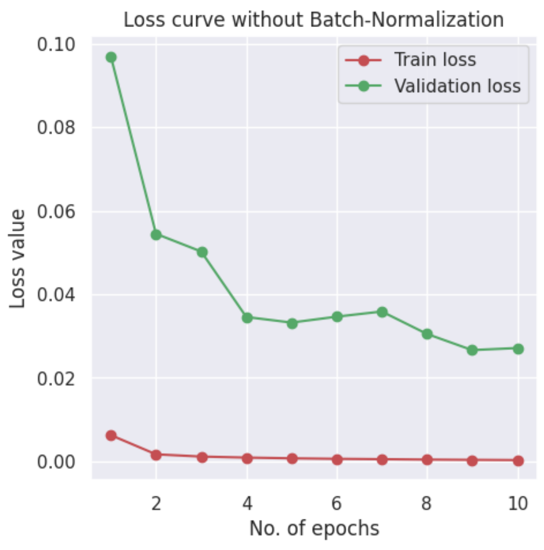
Programming assignment 2: MNIST classification using CNN

Arunima Sarkar

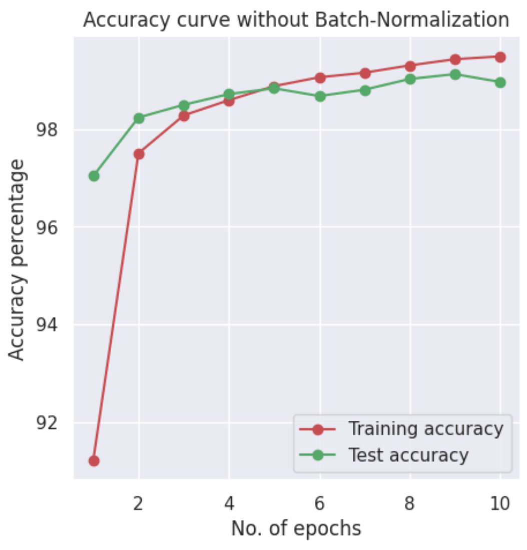
1 Question 1

1. The average prediction accuracy is 98.59% over the whole test set.

The plots for loss and accuracy is as follows:



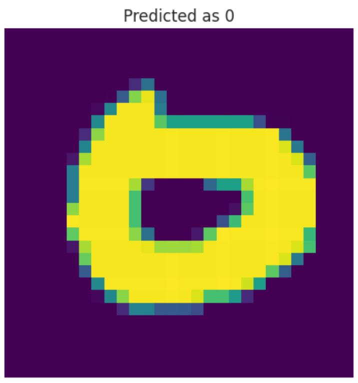
((a)) Loss curve



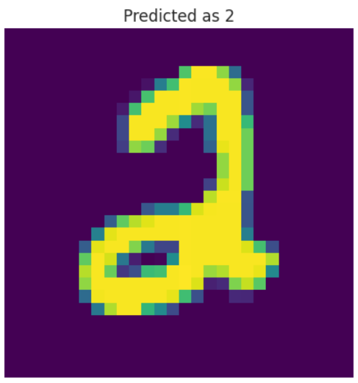
((b)) Accuracy curve

Figure 1: Plots for loss and accuracy

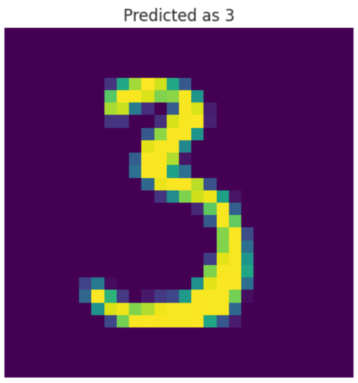
2. The random plots for some images are :



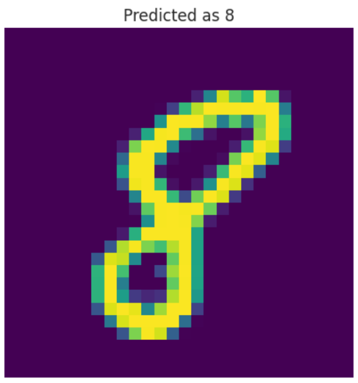
((a)) True label=0



((b)) True label=2



((c)) True label=3



((d)) True label=8

Figure 2: Images after random selection

3. The input and output dimensions at each layer:

   1. First convolutional layer
      - input size : $28 * 28 * 1$
      - output size : $28 * 28 * 32$
   2. Maxpool layer
      - input size : $28 * 28$
      - output size : $14 * 14$
   3. Second convolutional layer

- input size : $32 * 14 * 14$
- output size : $14 * 14$ for each filter

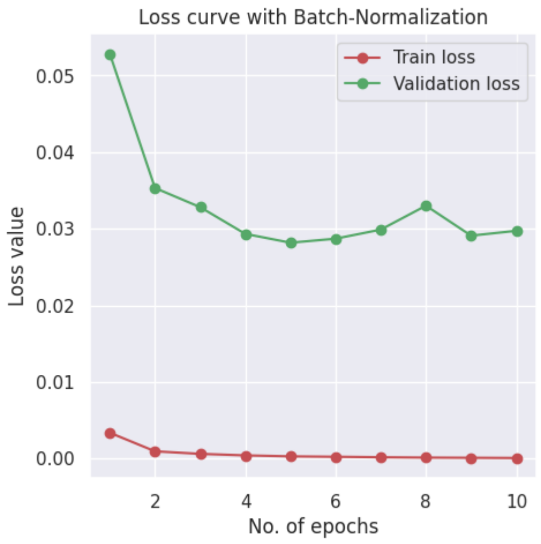
4. Maxpool layer
   - input size : $14 * 14$
   - output size : $7 * 7$
5. First fully connected layer
   - input size : $32 * 7 * 7$
   - output size : 500
6. Second fully connected layer
   - input size : 500
   - output size : 10

4. Number of parameters:
   1. First convolutional layer: weights=$3 * 3 * 32$=288, bias=32, total parameters=320
   2. Second convolutional layer: weights=$32 * 3 * 3 * 32$=9216,bias=32, total parameters=9248
   3. First fully connected layer: weights=$7 * 7 * 32 * 500$=784000,bias=500, total parameters=784500
   4. Second fully connected layer: weights=$500 * 10$=5000,bias=10, total parameters=5010

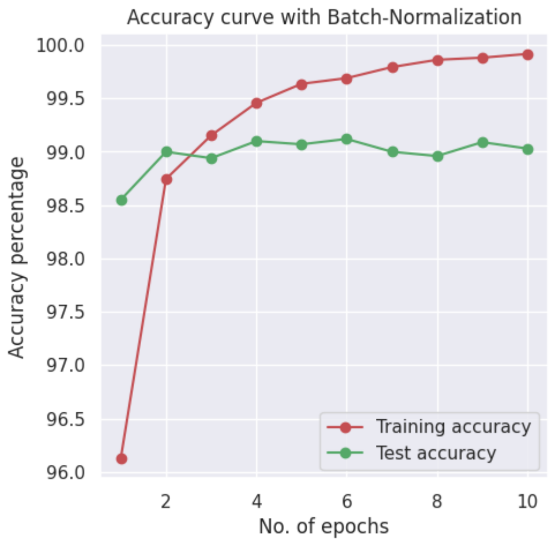
Thus total number of parameters are 799078. Fully connected layer has 789510 parameters whereas convolutional layers have 9568 parameters.

5. The network has 9564 neurons with 9054 neurons in convolutional layer and 510 in fully-connected layer.

6. Batch-Normalization



((a)) Loss curve



((b)) Accuracy curve

Figure 3: Plots for loss and accuracy

The validation loss doesn't improve but the overall test accuracy improves a bit. Without batch normalization the accuracy was 98.52% but with batch normalization the accuracy is 98.98%. However the training time increases from 59 seconds to 65 seconds.

2 Question 2

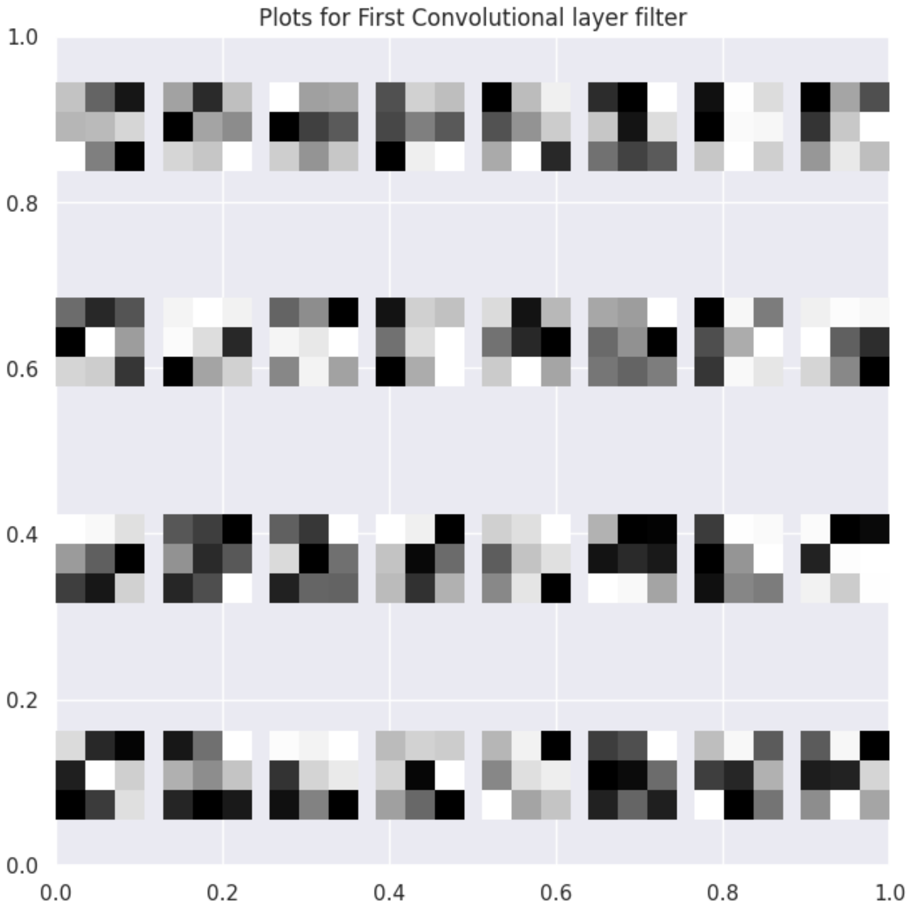
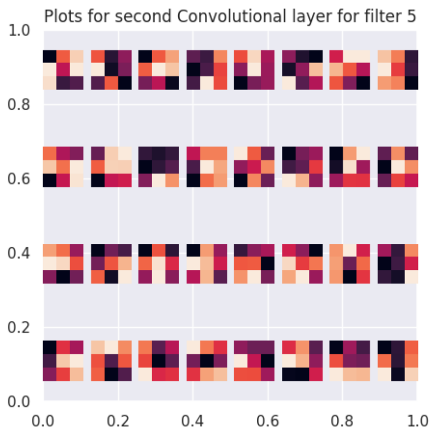


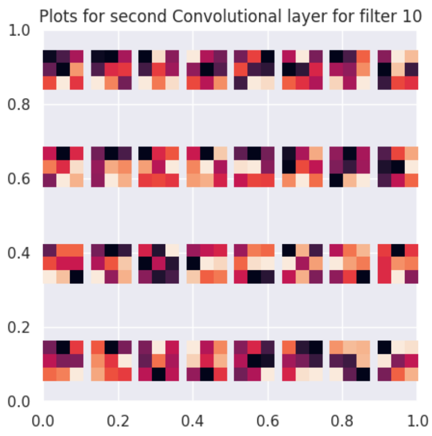
Figure 4: First convolutional layer filters

1. The first convolutional layer has 32 filter so all are shown together in a grid of 4*8. One inference can be drawn is that these filters are learning to recognise the low level features like edges which are evident from the feature maps of the first layer as shown in Figure 6. Filters with a darker patch in the middle and progressively lighter ones around it will sharpen the images which coincides with the concept of image sharpening filters. Similarly images with a horizontal or vertical streaks of light or dark patches will highlight the horizontal or vertical edges.

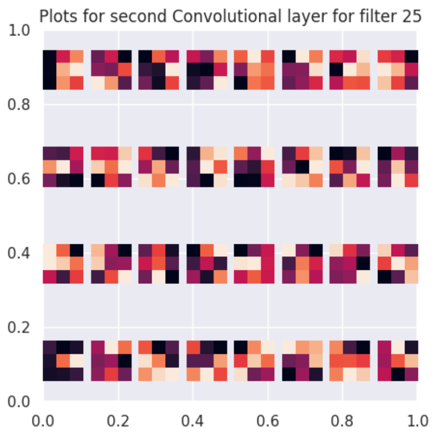
2. The output from first layer has shape 32*28*28 and the next convolution layer has 32 filters so each filter has 32 channels. Thus, it is not possible to show all of them but some are shown and inferences are drawn.



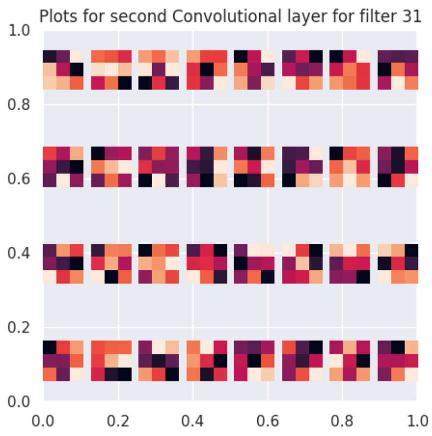
((a)) Filter 5



((b)) Filter 10



((c)) Filter 25



((d)) Filter 31

Figure 5: Images of 4 different filters out of 32 filters

The filter 5 still have some vertical and horizontal patches that indicate some edge information is getting filtered but as we go deeper on filter 10,25 or 31 patches doesn't represent clear well defined kernel nor what they are learning.

3. **Feature maps**:
Here I have shown some feature maps from first and second convolutional layers. The ones from the first layer is,

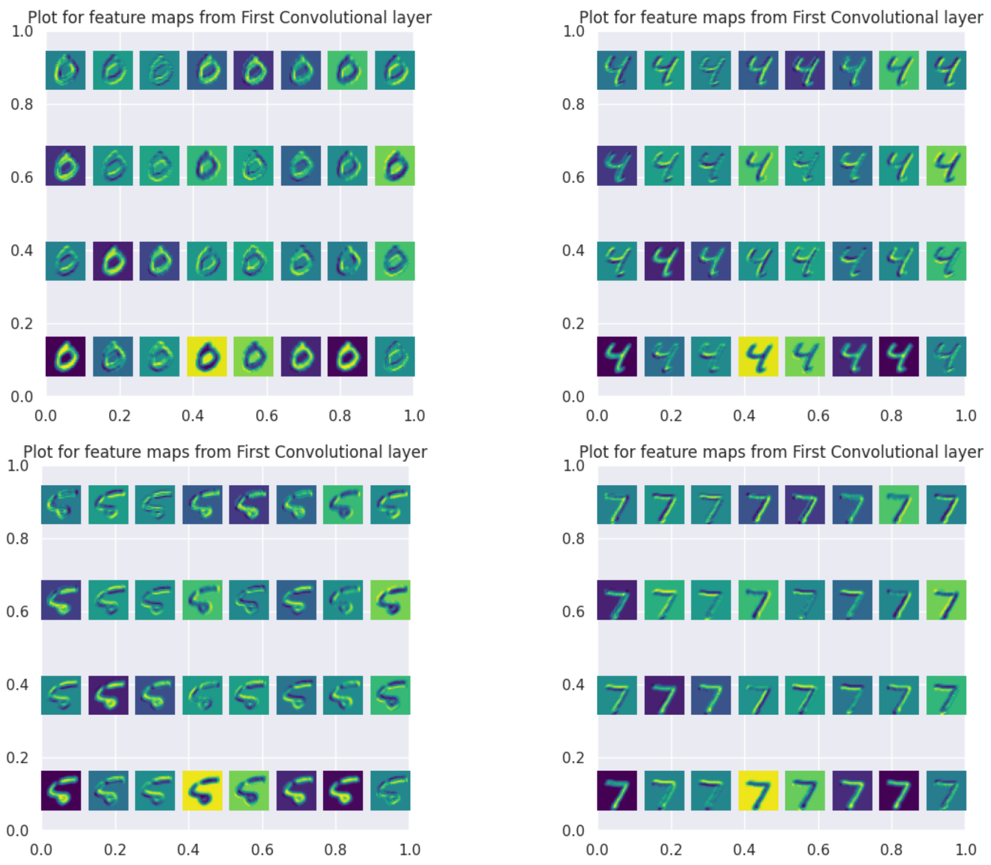


Figure 6: Images of 4 different feature maps for different inputs, namely 0,4 5 7

We can see that spatial context is clear shape and structure is evidently understood. This concludes our filter output (as shown in Figure 4) assumption from first convolutional layer.

The maps from second convolutional layer is,

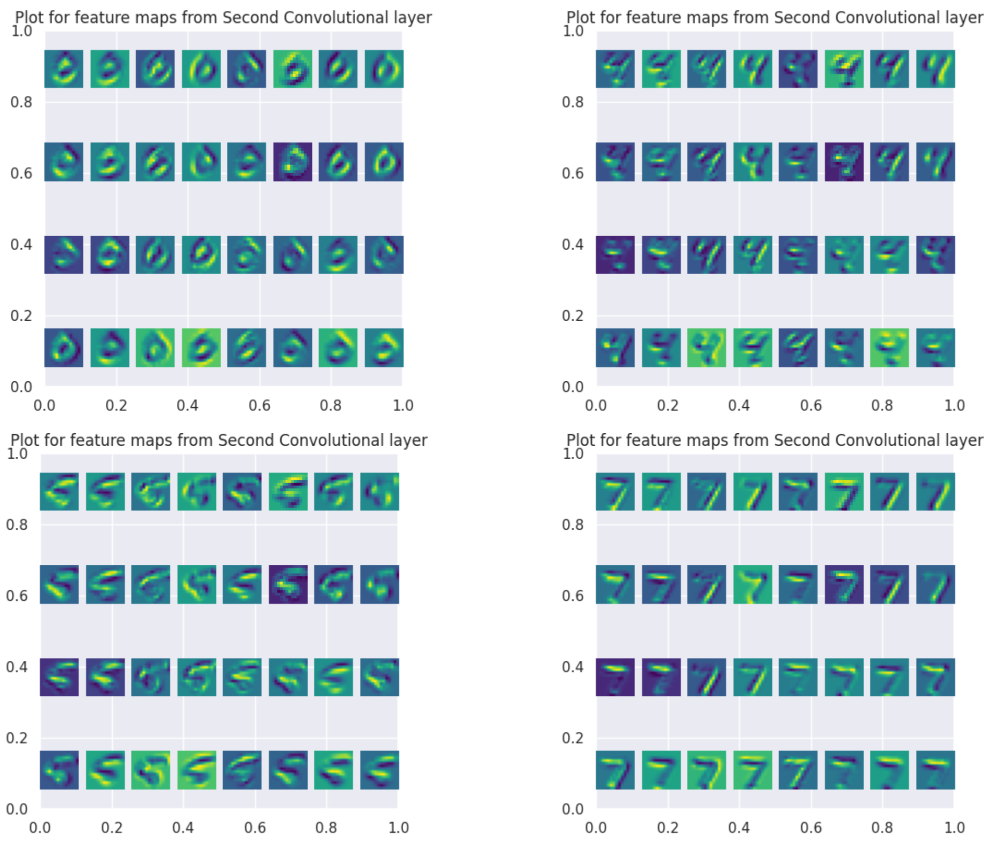
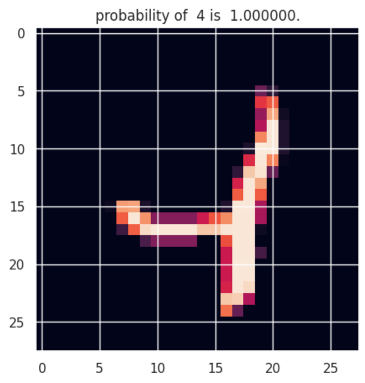


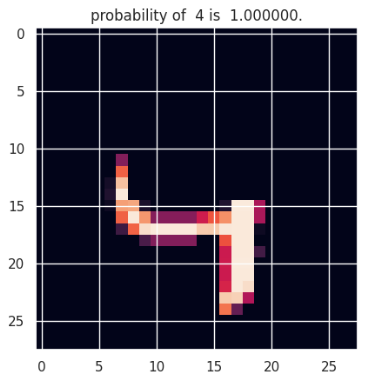
Figure 7: Images of 4 different feature maps for different inputs, namely 0,4 5 7

It is evident that the edges are a bit blurry and the output activations and not localized ,which indicates the network might be learning some global features and also some sophisticated features apart from sharpening edges. Moreover all of the filters are giving response thus affirming that our network doesn't have dead neurons which might hinder the gradients from flowing back, which indicates that out network is learning well on the learning rate given.

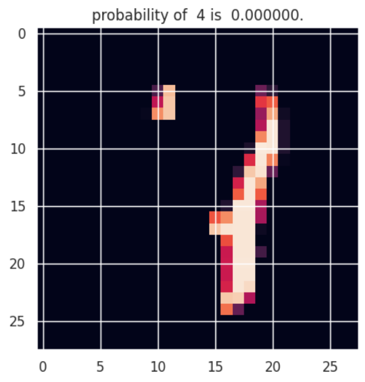
4. Occlusion experiments : Plotting 10 numbers and their probability heat map



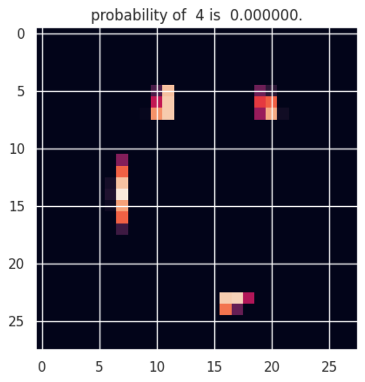
((a))



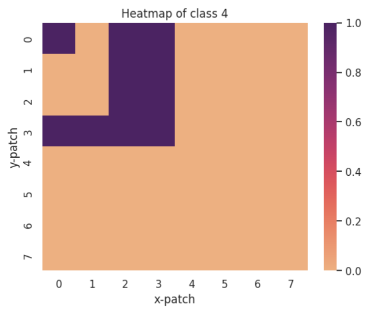
((b))



((c))

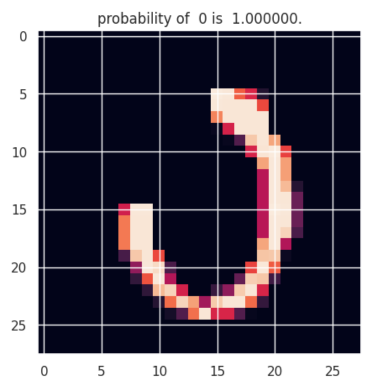


((d))

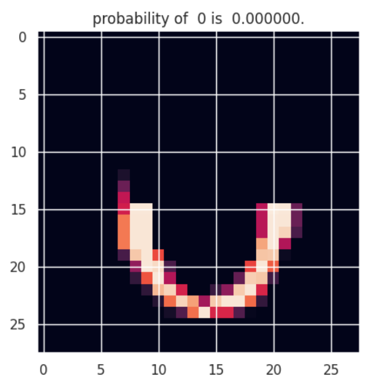


((e))

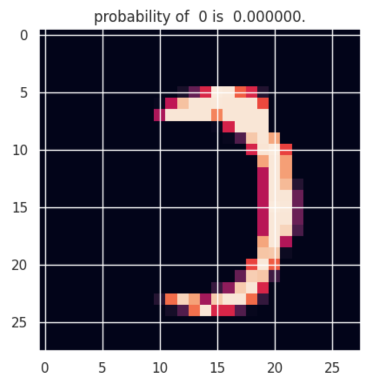
Figure 8: Occlusion on 4 with patch size=15



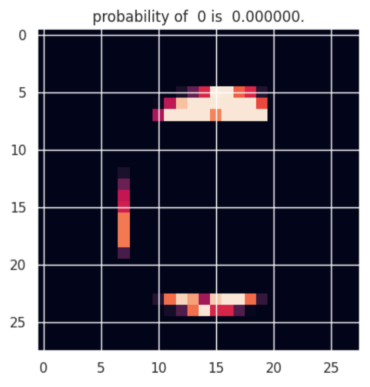
((a))



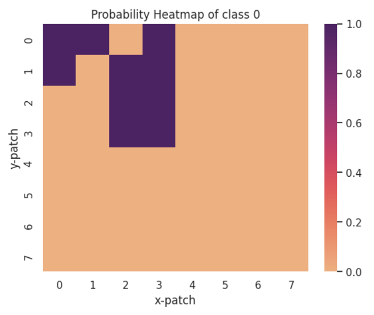
((b))



((c))

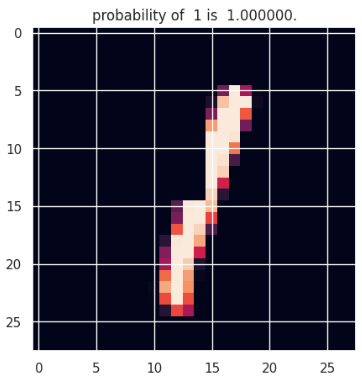


((d))

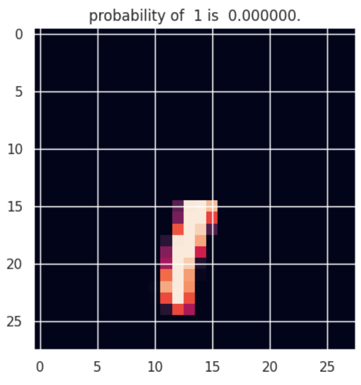


((e))

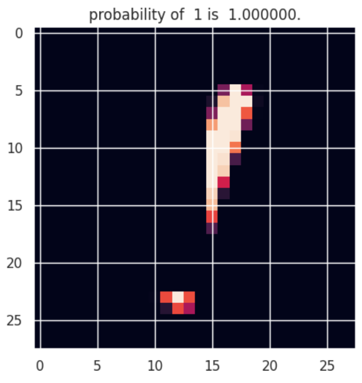
Figure 9: Occlusion on 0 with patch size=15



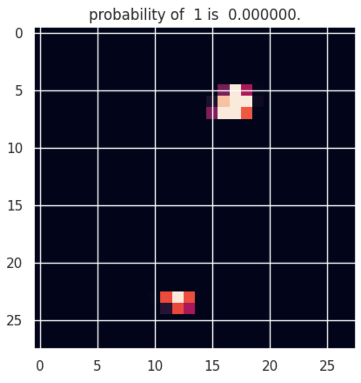
((a))



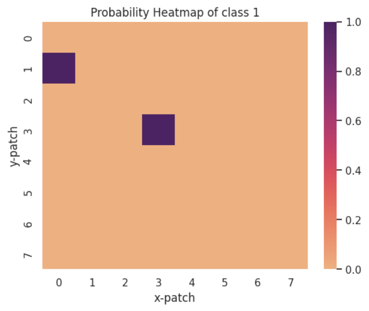
((b))



((c))

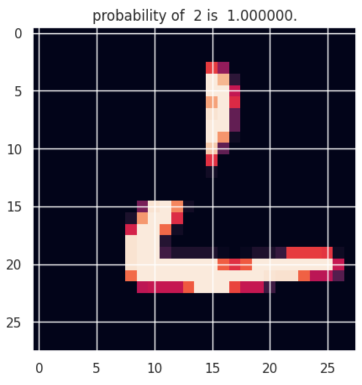


((d))

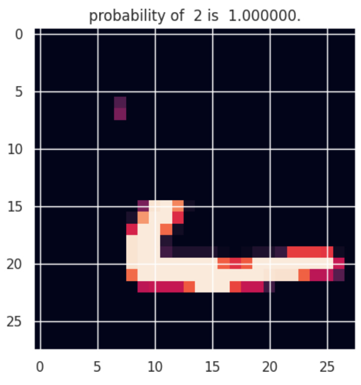


((e))

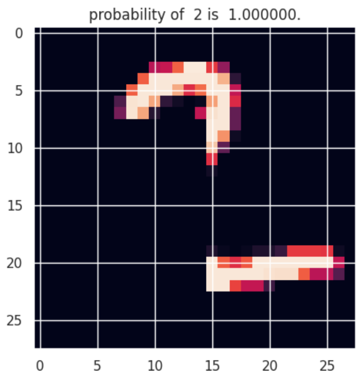
Figure 10: Occlusion on 1 with patch size=15



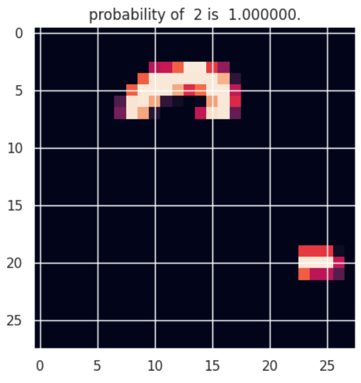
((a))



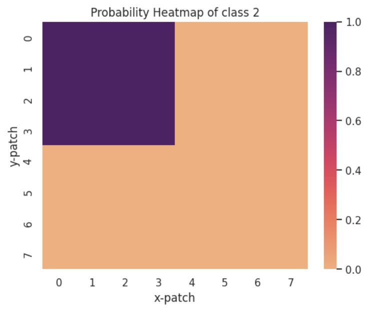
((b))



((c))

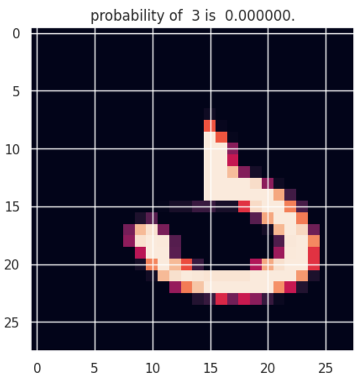


((d))

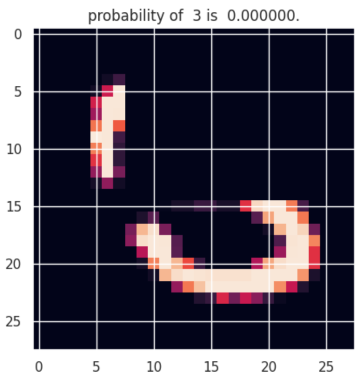


((e))

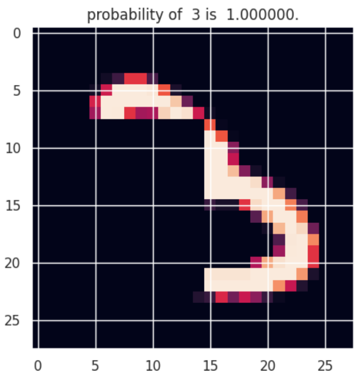
Figure 11: Occlusion on 2 with patch size=15



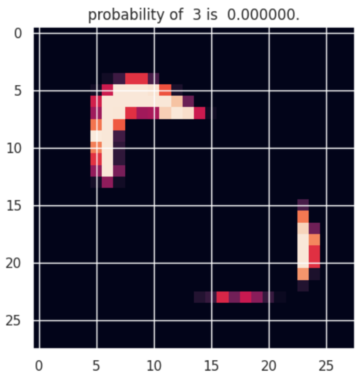
((a))



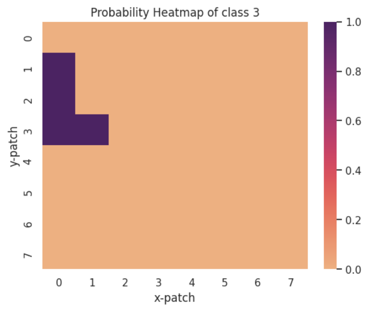
((b))



((c))

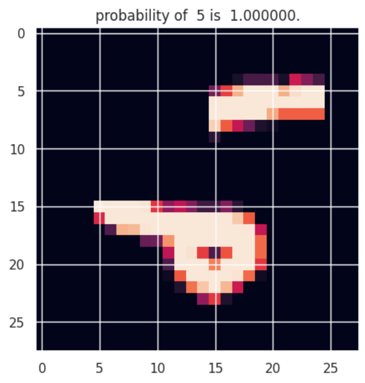


((d))

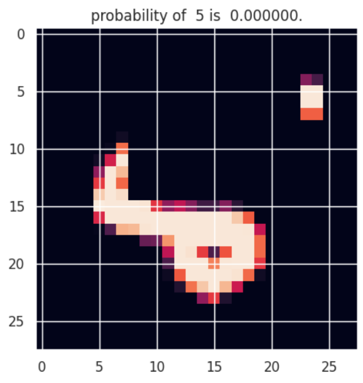


((e))

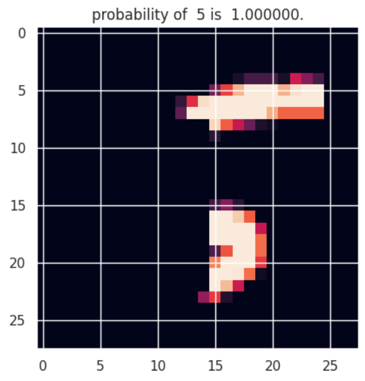
Figure 12: Occlusion on 3 with patch size=15



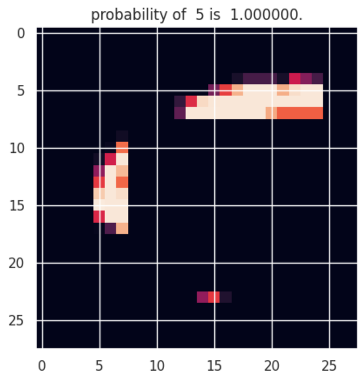
((a))



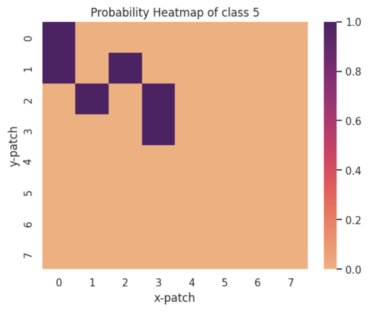
((b))



((c))

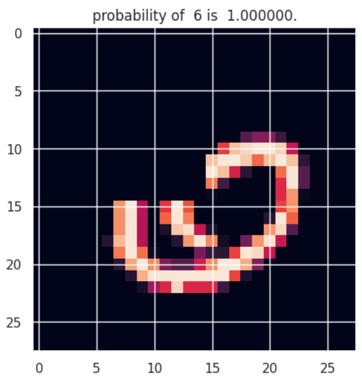


((d))

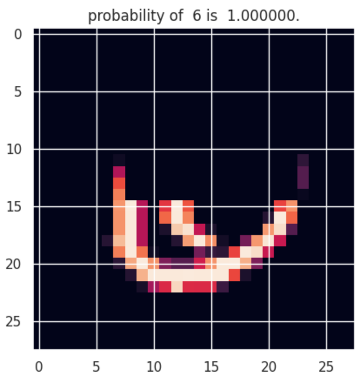


((e))

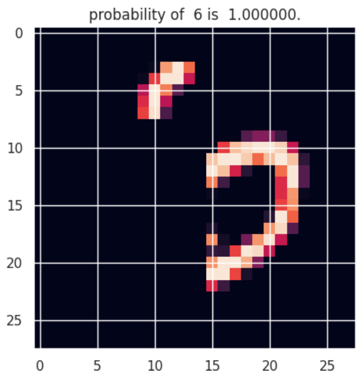
Figure 13: Occlusion on 5 with patch size=15



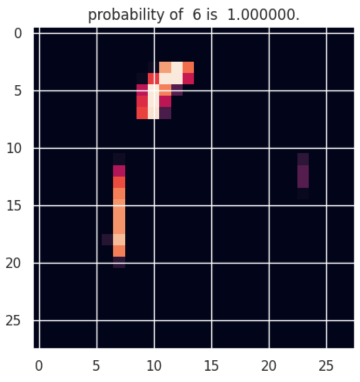
((a))



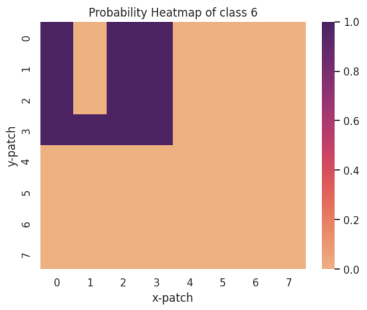
((b))



((c))

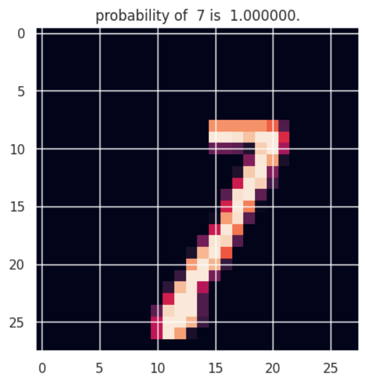


((d))

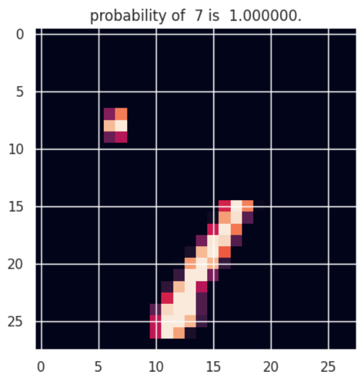


((e))

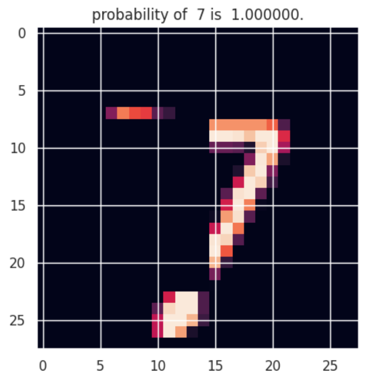
Figure 14: Occlusion on 6 with patch size=15



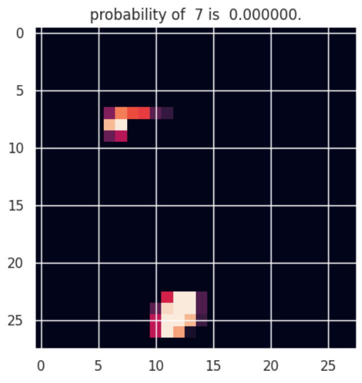
((a))



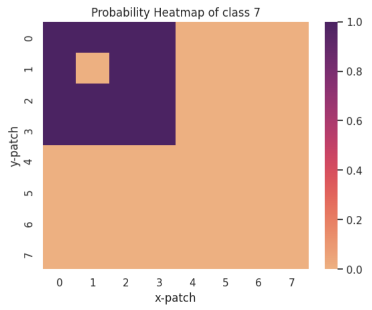
((b))



((c))

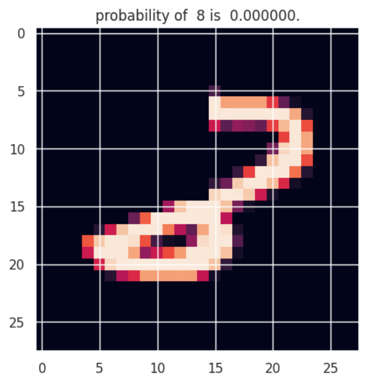


((d))

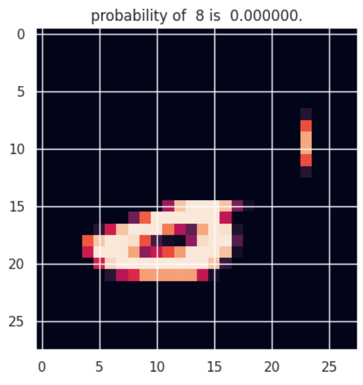


((e))

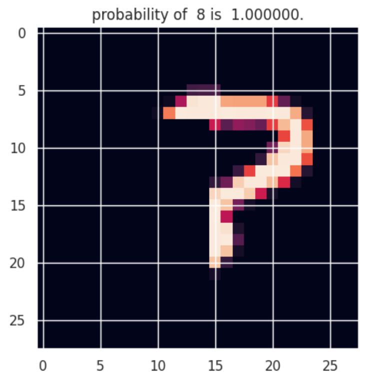
Figure 15: Occlusion on 7 with patch size=15



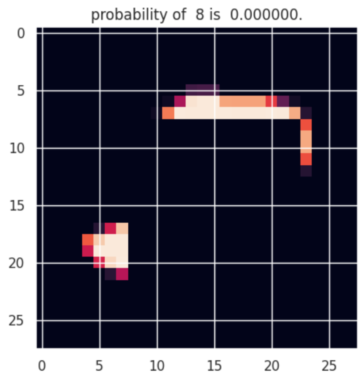
((a))



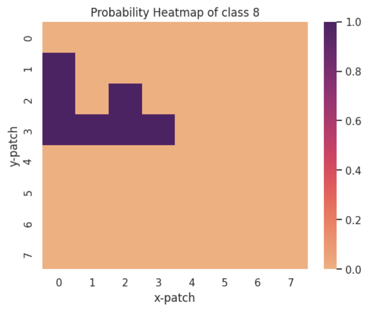
((b))



((c))

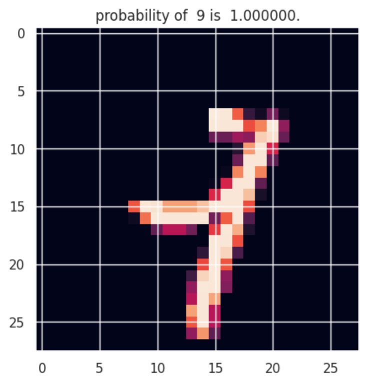


((d))

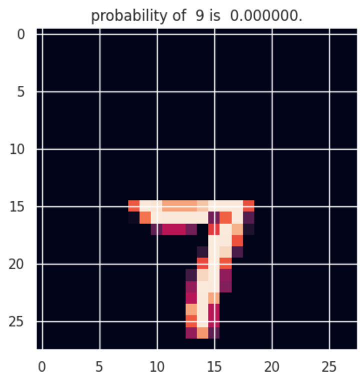


((e))

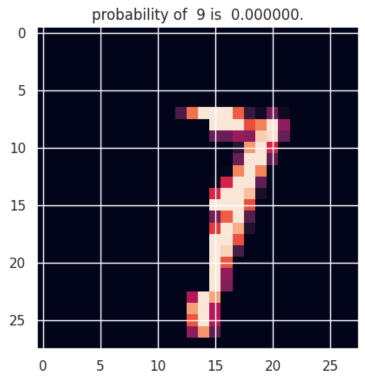
Figure 16: Occlusion on 8 with patch size=15



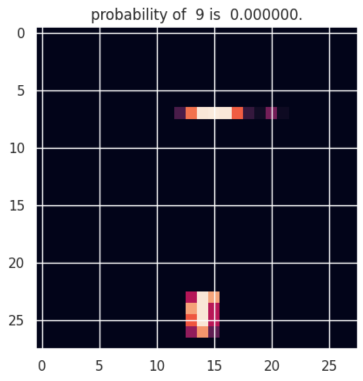
((a))



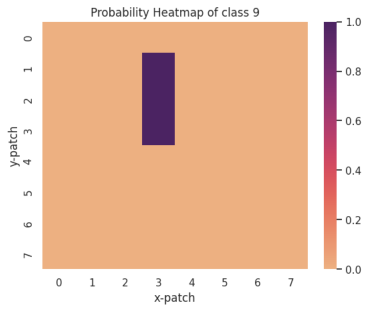
((b))



((c))



((d))



((e))

Figure 17: Occlusion on 9 with patch size=15

The occlusion experiments confirm that learning is not fully meaningful as expected because 3 when occluded gives the predicted class as 8 thus for 3 and 8 the learning is similar, moreover a lot of time when the patch is occluding a certain section of the image , for example like middle left or upper corner for digit 9 the prediction fails and such fails are evident for most of the other digits too. This proves that learning does depend more on certain regions compared to other. However in our case the patch size is big and is hindering most of the important pixel when probability of outcome goes zero which indicates that my network learned from the right regions however certain discrepancies still exist.

3 Question 3: Adversarial attack

3.1 Non-Targeted attack

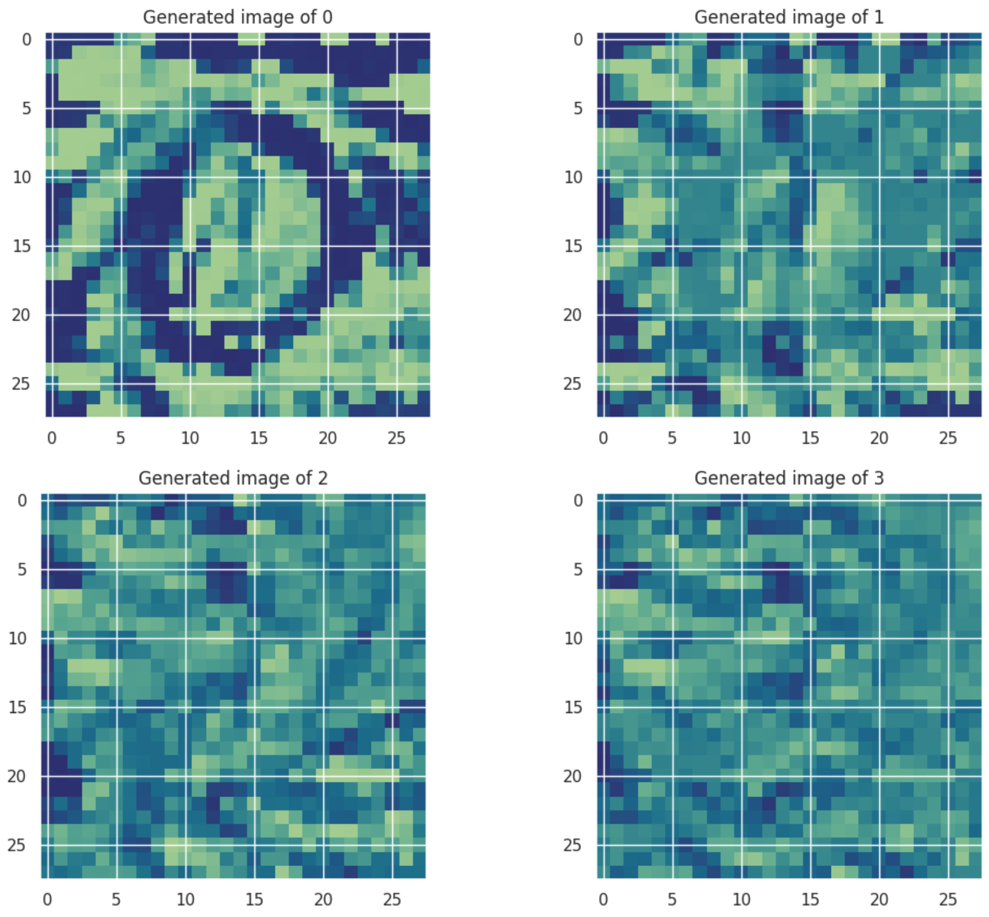


Figure 18: Generated adversarial images 0,1,2,3

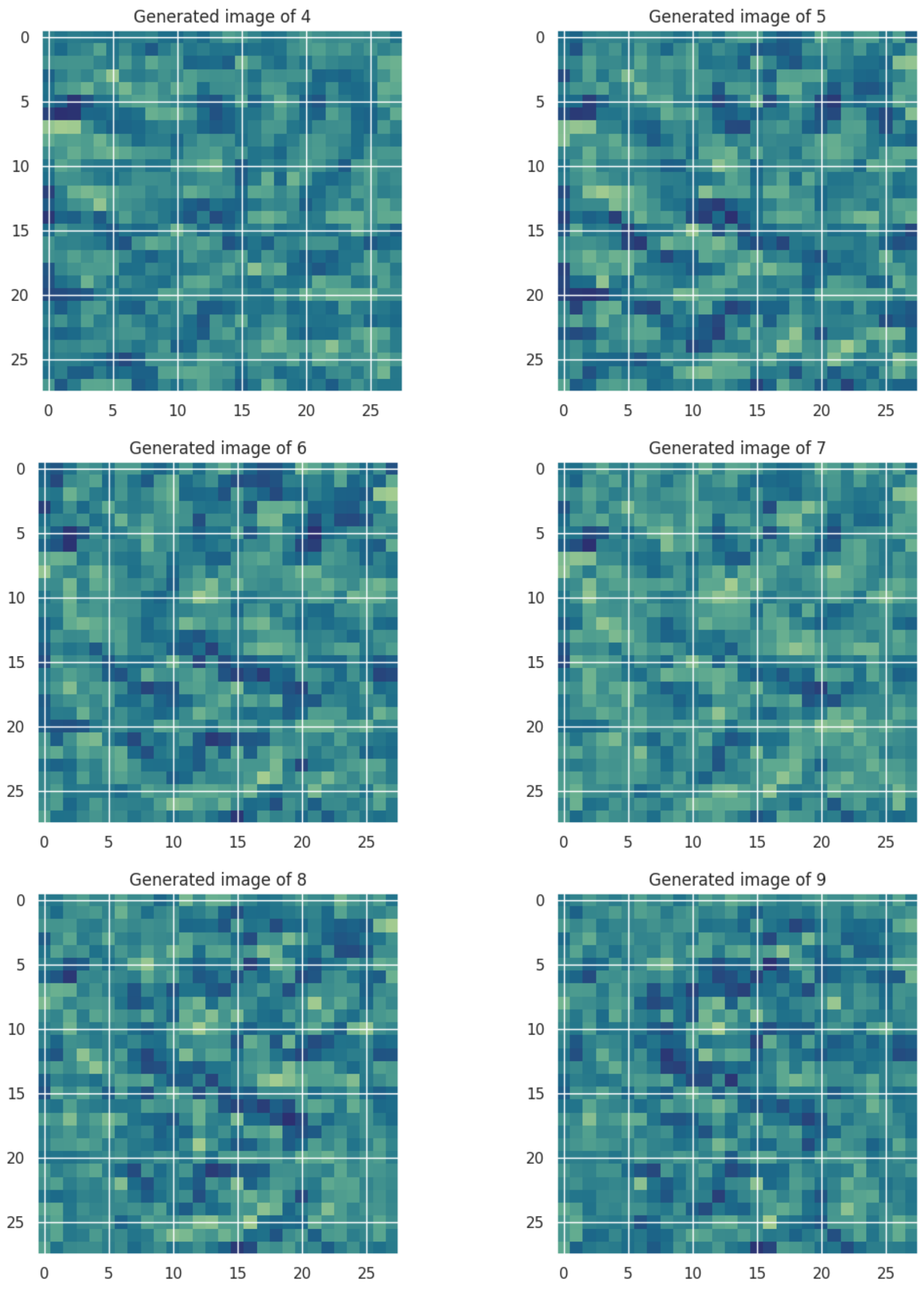


Figure 19: Generated adversarial images 4,5,6,7,8,9

Cost function plots

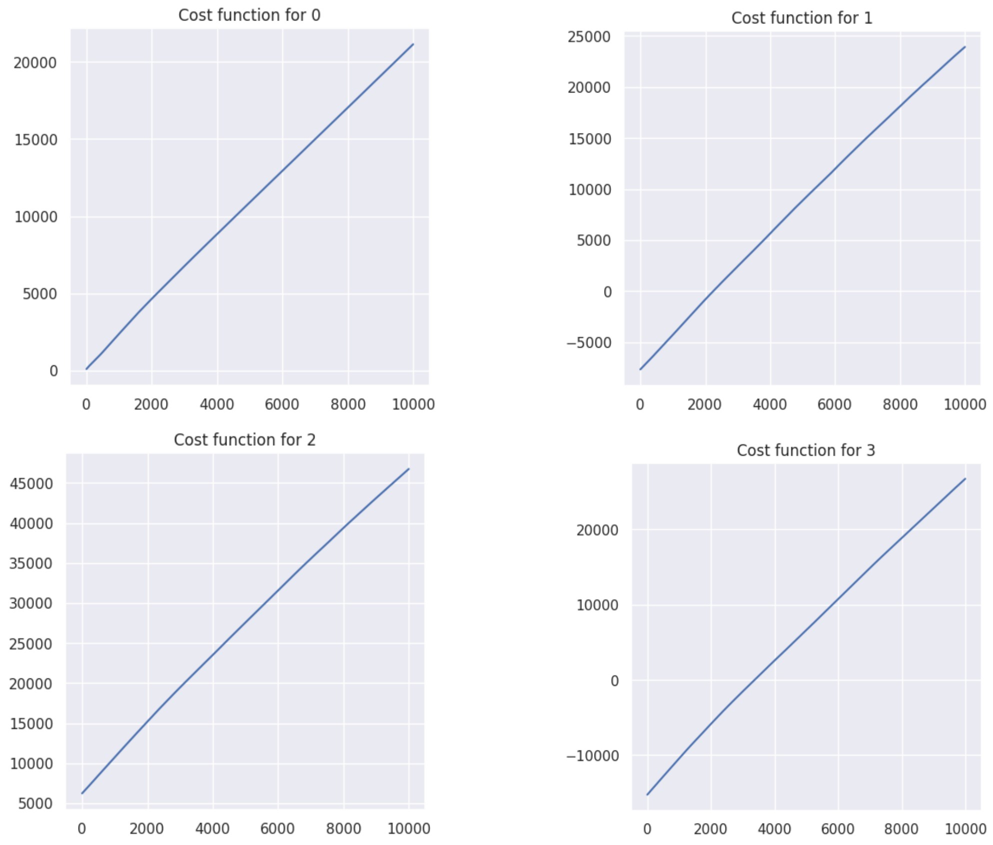


Figure 20: Cost function for Generated adversarial images 0,1,2,3

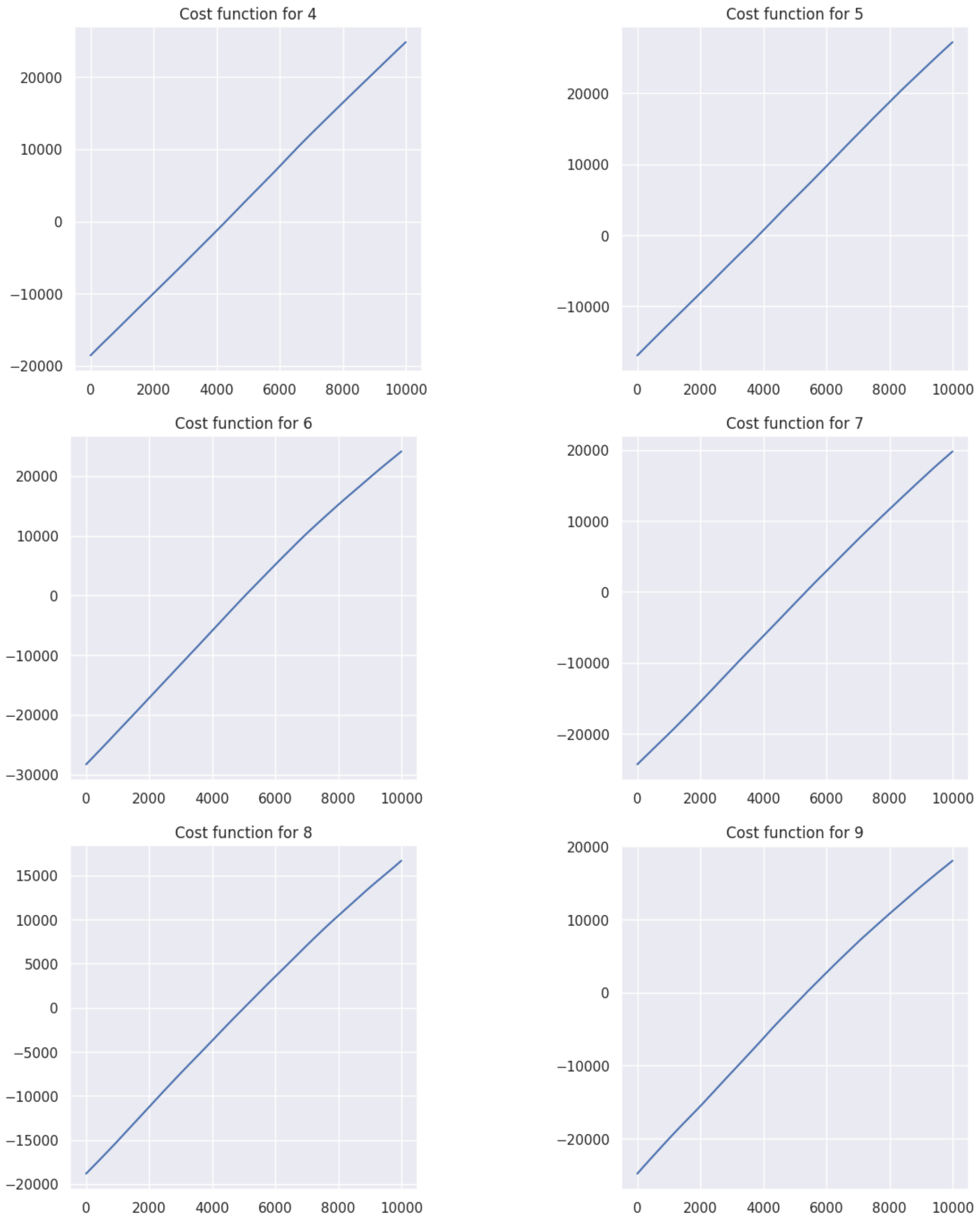


Figure 21: Cost function for Generated adversarial images 4,5,6,7,8,9

Inferences drawn from Non-Targeted attack:

- The network predict the target class with high confidence all the time. Moreover probability reaches 1 within some 1500 iterations only. This shows that how humans learn and how machines learn are very different from one another.
- The generated image doesn't look like a number. Even I ran the code for double the number of iterations but yet no visually meaningful number images were drawn out. Number 0 is still observable but for other images only fragments are visible. For 6 we can see some curve and for 8 a blob is faintly legible, however 1 is not obvious at the same time. One reason for images not appearing even-though target class is maximised is because we are not fully using all the features.Our images in the dataset is a part of very high dimensional space so while learning from the image is concerned and classifying based on it still gives very accurate results but when we try to modify the noise tensor to match an image then it fails to learn from all the features as everything is not captured in our model.
- The cost function is increasing for all the images.

3.2 Targeted Attack

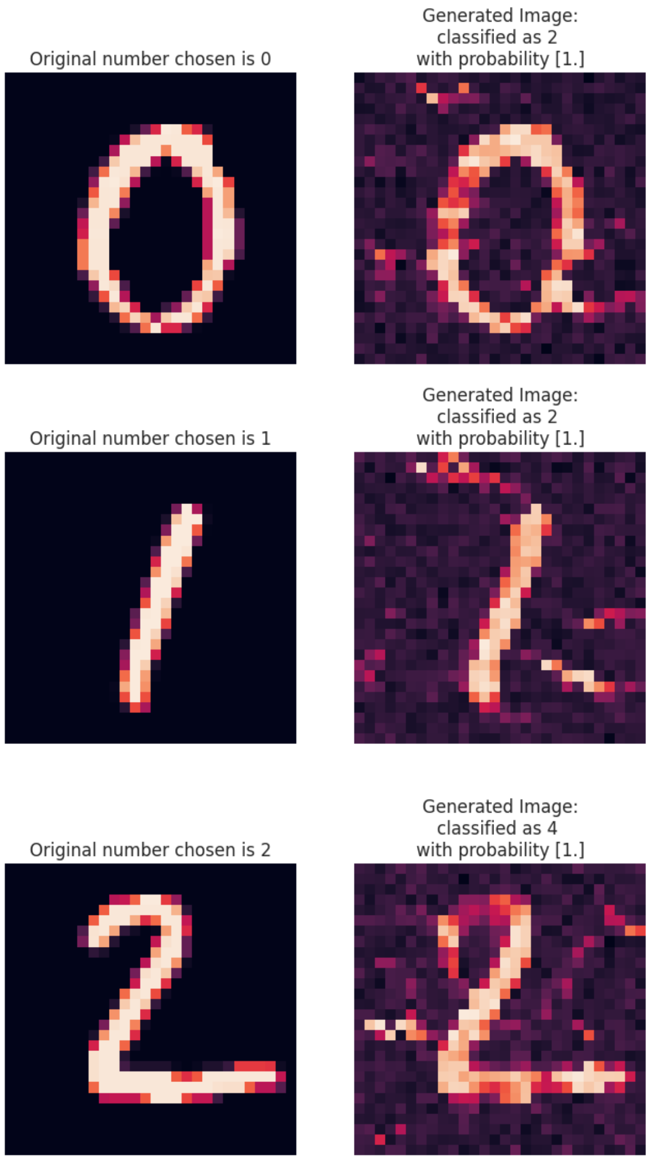

Original number chosen is 0
Generated Image: classified as 2 with probability [1.]
Original number chosen is 1
Generated Image: classified as 2 with probability [1.]
Original number chosen is 2
Generated Image: classified as 4 with probability [1.]

Figure 22: Generated adversarial images 0,1,2

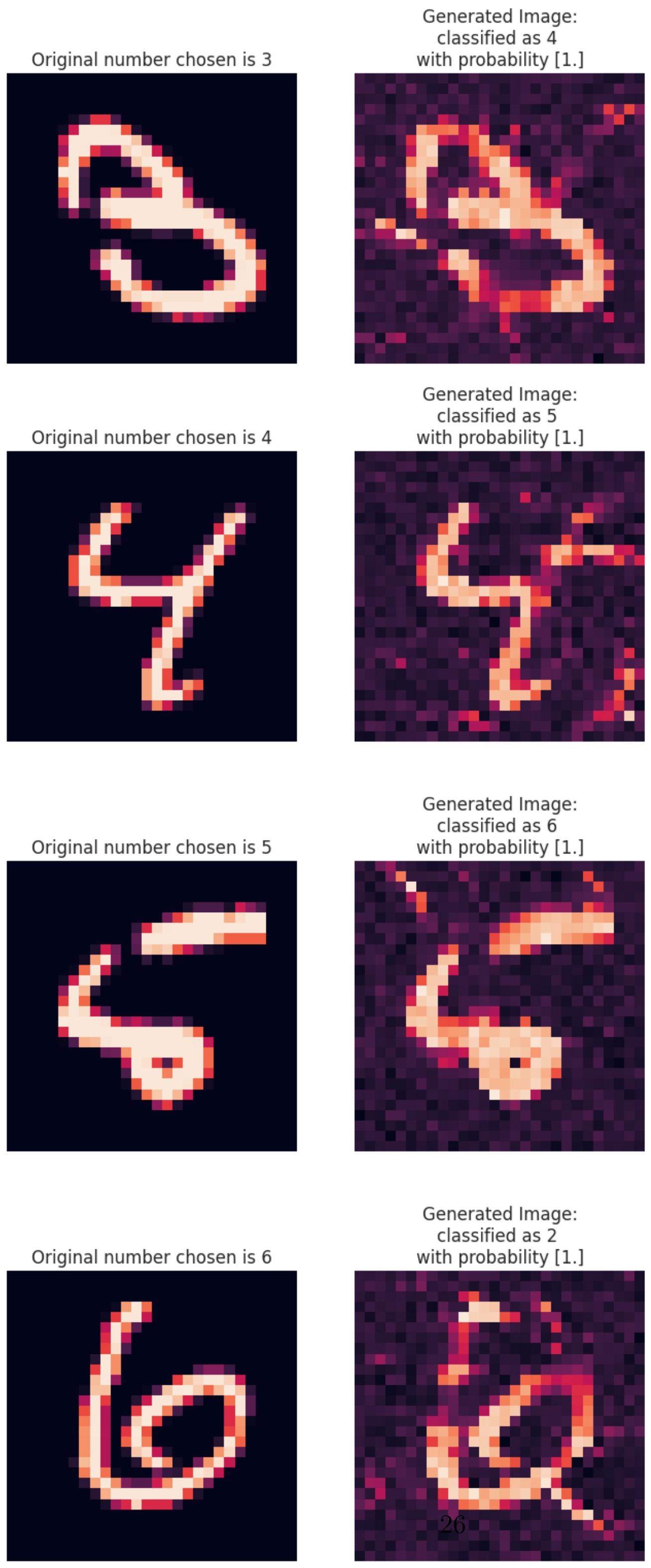

Original number chosen is 3
Generated Image: classified as 4 with probability [1.]
Original number chosen is 4
Generated Image: classified as 5 with probability [1.]
Original number chosen is 5
Generated Image: classified as 6 with probability [1.]
Original number chosen is 6
Generated Image: classified as 2 with probability [1.]

Figure 23: Generated adversarial images 3,4,5,6

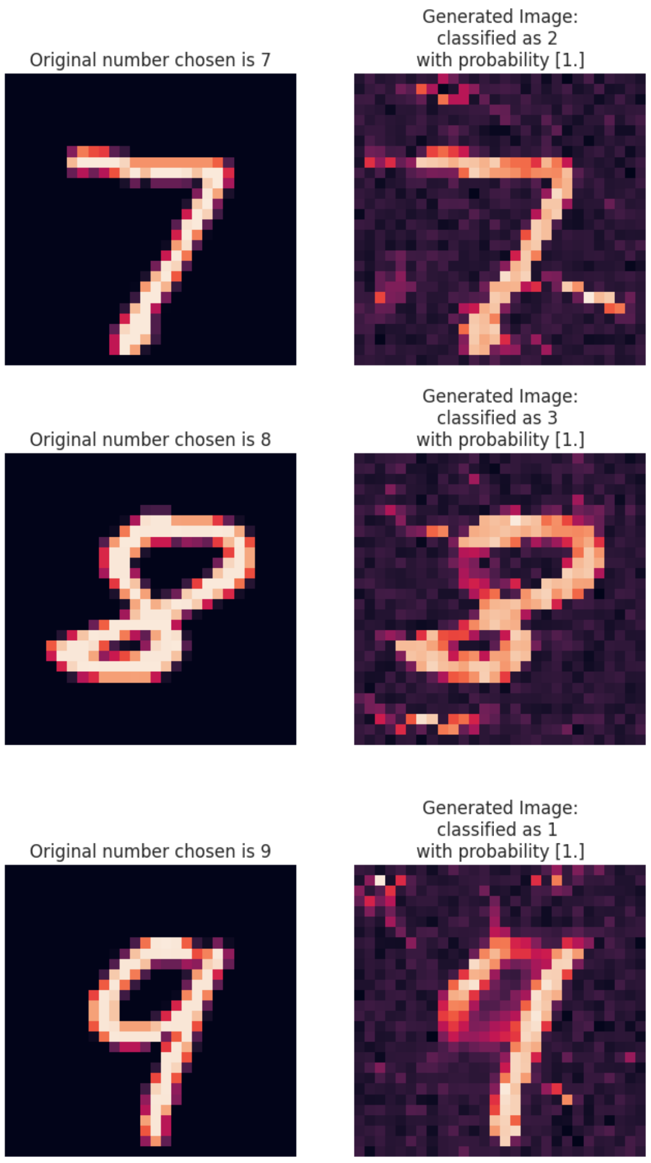


Figure 24: Generated adversarial images 7,8,9

The generated images look alike the original image but the classifier is fooled to predict it as other numbers.However if its creating the number I have trained it upon then it has learned the right features but taught to classify it as some other number.

3.3 Adding Noise

The corresponding plots after adding noise are:

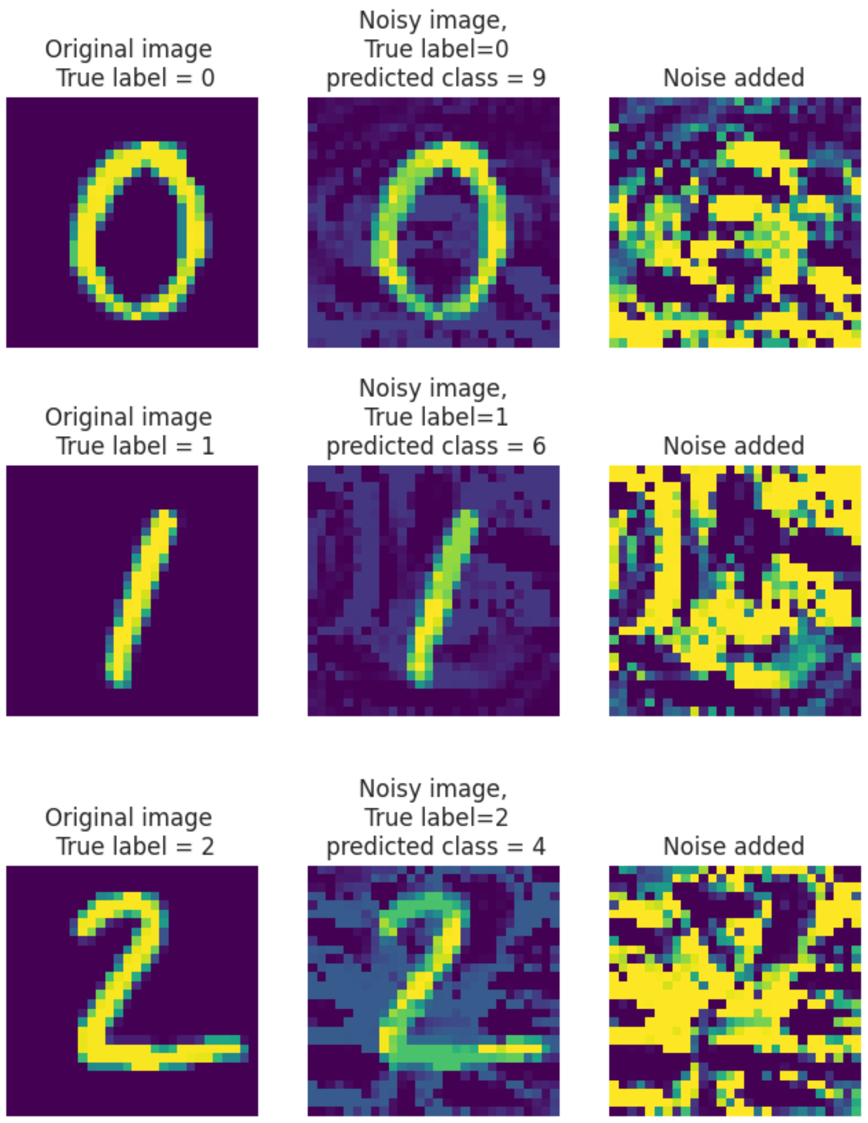


Figure 25: Plots describing noise added on images 0,1,2

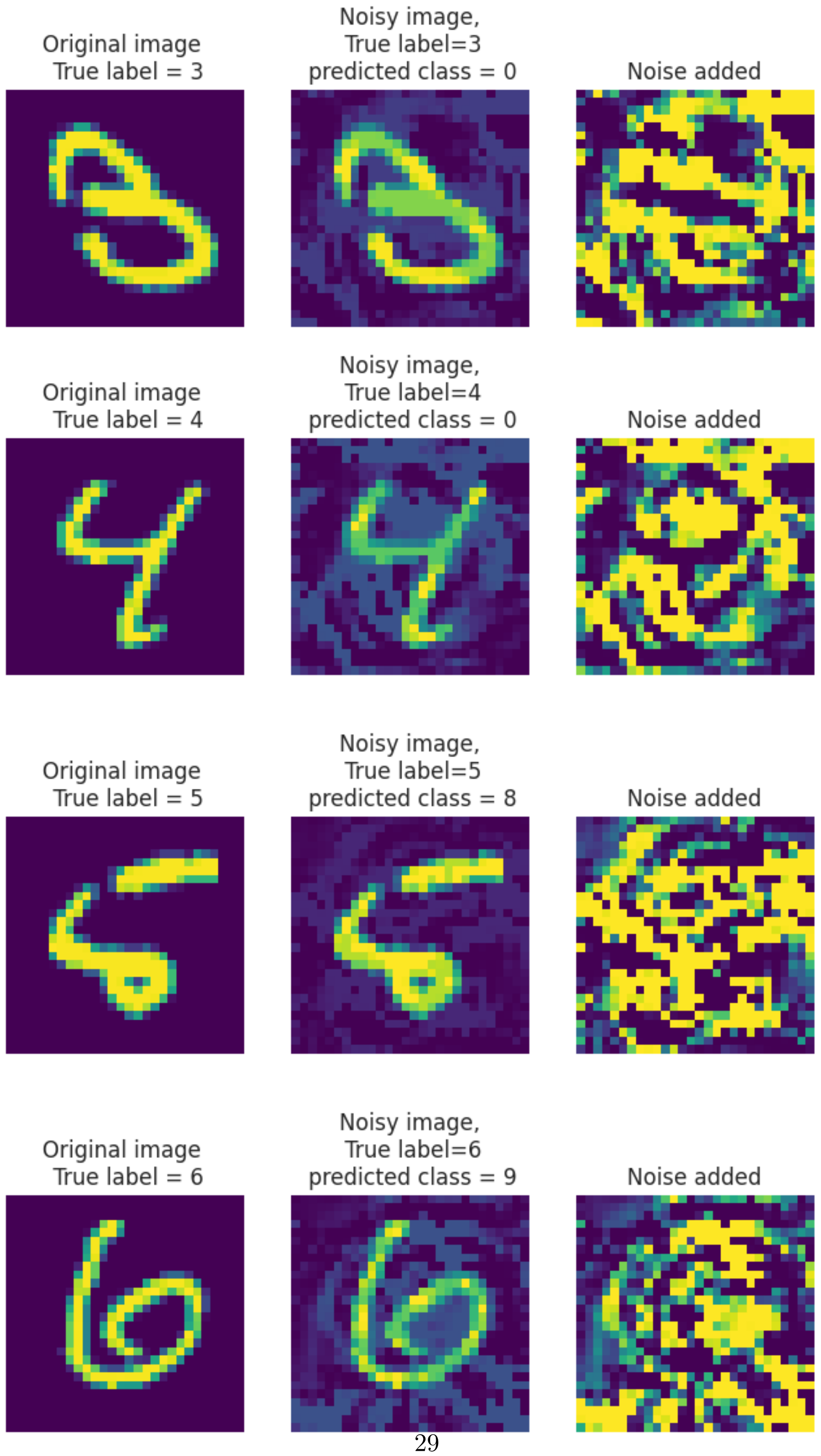


Figure 26: Plots describing noise added on images 3,4,5,6

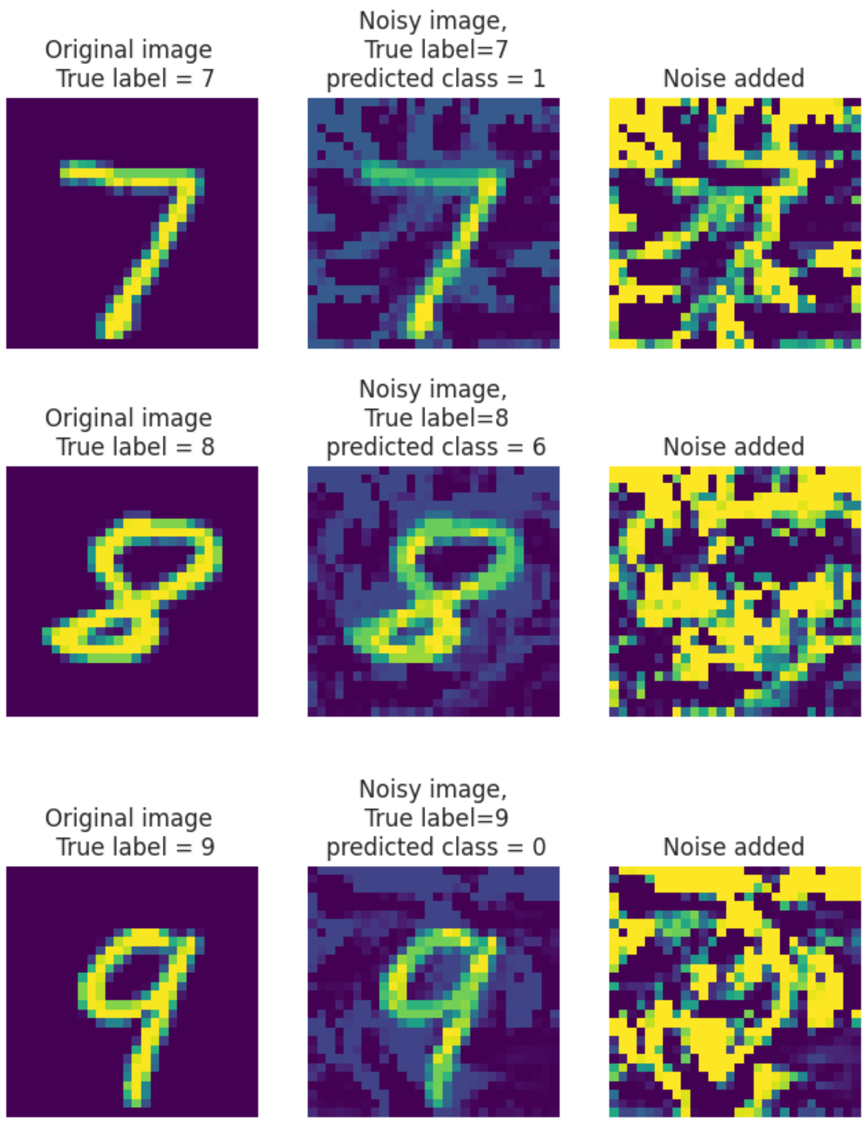


Figure 27: Plots describing noise added on images 7,8,9

2. I have extracted the noise of some classes and added it to other images which are not the original image and all the time it gave the output label as same as the predicted class which it has been misclassified as. Thus I understood that even if I change the original image to another image the predicted class will be what the network was trained upon the noise. The results are reported in the notebook