if find suitable inputs to RNN network, it could be able to perform number of simple operations like increment counter, decrement counter etc.

2.5 References

Adversarial Reprogramming of Neural Networks - Ian Goodfellow, Gamaleldin F.Elsayed - ICLR 2019