Report File
CSL4020
Assignment -1
Name: Ritam Sharma (B20BB0030)

D=05,M=04,Y=02

A=0,B=3,C=0

ABC=030

FIRST=RITAM

LAST= SHARMA

PROG=BTECH

DATASET: FASHIONMNIST

WEIGHT INITIAL: Xavier(M=04)

data aug: rotate by 10 degrees and gaussian noise. (D=05)

Pooling: Avg pooling (M=04)

Target classification : (D+M+Y=5+4+2=11) your target 5 classes will be 1,3,5,7,9.

Question 1

Problem Statement

The goal is to train a CNN to classify 1,3,5,7,9. The data is augmented with GaussianNoise and random flipping

Hyperparameter tuning

1. **Dataset consideration**

   The CNN model was trained firstly without augmented data and then secondly with the augmented data added, every other parameter is kept the same

   Hyperparameters:

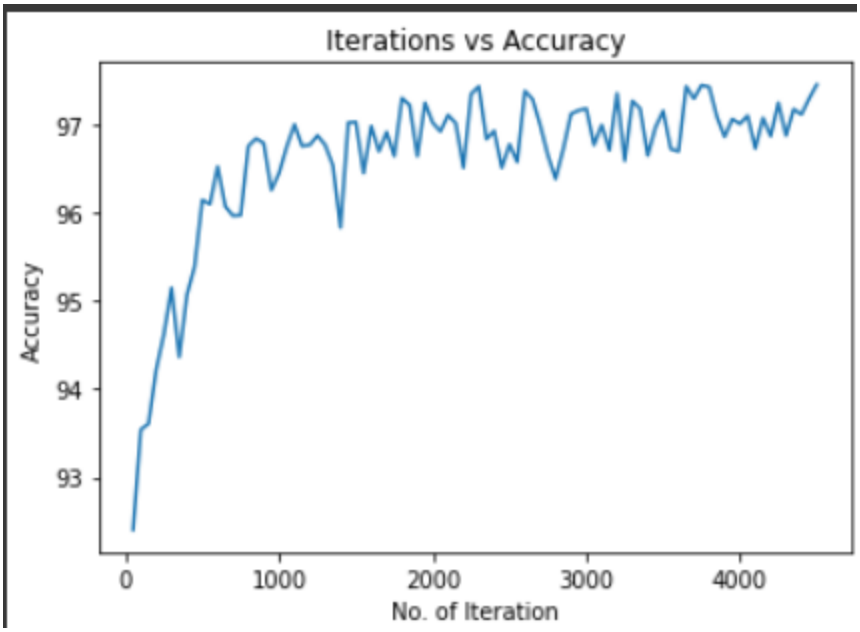
epoch	5
batch	100
optimizer	adam
learning rate	.001

▼ run 1

Architecture

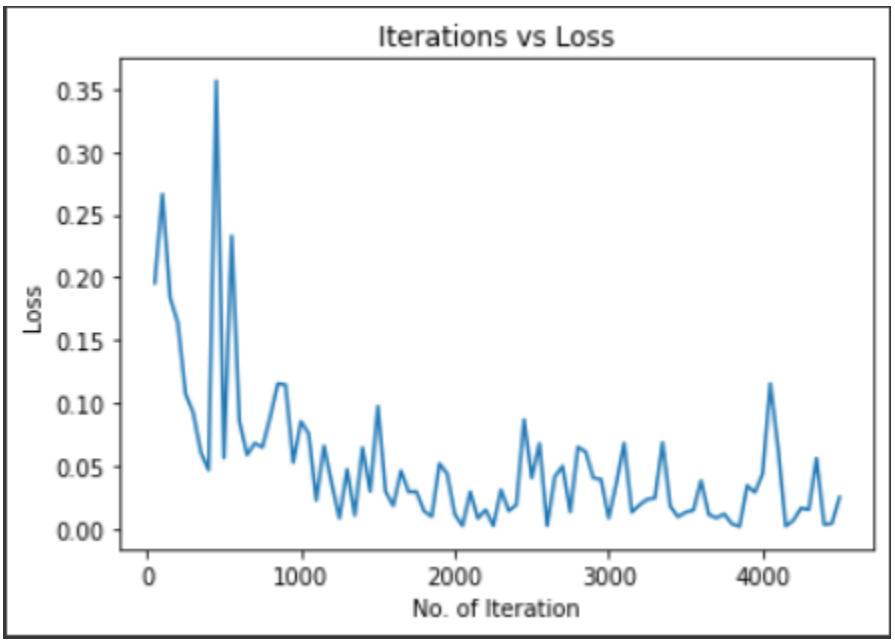
```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1          [-1, 10, 26, 26]             100
       BatchNorm2d-2          [-1, 10, 26, 26]              20
              ReLU-3          [-1, 10, 26, 26]               0
         MaxPool2d-4          [-1, 10, 13, 13]               0
            Conv2d-5          [-1, 20, 11, 11]           1,820
       BatchNorm2d-6          [-1, 20, 11, 11]              40
              ReLU-7          [-1, 20, 11, 11]               0
            Conv2d-8            [-1, 40, 9, 9]           7,240
       BatchNorm2d-9            [-1, 40, 9, 9]              80
             ReLU-10            [-1, 40, 9, 9]               0
           Conv2d-11            [-1, 80, 7, 7]          28,880
      BatchNorm2d-12            [-1, 80, 7, 7]             160
             ReLU-13            [-1, 80, 7, 7]               0
        MaxPool2d-14            [-1, 80, 3, 3]               0
           Linear-15                 [-1, 265]         191,065
        Dropout2d-16                 [-1, 265]               0
           Linear-17                   [-1, 5]           1,330
================================================================
Total params: 230,735
Trainable params: 230,735
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.00
Forward/backward pass size (MB): 0.40
Params size (MB): 0.88
Estimated Total Size (MB): 1.28
----------------------------------------------------------------
```


training loss
testing loss
0.18
0.16
0.14
0.12
0.10
0.08
0.06
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0


Iterations vs Accuracy
Accuracy
97
96
95
94
93
0
1000
2000
3000
4000
No. of Iteration

```
Classification report for CNN :
              precision    recall  f1-score   support

           0       0.99      0.98      0.98    270000
           1       0.98      0.98      0.98    270000
           2       0.97      0.97      0.97    270000
           3       0.94      0.95      0.94    270000
           4       0.96      0.95      0.96    270000

    accuracy                           0.97   1350000
   macro avg       0.97      0.97      0.97   1350000
weighted avg       0.97      0.97      0.97   1350000
```

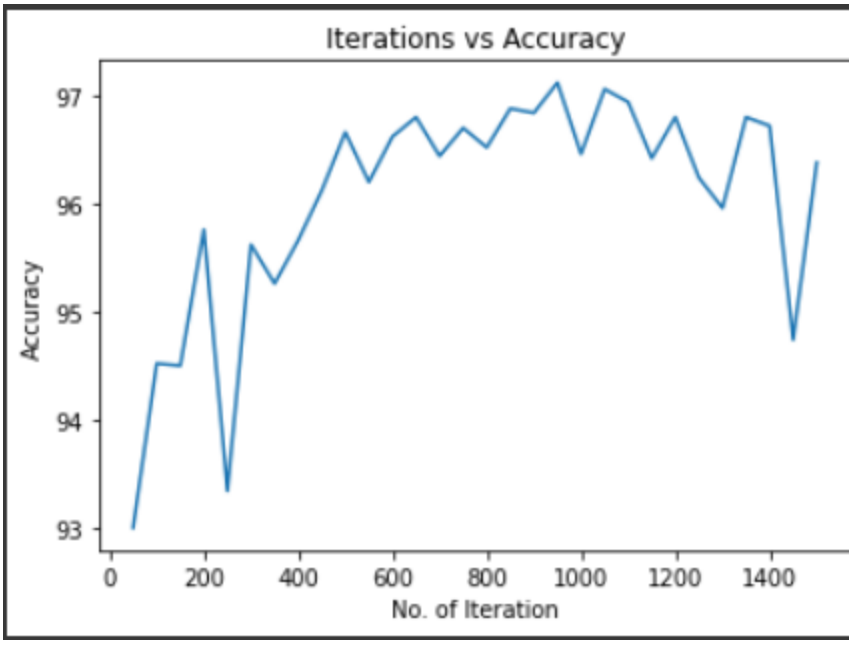
```
Accuracy of Trouser: 98.50%
Accuracy of dress: 98.93%
Accuracy of sandle: 97.03%
Accuracy of sneaker: 97.73%
Accuracy of ankle boot: 94.63%
```

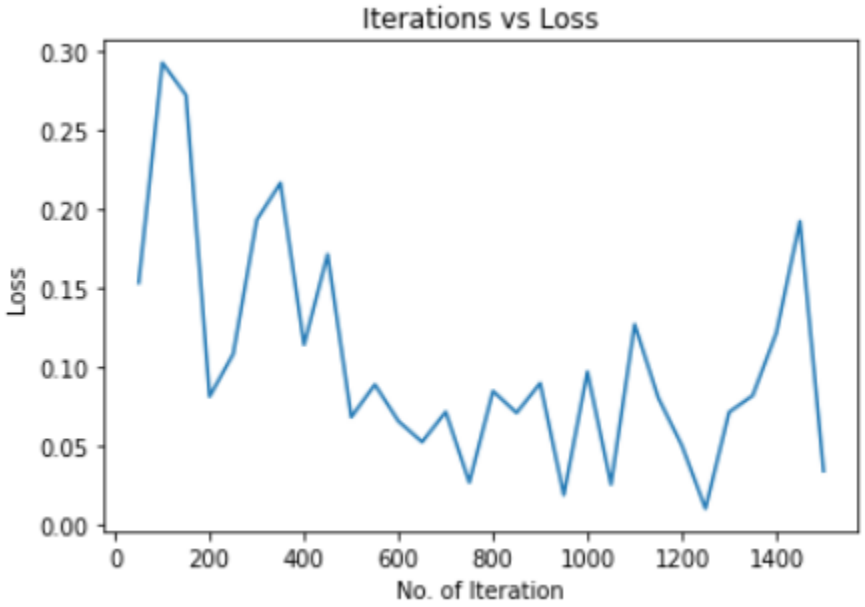


▼ run 2

without augmented data same as run 1 (accuracy decreases by 1 %)

```
Classification report for CNN :
              precision    recall  f1-score   support

           0       0.98      0.97      0.98     30000
           1       0.97      0.98      0.98     30000
           2       0.96      0.96      0.96     30000
           3       0.93      0.94      0.94     30000
           4       0.95      0.95      0.95     30000

    accuracy                           0.96    150000
   macro avg       0.96      0.96      0.96    150000
weighted avg       0.96      0.96      0.96    150000
```



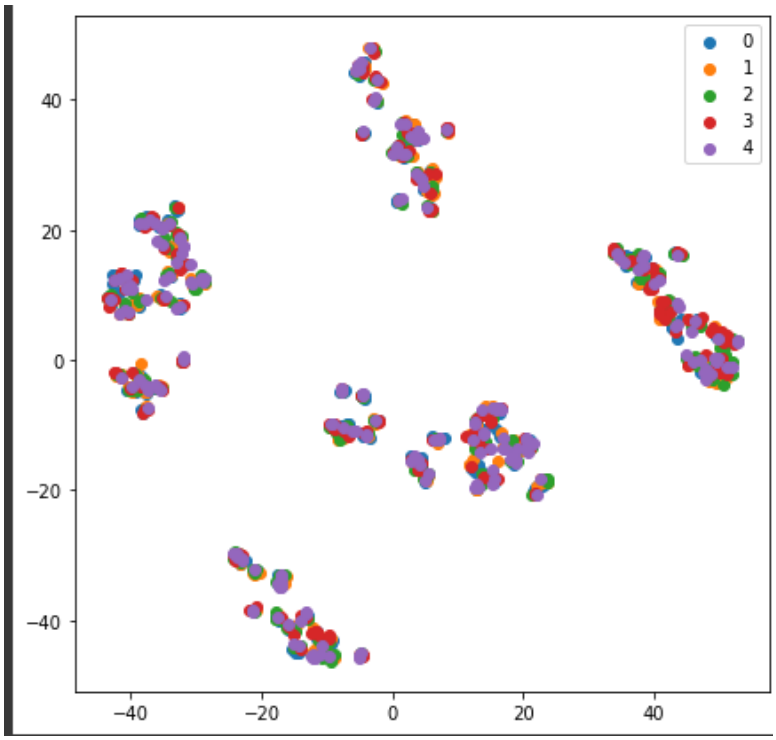


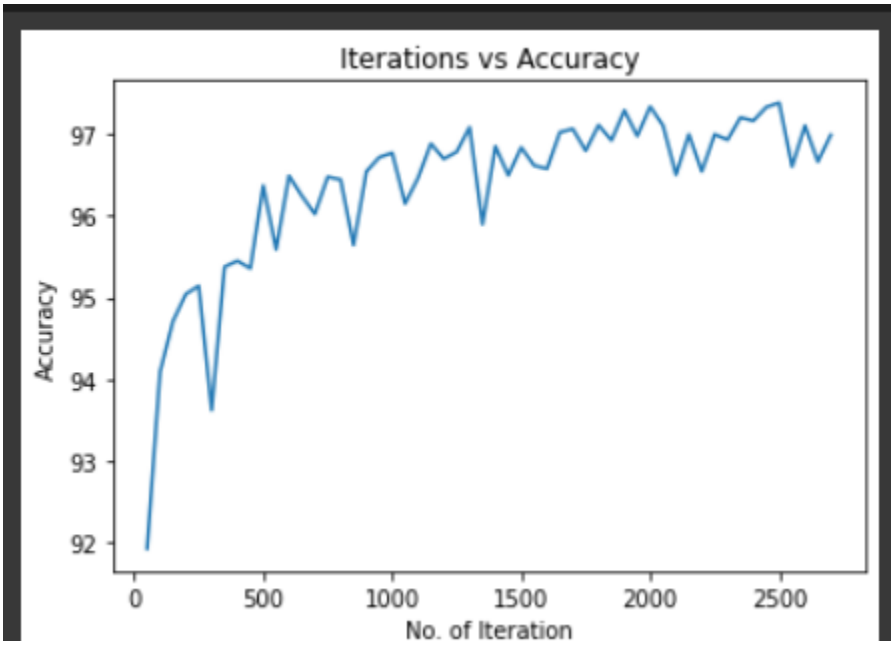
💡 The addition of augmented data has increased the accuracy by 1 %, therefore for further analysis augmented data is added

2. **epoch tuning**
   The CNN model was trained firstly with epochs 3,5,8,10 every other parameter is kept the same
   Hyperparameters:

batch	100
optimizer	adam
learning rate	.001

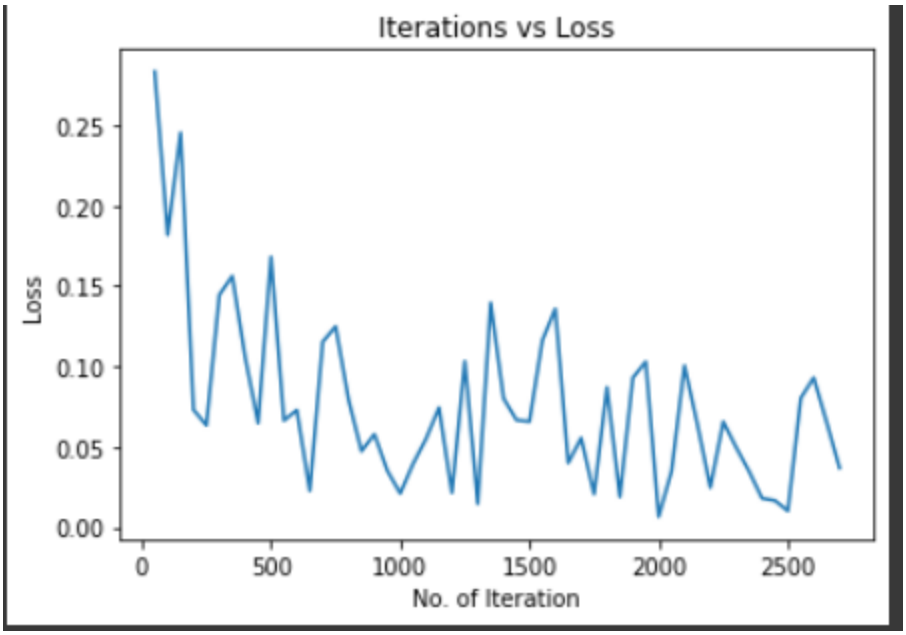
▼ run 1 for epoch = 3


0
1
2
3
4
40
20
0
−20
−40
−40
−20
0
20
40


Iterations vs Accuracy
Accuracy
97
96
95
94
93
92
0
500
1000
1500
2000
2500
No. of Iteration

Classification report for CNN :

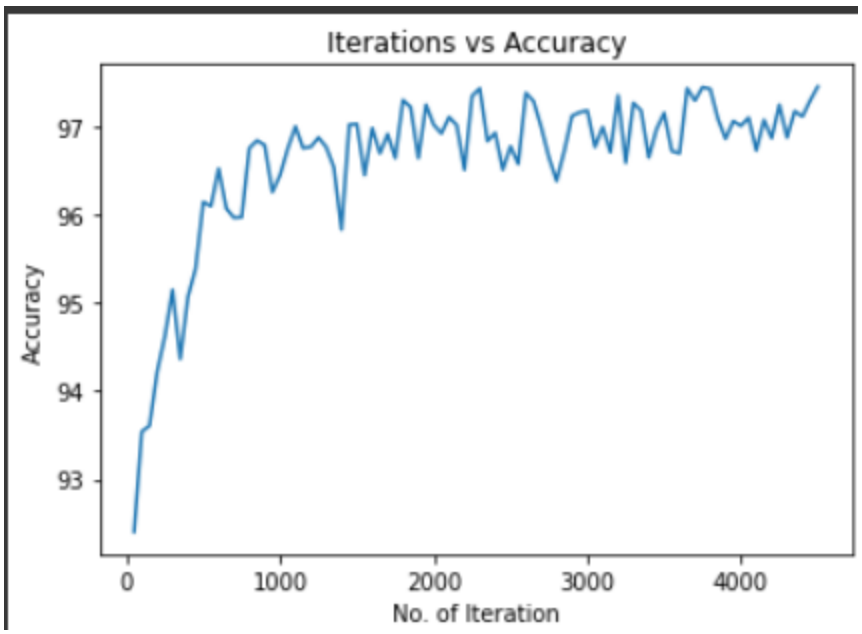
	precision	recall	f1-score	support
0	0.98	0.97	0.98	30000
1	0.97	0.98	0.98	30000
2	0.96	0.96	0.96	30000
3	0.93	0.94	0.94	30000
4	0.95	0.95	0.95	30000
accuracy			0.96	150000
macro avg	0.96	0.96	0.96	150000
weighted avg	0.96	0.96	0.96	150000

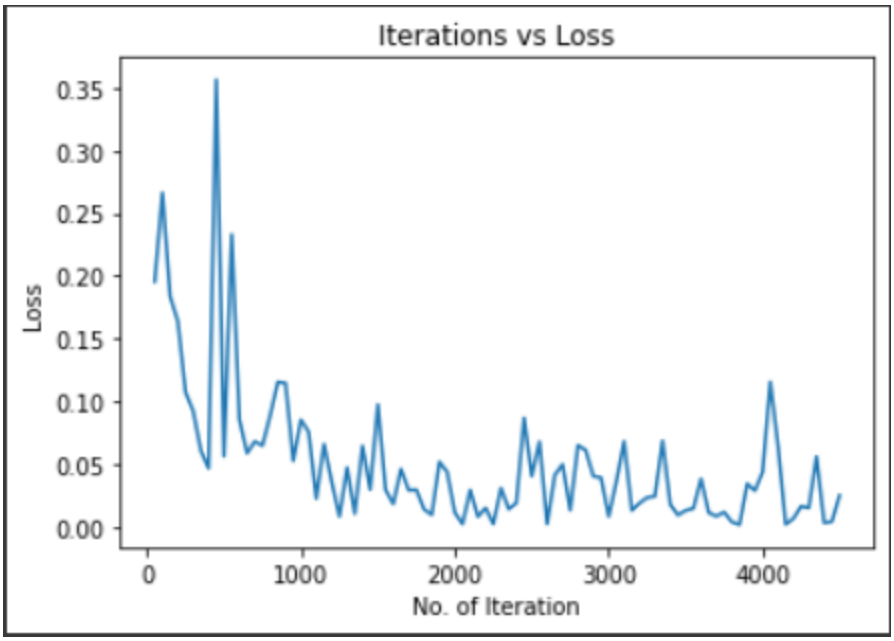


▼ run 2 epoch=5

Architecture


training loss
testing loss
0.18
0.16
0.14
0.12
0.10
0.08
0.06
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0

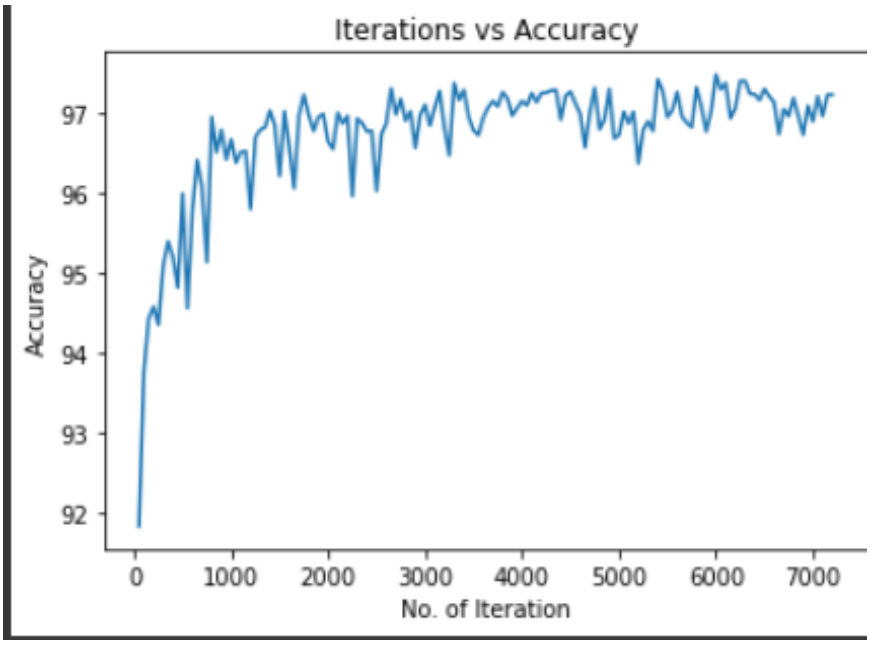

Iterations vs Accuracy
Accuracy
97
96
95
94
93
0
1000
2000
3000
4000
No. of Iteration




```
Classification report for CNN :
              precision    recall  f1-score   support

           0       0.99      0.98      0.98    270000
           1       0.98      0.98      0.98    270000
           2       0.97      0.97      0.97    270000
           3       0.94      0.95      0.94    270000
           4       0.96      0.95      0.96    270000

    accuracy                           0.97   1350000
   macro avg       0.97      0.97      0.97   1350000
weighted avg       0.97      0.97      0.97   1350000
```

```
Accuracy of Trouser: 98.50%
Accuracy of dress: 98.93%
Accuracy of sandle: 97.03%
Accuracy of sneaker: 97.73%
Accuracy of ankle boot: 94.63%
```

▼ run 4 epoch=8

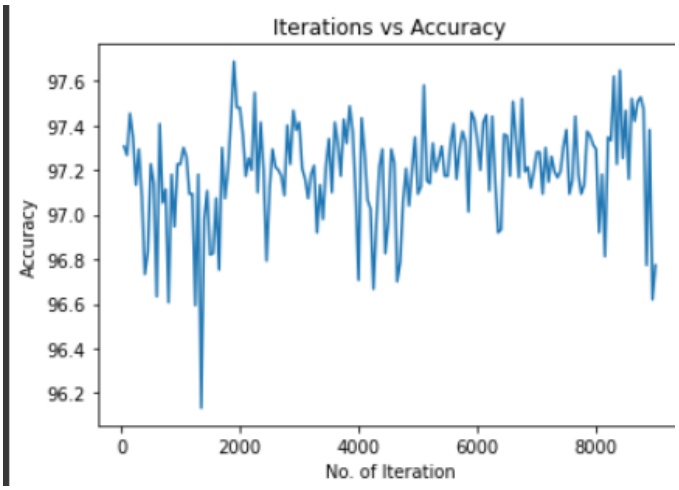


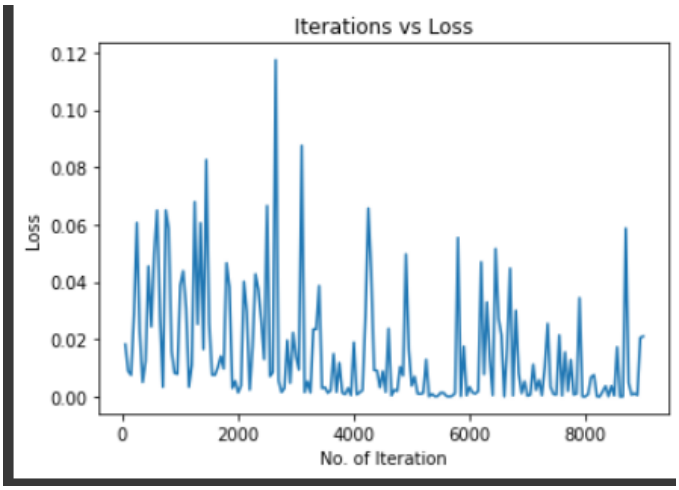

```
Accuracy of Trouser: 98.00%
Accuracy of dress: 99.57%
Accuracy of sandle: 97.27%
Accuracy of sneaker: 95.47%
Accuracy of ankle boot: 96.10%
```

```
Classification report for CNN :
                precision    recall  f1-score   support

             0       0.99      0.98      0.98    432000
             1       0.98      0.99      0.98    432000
             2       0.97      0.96      0.97    432000
             3       0.94      0.95      0.95    432000
             4       0.96      0.95      0.96    432000

      accuracy                           0.97   2160000
     macro avg       0.97      0.97      0.97   2160000
  weighted avg       0.97      0.97      0.97   2160000
```

▼ run 5 epoch=10

Very noisy learning curves

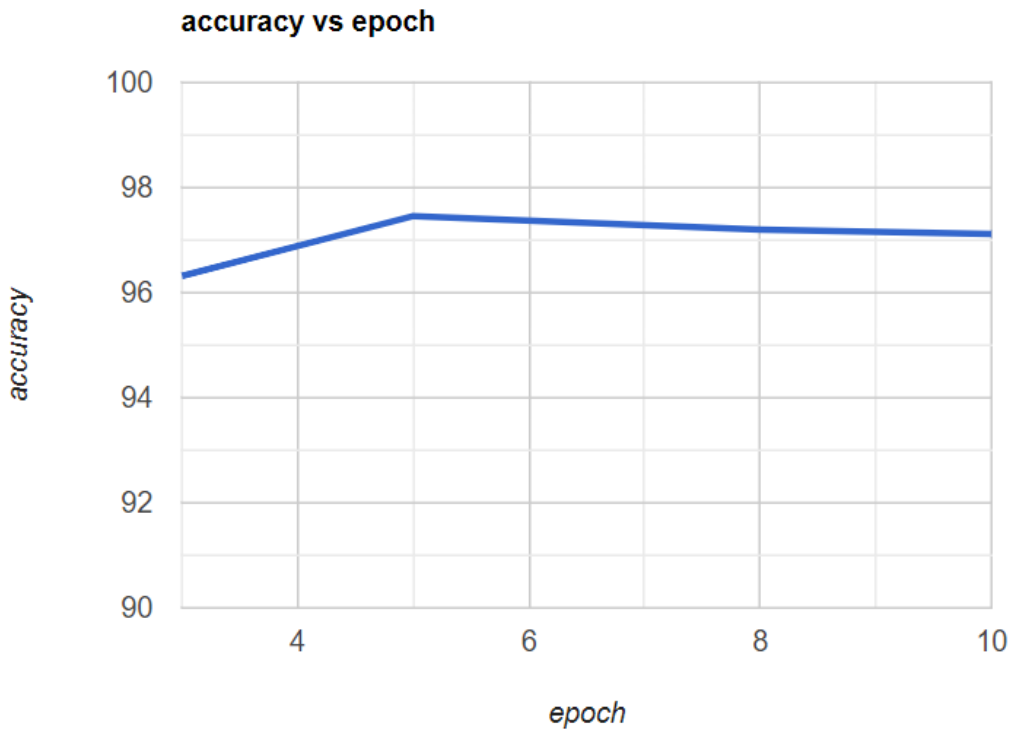




Classification report for CNN :

	precision	recall	f1-score	support
0	0.99	0.98	0.99	540000
1	0.98	0.99	0.99	540000
2	0.98	0.97	0.97	540000
3	0.95	0.96	0.95	540000
4	0.96	0.96	0.96	540000
accuracy			0.97	2700000
macro avg	0.97	0.97	0.97	2700000
weighted avg	0.97	0.97	0.97	2700000

```
Accuracy of Trouser: 98.63%
Accuracy of dress: 97.40%
Accuracy of sandle: 95.43%
Accuracy of sneaker: 96.30%
Accuracy of ankle boot: 96.17%
```



💡 The Best epoch was 5 , since for 10 the learning curves were really noisy but for 3 the accuracy was low

3. **Optimizer tuning**: The CNN model was trained firstly with SGD then Adam every other parameter is kept the same

   ▼ run 1 (SGD)

   epoch: 5

   batch : 100

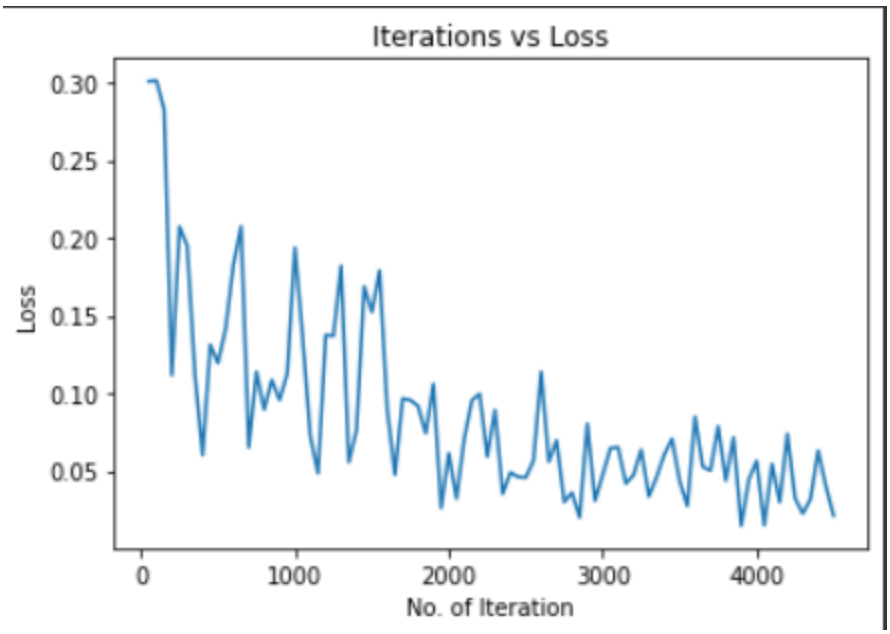
   optimizer : sgd

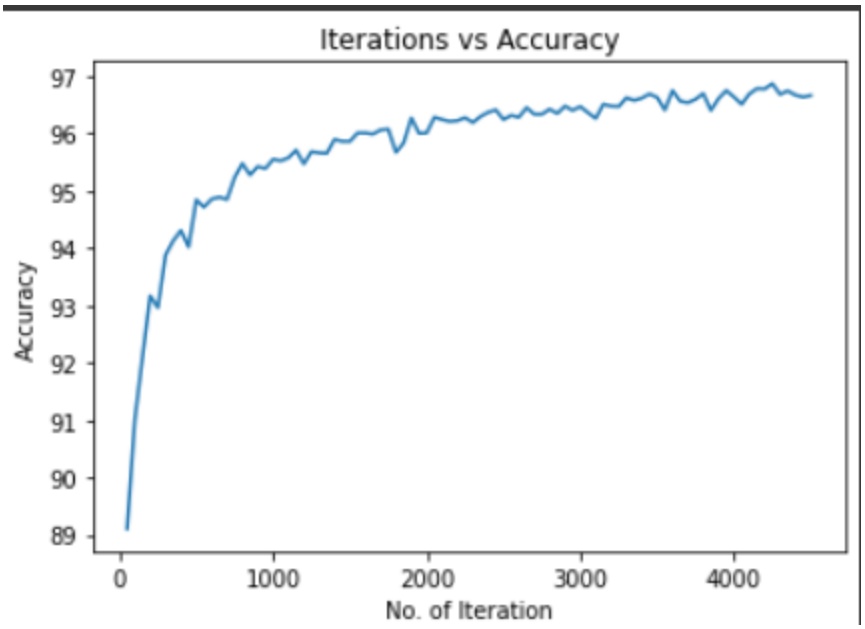
   1 % dec than adam

   learning rate : .001

```
Classification report for CNN :
              precision    recall  f1-score   support

           0       0.99      0.97      0.98    270000
           1       0.97      0.98      0.98    270000
           2       0.96      0.95      0.95    270000
           3       0.93      0.94      0.93    270000
           4       0.95      0.95      0.95    270000

    accuracy                           0.96   1350000
   macro avg       0.96      0.96      0.96   1350000
weighted avg       0.96      0.96      0.96   1350000
```

```
Accuracy of Trouser: 97.70%
Accuracy of dress: 98.63%
Accuracy of sandle: 96.10%
Accuracy of sneaker: 95.47%
Accuracy of ankle boot: 95.87%
```





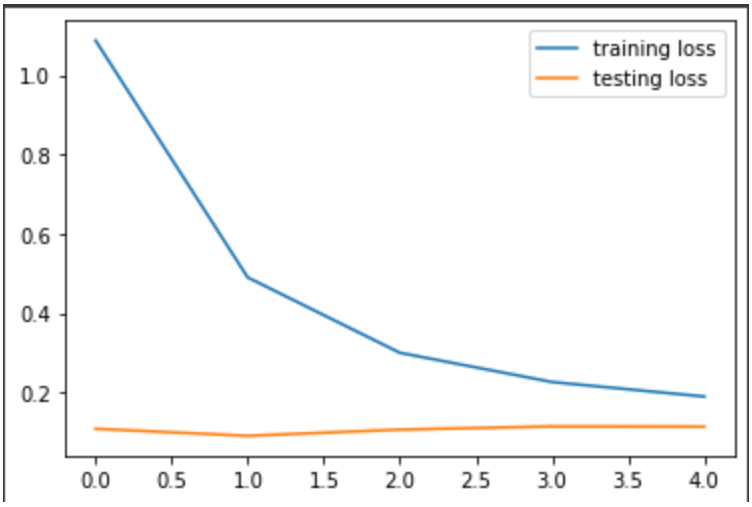
▼ run 2 (Adam)

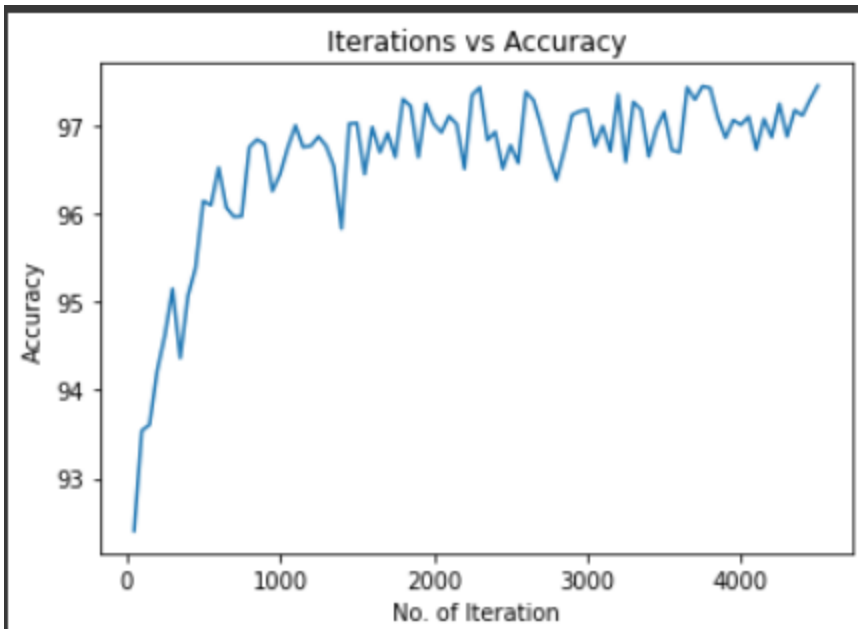
epoch: 5

batch : 100

optimizer : adam

learning rate : .001

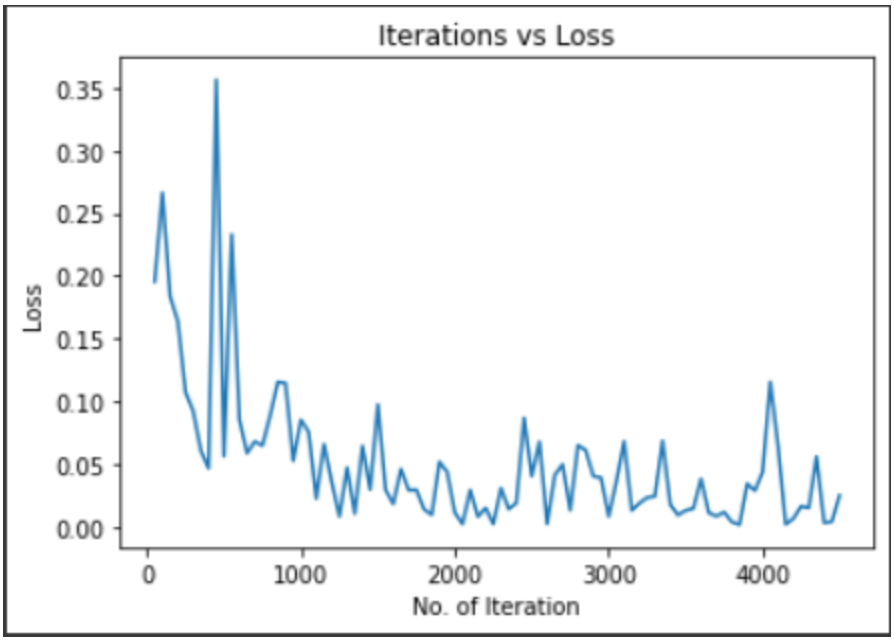




Classification report for CNN :

	precision	recall	f1-score	support
0	0.99	0.98	0.98	270000
1	0.98	0.98	0.98	270000
2	0.97	0.97	0.97	270000
3	0.94	0.95	0.94	270000
4	0.96	0.95	0.96	270000
accuracy			0.97	1350000
macro avg	0.97	0.97	0.97	1350000
weighted avg	0.97	0.97	0.97	1350000

```
Accuracy of Trouser: 98.50%
Accuracy of dress: 98.93%
Accuracy of sandle: 97.03%
Accuracy of sneaker: 97.73%
Accuracy of ankle boot: 94.63%
```



💡 Adam optimizer works better by 1% than SGD

Question 2

Problem Statement

The goal is to train an autoencoder and use the features from the encoder part to classify 1,3,5,7,9

The autoencoder is firstly trained to remove noise from data, then the classifier layer along with the encoder layer are trained to classify the classes 1 3 7 9

Hyperparameter tuning

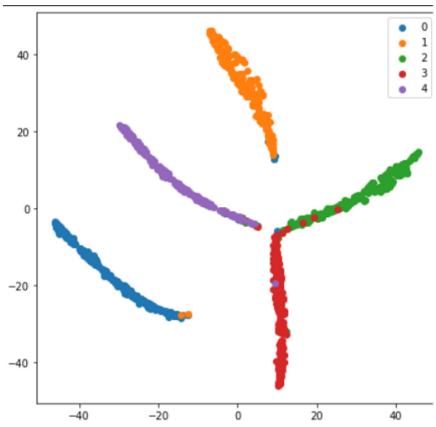
1. **learning rate tuning** For runs with the following hyperparameter, the following results was received

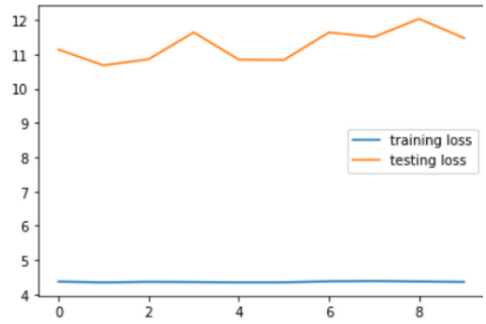
▼ run 1

Hyperparameters

learning rate	0.001
epoch	10

batch size	100

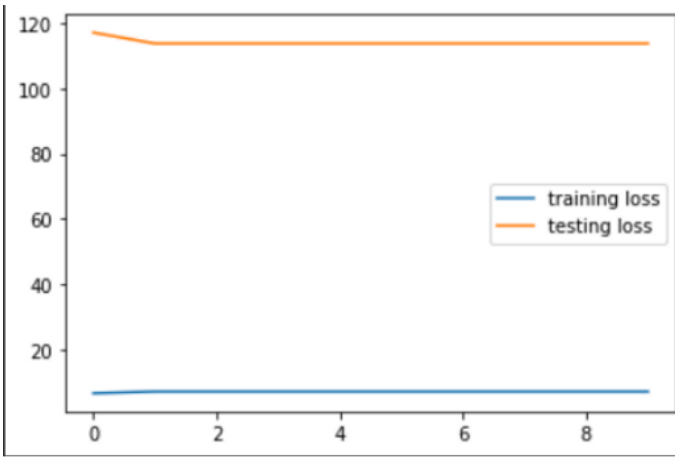


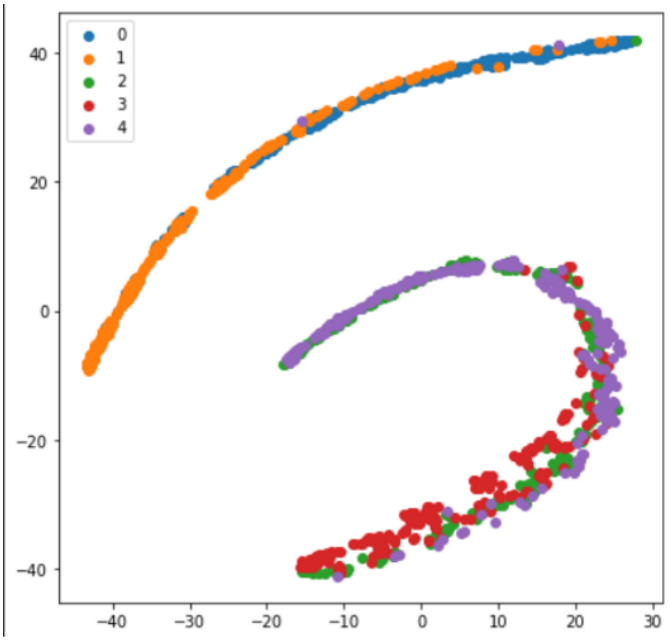


▼ run 2

Hyperparameters

learning rate	0.01
epoch	10
batch size	100






```
Test accuracy: 39.84%
```

Learning rate of 0.001 is preferred over 0.01 as the former has a much better tnse plot and accuracy

2. **model architecture tuning**, to models architecture, the following model architectures were tried

▼ run 1

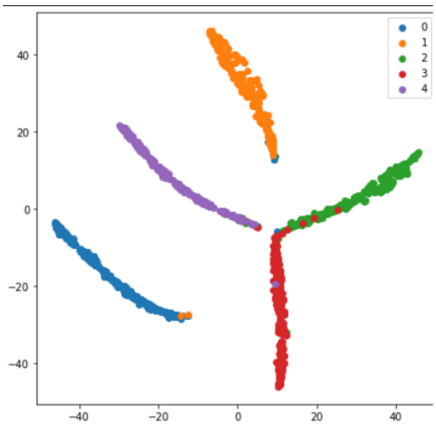
Architecture

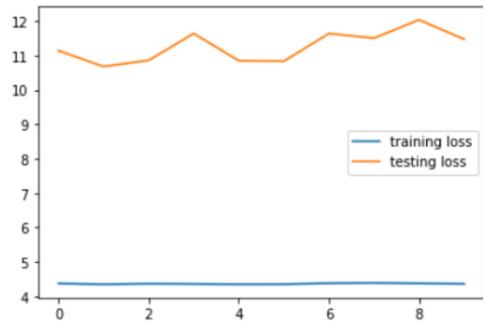
of autoencoder since the classifier architecture is almost fixed

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Linear-1                  [-1, 512]         401,920
              ReLU-2                  [-1, 512]               0
            Linear-3                  [-1, 256]         131,328
              ReLU-4                  [-1, 256]               0
            Linear-5                   [-1, 64]          16,448
              ReLU-6                   [-1, 64]               0
            Linear-7                  [-1, 256]          16,640
              ReLU-8                  [-1, 256]               0
            Linear-9                  [-1, 512]         131,584
             ReLU-10                  [-1, 512]               0
           Linear-11                  [-1, 784]         402,192
             ReLU-12                  [-1, 784]               0
================================================================
Total params: 1,100,112
Trainable params: 1,100,112
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.00
Forward/backward pass size (MB): 0.04
Params size (MB): 4.20
Estimated Total Size (MB): 4.24
----------------------------------------------------------------
```

Hyperparameters

learning rate	0.001
epoch	10
batch size	100





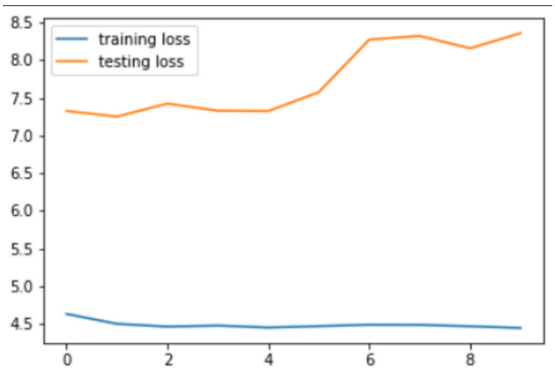

```
Test accuracy: 96.20%
```

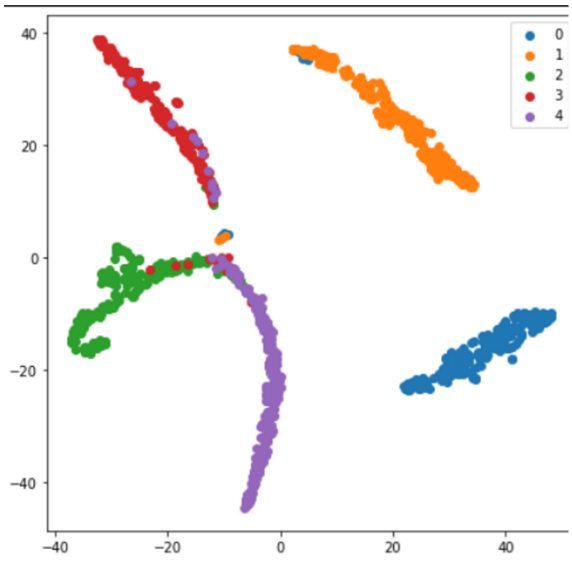
▼ run 2

this is the thicker model

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Linear-1                  [-1, 600]         471,000
              ReLU-2                  [-1, 600]               0
            Linear-3                  [-1, 550]         330,550
              ReLU-4                  [-1, 550]               0
            Linear-5                  [-1, 512]         282,112
              ReLU-6                  [-1, 512]               0
            Linear-7                  [-1, 550]         282,150
              ReLU-8                  [-1, 550]               0
            Linear-9                  [-1, 600]         330,600
             ReLU-10                  [-1, 600]               0
           Linear-11                  [-1, 784]         471,184
             ReLU-12                  [-1, 784]               0
================================================================
Total params: 2,167,596
Trainable params: 2,167,596
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.00
Forward/backward pass size (MB): 0.05
Params size (MB): 8.27
Estimated Total Size (MB): 8.33
```

learning rate	0.001
epoch	10
batch size	100






```
Test accuracy: 95.60%
```

💡 the learning curve in run 2 is showing signs of overfitting since the test loss keeps increasing while the training loss is really low, furthermore, there is a decrease in in accuracy on increasing the parameters therefore run1 has the better architecture

3. **epoch tuning**

   ▼ run1

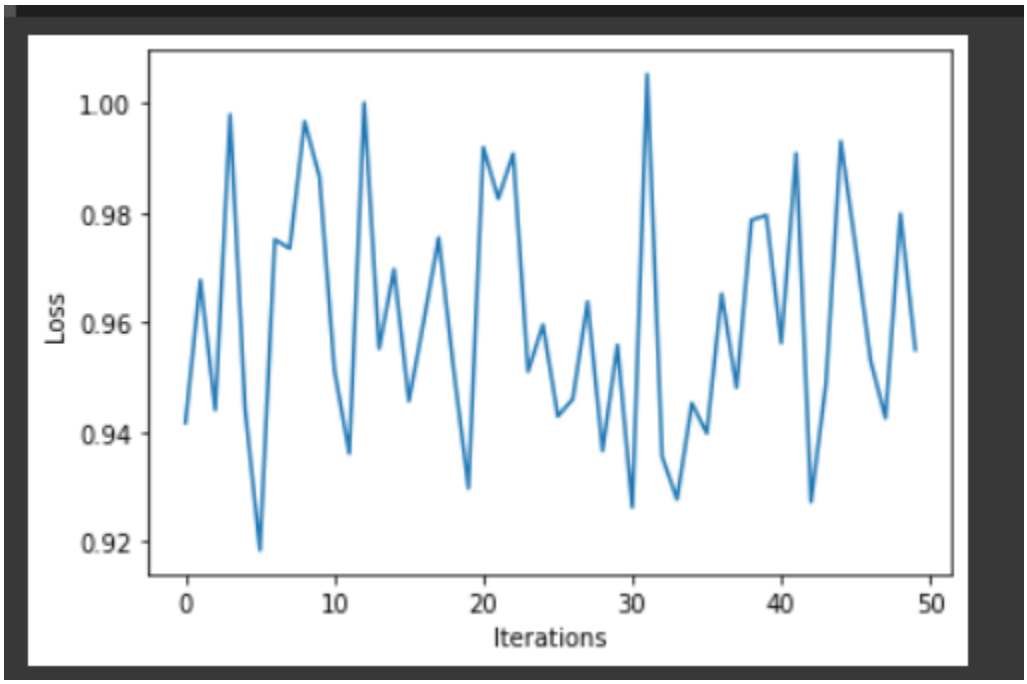
   train epoch : 2

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Linear-1                  [-1, 512]         401,920
              ReLU-2                  [-1, 512]               0
            Linear-3                  [-1, 256]         131,328
              ReLU-4                  [-1, 256]               0
            Linear-5                   [-1, 64]          16,448
              ReLU-6                   [-1, 64]               0
            Linear-7                  [-1, 256]          16,640
              ReLU-8                  [-1, 256]               0
            Linear-9                  [-1, 512]         131,584
             ReLU-10                  [-1, 512]               0
           Linear-11                  [-1, 784]         402,192
             ReLU-12                  [-1, 784]               0
================================================================
Total params: 1,100,112
Trainable params: 1,100,112
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.00
Forward/backward pass size (MB): 0.04
Params size (MB): 4.20
Estimated Total Size (MB): 4.24
----------------------------------------------------------------
```

An underfit model can be identified from the learning curve of the training loss only.

It may show a flat line or noisy values of relatively high loss, indicating that the model was unable to learn the training dataset at all.

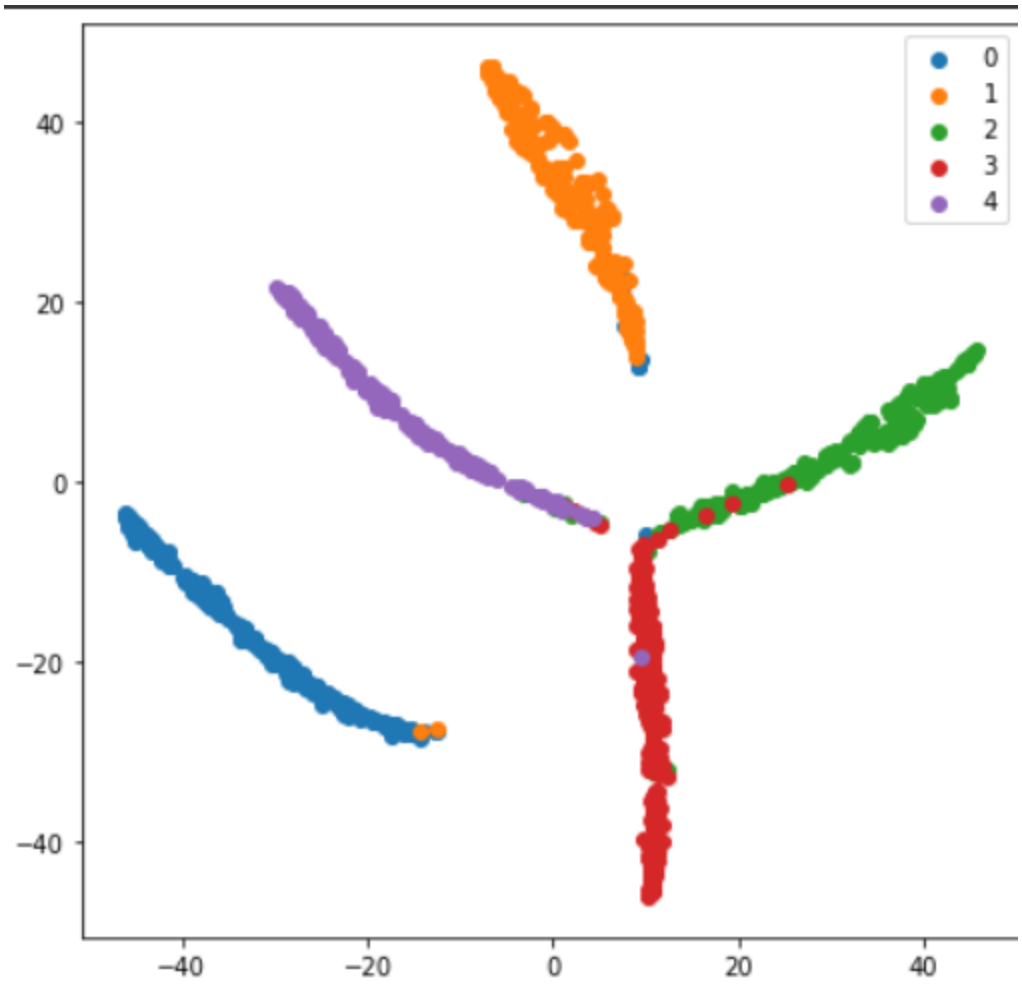
therefore the following graph suggests that the the model is underfitting so we increase the epochs

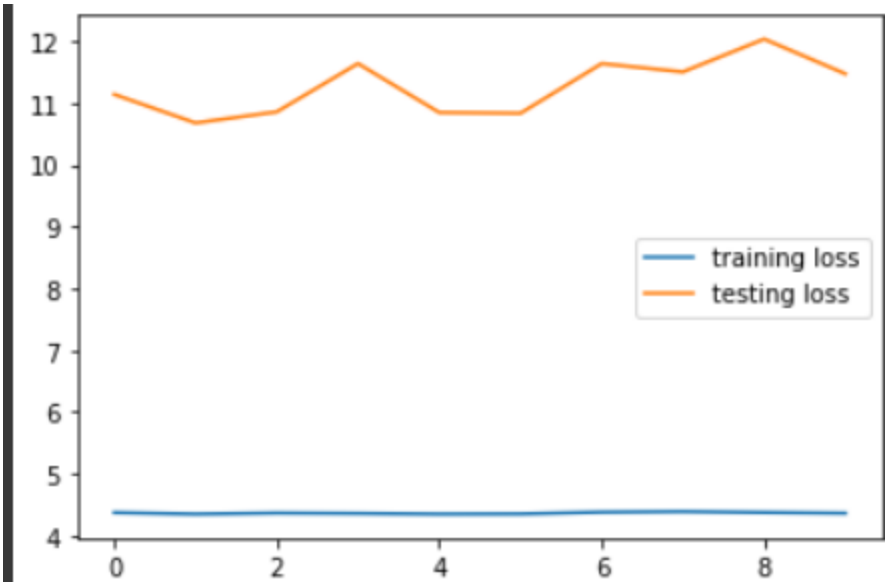



```
Test accuracy: 94.36%
```

▼ run2

epoch : 2





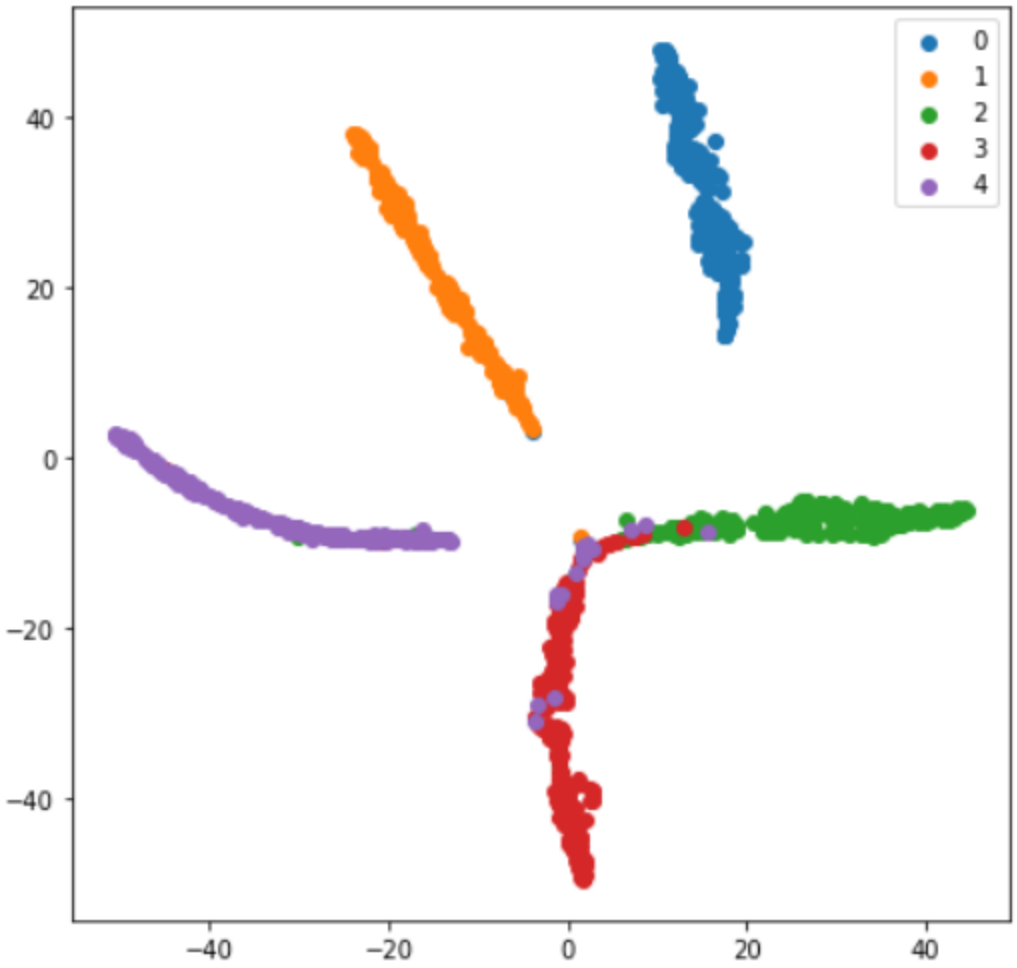

```
Test accuracy: 96.20%
```

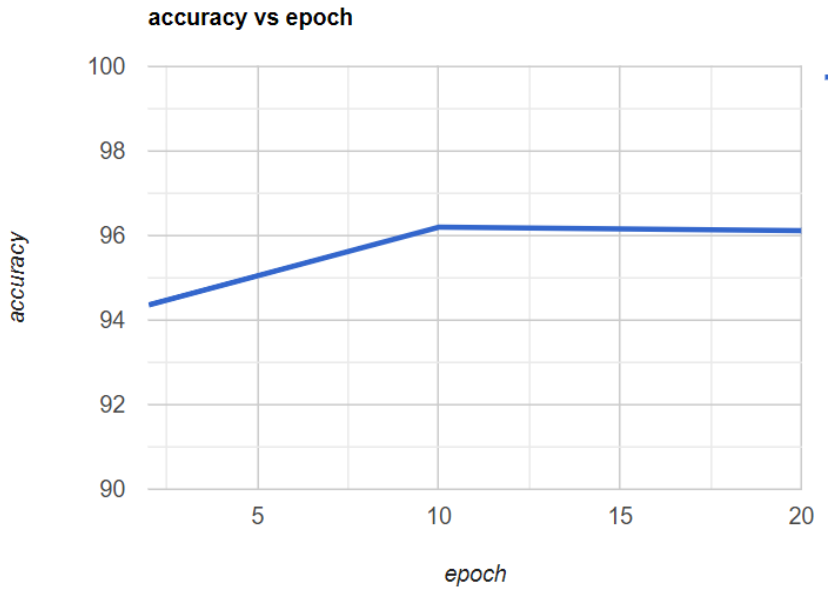
▼ run 3

epoch 20

(overfitting)


```
Test accuracy: 96.12%
```


0
1
2
3
4
40
20
0
−20
−40
−40
−20
0
20
40


accuracy vs epoch
100
98
96
94
92
90
accuracy
5
10
15
20
epoch

💡 as observed in the learning curves, epoch 2 was underfitting while epoch 20 was overfitting therefore 10 is the optimum epoch

Conclusion

CNN was performing quite better as compared to auto encoder