CIS 680 HW 1

Venkata Sai Nikhil, Thodupunuri

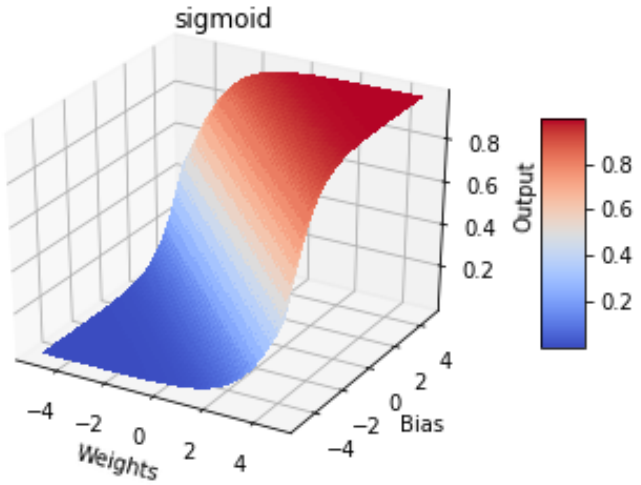
62716136

collaborators: Disha Jindal

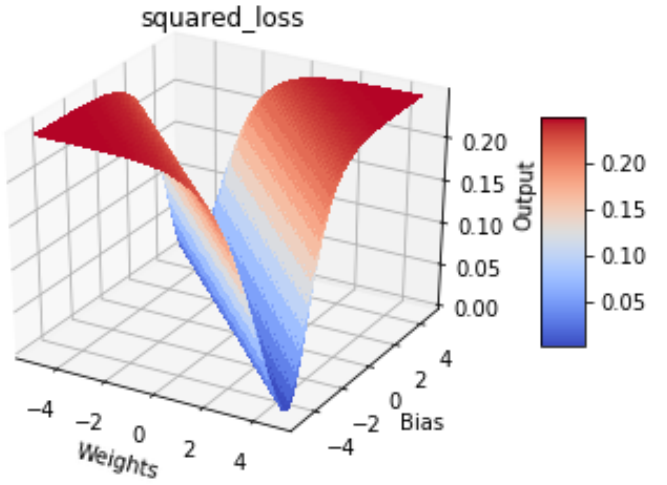
1 Problem 1

Note: For all the plots, Weights and Bias are in the range [-5, 5] with an interval of 0.25.

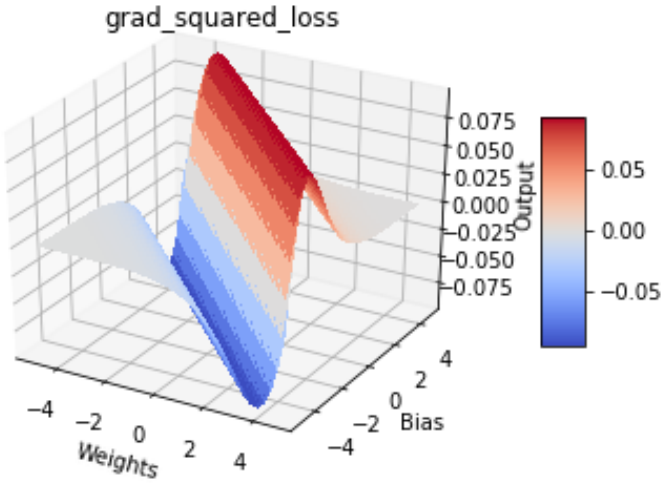
1.1 Sigmoid Function



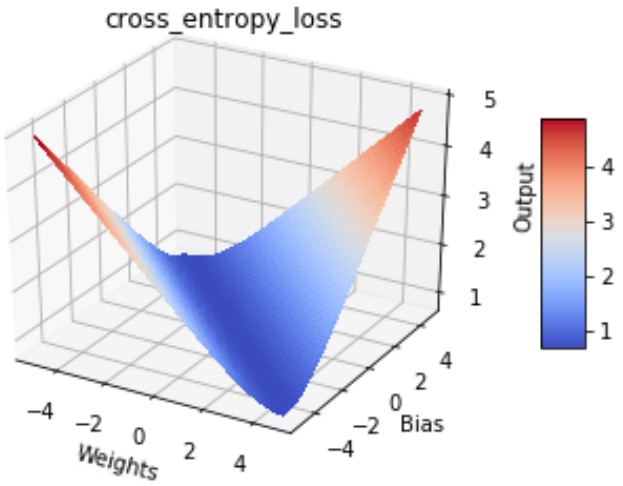
1.2 L2 Loss



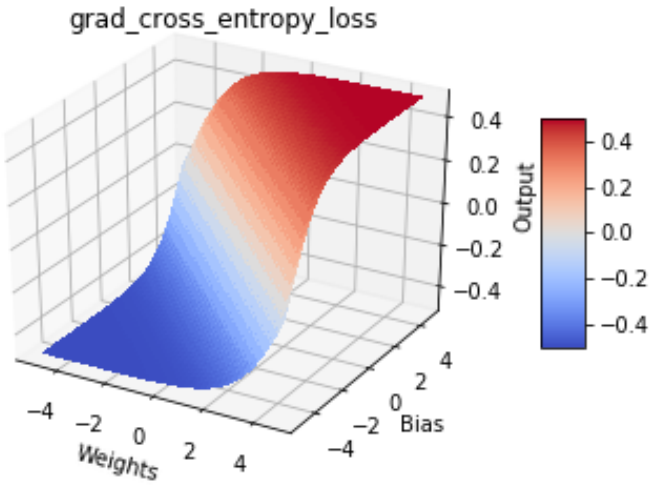
1.3 Gradient of L2 Loss



1.4 Cross-entropy Loss



1.5 Gradient of Cross-entropy Loss



1.6 Observations

L2 Loss function :

$$L2 = (y' - y)^2$$

Here y' is the prediction and y is the actual label from the data set.

Cross-Entropy Loss Function :

$$CEL = -(\ y * log(y')\ +\ (1 - y) * log(1 - y'))$$

Here y' is the prediction and y is the actual label from the data set.

- The loss value for Cross entropy is higher than L2 for same inputs. This means Cross entropy loss punishes mis-classified samples more than L2 loss.

Explanation:
y', y are in the range [0,1]. $|y' - y|$ will also be in range [0,1]. In L2 loss, $(y' - y)^2$ will reduce the loss value. But in cross entropy, $log(y')$ and $log(1 - y')$ will be in range $[-\infty, 0]$. y' and (1-y) are in the range $[0, 1]$. We have negative sign as well, So each term in cross entropy loss is a product of two values, where one is range [0,1] and other in $[0, \infty]$.

L2 loss gradient with sigmoid as activation:

$$L2\_grad = 2 * (S(wx + b) - y) * S(wx + b) * (1 - S(wx + b)) * x$$

Note: The above gradients are with respect to weights.
Cross Entropy loss gradient with sigmoid as activation:

$$CEL\_grad = -1*(y/S(wx+b))-(1-y)/(1-S(wx+b))*(1-S(wx+b))*S(wx+b)*x$$

$$= -1 * (y - S(wx + b)) * x$$

Here S(wx+b) is the sigmoid function, weight w, bias b and input x
The magnitude for CEL gradient is higher than L2 gradient. This will help in faster learning. L2 gradient plot is flat at the ends. As the gradient in these regions are very low, it takes lost of updates to reach the minima.

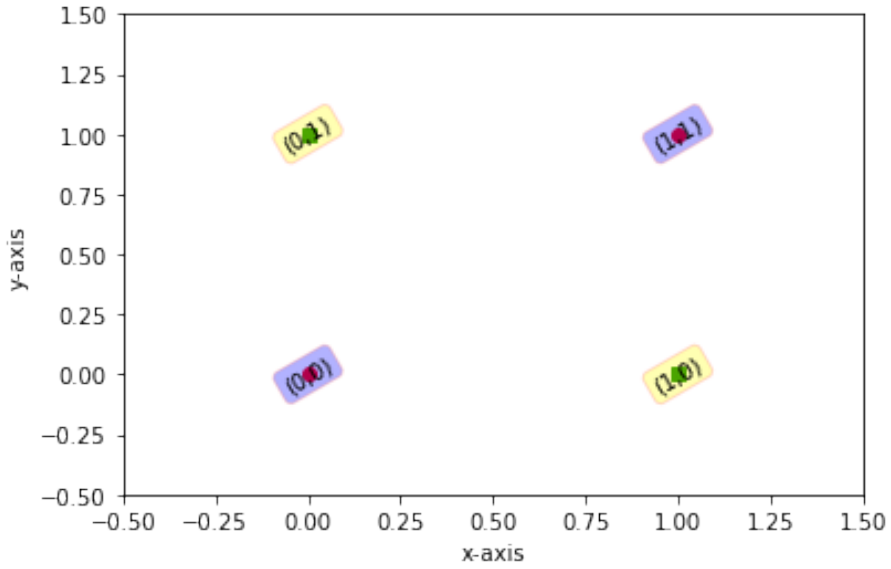
2 Problem 2

2.1 XOR Formulation

Truth Table:

X_1	X_2	$Y = X1 \oplus X2$
0	0	0
0	1	1
1	0	1
1	1	0

2D plot:



From the above plot, we can see that XOR problem doesn't have linearly separable solution.

To find a linearly separable plane for the XOR problem, The inputs have to be transformed into a new space, where there is a linearly separable solution.

We model the XOR by a neural network, with hidden layers. First component will act like a transformation component, the final layer with sigmoid will act as a linear classifier. Note: The transformation component has to be nonlinear.

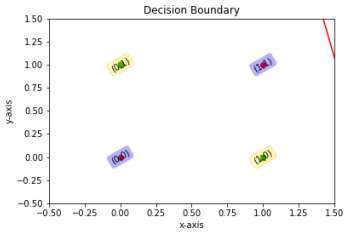
Training this neural network implies finding a non-linear mapping from (x_1, x_2) to (h_1, h_2), and also a linear classifier. The nonlinear component, hyperbolic tangent function provides the required nonlinear feature transformation. [Other components of the network expect the final sigmoid layer are just linear transformations].

Modeling:
Feature transformation component: A fully connected layer, followed by a nonlinear activation function will act as a non-linear transformation.

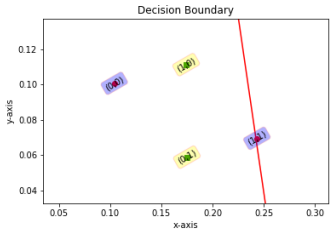
Linear classifier component: A fully connected layer followed by a sigmoid activation (one output node) will act like a linear classifier. Fully connected layer is nothing but the hyper plane separating the samples. If the sigmoid output is > 0.5, this implies, the sample is on the positive side of the line. If it is < 0.5, then sample is on the negative side of the line.

Optimization setting: Finding the weights of the "Feature transformation component" and "Linear classifier component" is the required goal. Treating the xor truth table as 4 inputs, The goal is to find the decision boundary which minimizes the loss function. We use cross-entropy loss as the loss function and gradient descent to find the weights.

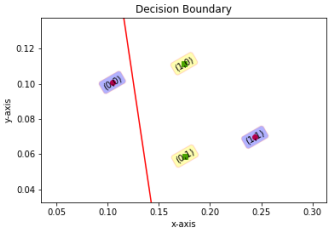
2.2 Decision boundaries over the epochs



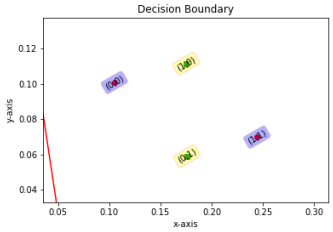
Epoch 0



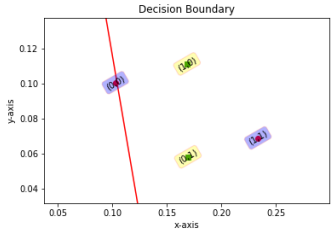
Epoch 11



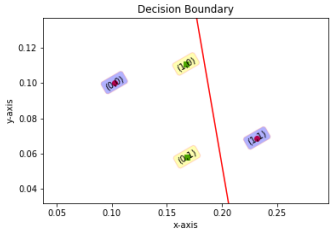
Epoch 12



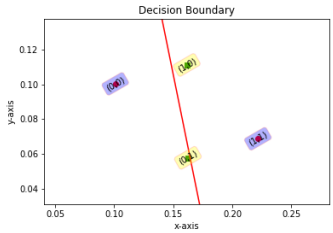
Epoch 13



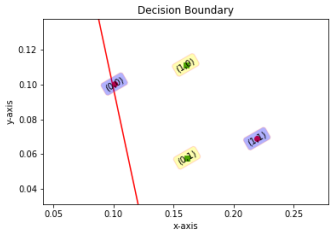
Epoch 30



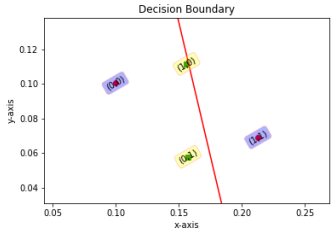
Epoch 32



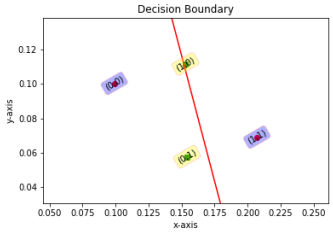
Epoch 52



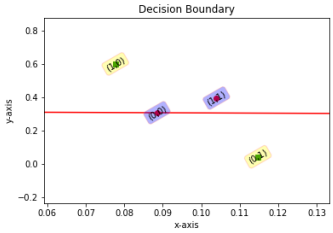
Epoch 58



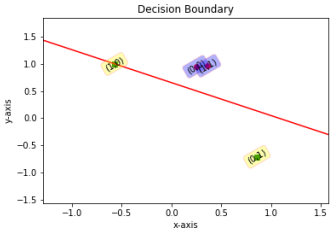
Epoch 73



Epoch 89



(a) Epoch 1762



(b) Epoch 1899

Note: The last image, (Epoch 1899) is the solution.

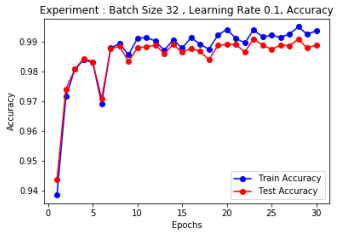
2.3

- Without non-linearity, Neural network will not find a solution.
- Without a non-linear activation, The neural net is nothing but a stack of linear transformation units, which will acts a linear classifier (The two linear transformation units can be combined into single one).
- We know that there is no linear separable solution, Hence the network cannot classify perfectly.

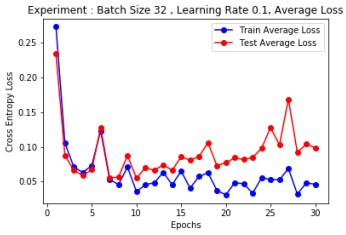
3 Problem 3

Network Performance with different hyper-parameters:

Batch Size: 32, LR = 0.1

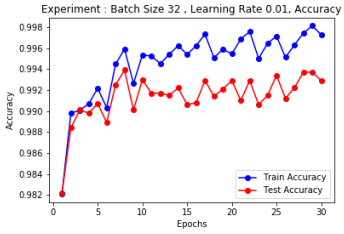


Accuracy

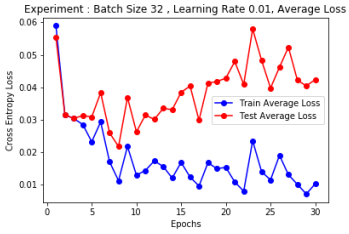


Loss

Batch Size: 32, LR = 0.01

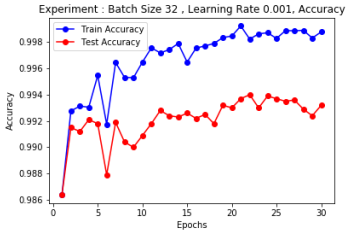


Accuracy



Loss

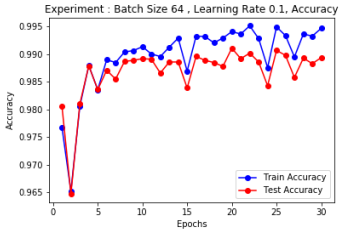
Batch Size: 32, LR = 0.001



Accuracy

Loss

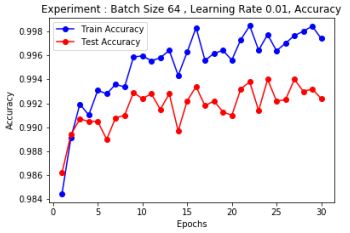
Batch Size: 64, LR = 0.1



Accuracy

Loss

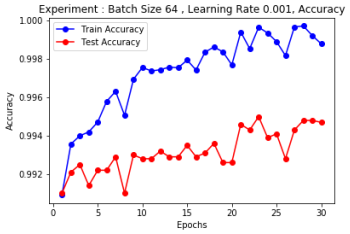
Batch Size: 64, LR = 0.01



Accuracy

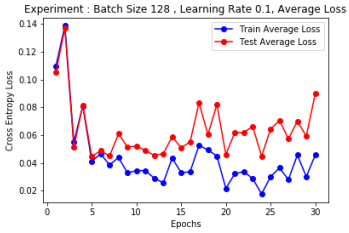
Loss

Batch Size: 64, LR = 0.001



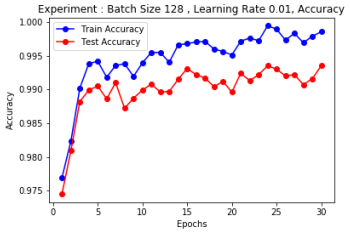
Accuracy Loss

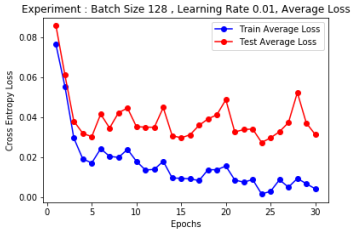
Batch Size: 128, LR = 0.1



Accuracy Loss

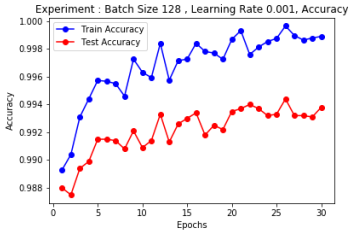
Batch Size: 128, LR = 0.01



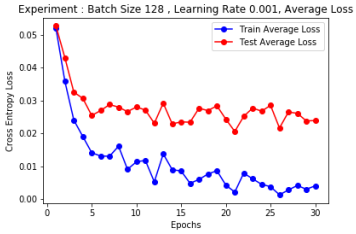


Accuracy Loss

Batch Size: 128, LR = 0.001

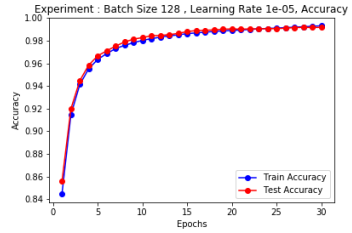


Accuracy

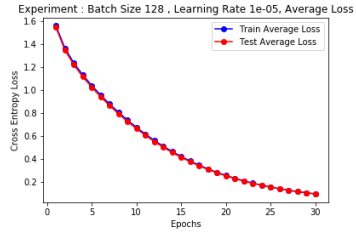


Loss

Batch Size: 128, LR = 0.00001 (Very low learning rate example)



Accuracy



Loss

Observations :

- With longer epochs (Not early termination), The network tends to over-fit. After some epochs, The testing accuracy is not decreasing(it's stagnant).

- Increasing batch size displays more stable learning. (For same learning late, Batch size 32, 64 has fluctuations. They have rapid learning but subjected to volatility).

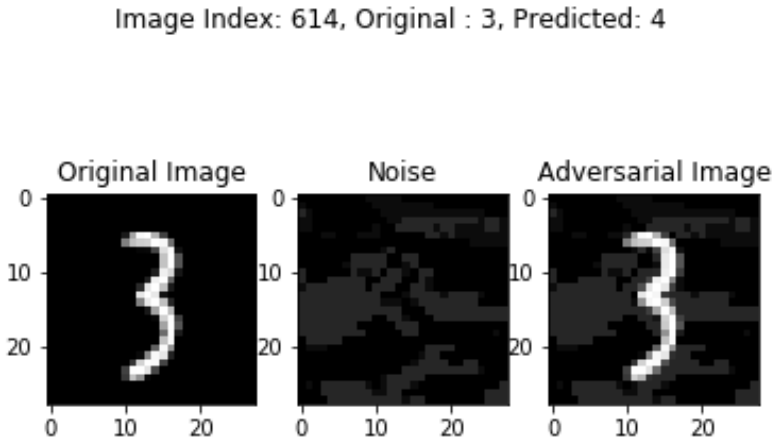
- For low learning rate, The loss plots are smooth, but takes more to find minima. For higher learning rate, There is a rapid decrease initially, but getting saturated eventually.

4 Problem 4

Note: Pixel values for Original and Adversarial images are in range [-1, 1], Noise is in range [0, 2].

4.1

Adversarial images generated:



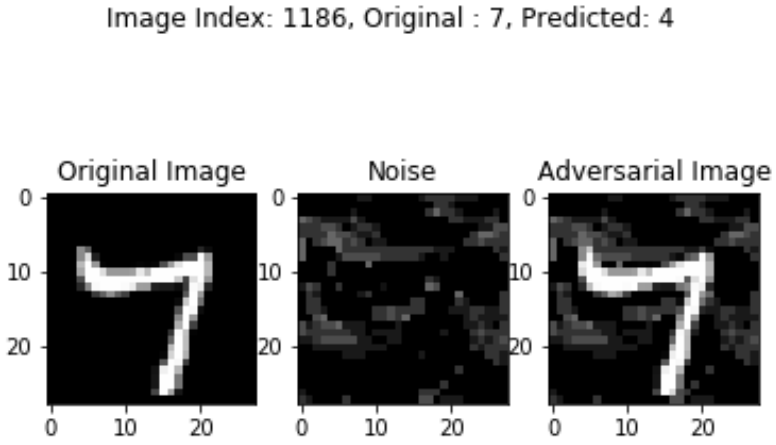


Image Index: 1486, Original : 9, Predicted: 4

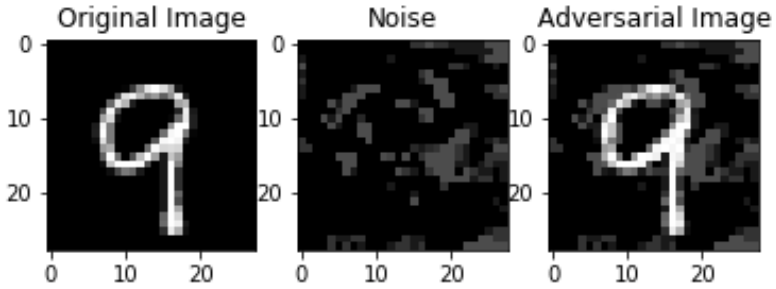
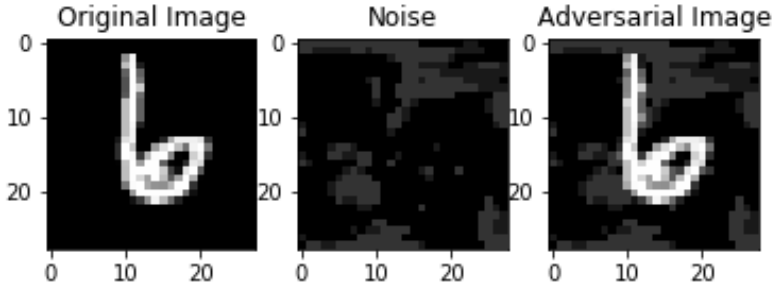
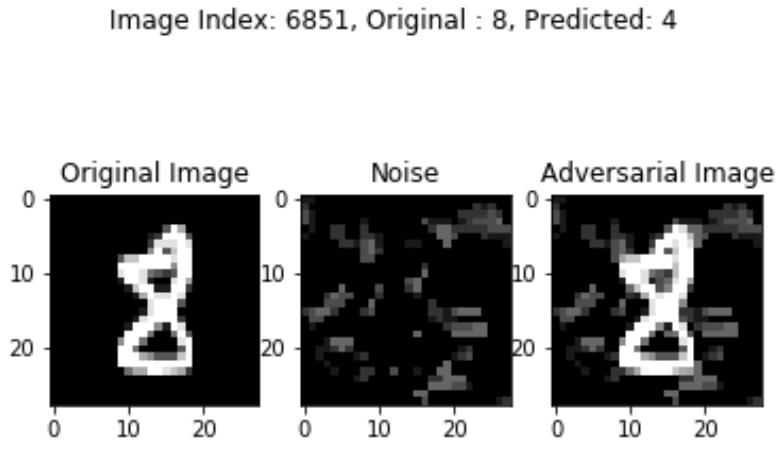


Image Index: 2471, Original : 6, Predicted: 5





4.2

Target label for this task is generated randomly.

Update equation :

$$I_\epsilon = I_\epsilon - sign(\nabla_{I_\epsilon} L(I_\epsilon, y_{target})),$$

Adversarial images generated:

Image Index: 5, Original : 1, Target: 8

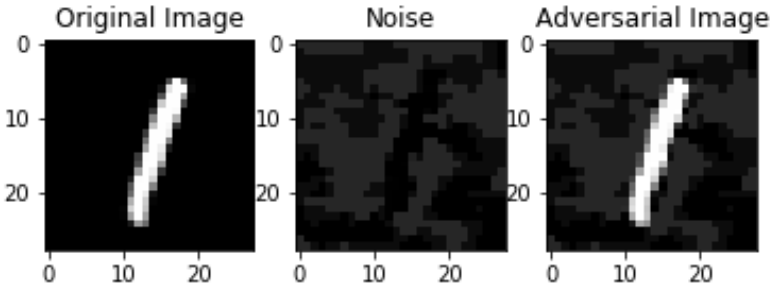


Image Index: 114, Original : 7, Target: 9

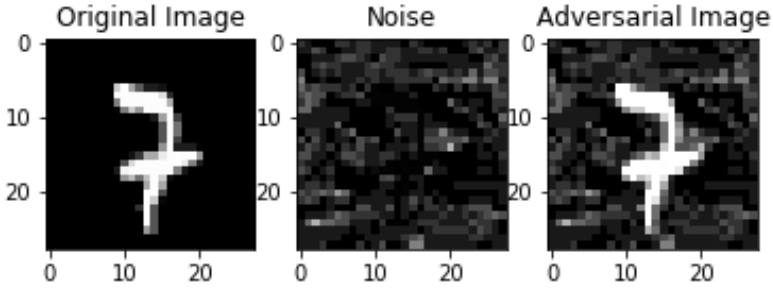


Image Index: 5275, Original : 5, Target: 9

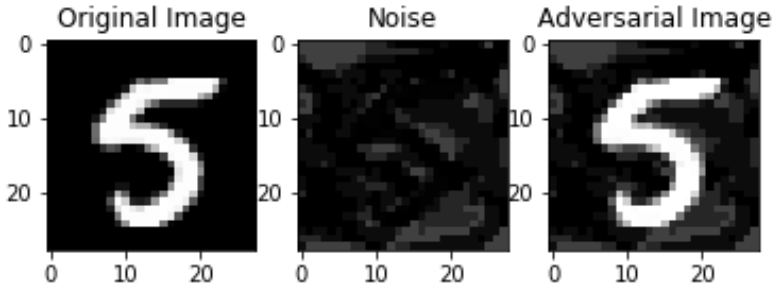
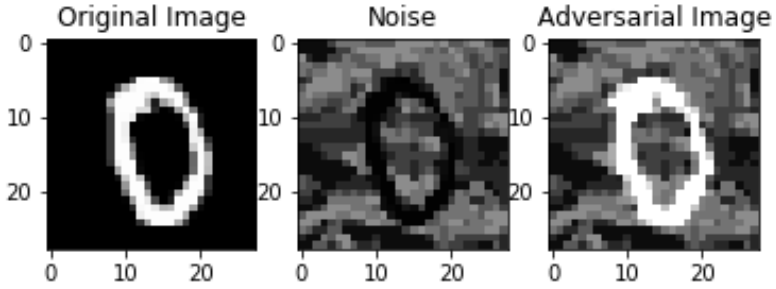
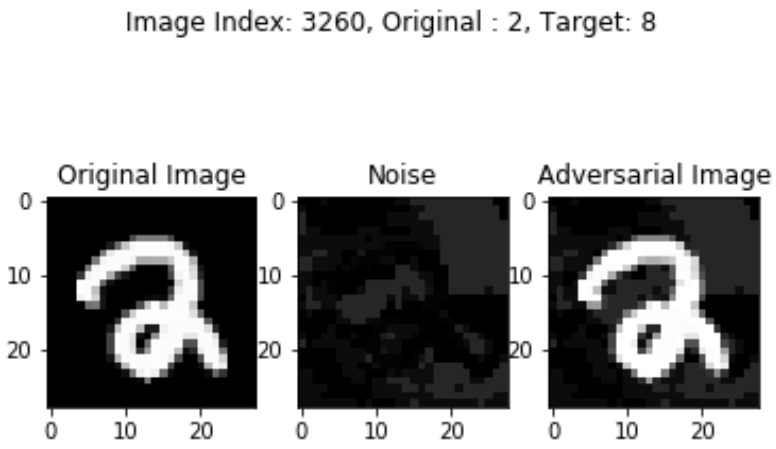


Image Index: 1762, Original : 0, Target: 8





4.3 Observations

- Naming Convention: Lets say "model 1" be the model used to generate Adversarial images. "model 2" be the model which is re-trained separately on MNIST.

- Testing the above generated Adversarial images on model 2:

  - Using Samples from 4.1,
    - Total adversarial images used: 7300
    - Accuracy: 0.81322427.
  - Using Samples from 4.2,
    - Total adversarial images used: 5700
    - Accuracy: 0.64104336.

- We can see that the generated adversarial images are not universal. Though these are misclassified in model 1 (Accuracy: 0), model 2 classifies many of them correctly.

- Approaches mentioned (4.1, 4.2) is not a universal approach. They are very specific to the trained network.

- This means neural network solution is not a generalized solution. What they do is just minimizing loss.
- Formulating this classification problem as Minimizing loss is not resulting in a perfect model. Theoretically this is solved with infinite training samples. This can be also viewed as over fitting.
- This exposes the vulnerability of neural networks. They are very sensitive.
- Some ways to overcome this vulnerability:
  - Retrain the network with generated adversarial images. This acts as a regularization.
  - Have a component in loss function which formulates adversarial loss. Minimizing this compound loss function will make network robust(not completely) to adversarial images.