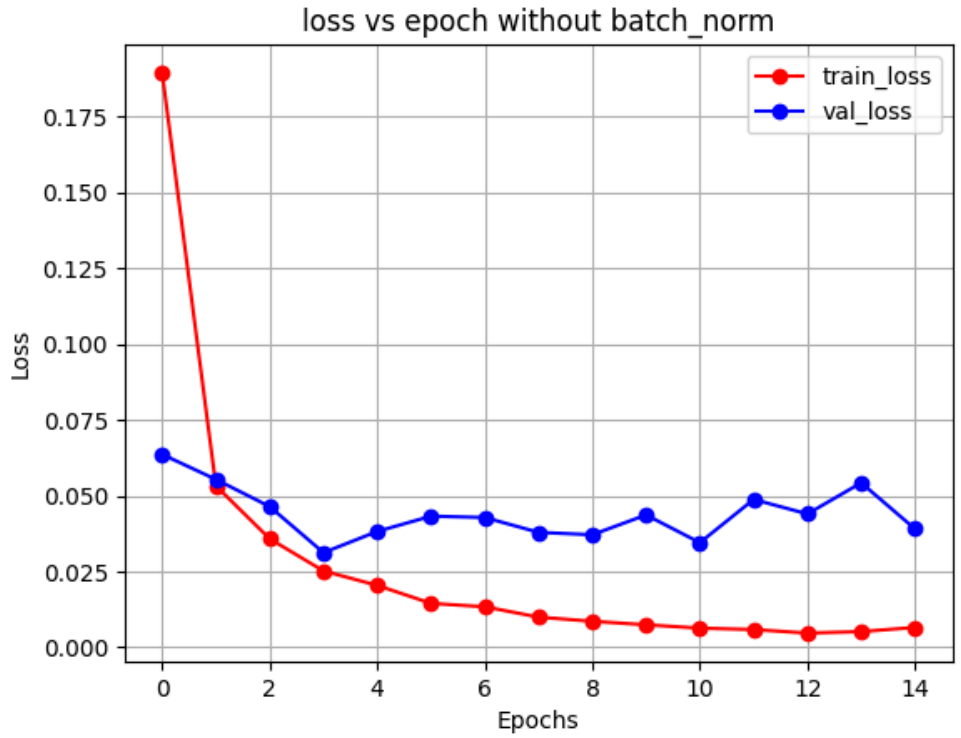
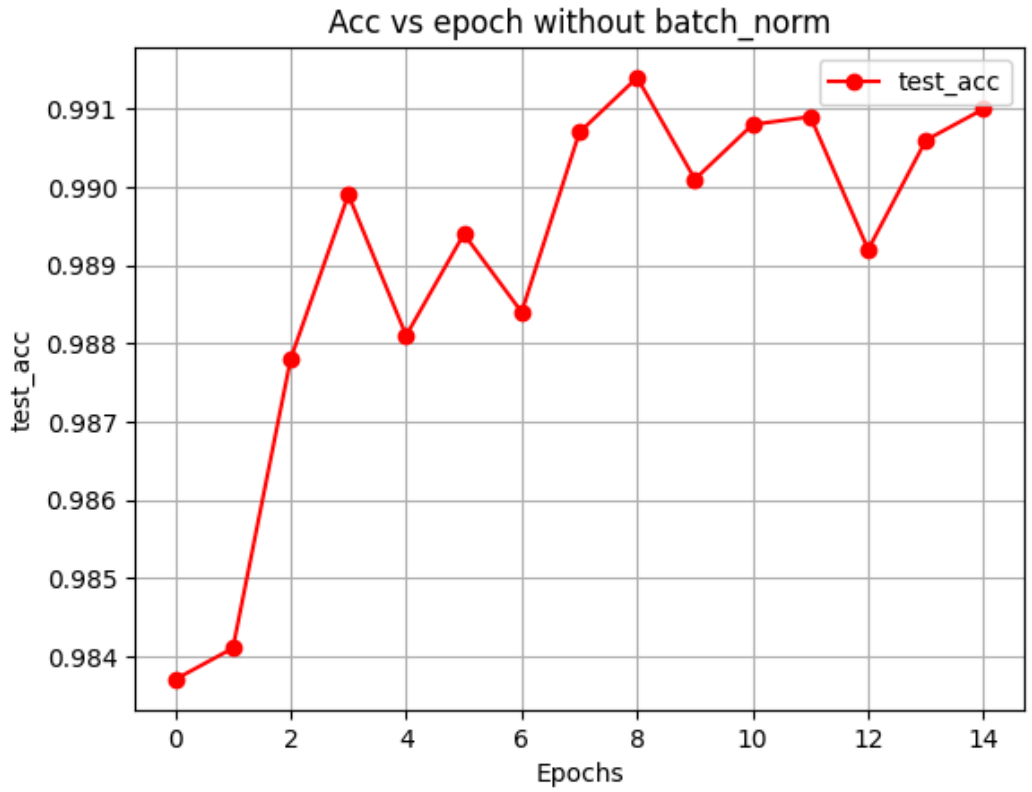
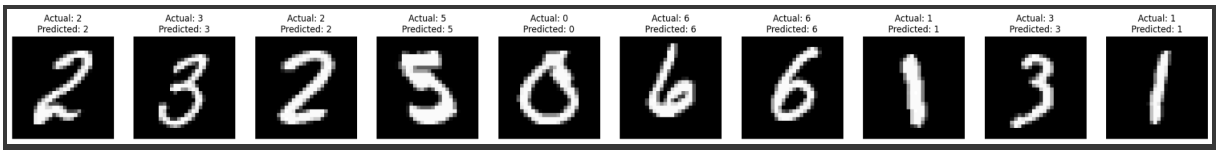
1.1 . PLOTS OF TRAINING ERROR, VALIDATION ERROR AND PREDICTION ACCURACY





PREDICTION ACCURACY FOR THE WHOLE TEST SET = 99.1 %

1.2 . DISPLAYING RANDOMLY SELECTED TEST IMAGES



1.3 . DIMENSIONS OF INPUT AND OUTPUT AT EACH LAYER

```
LAYER_1: INPUT:[BS,1,28,28] OUTPUT:[BS,32,28,28] -CONV1

LAYER_2: INPUT:[BS,32,28,28] OUTPUT:[BS,32,14,14] -MAXPOOL

LAYER_3: INPUT:[BS,32,14,14] OUTPUT:[BS,32,14,14] -CONV2

LAYER_4: INPUT:[BS,32,14,14] OUTPUT:[BS,32,7,7] -MAXPOOL

FC (NEURON NUMBERS):      INPUT:[BS,1568] OUTPUT:[BS,500]

FC (NEURON NUMBERS):      INPUT:[BS,500] OUTPUT:[BS,10]
```

BS - Batch size

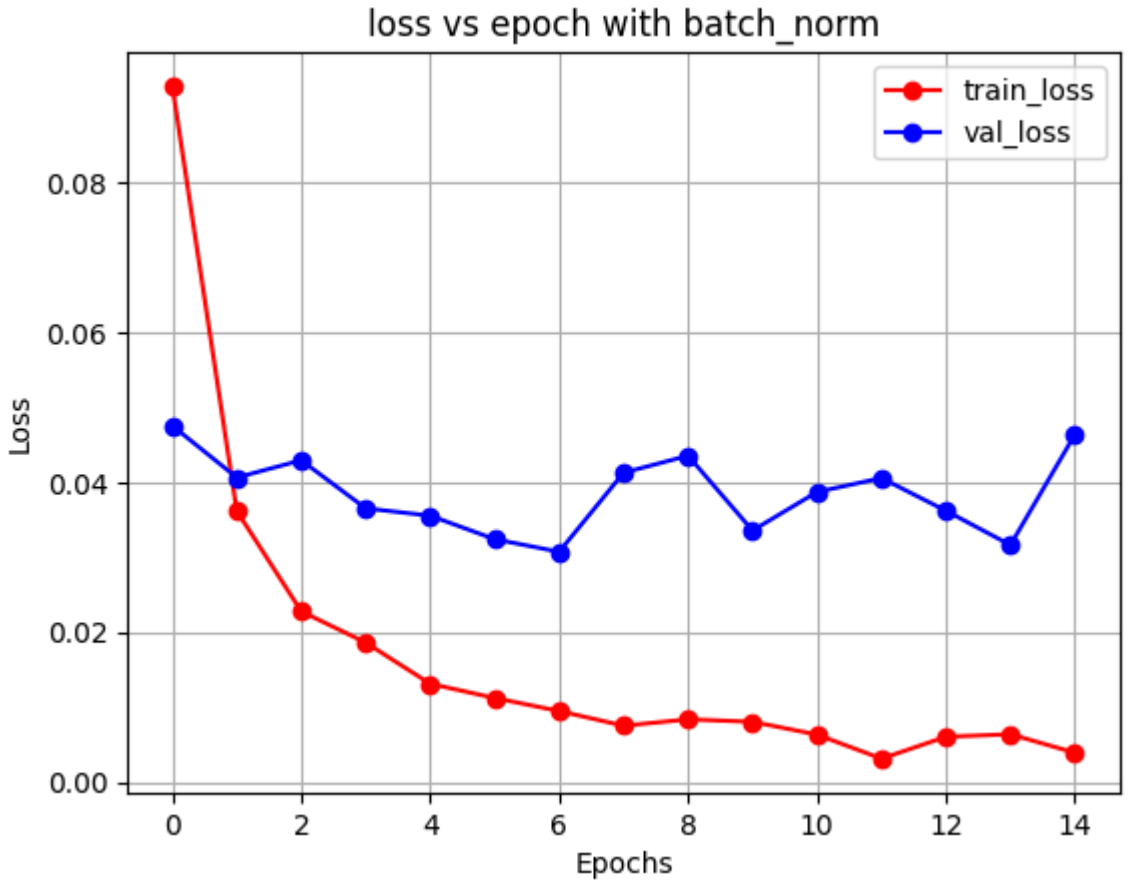
1.4. PARAMETERS OF THE NETWORK

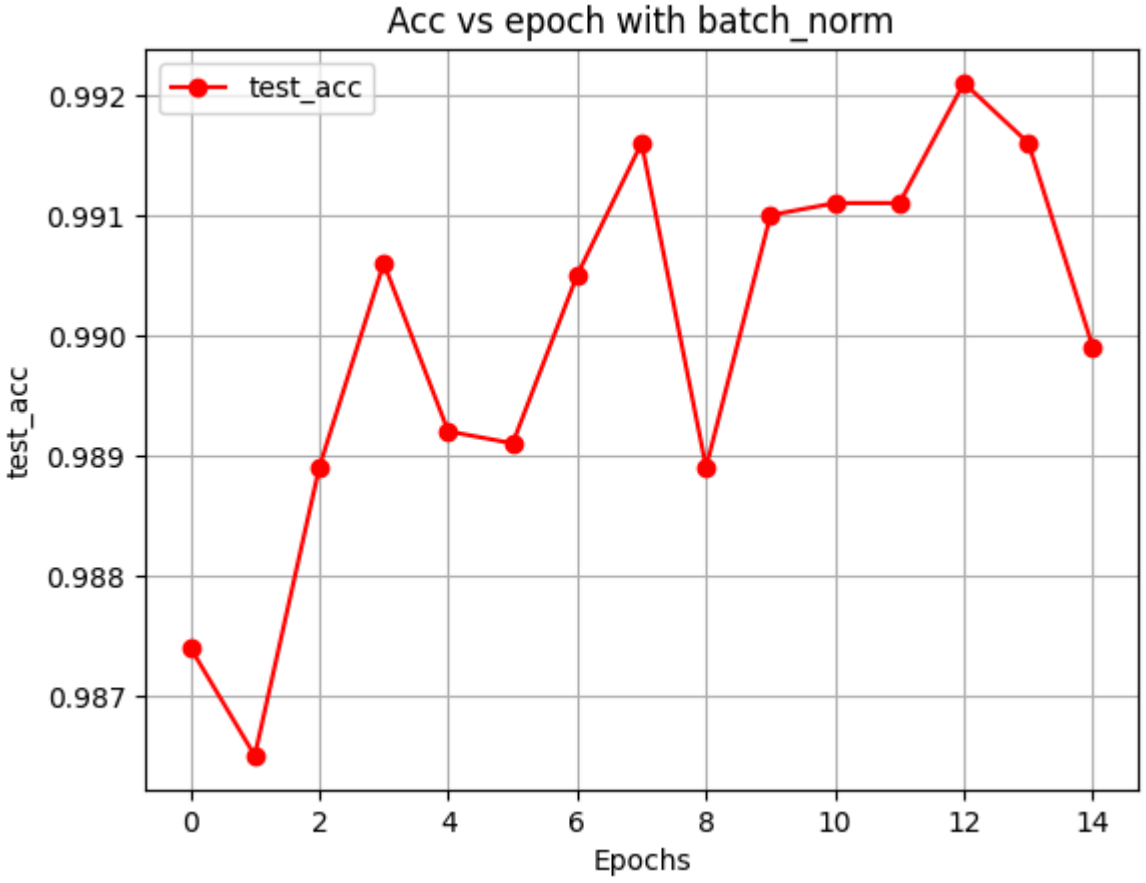
```
conv_params:9568 | fc_params:789510
Convolutional layers parameters: 9,568
Fully connected layers parameters: 789,510
Total trainable parameters: 799,078
```

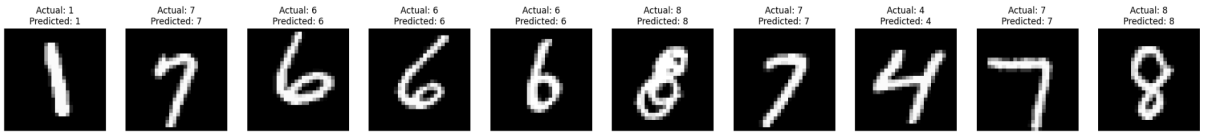
1.5: NETWORK PARAMETER COUNT:

- CONV_1_NEURONS: 32 * 28 * 28 = 25088
- CONV_2_NEURONS: 32 * 14 * 14 = 6272
- TOTAL CONVOLUTIONAL NEURON=31,360
- FC_NEURONS: 500 + 10 = 510
- TOTAL NEURONS: 31870

1.6 . MODEL CREATED WITH BATCH - NORMALIZATION


loss vs epoch with batch_norm
train_loss
val_loss
Loss
Epochs
0.00
0.02
0.04
0.06
0.08
0
2
4
6
8
10
12
14


Acc vs epoch with batch_norm
test_acc
test_acc
Epochs
0.987
0.988
0.989
0.990
0.991
0.992
0
2
4
6
8
10
12
14



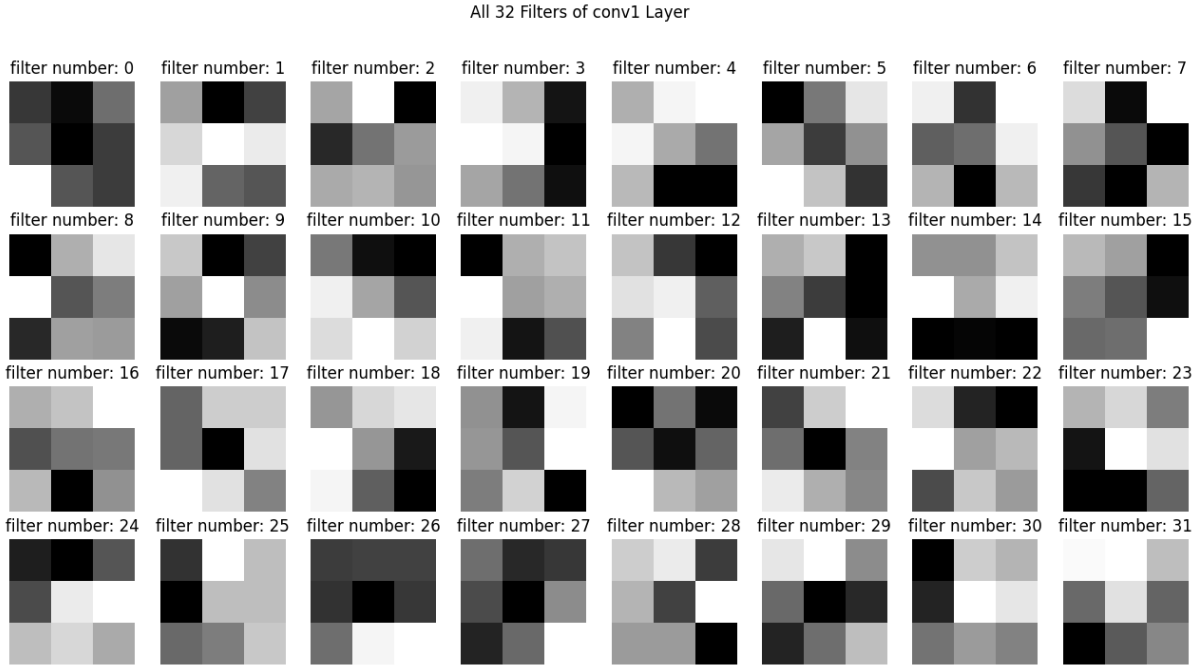
Result:

- Batch Normalization did not improve the test accuracy to a greater extent. Both accuracy are almost the same. The best test accuracy with BN is 99.21% and without BN is 99.14%

- Batch Normalization would have increased the training time slightly due to extra computations.

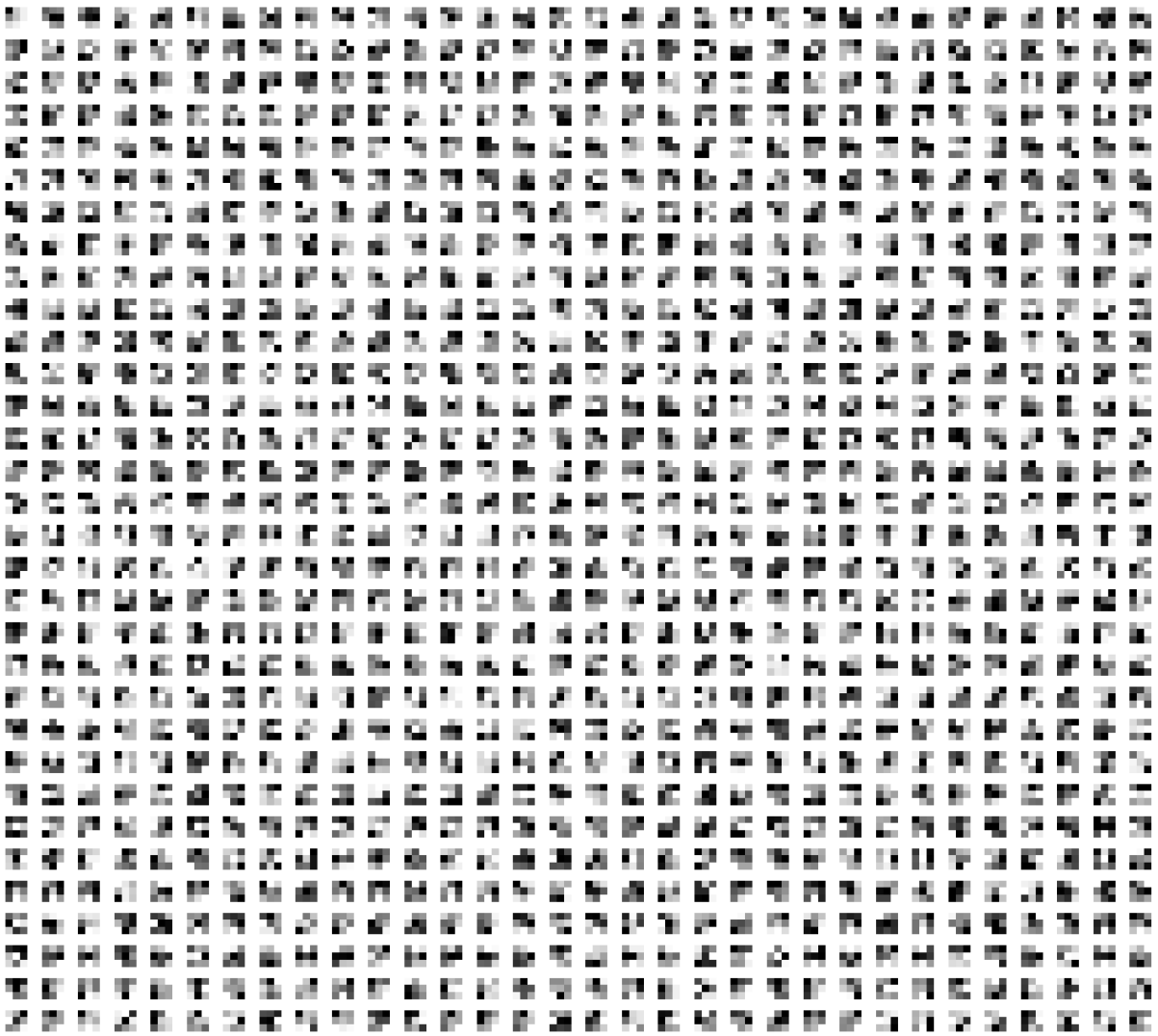
2. VISUALIZING CONVOLUTIONAL NEURAL NETWORK:

2.1 CONV1 LAYER FILTERS



2.2 FILTERS OF HIGHER LAYER

All 1024 Channel-Filters of conv2 Layer



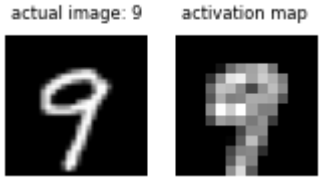
OBSERVATION FOR 2.1 & 2.2

Comparison between Conv1 and Conv2 Filters:

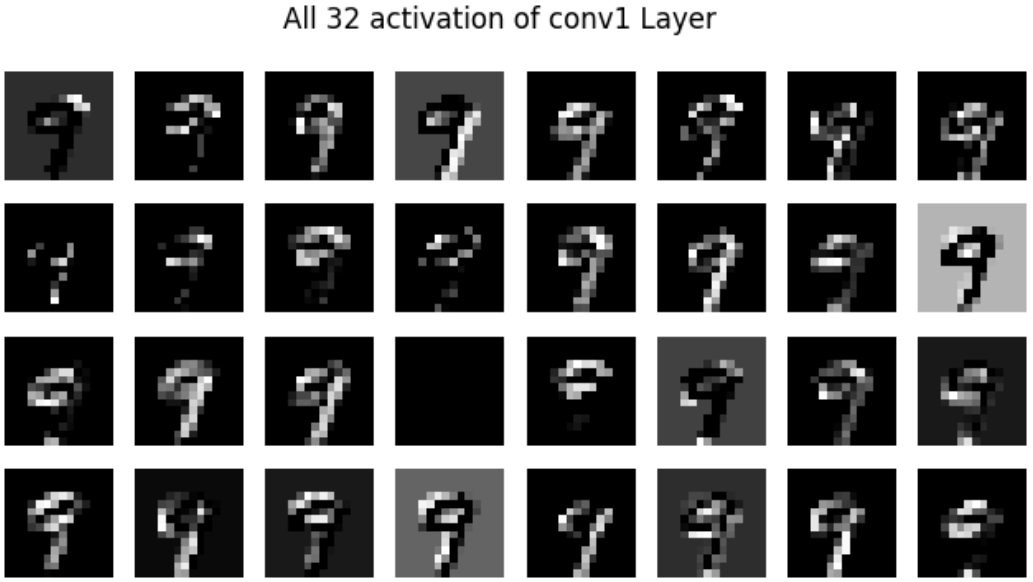
- Conv1 filters (learned from raw pixels) are simpler and capture basic patterns like edges, gradients, and simple textures.
- Conv2 filters operate on the feature maps from Conv1. They learn to combine the basic patterns from the first layer into more complex and abstract features. As a result, they often appear more intricate and less directly interpretable than Conv1 filters

2.3. VISUALIZING THE ACTIVATIONS

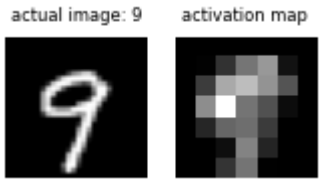
- VISUALIZING CONV1 LAYER ACTIVATION
    - Concatenated activation output



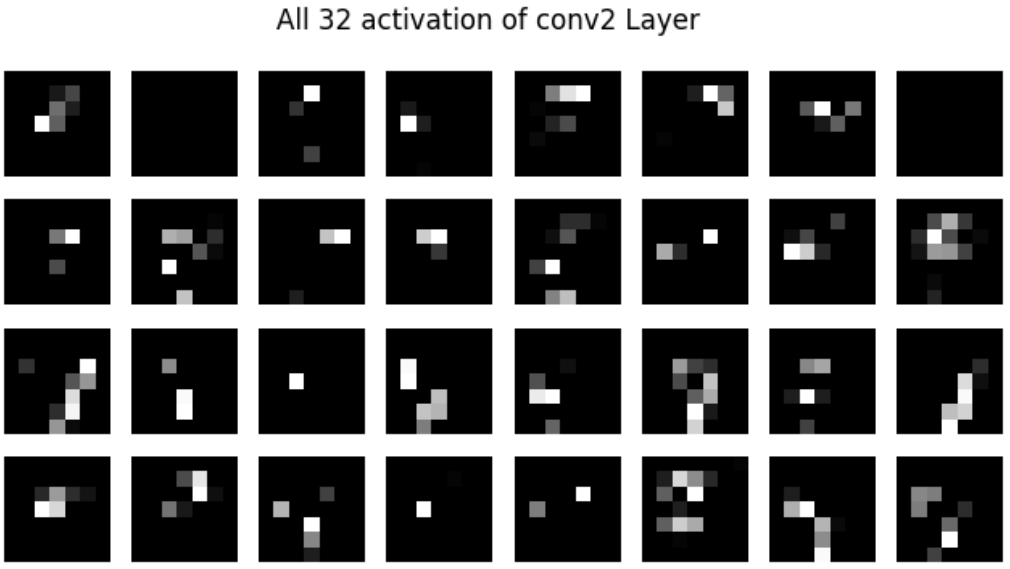
- All 32 activation output



- VISUALIZING CONV2 LAYER ACTIVATION
  - Concatenated activation output



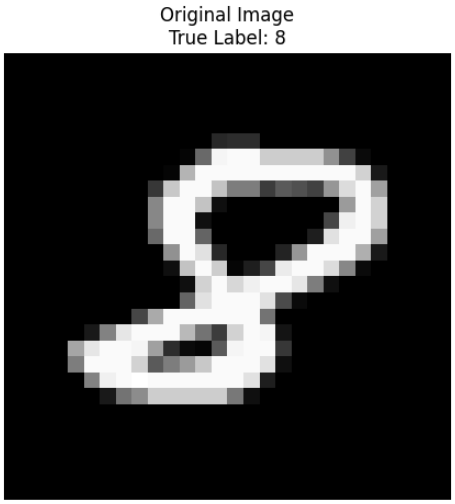
  - All 32 activation output

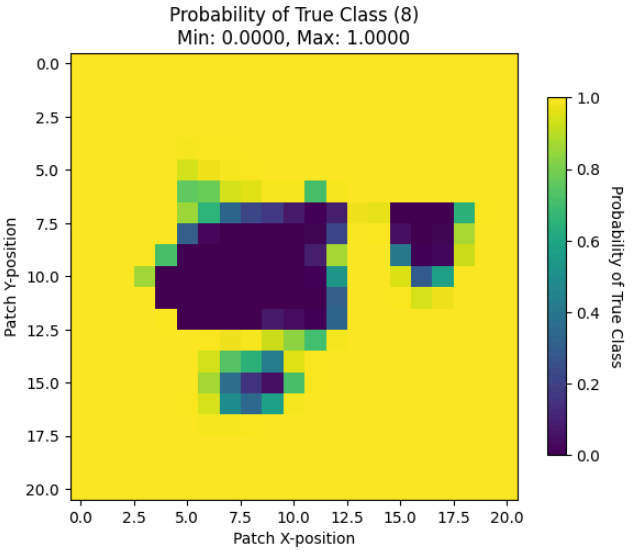


OBSERVATION FOR 2.3.

- As we go deeper into the network, the activations become more abstract and less visually interpretable.
- The Conv1 activations highlight simple features like edges and corners of the digit.
- The Conv2 activations are responses to more complex patterns built from the first layer's features. The spatial resolution (size of the feature map) is also smaller, meaning the features are more compressed.

2.4. OCCLUDING PARTS OF THE IMAGE





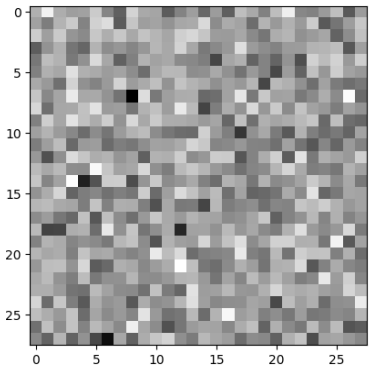
OBSERVATION FOR 2.4.

The occlusion experiment demonstrates that the model's confidence in the true class drops significantly only when the occluding patch covers the actual strokes of the digit '8'. This shows the network has learned to focus on the semantically important features of the digit itself, rather than the background. Therefore, the experiment confirms that the model's learning is meaningful.

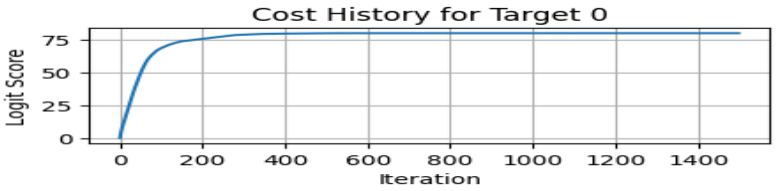
ADVERSARIAL EXAMPLES

3.1. NON TARGETED ATTACK

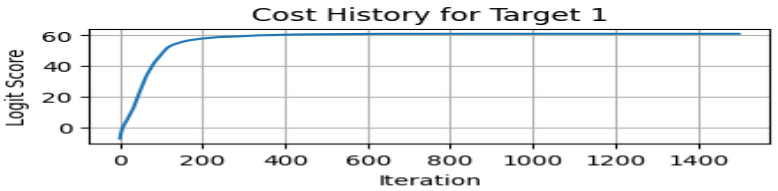
RANDOM NOISE USED AS INPUT:



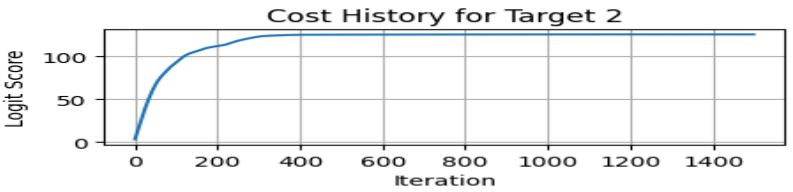
Target: 0
Final Prob: 1.0000



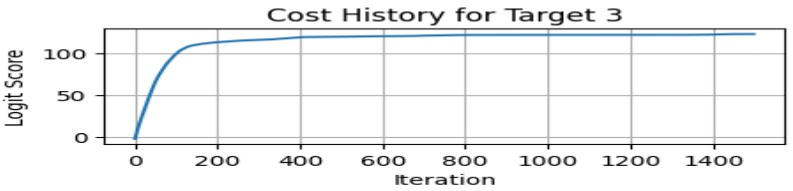
Target: 1
Final Prob: 1.0000



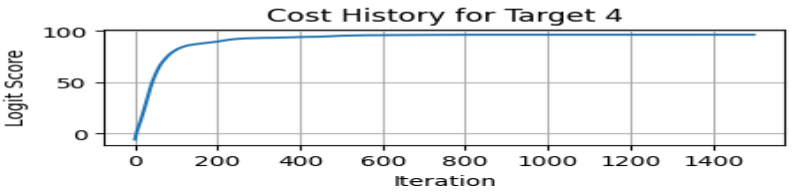
Target: 2
Final Prob: 1.0000



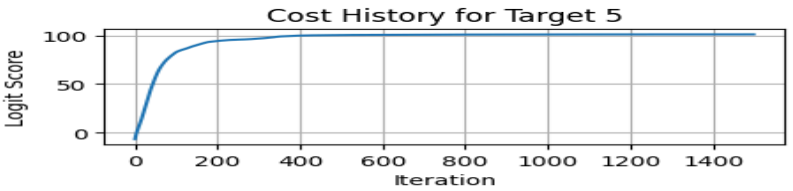
Target: 3
Final Prob: 1.0000



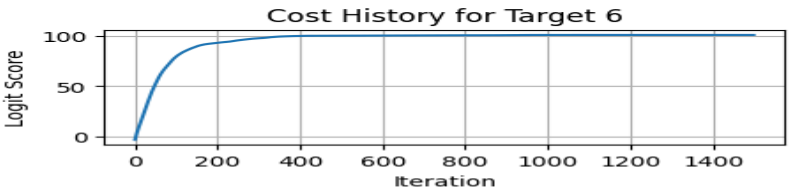
Target: 4
Final Prob: 1.0000



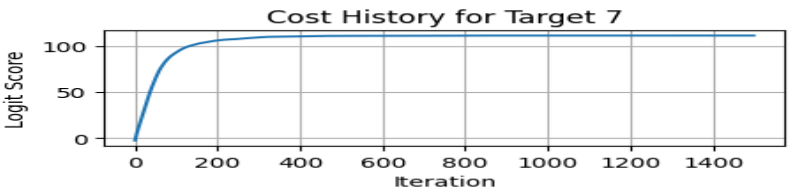
Target: 5
Final Prob: 1.0000



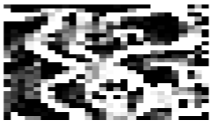
Target: 6
Final Prob: 1.0000

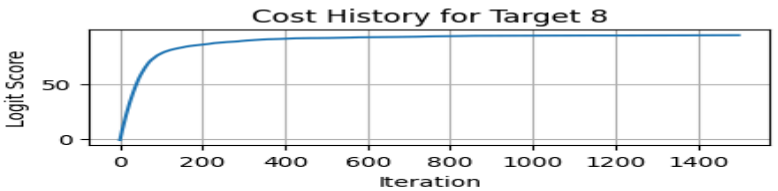


Target: 7
Final Prob: 1.0000

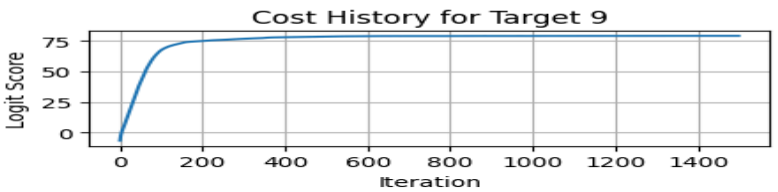


Target: 8
Final Prob: 1.0000





Target: 9
Final Prob: 1.0000

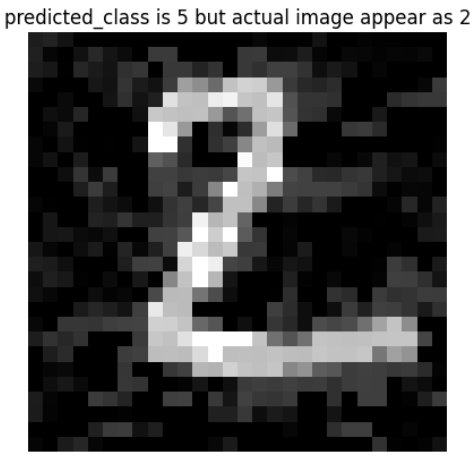


3.1.2 Yes , The model is predicting the target_class with very high confidence(as PROBABILITY is 1)

3.1.3 No, they don't look like numbers. They look like noisy patterns. This is because the model is choosing the most efficient pattern of pixels that trigger the target_class. Gradient ascent finds the perfect , minimalist combination of the features that represent target_class.This "perfect" pattern doesn't need to form a coherent shape that a human would recognize; it just needs to be an optimal trigger for the network's learned logic.

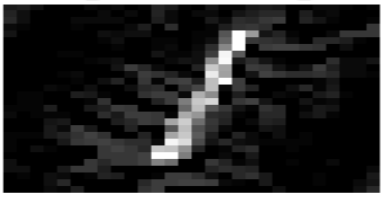
3.1.4 The cost function is increasing. This is expected because we are using gradient ASCENT to maximize the logit value for the target class. The plot confirms this upward trend.

3.2. TARGETED ATTACK

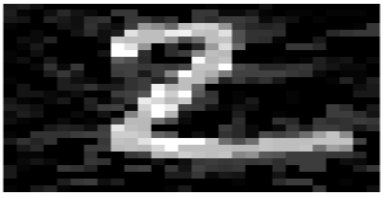


predicted_class-5 & actual_image-0

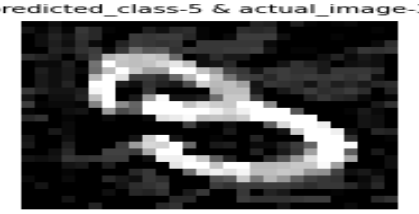
predicted_class-5 & actual_image-1



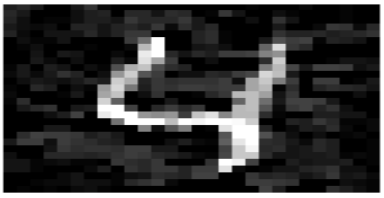
predicted_class-5 & actual_image-2



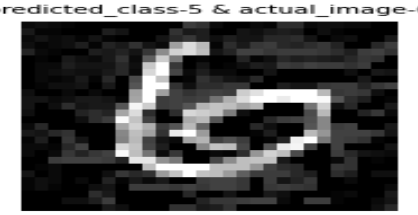
predicted_class-5 & actual_image-3



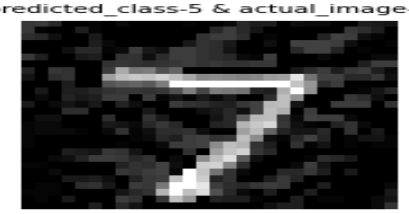
predicted_class-5 & actual_image-4



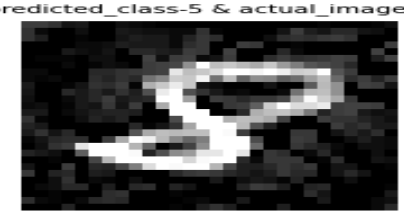
predicted_class-5 & actual_image-6



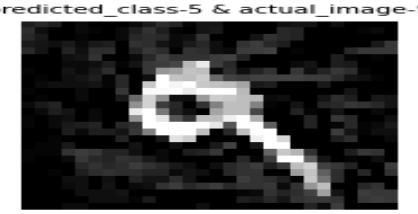
predicted_class-5 & actual_image-7



predicted_class-5 & actual_image-8



predicted_class-5 & actual_image-9



3.2.1 AS SHOWN ABOVE, THE GENERATED IMAGE CLEARLY RESEMBLES THE DIGITS. DUE TO MSE TERM IN THE COST FUNCTION, THE OPTIMIZATION PROCESS FORCES THE IMAGE TO BE VISUALLY SIMILAR TO THE TARGET IMAGE. BUT IT CLASSIFIES IT AS SOME OTHER NUMBER (TARGET_CLASS) BECAUSE OF LOGITS[TARGET_CLASS] IN LOSS FUNCTION.