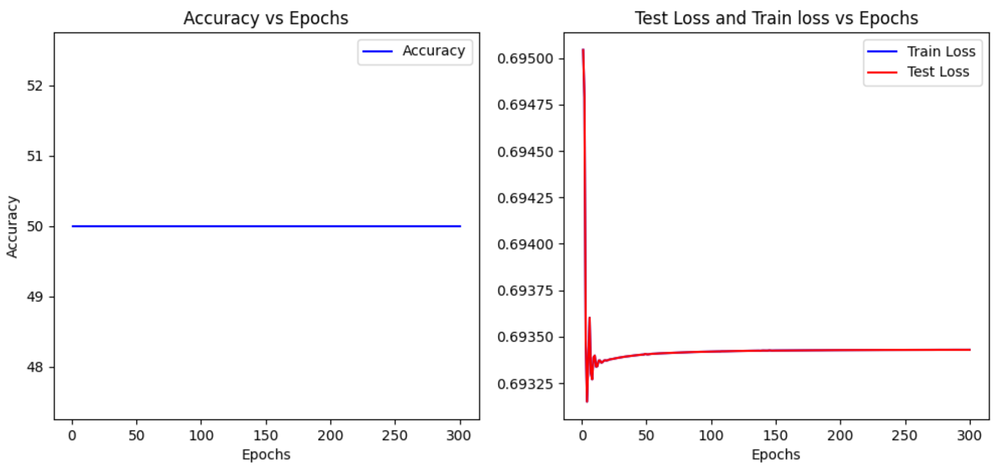
COL774 Assignment - 2.2

Convolutional Neural Networks

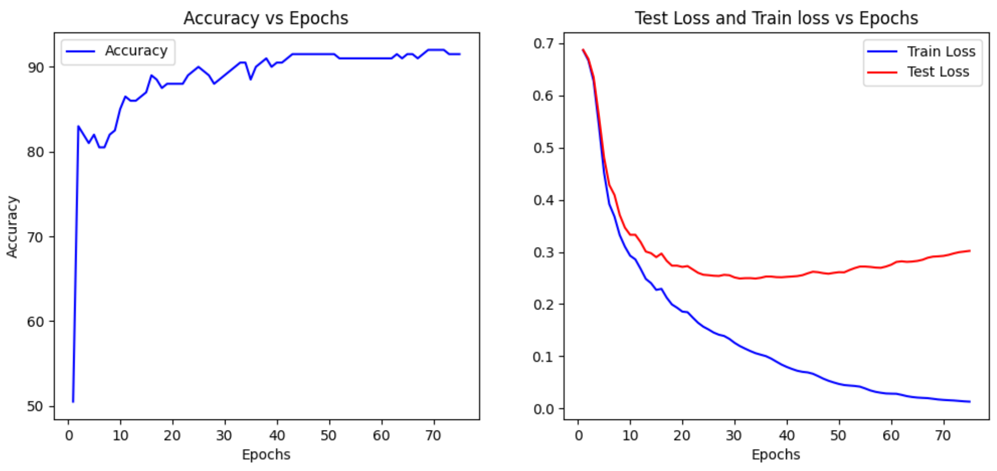
Divam Manchanda - 2022ME21336

Yash Bansal - 2022CS51133

Part a)



Results from Assignment 2.1 - Fully Connected Neural Network with 300 Epochs



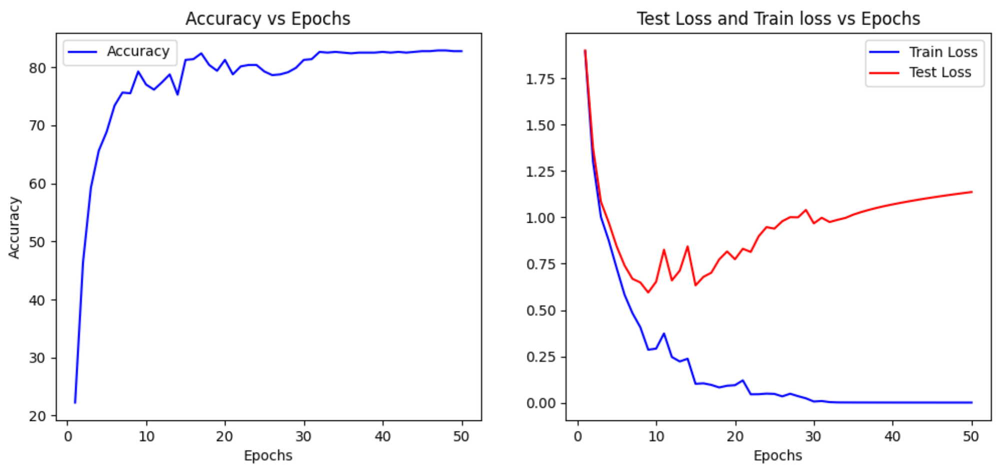
Results from Assignment 2.2 - CNN with 75 Epochs

Significantly improved results obtained via the CNN :-

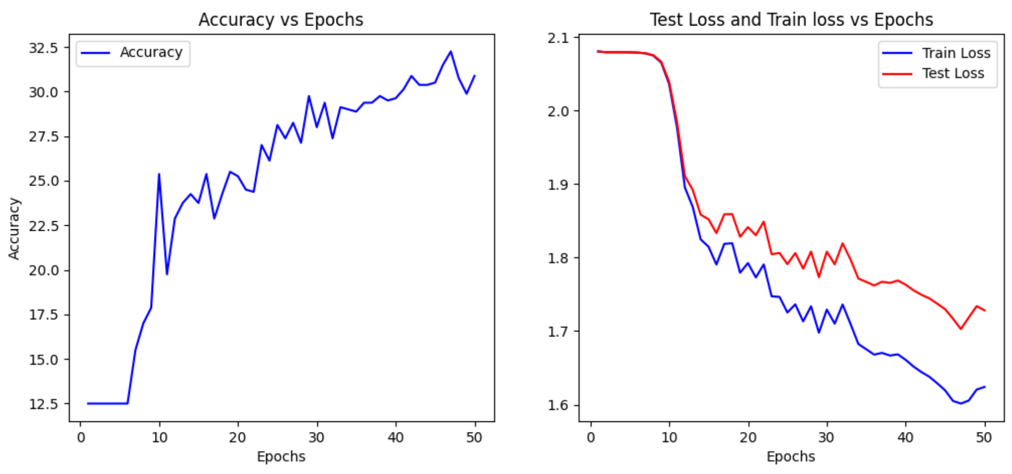
- Public test accuracy after 8 epochs :- 80.5 %
- Maximum test accuracy achieved at 40 epochs:- accuracy of 91.5% using the given architecture
- After 40 epochs, overfitting started to occur and test loss starts to increase, final accuracy stable at 91.5 %
- For CNN, approx 2 secs take per epoch:- so max accuracy for 91.5 % achieved in around 80 secs
  required plots shown

comparison with fully connected nn specification given in Assignment 2.1:-
Fully connected NN can not achieve accuracy of more than 50% as after just 15 epochs, the train loss cannot decrease any further.

Part b)



Results from Assignment 2.2 - CNN with 50 Epochs



Results from Assignment 2.1 - Fully Connected Neural Network with 300 Epochs

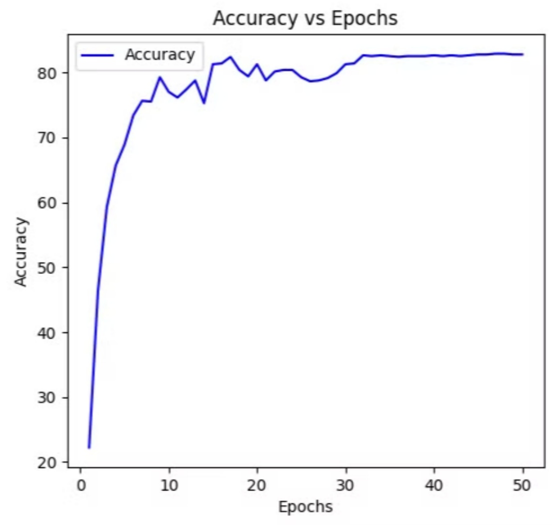
CNN results

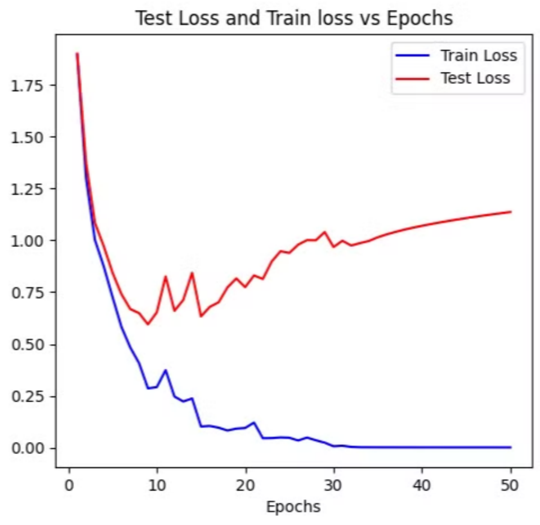
- Public test accuracy after 8 epochs:- 77.12 %
- Maximum test accuracy achieved at 30 epochs:- accuracy of 82.5% using the given architecture
- Final accuracy gets stable at 82.5% after 30 epochs
- For cnn(given architecture), takes approx 15 secs per epoch:- so max accuracy for 82.5 % achieved in around 80 secs

comparison with fully connected nn specification given in Assignment 2.1:-
Takes approx 0.3-0.4 secs per epoch
Accuracy highly fluctuation, cannot cross 50% in 300 epochs
Accuracy still fluctaution between 30-35% till 1500 epochs:- not at all near cnn

Part c)

c.1) Experimenting with batch sizes - part b) architecture.

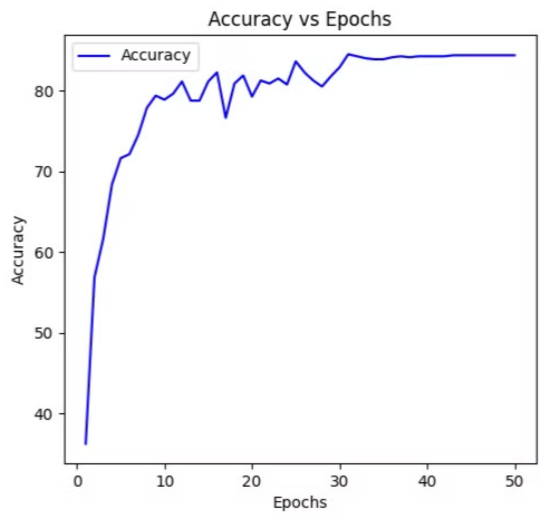


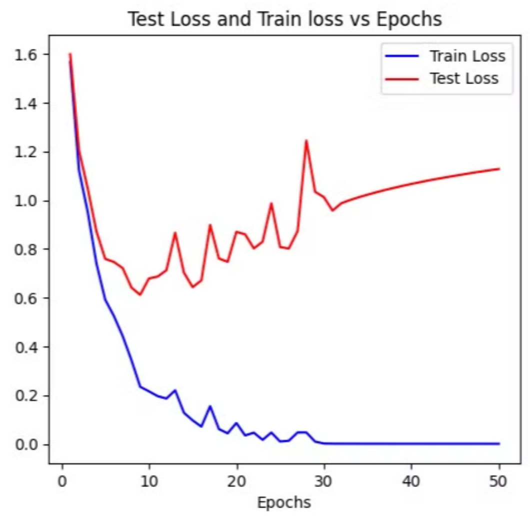


Parameters -

- batch size = 128
- learning rate = 0.001
- number of epochs = 50

Accuracy for the test set stagnates at 82% after ~30 epochs.

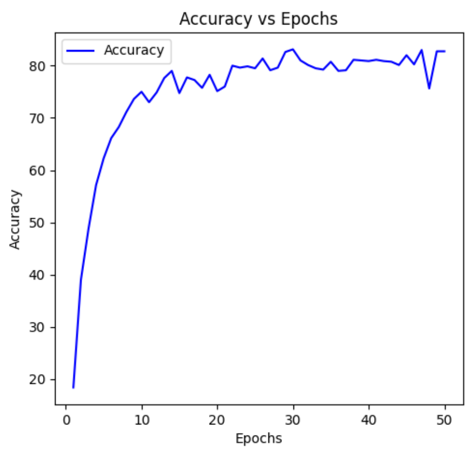


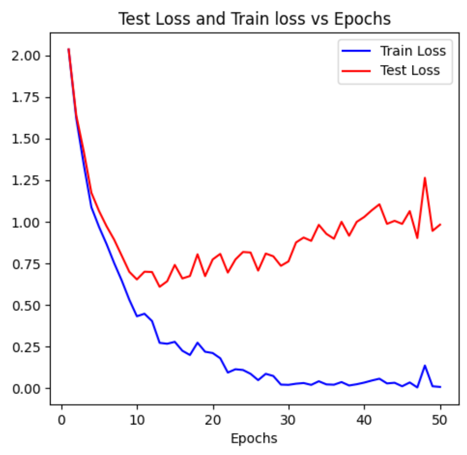


Parameters - (batch size reduced)

- batch size = 100
- learning rate = 0.001
- number of epochs = 50

Accuracy for the test set stagnates at 84% after ~30 epochs.

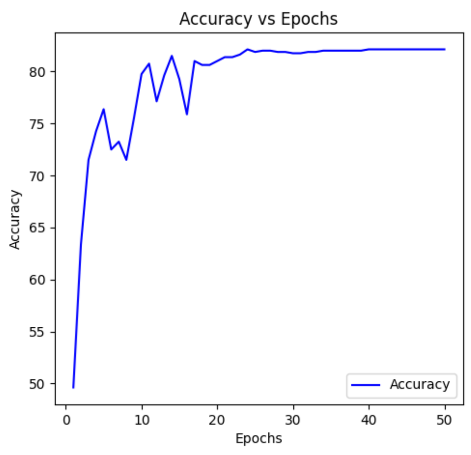


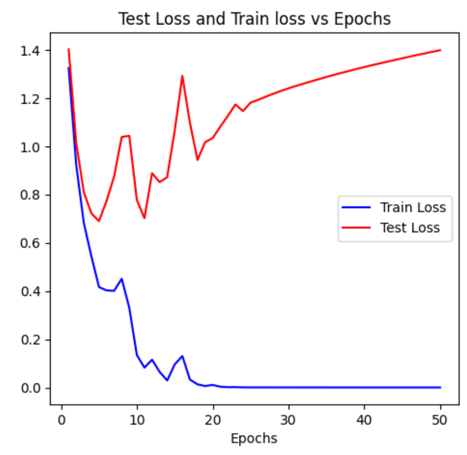


Parameters - (batch size increased)

- batch size = 256
- learning rate = 0.001
- number of epochs = 50

larger batch size, can't converge quickly enough





Parameters - (further lowered the batch size)

- batch size = 50
- learning rate = 0.001
- number of epochs = 50

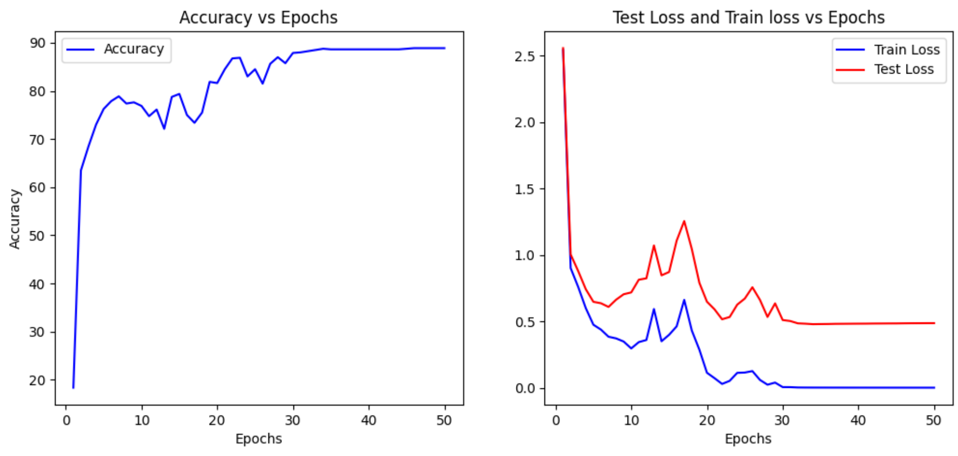
Lower batch size converges to 82% quickly but is unable to attain a higher accuracy.

💡 General observations from above: -

- Larger batch sizes converge to give higher accuracy on the test set.
- At the same time, larger sizes also take greater time to converge.
- Taking this into account, the batch size was **set to be 100**

c.2) Adding Batch Normalization - Part b) Architecture

In the following iterations, batch normalization was done before applying activation functions.



Batch normalization is applied to CNN layers only.

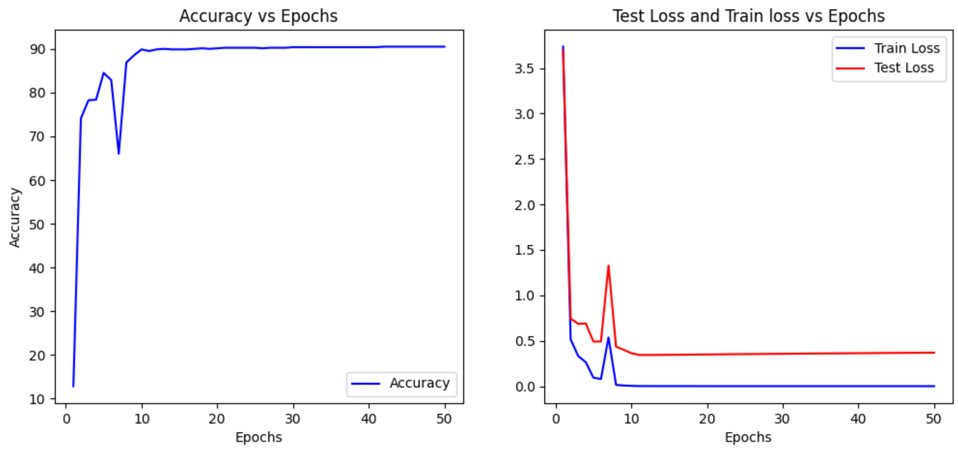
Parameters:

- batch size = 100
- learning rate = 0.001
- number of epochs = 50

Inferences:-

- Initially, accuracy fluctuates between 83-88%
    - The lower bound already surpasses the results obtained before batch normalization.
- achieved an accuracy of 88% by 40 epochs, then stagnated
    - takes longer time to converge, but is able to converge within the given time, with significant improvement
- Final accuracy stable at 88.6% by 50 epochs
- **Normalization also deals with overfitting.**
    - Overfitting reduced as apparent from the graph:- test loss becomes stable.
    - Without batch normalization, drastic oscillations in the train loss, which began to increase post a certain point
    - Post batch normalization, oscillations significantly reduced, and rate of increase in test loss very low.

💡 Batch Normalization therefore implemented here onwards.



Batch normalization for all the layers - CNN and linear layers

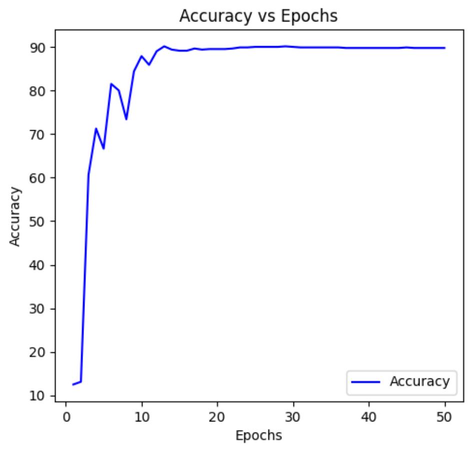
Parameters

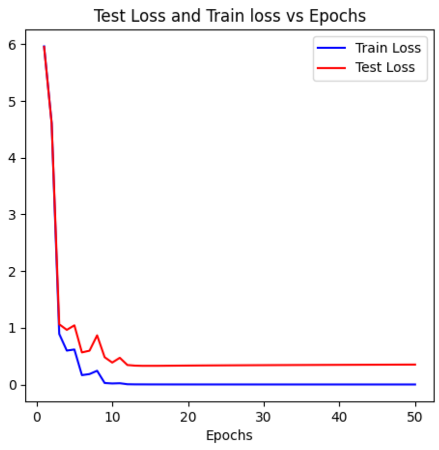
- batch size = 100

- learning rate = 0.001
- number of epochs = 50

Observations:-

- Normalizing linear layer output increased accuracy by 1%
- Significantly faster convergence observed - stable value of 90.5 % attained in 15 epochs.
- Post batch normalization on linear layer, train loss after 30-40 epochs converged to near 0 values
- Test loss also stabilized and converged to a low value





Batch normalization for all layers, Batch size increased

parameters -

- batch size = 200
  learning rate = 0.001
  number of epochs = 50

In section 1.1, the primary issue with higher batch size was slow rate of convergence, which batch normalization was able to mitigate.

at the same time since higher batch sizes were giving better accuracy, and since now it was feasible to get them to converge, we again tried to increase batch sizes

Result: Achieved nearly the same accuracy but at a greater number of epochs

💡 The batch size still kept 100 for models with batch normalizations

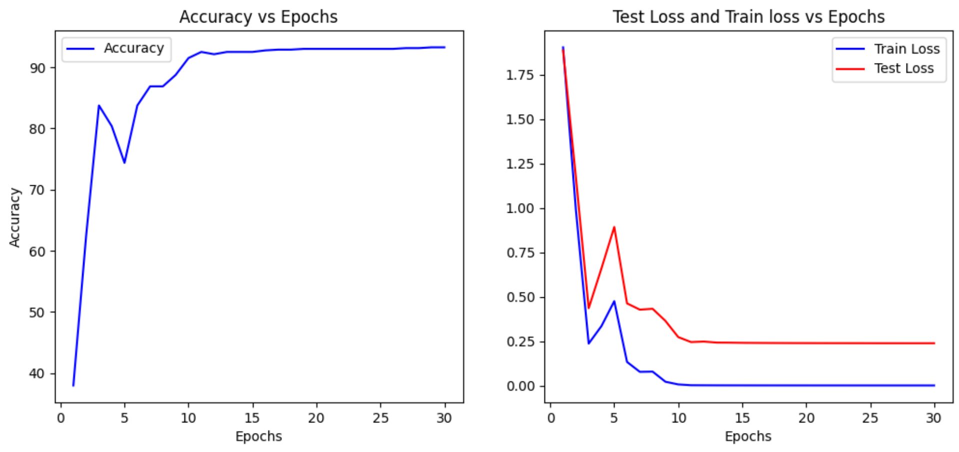
c.3) Adding hidden layers - finding the optimal architecture

The number of epochs were capped at 30 in the following iterations, as model parameters converged sufficiently before then.

Also for all the following iterations, the following parameters were taken:

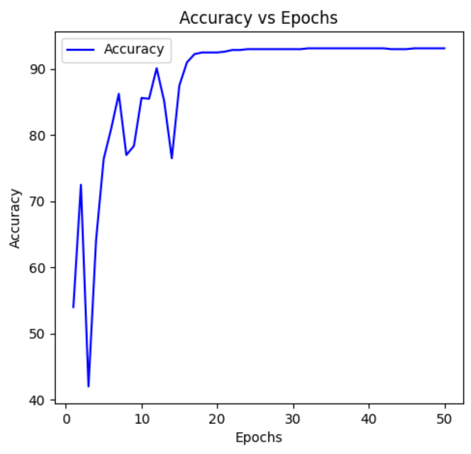
- batch size = 100
- learning rate = 0.001
- number of epochs = 30

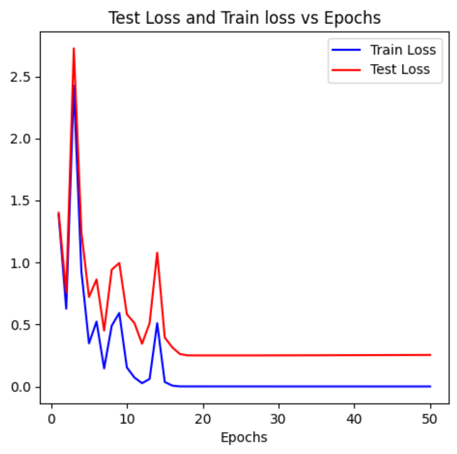
💡 in each layer we add, we double the amount of neurons



Added one more layer to CNN - output layers - 256

Achieved 93% accuracy - significant improvement



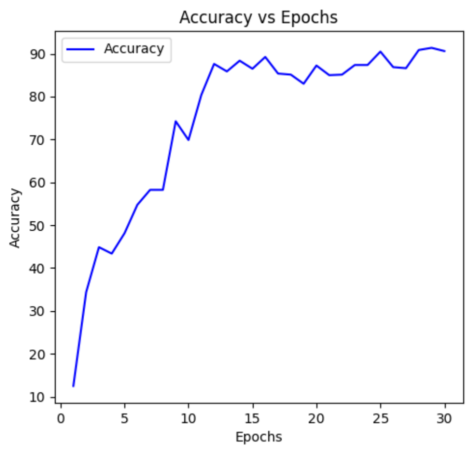


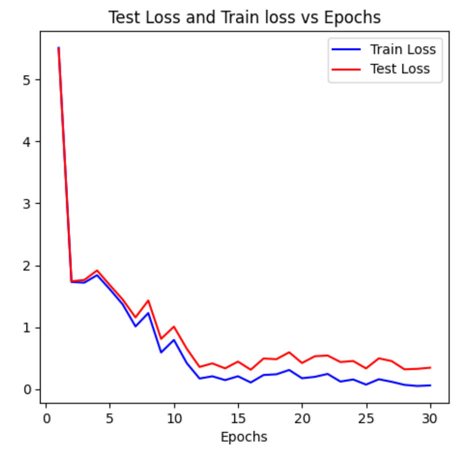
- Adding more layers to the CNN as adding layers gave very promising results
  Achieved 93% accuracy by 25 epochs
- output layers - 256 for convolution layer

Stable at 93.12 % accuracy

💡 Some other things tried at this point but not implemented after this:

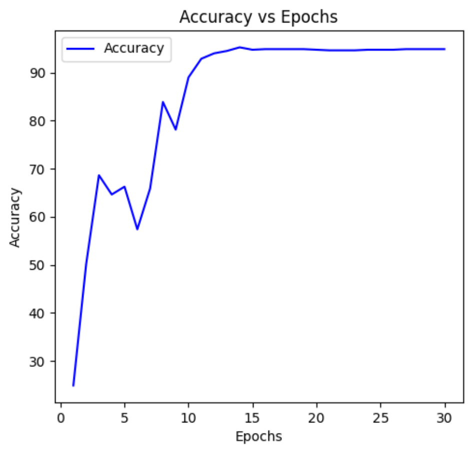
- We tried to decrease stride of the first layer, but due to significant increase in run time, this approach was dropped
- added another linear layer, but no significant difference obtained, on the contrary, accuracy reduced by 0.5%

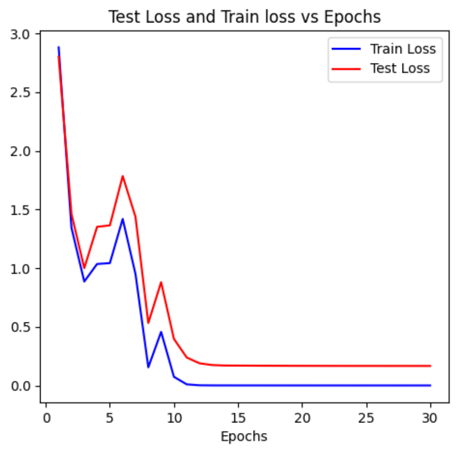




Added one more convolution layer - size 512 to the CNN model

94.5 % accuracy achieved in 15 epochs





Another Convolution layer - size 1024 - added

Accuracy of 95.25% obtained

Beyond this, adding another convolution made the model slower.

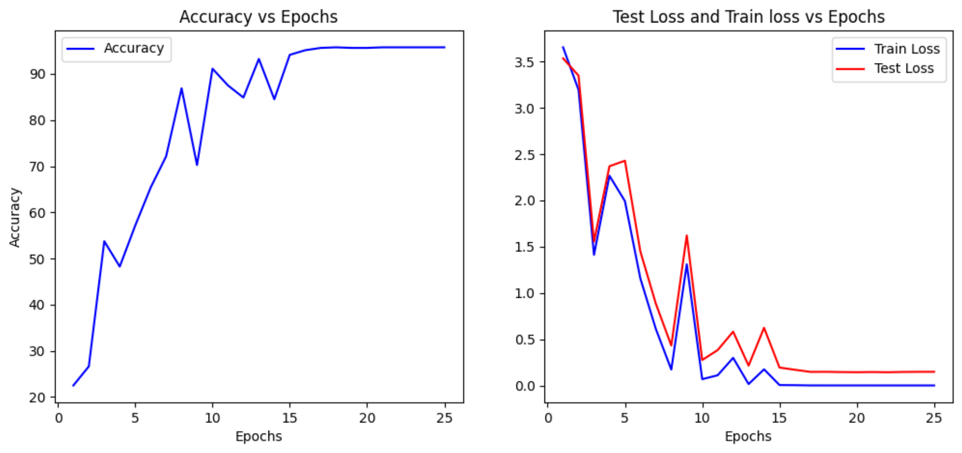
Therefore final model architecture:

- CNN layer - 32, kernel - 3
- CNN layer - 64, kernel - 3
- CNN layer - 128, kernel - 3
- CNN layer - 256, kernel - 3
- CNN layer - 512, kernel - 3
- CNN layer - 1024, kernel - 3

all CNN layers were followed by batch normalization, RELU , and then max pooling with kernel 2

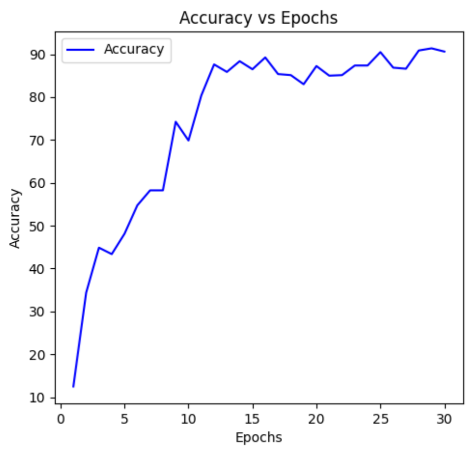
- linear layer - size - (1024*3*10, 512) with batch normalization, and RELU
- this linear layer also contains dropout which is discussed in the next section
- Final softmax layer size - (512, 8)

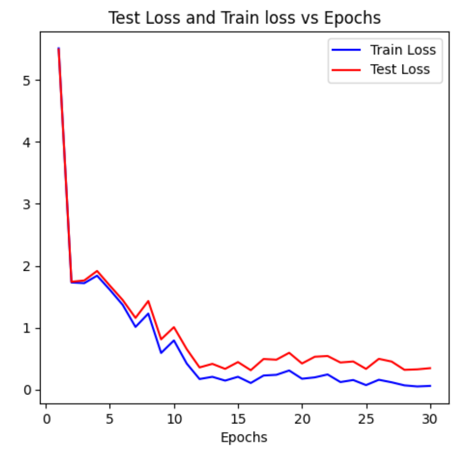
c.4) Adding Dropout



Tried dropout in the last layer:- probability p = 0.5

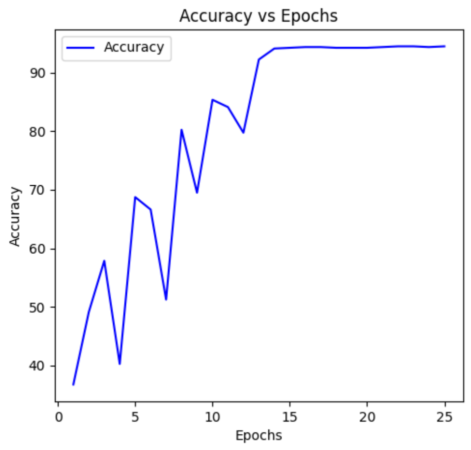
Achieved 95.7% within 20 epochs

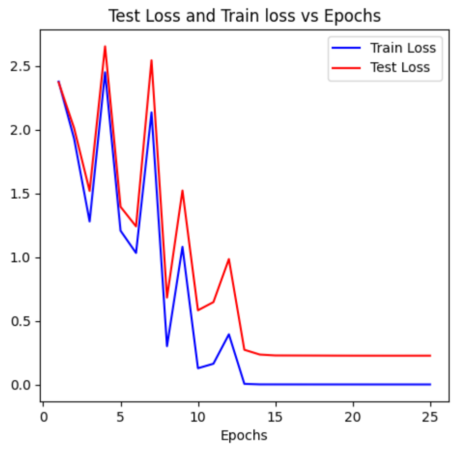




dropout in all the layers(dropout probability 0.4)

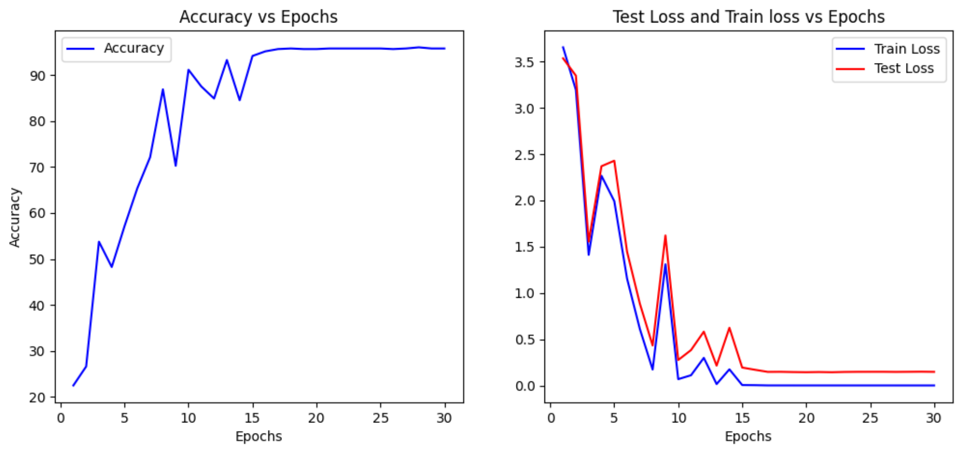
- very bad accuracy of only 90% by 20 epochs and still not stable(oscillated between 85-90%)
- Train loss also not decreasing significantly
- **Hence dropped**





💡 Applied laplacian edge detection kernel at the start of each forward prop
so that the CNN model takes in only the edges, not the whole image
Got stable at 94 %:- not a good method at most of the edges are lost at preprocessing

c.5) Final Model



as stated earlier -

💡 Beyond this, adding another convolution made the model slower.

Therefore final model architecture:

- CNN layer - 32, kernel - 3
- CNN layer - 64, kernel - 3
- CNN layer - 128, kernel - 3
- CNN layer - 256, kernel - 3
- CNN layer - 512, kernel - 3
- CNN layer - 1024, kernel - 3

all CNN layers were followed by batch normalization, RELU , and then max pooling with kernel 2

- linear layer - size - (1024∗3∗10, 512) with batch normalization, and RELU
- this linear layer also contains dropout in last linear layer with probability 0.5.
- Final softmax layer size - (512, 8)