# CS671 Assignment-2

# GROUP - 8

Akash Karnatak (B19147)
Laishram Ponghtangamba (B19011)
Abhishek Mishra (B19231)

## Tasks based on different optimizers:

FCNN with 3 hidden layers consisting of 64 neurons each was trained on a subset of MNIST data using different optimizers. Observations of the experiment performed are given below.

### i)

| Optimizer | Epochs |
|---|---|
| SGD | 50 |
| Vanilla GD | 131 |
| NAG | 76 |
| RMSProp | 22 |
| Adam | 19 |

Table 1. *Number of epochs until convergence for different optimizers*

FCNN with adam optimizer took the least number of epochs to converge.

### ii)

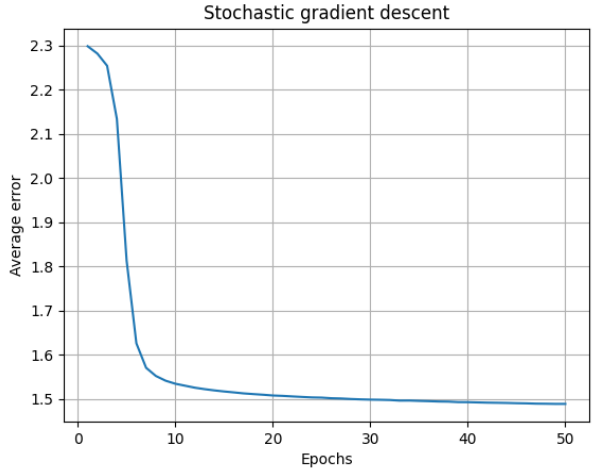



Fig. 1 *Average training error vs epochs for SGD*

For stochastic gradient descent algorithm a learning rate of 0.001 was used. It took a very long time (17 minutes) for the algorithm to converge because of the low learning rate and large number of update steps required.

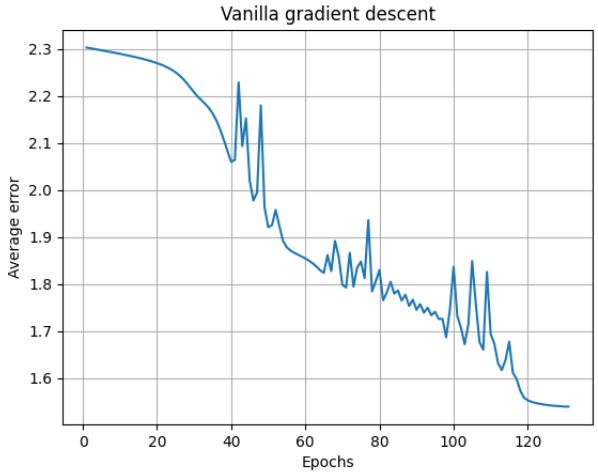



Fig. 2 *Average training error vs epochs for Vanilla GD*

For the vanilla gradient descent algorithm a learning rate of 1 was used. Since each epoch updates the parameters only once, a smaller learning rate would result in a very small update in parameters. Thus MSE does not change much after one epoch and as a result the model converges within one epoch with a very poor accuracy. Therefore a higher learning rate was used.

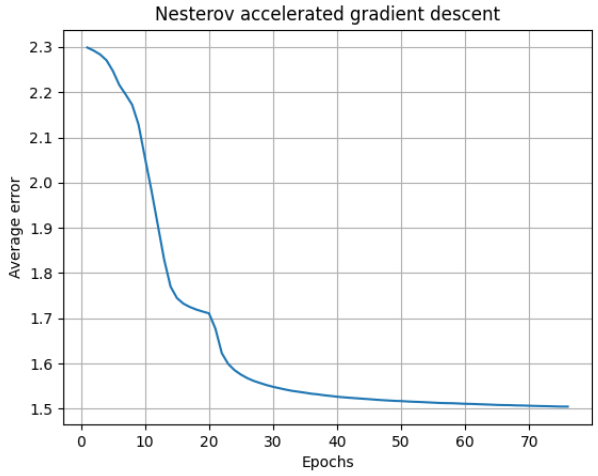



Fig. 3 *Average training error vs epochs for NAG*

For the NAG algorithm a learning rate of 0.001 and momentum of 0.9 was used. NAG took around 1 minute 30 seconds to converge which is roughly 11 times faster than SGD algorithm.

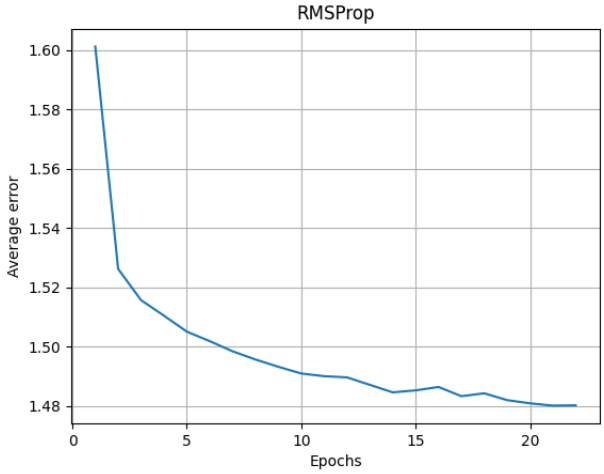



Fig. 4 *Average training error vs epochs for RMSProp*

RMSProp without momentum and with a learning rate of 0.001 and beta of 0.9 took around 19 seconds to converge.

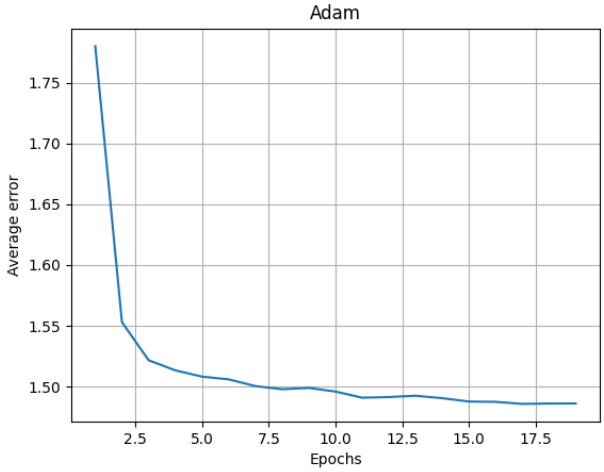



Fig. 5 *Average training error vs epochs for Adam*

For adam algorithm a learning rate of 0.001 with betas 0.9 and 0.999 was used. It took around 21 seconds for the algorithm to converge.

iii)

| Optimizer | Training accuracy | Validation accuracy |
|---|---|---|
| SGD | 97.45% | 95.47% |
| Vanilla GD | 92.73% | 92.70% |
| NAG | 96.14% | 94.91% |
| RMSProp | 98.09% | 97.71% |
| Adam | 97.48% | 96.73% |

Table 2. *Training and validation accuracy for different optimizers*

Among different optimizers RMSProp performed the best in terms of validation accuracy.

## iv)

The best architecture based on validation accuracy is RMSProp. Training accuracy for RMSProp is 98.09% and test accuracy is 97.58%.

| | | **Predicted class** | | | | |
|---|---|---|---|---|---|---|
| | | **1** | **3** | **4** | **8** | **9** |
| **Actual class** | **1** | 2249 | 16 | 4 | 6 | 2 |
| | **3** | 4 | 2240 | 1 | 19 | 13 |
| | **4** | 1 | 2 | 2247 | 2 | 25 |
| | **8** | 4 | 20 | 0 | 2244 | 9 |
| | **9** | 2 | 23 | 12 | 6 | 2234 |

Table 3. *Confusion matrix of RMSProp on training data*

| | | **Predicted class** | | | | |
|---|---|---|---|---|---|---|
| | | **1** | **3** | **4** | **8** | **9** |
| **Actual class** | **1** | 748 | 8 | 0 | 3 | 0 |
| | **3** | 1 | 742 | 1 | 11 | 4 |
| | **4** | 0 | 1 | 742 | 2 | 14 |
| | **8** | 1 | 13 | 3 | 736 | 6 |
| | **9** | 3 | 6 | 12 | 3 | 735 |

Table 4. *Confusion matrix of RMSProp on test data*

# Tasks based on autoencoder:

## 1-Layer AutoEncoder

We built a 1-layer autoencoder with the single hidden layer having 64 neurons. MSE was used as loss function and adam optimizer(with default parameters was used) was used.
The training was stopped when the error between successive epochs falls below 1e-4 .
We also tried 100,180 neurons and their accuracy was 97% and 99.96% respectively but their training time was relatively longer.

### RMSE-Error :

The MSE loss for training data was found to be 0.0051, while it was 0.0057 for the validation data.

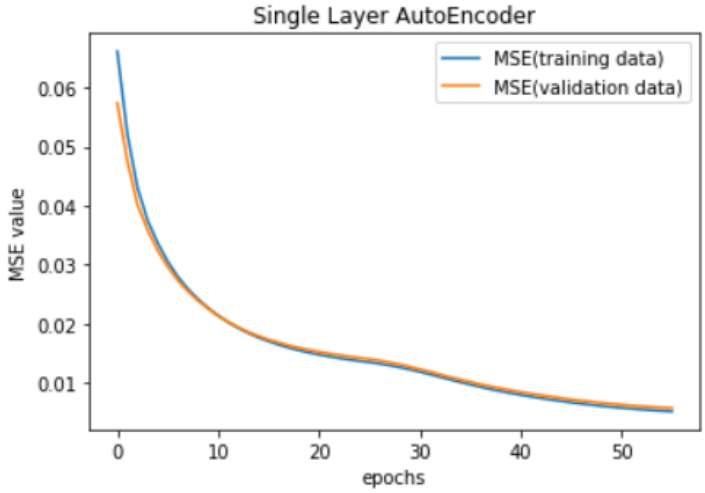



Fig. *Average training error vs epochs for Single Layer Autoencoder*

### Test Reconstruction Error :

Reconstruction error on the test data was found to be 0.005578 (MSE).

### Original Vs Reconstructed :

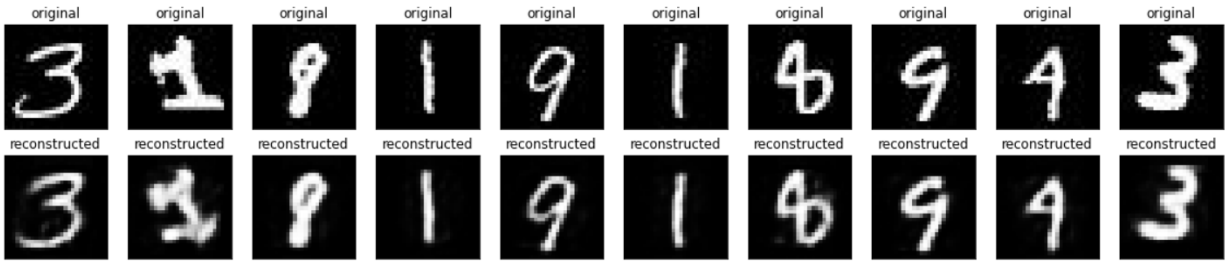



Fig. 5 *Original vs Reconstructed Images for Single layer autoencoder*

We can observe that even with using only 64 input features out of 784, the autoencoder was able to fairly detect the digits. So the model was able to detect the important features very well.

## Weight Visualization :

Here each neuron is learning some particular feature of the digit and it will get most fired when that particular digit is presented as the input.

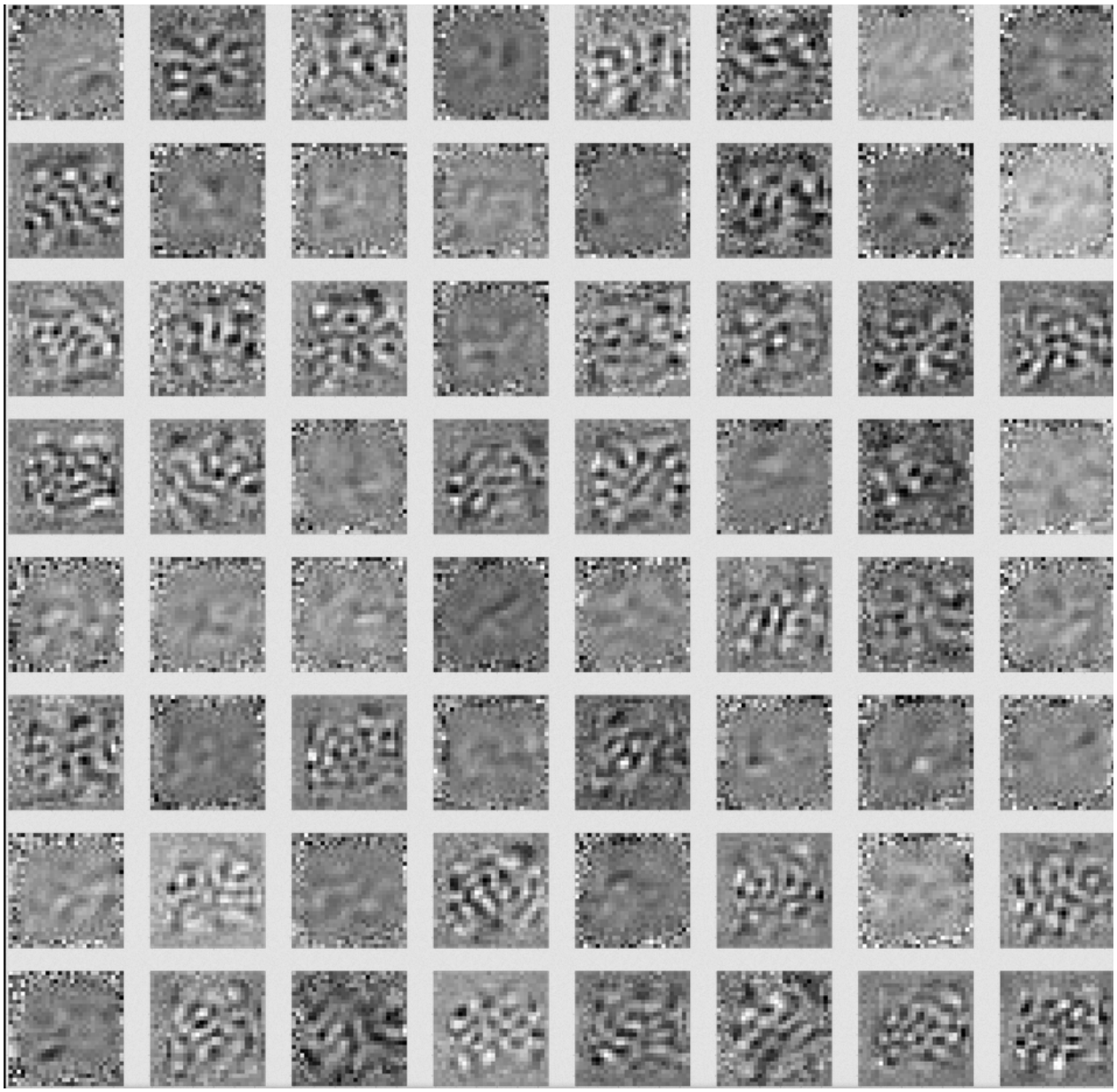


Fig. *Weight Visualization for Single Layer AutoEncoder*

## CLASSIFICATION USING ENCODED REPRESENTATION :

64 neurons were used in the hidden layer for the classification. The sparseCategoricalCrossentropy was used as the error function and adam was used as optimizer.
The training was stopped when the error between successive epochs fell below 1e-4 for training data.

## Accuracy on Training and Validation data :

The loss on training data was found to be 0.1676, accuracy on training data was found to be 94.40% while error on validation data was found to be  0.1628 and accuracy was found to be 94.68%.

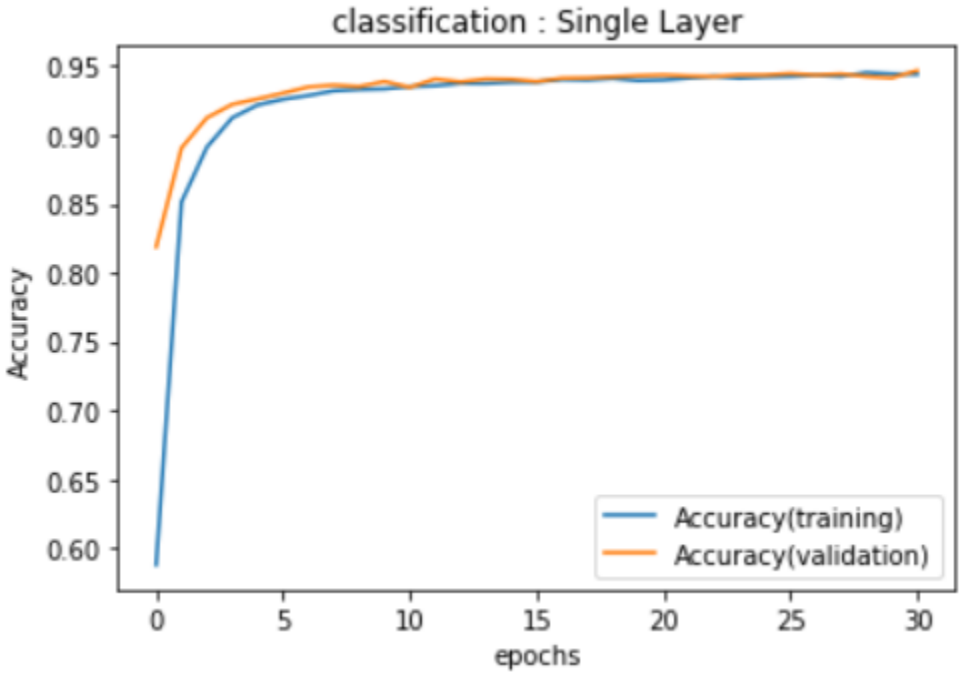



Fig. *Average training accuracy vs epochs for Single Layer Autoencoder*

## Accuracy on Validation Data and Test Data :

Loss on Validation data was found to be 0.1628 while it was found to be 0.1635 for test dataset.

Accuracy on Validation data was found to be 94.68% and was found to be 94.387% for the test dataset.

## CONFUSION MATRIX:

We can observe that most of the values lying across the diagonal of the matrix which signifies that our model is giving high True positive. This is indeed confirmed by 94.68% accuracy that we are getting.

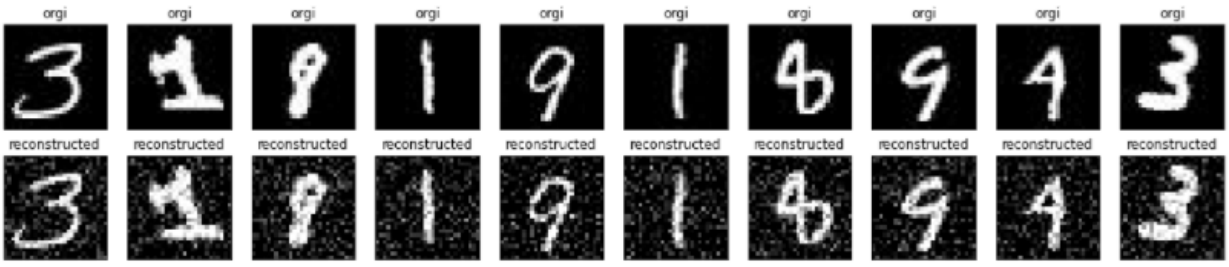
```
--------------------TRAIN CONFUSION MATRIX----------------------

[[2237   15    1   19    5]
 [  19 2153    5   67   33]
 [  17    6 2113   15  126]
 [  48   83   11 2110   25]
 [   6   51   54   22 2144]]

--------------------VAL CONFUSION MATRIX----------------------

[[746   2   1   8   2]
 [  8 712   6  23  10]
 [  5   0 723   3  28]
 [ 11  29  10 701   8]
 [  4  14  22  11 708]]

--------------------TEST CONFUSION MATRIX----------------------

[[745   5   0   9   0]
 [  7 711   6  28   7]
 [  2   2 706   5  44]
 [ 15  33   8 691  12]
 [  7  11  15  12 714]]
```

# DENOISING AUTOENCODER

A random noise with 20/40% probability was added on the go during training to the input data . Bottleneck layer contains 64 neurons and the sigmoid activation function was used in all the layers.
For building Autoencoder MSE was used as the error function.

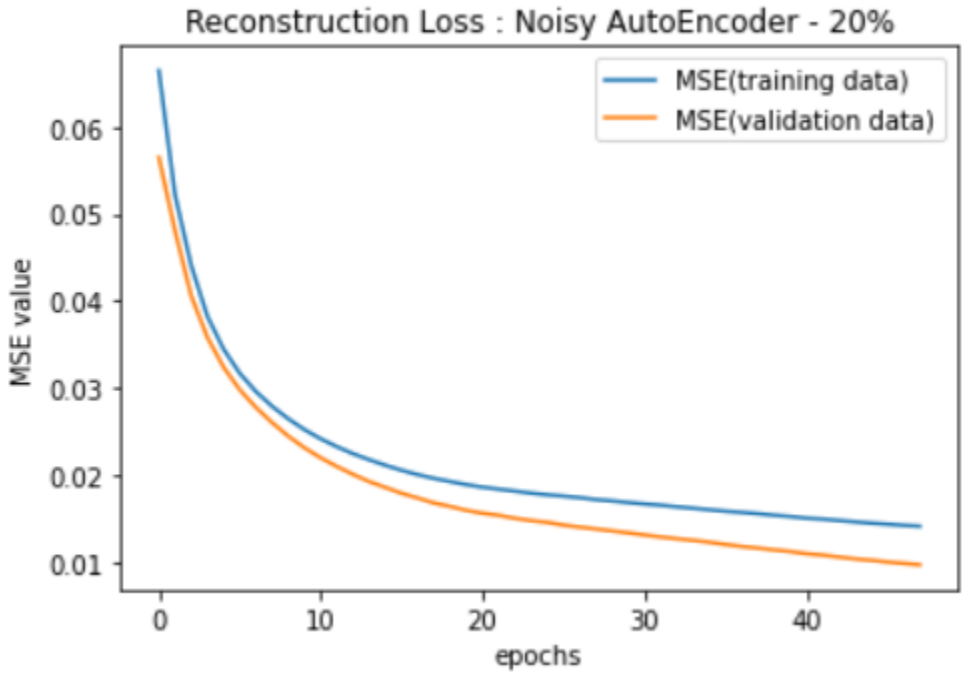
## 20% Denoising Autoencoder:

After adding 20% noise to the input data , the noised data looks like follows:



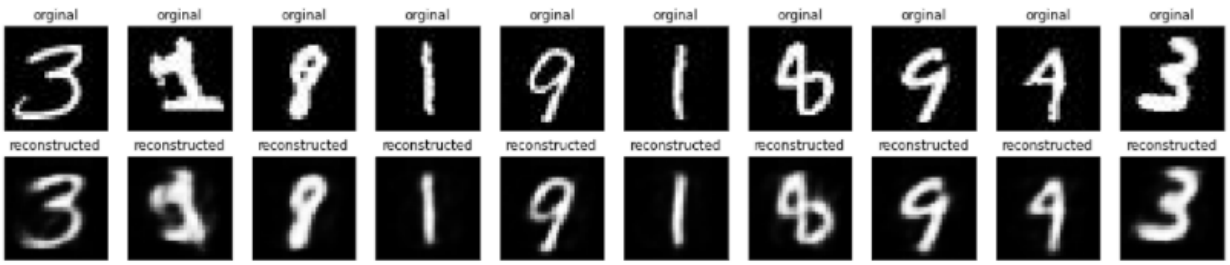
## MSE Error on Training and Validation data :

The MSE loss on training data was found to be 0.0141, while MSE error on validation data was found to be 0.0097.




## Original Vs Reconstructed :

We can observe that even when the input data was noisy , the model was fairly able to detect the input features even with using only 64 of the features.

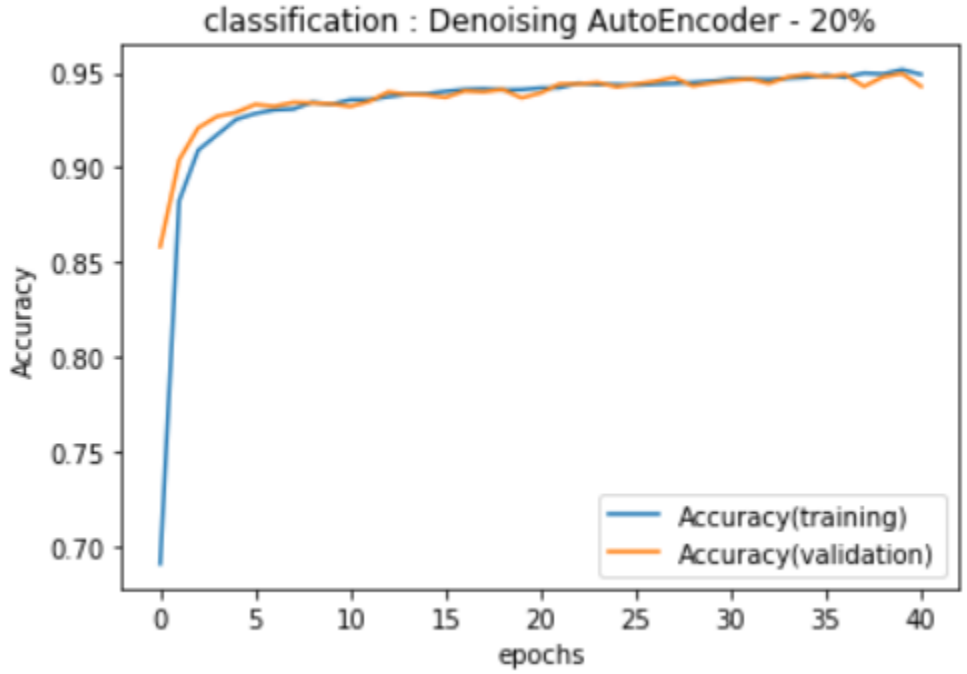



## CLASSIFICATION WITH DENOISING(20%) AUTOENCODER :

A single layer NN with 64 neurons was built. Adam was used as optimizer and sparse_Categorical_Crossentropy was used as the loss function.

## Accuracy on Training and Validation data :

The loss on training data was found to be 0.1486, accuracy on training data was found to be 94.91% while error on validation data was found to be  0.1606 and accuracy was found to be 94.28%.




## Accuracy on Validation Data and Test Data :

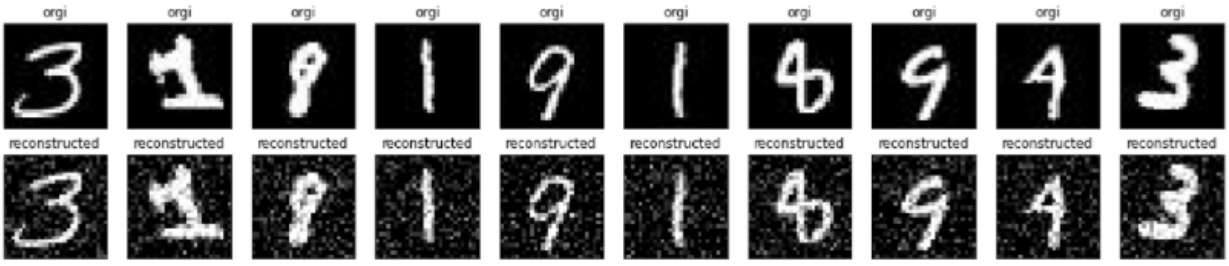
Loss on Validation data was found to be 0.1606 while it was found to be 0.1559 for the test dataset.

Accuracy on Validation data was found to be 94.28% and was found to be 94.60% for the test dataset.

## Weight Visualization :

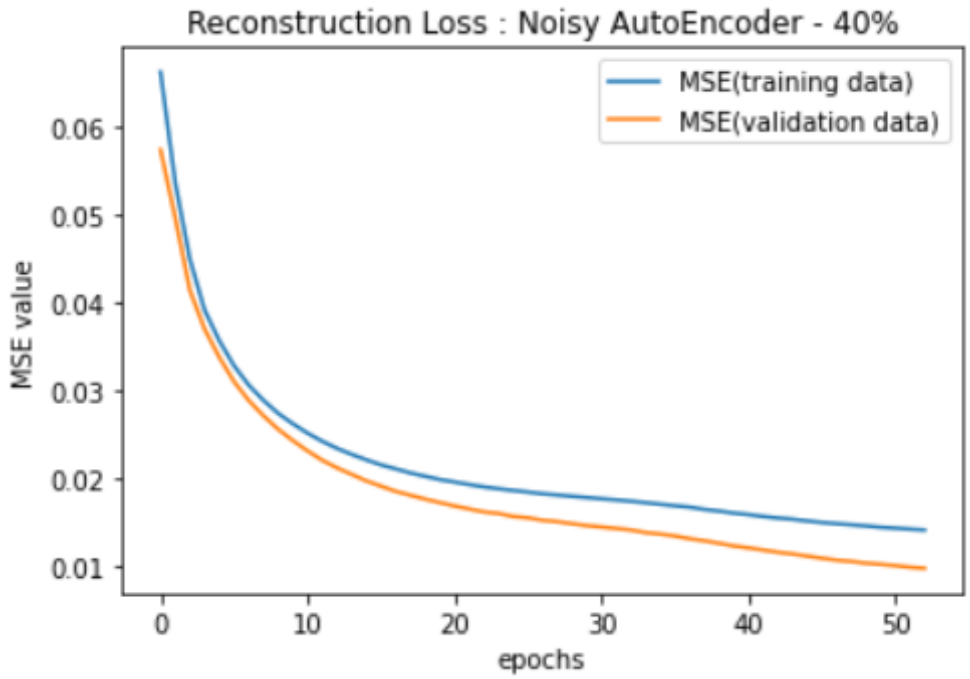
## 40% Denoising Autoencoder:

After adding 40% noise to the input data , the noised data looks like follows:



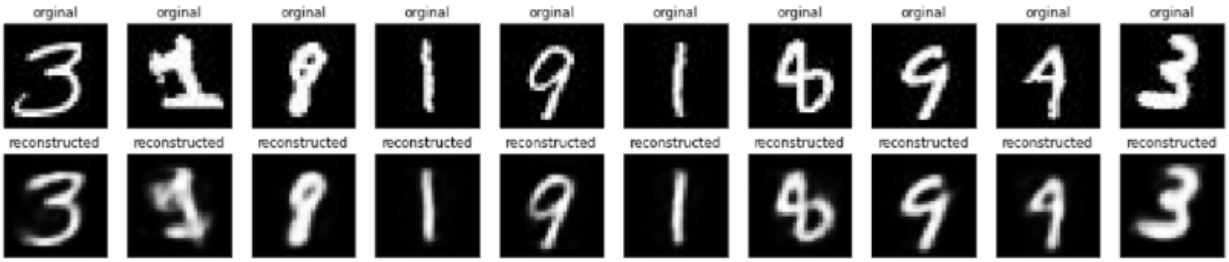

## MSE Error on Training and Validation data :

The MSE loss on training data was found to be 0.0142, while MSE error on validation data was found to be 0.0098.



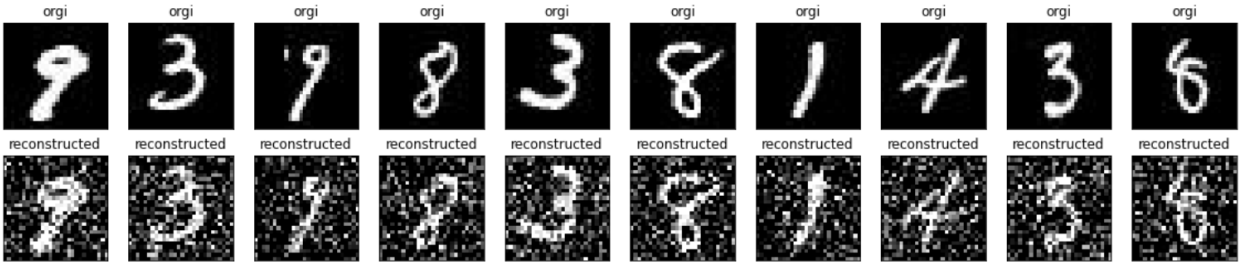

## Original Vs Reconstructed :

We can observe that even when the input data was noisy , the model was fairly able to detect the input features even with using only 64 of the features.



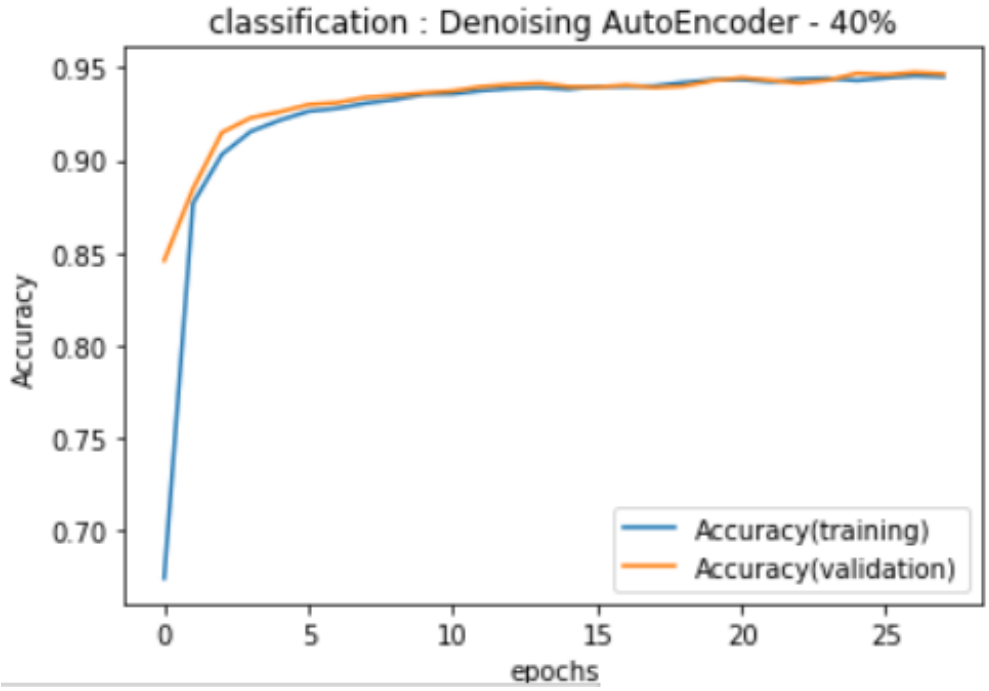

## CLASSIFICATION WITH DENOISING(40%) AUTOENCODER :

A single layer NN with 64 neurons was built. Adam was used as optimizer and sparse_Categorical_Crossentropy was used as the loss function.



## Accuracy on Training and Validation data :

The loss on training data was found to be 0.1682, accuracy on training data was found to be 94.49% while error on validation data was found to be  0.1611 and accuracy was found to be 94.65%.




## Accuracy on Validation Data and Test Data :

Loss on Validation data was found to be 0.1611 while it was found to be 0.1679 for the test dataset.

Accuracy on Validation data was found to be 94.65% and was found to be 94.73% for the test dataset.

## Comparison With 1st question accuracies:

Here the accuracy has been reduced by 2%(approx) as compared to accuracy when the classifier is built upon original data. But this error is understandable as we are using only 64 of the features out of Very large feature set that our image contains.
But when we built it on 180 neurons, accuracy of 99.96% was achieved on training dataset, which signifies our model learnt all important features of the image.

# 3 Hidden-Layer AutoEncoder :

The learning rate = 1e-3
For adam optimizer ,  betas=(0.9, 0.999), eps=1e-8

**i.**The convergence criteria is the difference between average error of successive epochs falling below a threshold 1e-4.

The average reconstruction error for training data and validation data for each of the architectures are:

| **Architecture** | **Average reconstruction error** |
|---|---|
| The architecture of the encoder is:<br>1st hidden layer has 80 neurons<br>2nd hidden layer has 50 neurons<br>3rd hidden layer has 80 neurons | After 35 Epochs ,<br>The mean square error of training is 0.01090<br>The mean square error of validation is 0.01141 |
| The architecture of the encoder is:<br>1st hidden layer has 40 neurons<br>2nd hidden layer has 32 neurons<br>3rd hidden layer has 40 neurons | After 41 Epochs,<br>The mean square error of training is 0.01534<br>The mean square error of validation is 0.01584 |
| The architecture of the encoder is:<br>1st hidden layer has 400 neurons<br>2nd hidden layer has 100 neurons<br>3rd hidden layer has 400 neurons | After 31 Epochs ,<br>The mean square error of training is 0.00568<br>The mean square error of validation is 0.00646 |
| The architecture of the encoder is:<br>1st hidden layer has 600 neurons<br>2nd hidden layer has 100 neurons<br>3rd hidden layer has 600 neurons | After 30 Epochs,<br>The mean square error of training is 0.00525<br>The mean square error of validation is 0.00622 |
| The architecture of the encoder is:<br>1st hidden layer has 450 neurons<br>2nd hidden layer has 270 neurons<br>3rd hidden layer has 450 neurons | After 27 Epochs,<br>The mean square error of training is 0.00496<br>The mean square error of validation is 0.00578 |

**ii.**
The convergence criteria is the difference between average error of successive epochs falling below a threshold 1e-4.

The best architecture among the tested architectures is selected by comparing the accuracies given by the same classifier which is trained on a fixed amount of epochs (10) for all architecture.

The architecture of the classifier to check the encoder efficiency is:

| **Architecture** | **Accuracy** |
|---|---|

| | |
|---|---|
| 1st hidden layer has 40 neurons<br>2nd hidden layer has 32 neurons<br>3rd hidden layer has 40 neurons | After 10 epochs ,<br>The accuracy of training of classifier is 92.59%<br>The accuracy of validation of classifier is 93.28% |
| 1st hidden layer has 80 neurons<br>2nd hidden layer has 50 neurons<br>3rd hidden layer has 80 neurons | After 10 epochs ,<br>The accuracy of training of classifier is 92.91%<br>The accuracy of validation of classifier is 93.20% |
| 1st hidden layer has 400 neurons<br>2nd hidden layer has 100 neurons<br>3rd hidden layer has 400 neurons | After 4 Epochs,<br>The accuracy of training of classifier is 92.99%<br>The accuracy of validation of classifier is 93.39% |
| 1st hidden layer has 600 neurons<br>2nd hidden layer has 100 neurons<br>3rd hidden layer has 600 neurons | After 4 Epochs,<br>The accuracy of training of classifier is 93.05%<br>The accuracy of validation of classifier is 93.39% |
| 1st hidden layer has 450 neurons<br>2nd hidden layer has 270 neurons<br>3rd hidden layer has 450 neurons | After 4 Epochs,<br>The accuracy of training of classifier is 94.38%<br>The accuracy of validation of classifier is 94.52% |

First , we fix the number of 1st and 3rd hidden layers' nodes (600) and we vary the number of 2nd hidden layer's nodes. The best accuracy is found when the 2nd hidden layer has 270 nodes. Then we fix the number of 2nd hidden layer's nodes and we vary the number of 1st and 3rd hidden layers's nodes.The best accuracy is found when the 1st and 3rd hidden layers have 450 nodes each.

So, , the architecture :
1st hidden layer has 450 neurons
2nd hidden layer has 270 neurons
The 3rd hidden layer has 450 neurons , and is chosen as the best architecture.

**Comparison of architectures with encoder and without encoder:**
We get similar accuracy in both architectures.

The test reconstruction error for the chosen best architectures is 0.00594.

**Inferences:**
When we fix the 2nd hidden layer to 270 nodes and vary the number in 1st and 3rd hidden layers , the accuracy drops sharply when the number of nodes decreases below 450. But the accuracy remains the same or somewhat decreases as the number of nodes increases above 450.

iii. The plots of average training reconstruction error (y-axis) vs. epochs (x-axis) for the best architecture for encoder with 3 hidden layers architecture.

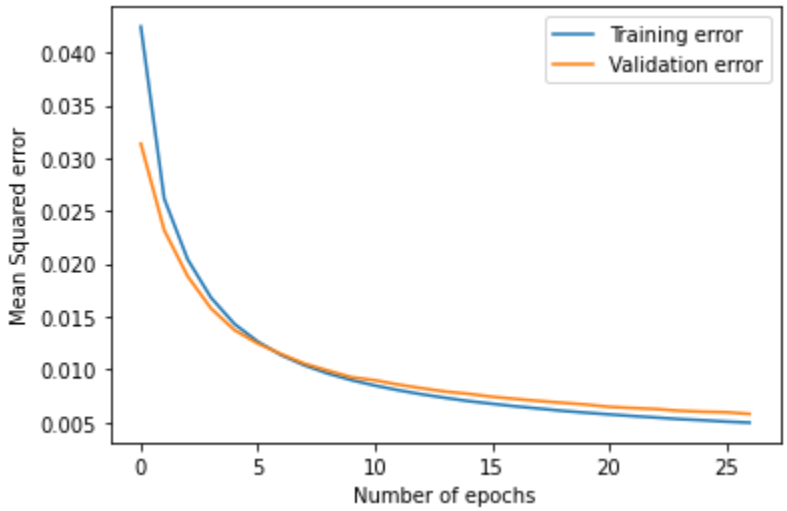



*Fig: Mean Squared Error vs Number of Epochs of reconstruction error of Training and Validation Data.*

iv. Reconstruction images of validation data using the best 3 hidden layers architecture.

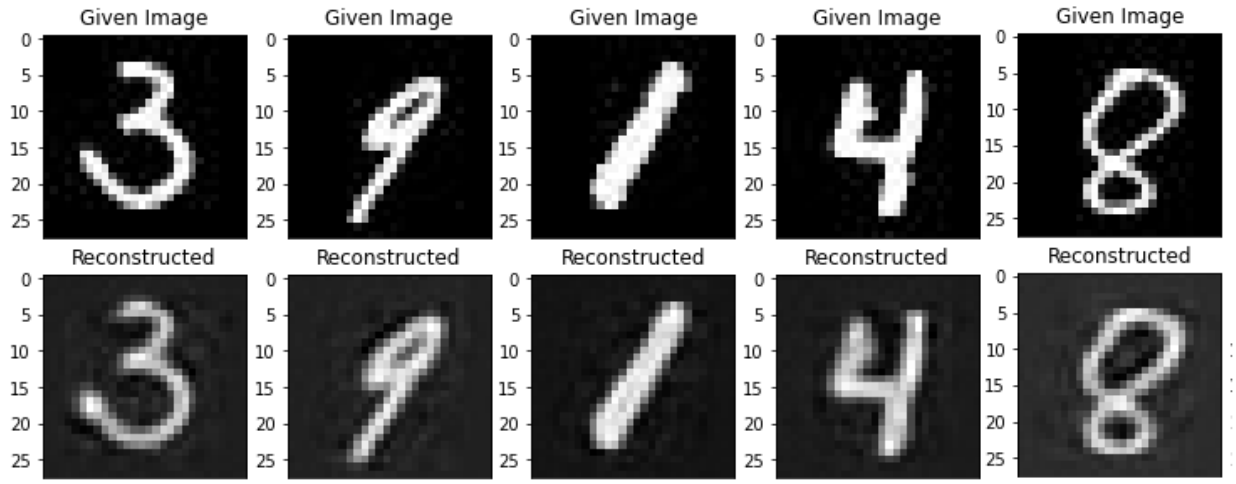



*Fig: Original and Reconstructed images of validation data using best encoder.*

Reconstruction images of training data using the best 3 hidden layers architecture.

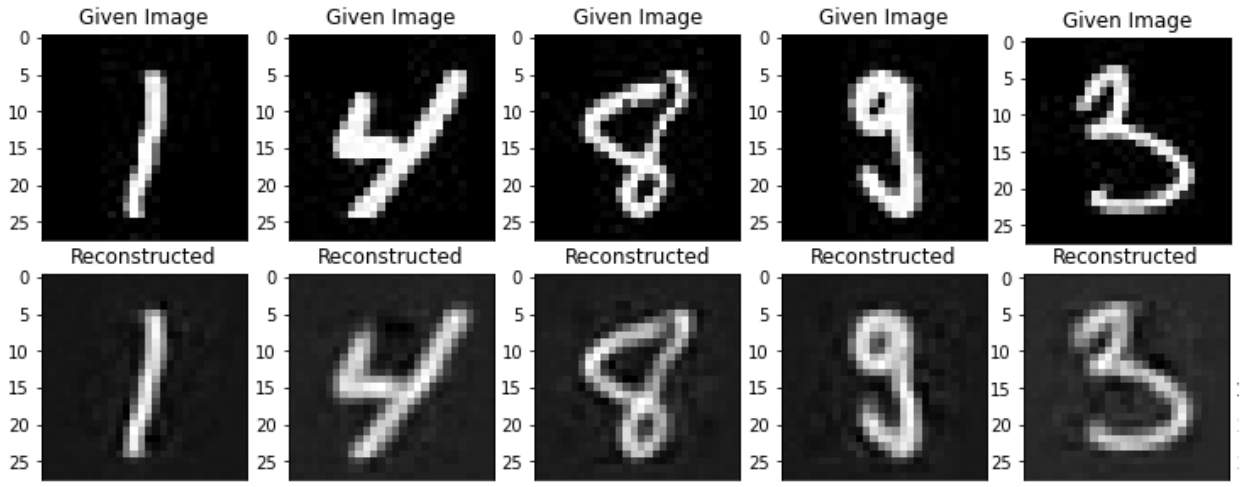



*Fig: Original and Reconstructed images of training data using best encoder.*

**vi.** The classification of MNIST images using the output of the best 3 hidden layers encoders as input to the classifier. The classifier is then trained with different architectures and the best architecture is chosen.

The best classifier architecture:
1st hidden layer = 64
2nd hidden layer = 64
3rd hidden layer = 64

By reaching the convergence criteria,
Epochs=27
The accuracy of training of classifier is 97.19%
The accuracy of validation of classifier is 97.10%
Testing accuracy=96.71%

The confusion matrix of the best classifier:

| | | **Predicted class** | | | | |
|---|---|---|---|---|---|---|
| | | **1** | **3** | **4** | **8** | **9** |
| **Actual class** | **1** | 2205 | 8 | 7 | 55 | 2 |
| | **3** | 0 | 2156 | 1 | 57 | 63 |
| | **4** | 1 | 1 | 2004 | 44 | 227 |
| | **8** | 1 | 9 | 3 | 2258 | 6 |
| | **9** | 0 | 7 | 15 | 22 | 2233 |

*Table: Confusion Matrix for the best classifier using encoder.*