DL ASSIGNMENT-2 Part-2

Our result from Part-1

Activation+Optimiser	Train Accuracy	Validation Accuracy
Tanh	73.96	69.6
ReLU	88.68	84.40
Sigmoid	19.96	17.40
ReLU + momentum	98.95	93.30
ReLU + NAG	99.11	93.55
ReLU+ AdaGrad	100	94.65
ReLU + RMSProp	92.33	86.5
ReLU+ Adam	19.05	19.05

Part-2

1. Implement the following weight initialization techniques from scratch and do a thorough analysis on the output of each one of them. [30 points]

- He
- Xavier

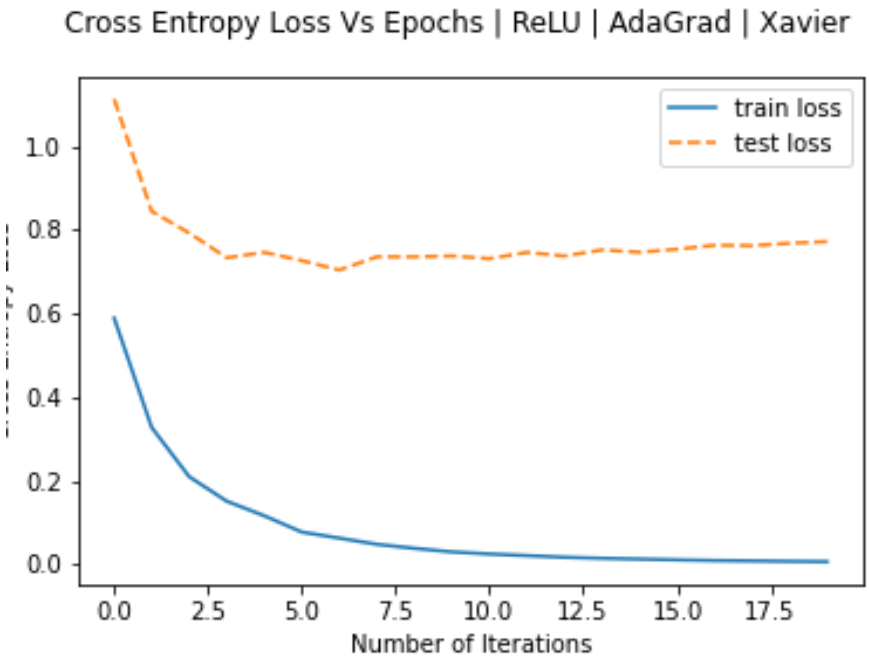
```
[2] model = pickle.load(open("Models\\relu_adagrad_he.pkl","rb"))...

    Train Accuracy: 0.9993
    Tests Accuracy: 0.868
    Train Loss 0.01248104209137784
    Val Loss 1.978930046701534

[3] model = pickle.load(open("Models\\relu_adagrad_xavier.pkl","rb"))...

    Train Accuracy: 1.0
    Tests Accuracy: 0.8985
    Train Loss 0.007727448412395178
    Val Loss 0.7716953766464049
```

Xavier performs better out of the two.

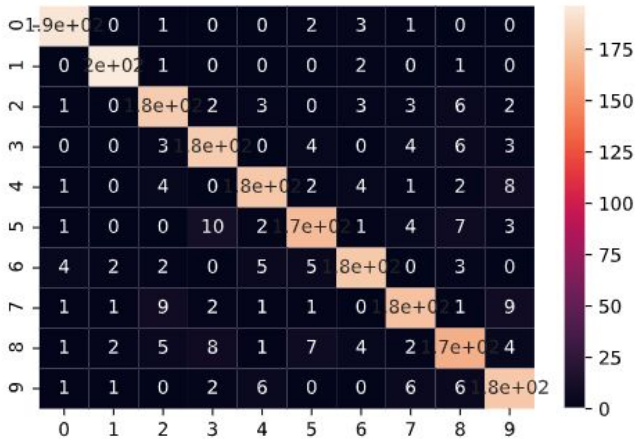
DL ASSIGNMENT-2 Part-2

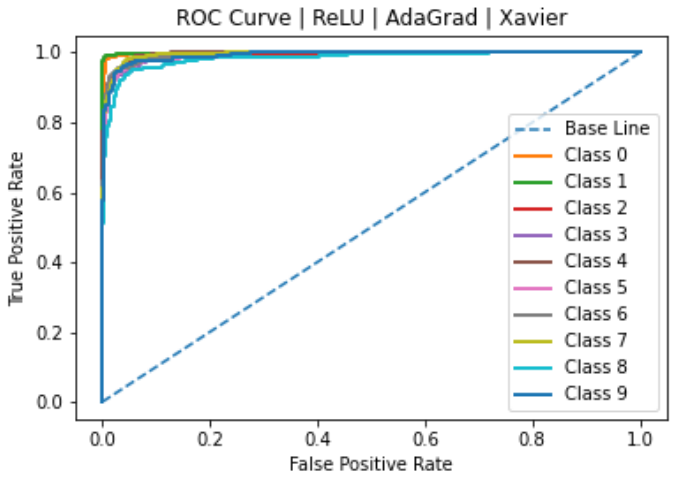
- **Activation=ReLU, Optimiser=AdaGrad, Initialisation=Xavier**

LR=0.01, EPOCHS=20



ACCURACY	
Training	100
Validation	89.85





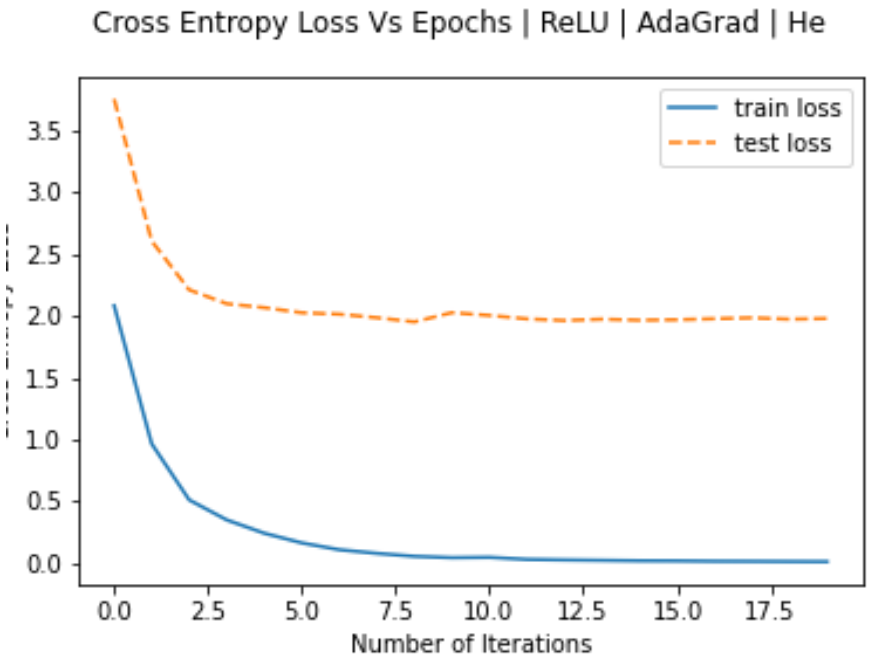
Note: Horizontal axis represents predicted labels while the vertical axis represents the true labels.

INFERENCE: We see that after Xavier initialisation convergence is much faster, as compared to random_init, also in latter cases there are spikes in Loss vs Epoch graph, however after Xavier initialisation the convergence is smooth. This may be because Xavier helps avoid exploding gradients and thus learning is efficient.

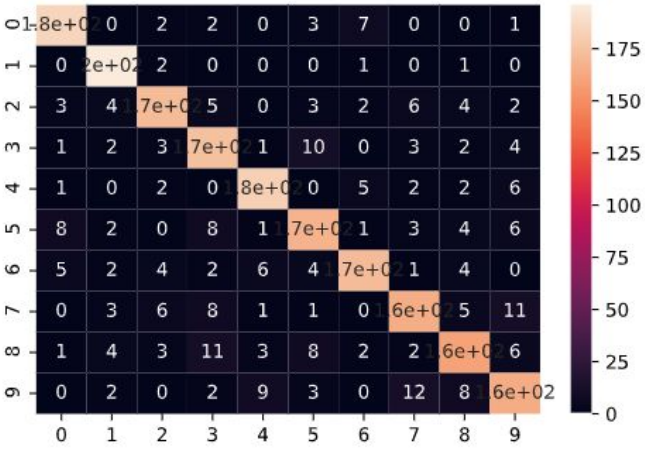
DL ASSIGNMENT-2 Part-2

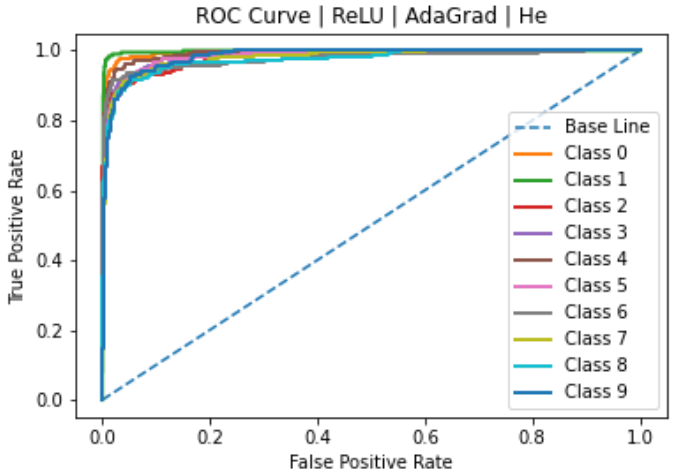
- **Activation=ReLU, Optimiser=AdaGrad, Initialisation=He**

LR=0.01, EPOCHS=20



ACCURACY	
Training	99.93
Validation	86.80





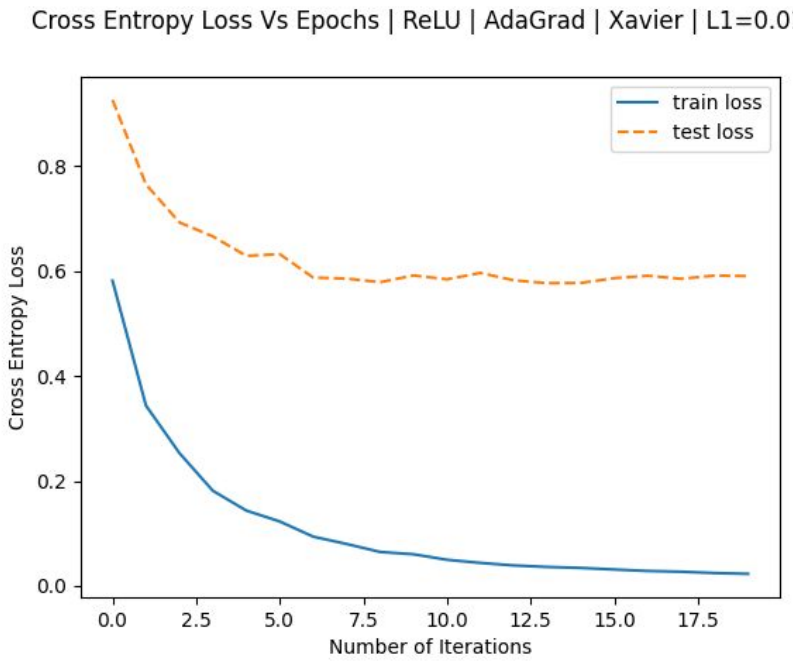
Note: Horizontal axis represents predicted labels while the vertical axis represents the true labels.

INFERENCE: We see that after He initialisation convergence is faster as compared to random_init but slower than Xavier, also Test loss is more in case of He. This may be because Xavier helps avoid exploding gradients and thus learning is efficient. He is although said to perform better with ReLU as it prevents dying neurons, but in our case Xavier is performing better maybe due to the dataset chosen, or because we implemented LReLU.

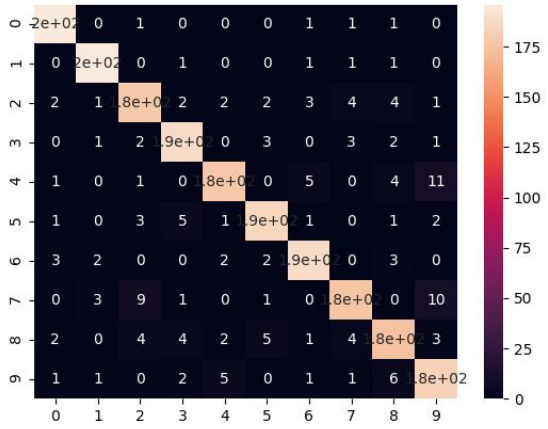
DL ASSIGNMENT-2 Part-2

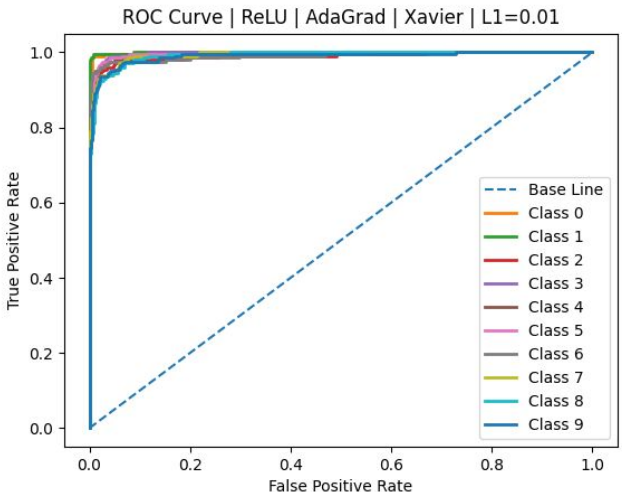
- **Activation=ReLU, Optimiser=AdaGrad, Initialisation=Xavier, Regularization=L1, λ=0.01**

LR=0.01, Epochs=20



ACCURACY	
Training	99.94
Validation	92.25





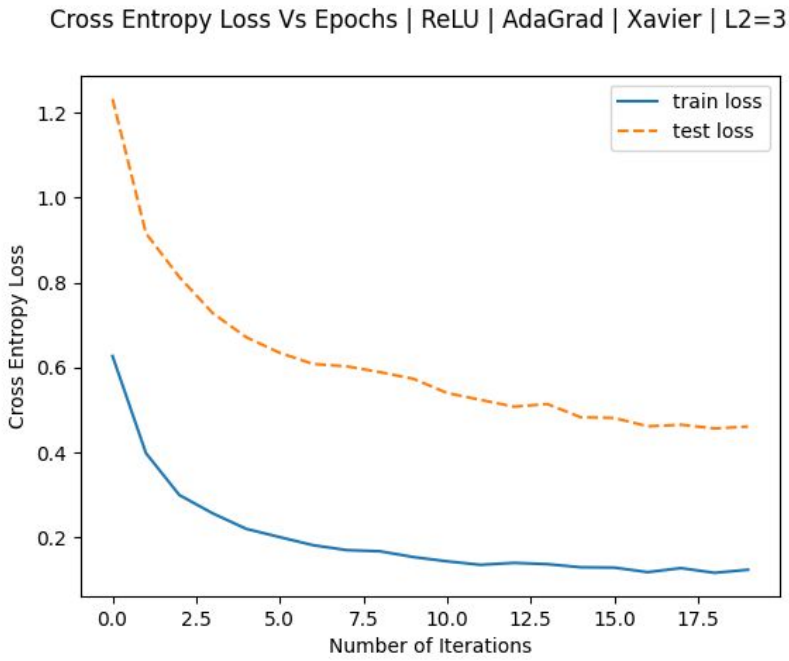
Note: Horizontal axis represents predicted labels while the vertical axis represents the true labels.

INFERENCE: With L1 loss, we do grid search for λ, and the difference between train and test loss decreases for 0.01. Validation accuracy however does not improve much. Model overfitting is avoided in this case, and this is the purpose of regularization.

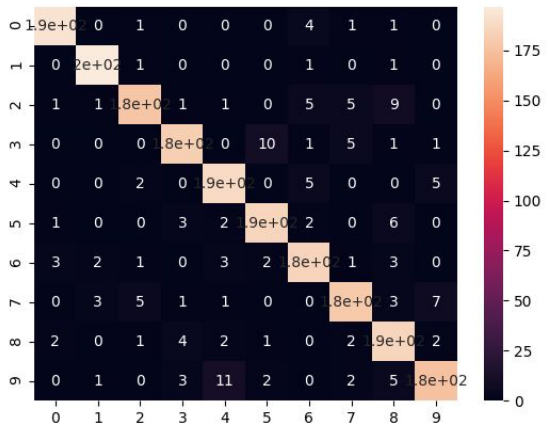
DL ASSIGNMENT-2 Part-2

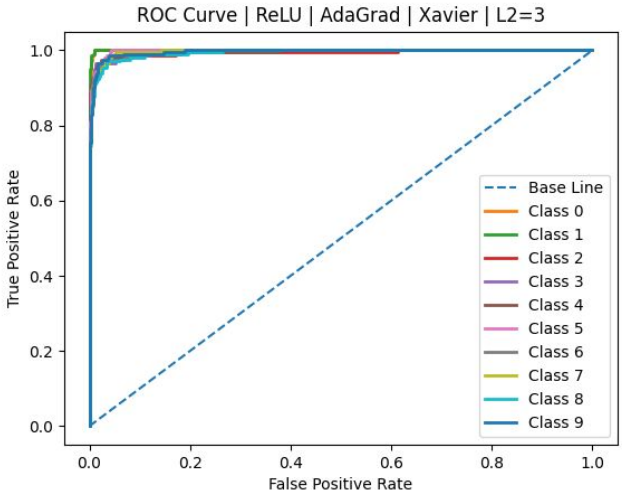
- **Activation=ReLU, Optimiser=AdaGrad, Initialisation=Xavier, Regularization=L2,λ=3**

  LR=0.01, Epochs=20



ACCURACY	
Training	99.07
Validation	92.5





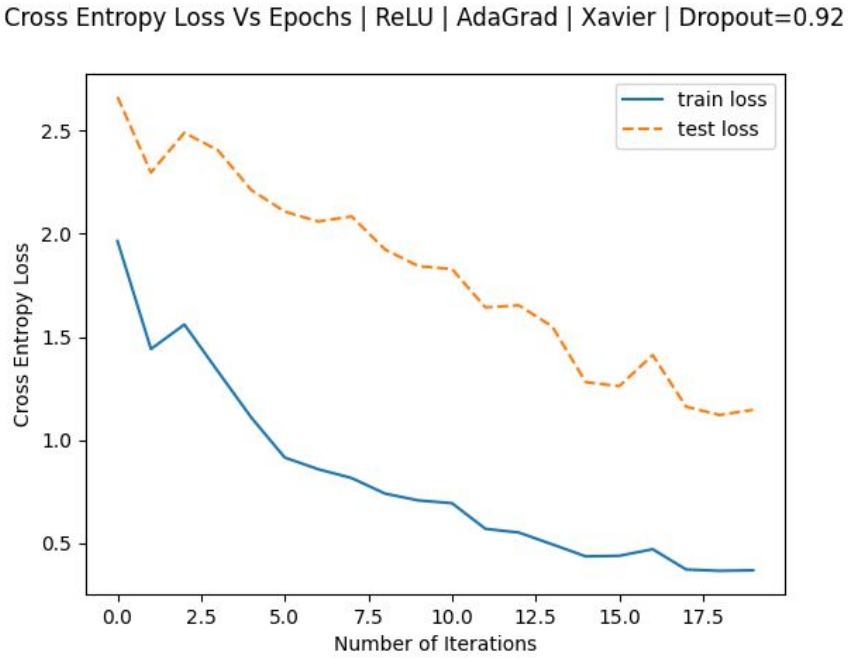
Note: Horizontal axis represents predicted labels while the vertical axis represents the true labels.

INFERENCE: We take λ as 3, which completely avoids overfitting and penalistion is heavy. The validation accuracy in L2 regularisation is higher and difference b/w losses is less. L2 regularisation also serves its purpose and helps in avoiding overfit.

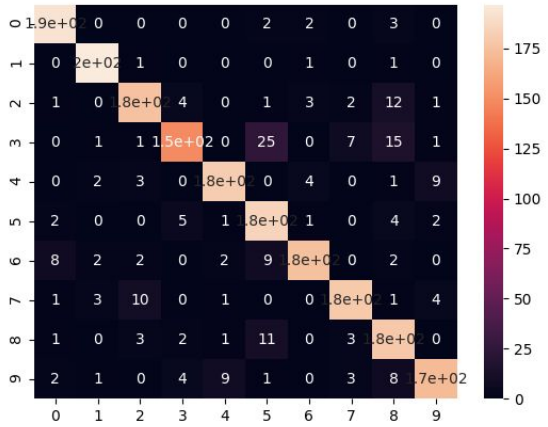
DL ASSIGNMENT-2 Part-2

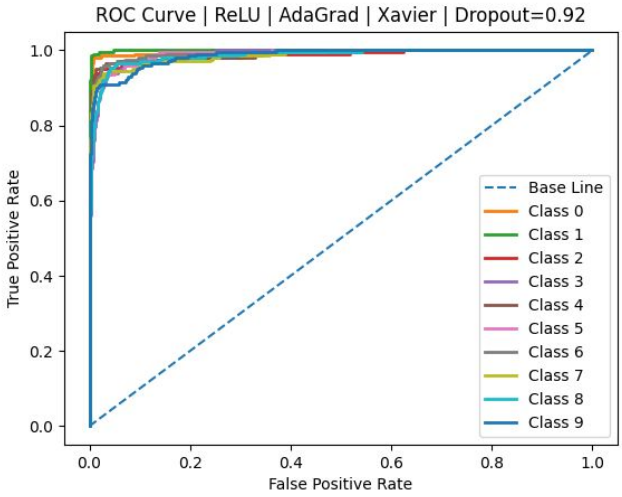
- **Activation=ReLU, Optimiser=AdaGrad, Initialisation=Xavier, Regularization=Dropout**

LR=0.01, Epochs=20



ACCURACY	
Training	94.87
Validation	89.40





Note: Horizontal axis represents predicted labels while the vertical axis represents the true labels.

DL ASSIGNMENT-2 Part-2

```
[5] model = pickle.load(open("Models\\relu_adagrad_l1.pkl","rb"))...

    Train Accuracy: 0.9994
    Tests Accuracy: 0.9225
    Train Loss 0.02300041451962646
    Val Loss 0.5904971106811704

[6] model = pickle.load(open("Models\\relu_adagrad_l2.pkl","rb"))...

    Train Accuracy: 0.9907
    Tests Accuracy: 0.925
    Train Loss 0.12424744259699466
    Val Loss 0.46127556756364285

[7] model = pickle.load(open("Models\\relu_adagrad_dp.pkl","rb"))...

    Train Accuracy: 0.9487
    Tests Accuracy: 0.894
    Train Loss 0.36909436763360315
    Val Loss 1.1464729901657162
```

INFERENCE:

Dropout seems to work well if the keep_probability is > 0.9. Any value less than 0.9 doesn't seem to work well with "Xavier" weights. It is due to the following reasons:

1) The model is already generalized and not over-fitting thus only a little bit tuning is done when probability is kept at the value of 0.92. While drop-out is used to prevent overfitting, excessive use of drop-out leads to bad training/ bad accuracy because loss of data occurs at the hidden layer neurons.
2) Xavier initialisation technique makes weights smaller as compared to random weight initialization, thus drop out is unable to affect the weights more. We can see that if we use random weights instead of xavier for the same configuration drop out 0.8 gives us promising results.
3) One of the reasons that drop out doesn't work well is the saturation of neurons in the hidden layers. As we use drop out, keep_prob also works as a scaling factor for the activation of neurons. For example if keep_prob is 0.5, activation will be 2 X times the initial thus some neurons reach saturation if the model is not overfitting a lot. This saturation leads to wrong results in case of dropout regularization.

DL ASSIGNMENT-2 Part-2

dropout=0.85, init=xavier

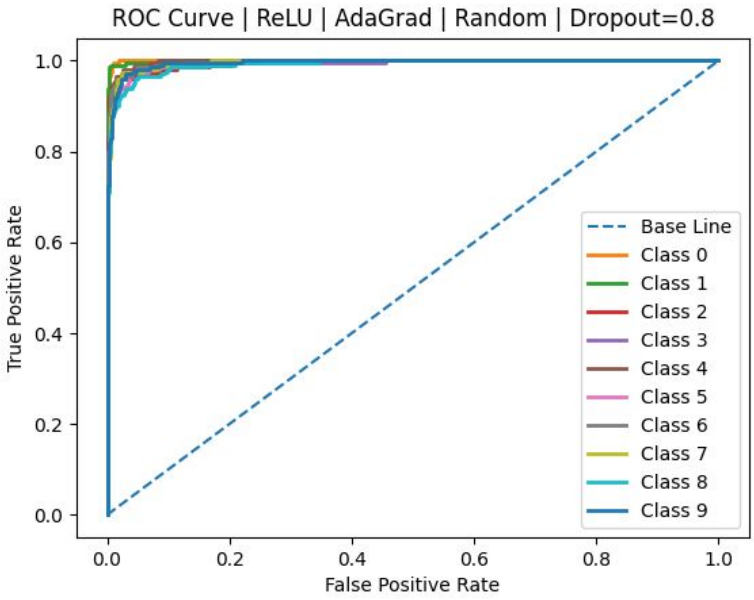
```
Train Accuracy: 0.6577
Tests Accuracy: 0.63
```

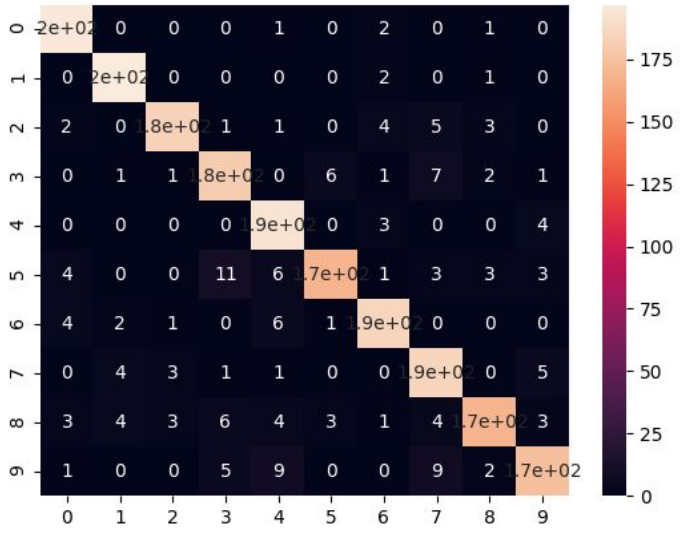
dropout=0.8, init=xavier

```
Train Accuracy: 0.6987
Tests Accuracy: 0.658
```

dropout = 0.8, init=random

```
Train Accuracy: 0.9659
Tests Accuracy: 0.9175
```





DL ASSIGNMENT-2 Part-2

Cross Entropy Loss Vs Epochs | ReLU | AdaGrad | Random | Dropout=0.8