Q 1.1

Softmax (x­­i) =

X is a vector and i is the index of each element . j is the range of elements in x.

Now consider x = x + c ,

=

=

=

From the above proof we can see that softmax is invariant to any translation c.

Softmax(x) = softmax (x + c)

The value -max( c ) , when used makes the enumerator range from (0 ,1] . So , with the translation , it helps to avoid the overflow during calculation and obtain the same result as it is invariant to translation.

Q 1.2 :

* The range of each element is 0 to 1 , not including 0. Sum over all elements is 1.
* Probability distribution with same number of elements in x.
* The first step is to calculate the exponential of each element in x. The second step to calculate the total value of all elements. The third step is to calculate the ratio for each element in x by dividing the above 2 from which you obtain the probability distribution of the elements in x.

1.3 :

Consider two successive linear regression layers.

h­1 = w1Tx + b1

h2 = w2Th1 + b2

now,

h2 = W2T(w1Tx + b1) + b2

considering layer3 ,

h3  = W3Th2 + b3

h3  = W3T(**W2T(w1Tx + b1)** )+ b2 + b3

from this, we can see that , without a non-linear Activation function, the multilayered Neural network is essentially a linear regression.

1.4.

Derivative of Sigmoid ,

= =

= \* 1-

=

Thus the derivative of sigmoid is expressed without using x directly.

1.5.

or

Derivatives with respect to scalars,

=

=

=

By taking derivative for matrices,

= =

= =

= =

1.6

1. Considering the sigmoid activation function, we know that the derivative of sigmoid is (0,1].

As the gradient of the activation update scales the backpropagation update and the value of the gradients is less than 1 and greater than 0 , the value of the gradient as it is multiplied over multiple values less than 1, tends to 0 as the layers keep increasing meaning the gradient **vanishes.**

1. The derivative of both the functions are seen below.

Tanh =

Add and subtract e­-2x  on the numerator, we get

=

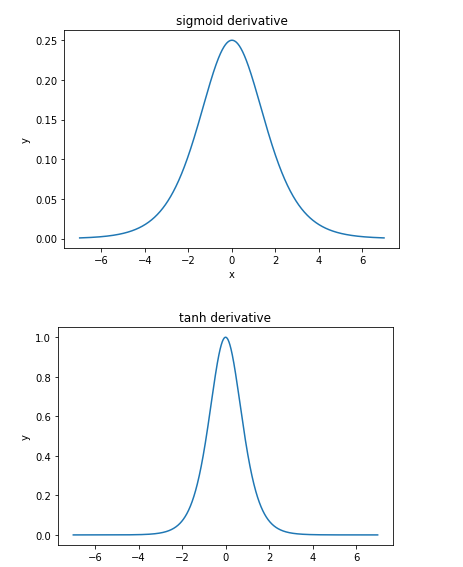
=

=

It can be see that , the output of tanh ranges from (-1,1) and is centered around 0 (symmetric) which helps to converge a lot faster than sigmoid.

Also, the gradient is also large at 0 for tanh than sigmoid which helps in reducing the chance In reaching a vanishing gradient. Thus, tanh is preferred over sigmoid for activation.

1. Comparison of derivatives of sigmoid and tanh is shown below.

****

Tanh has less of a vanishing gradient problem because of the higher gradient at 0 than sigmoid (can be seen from the image). Thus , it might take very larger number of layers to make the gradient vanish for tanh compared to sigmoid.

1. Tanh can be expressed in terms of sigmoid as ,

**tanh(x)=2σ(2x) – 1**

Q2.1.1

Having zero weights to start with will prevent the network from learning. The errors backpropagated through the network is proportional to the value of the weights. If all the weights are the same, then the backpropagated errors will be the same, and consequently, all of the weights will be updated by the same amount. To avoid this symmetry problem, the initial weights to the network should be unequal.

Thus, the output will be the same as input as the gradients were the same and the weights weren’t updated.

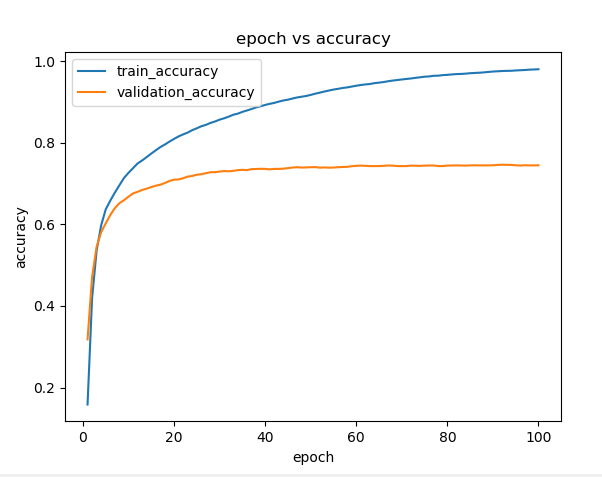
2.1.2

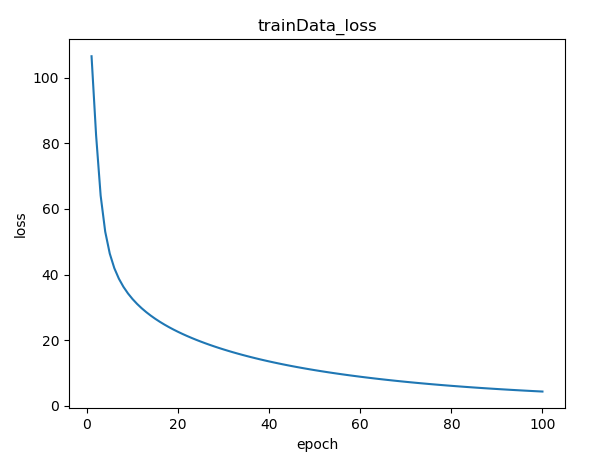
The random initialization increase the chance to reach to the optimal parameters that can lead to low loss. It helps to find an optimal set of weights for the mapping functions from the input data to the output.

The variance in the gradient has to be same for all the layers, thus the scaling depending on the layer size is done to make that possible.

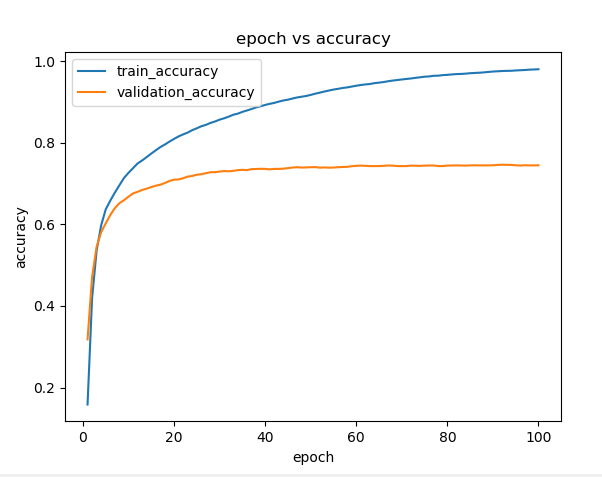
3.1.1:

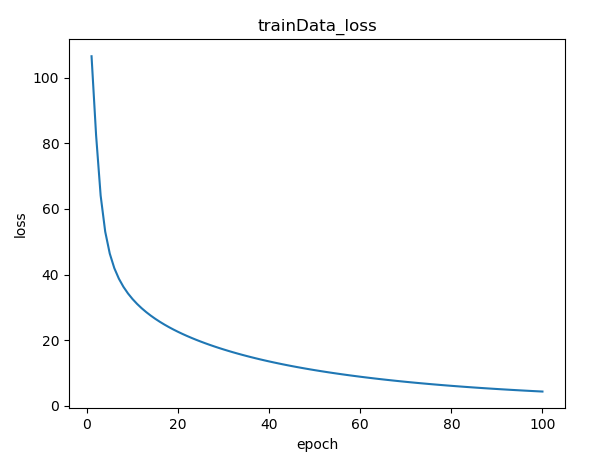
Learning rate = **0.003**



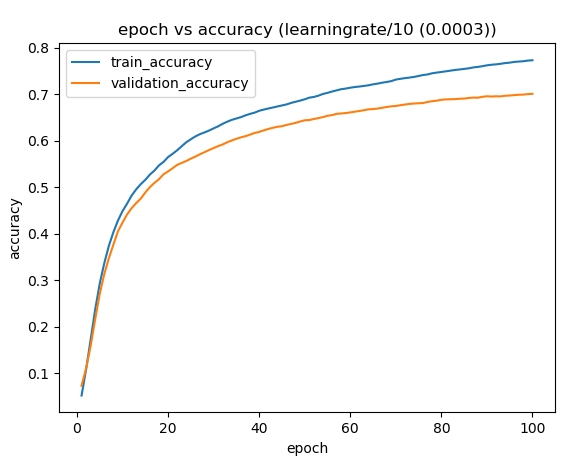


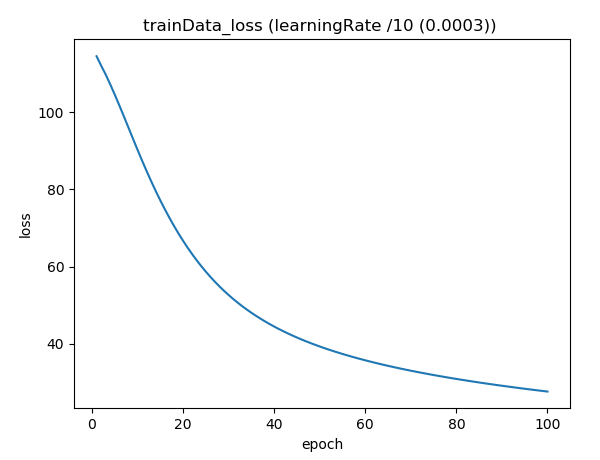
3.1.2

Default learning rate as 3.1.1 : **0.003**

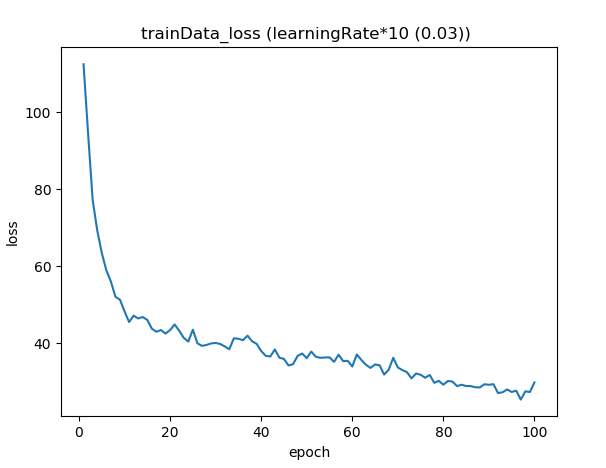


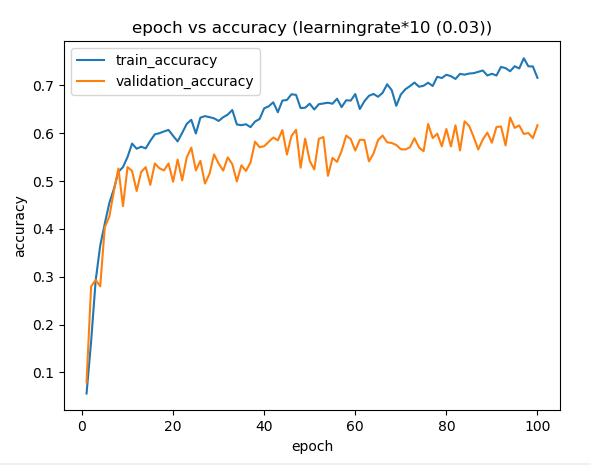
**2. Learning rate/10:**





**3. Learning rate/10**





Accuracy of the network for the best learning rate , 0.003 = 77%

When the optimum learning rate is not used, the global optimum takes relatively longer time to be obtained . In some cases it might get into a local minima and global minima can never be obtained. In our case, the accuracy dropped down from 97% for a learning rate of 0.003 to 75-80% for both the modified learning rates (i.e.) 0.03 and 0.0003.

3.1.3 : Weight visualization:

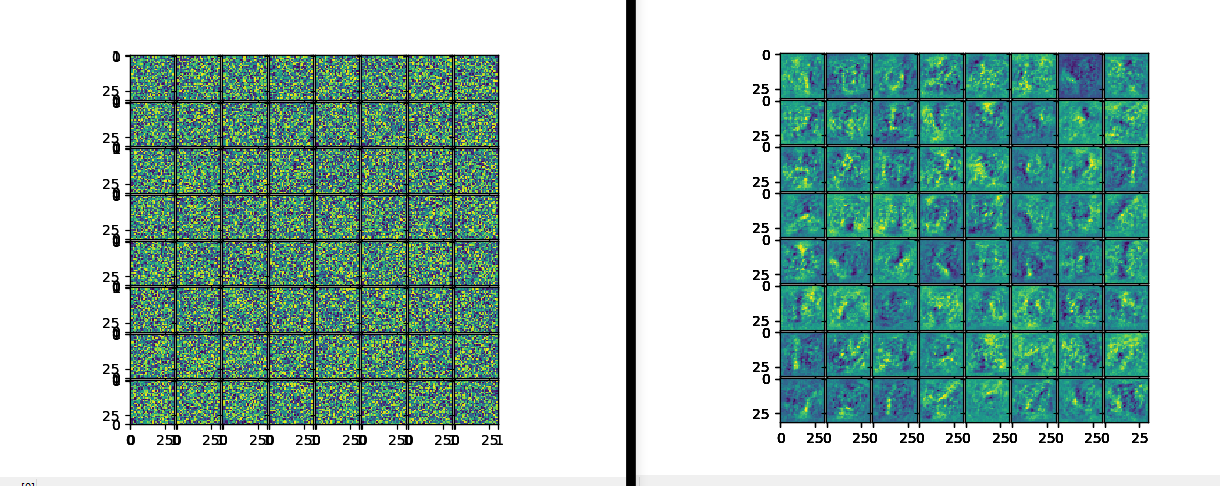
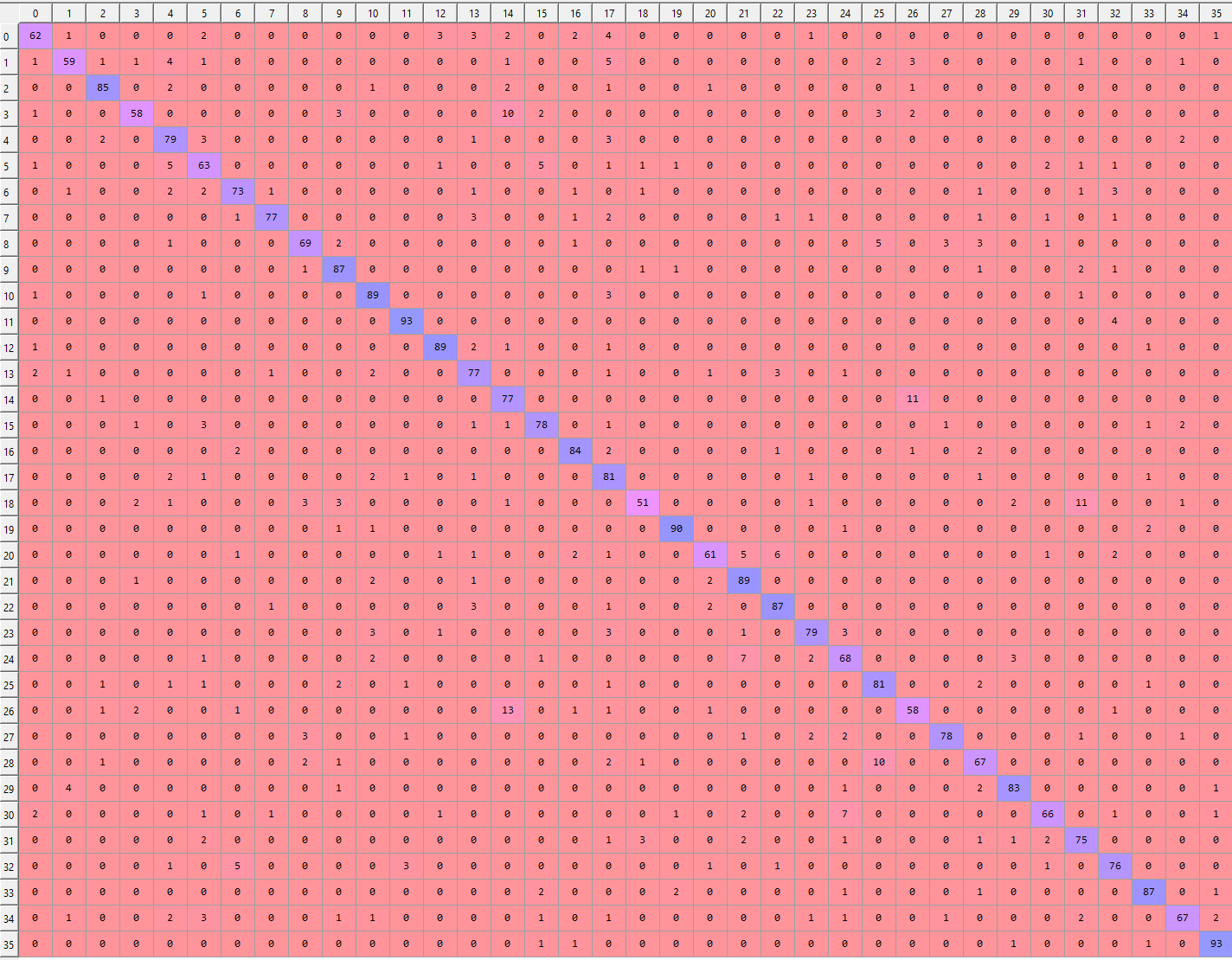


Fig.Weights during Initialization Fig.weights after training

Yes, I do notice the patterns .From the images it can be seen that the training has changed the weights from being random and unorderly to having a structure and pattern with lesser randomness.

3.1.4 X axis = LABEL

Y axis = Predicted class

13 is the most number of misclassifications happened between 2 classes. (i.e.) 14th class , has been predicted as 26th and 3rd class, meaning N has been wrongly classified as Z and D respectively.

4.Extract Text from Images

4.1 The two assumptions which were taken from the pictures were,

1. The structure of the letters is continuous, meaning the lines of the letters are attached to each other/ closed.

2. All the letters in the cropped out images are upper case letters and the training is not done for the lower case letters. As the training is not done for the lower case letters, the network wont predict anything other than the 26 Upper case and 0- 9 numbers.

The figure below has some of the examples on which the Network will essentially fail.



Fig. sample images for 4.1

4.3

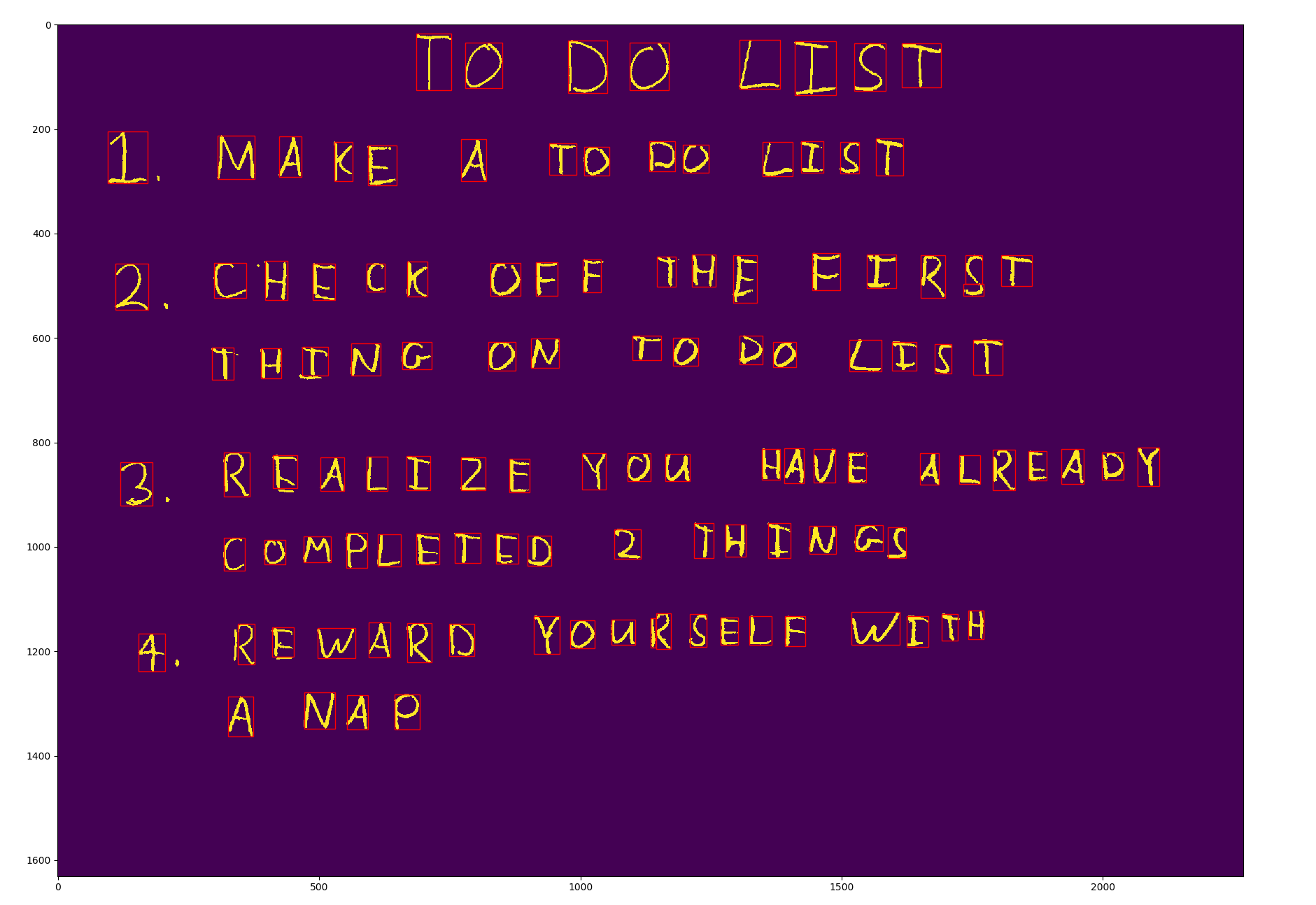


Fig. Image 1 used for training

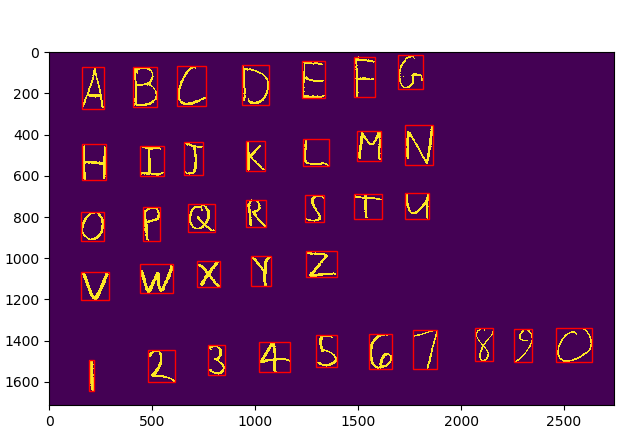


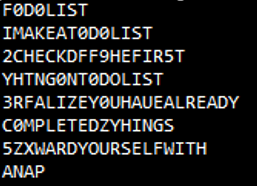
Fig. Image 2 used for training



Fig. Image 3 used for training



Fig. Image 4 used for training

4.4

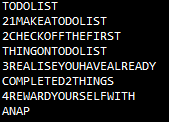
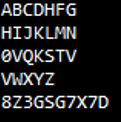


Fig: Predicted Fig: Original label

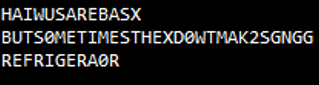
Fig: Predicted Fig: Original label



Fig: Predicted Fig: Original label

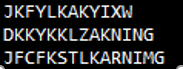
 

Fig: Predicted Fig: Original label

The above are the output of the segmented bounded box of the letters.

5.2

From the plot it can be seen that the loss decreases drastically in the beginning and it becomes gradual and converges at the end.

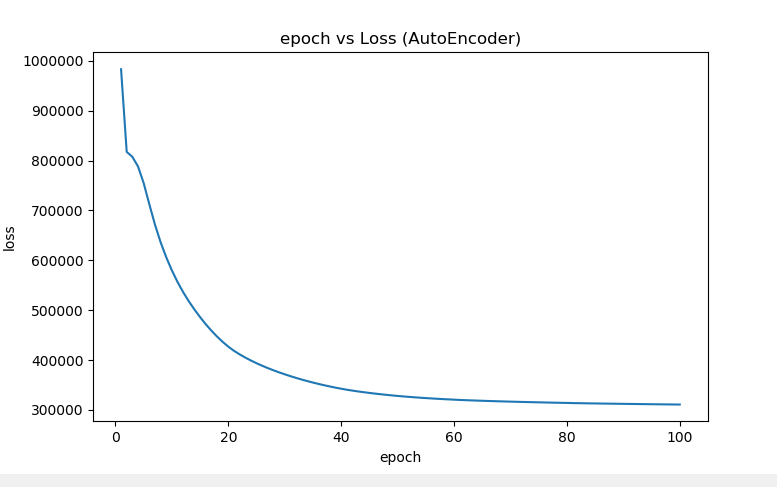
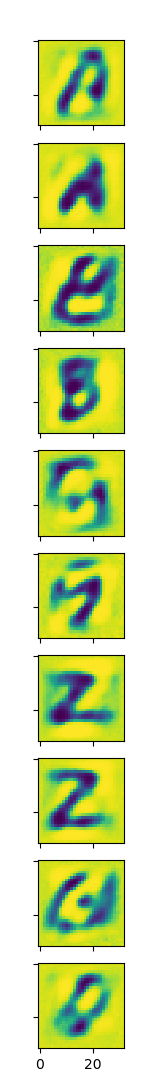
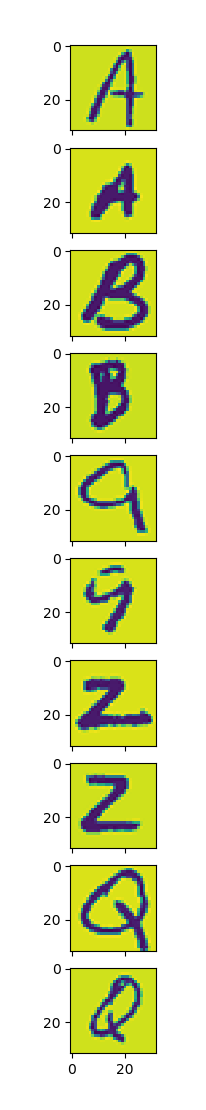


Fig. Epoch vs CrossEntropy loss

5.3.1



The output images are blurred to an extent and there is quite a bit of noise around the pixels, but it is still decipherable.

5.3.2

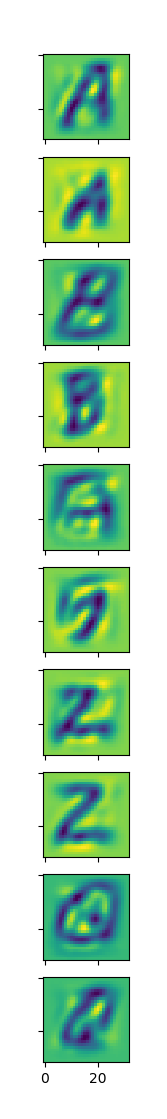
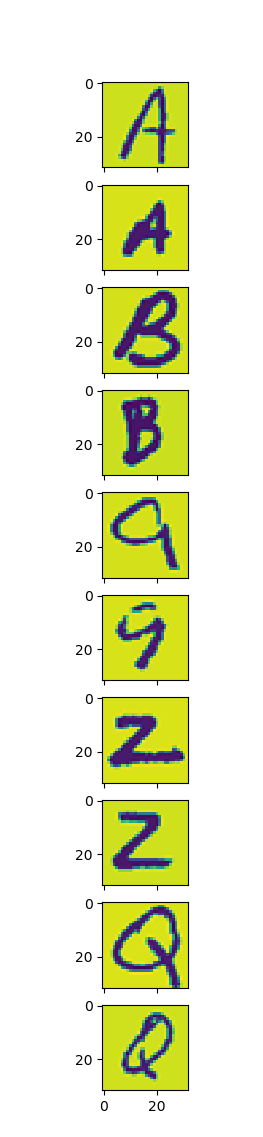
PSNR average **= 14.658335624612466**

6.1

The projection matrix has a shape of **(1024,32)**

And has a rank of **32**

6.2



There are noises in the reconstructed images around the background compared to the original images. The appearances of the characters are roughly the same as the original one. Also, the reconstruction result from autoencoder looks better than PCA.

**PSNR\_average with PCA = 15.471112009223548**

7.1.1

Given are the loss and accuracy plots of trainset for 7.1.1

**Fully Connected network- NIST36**

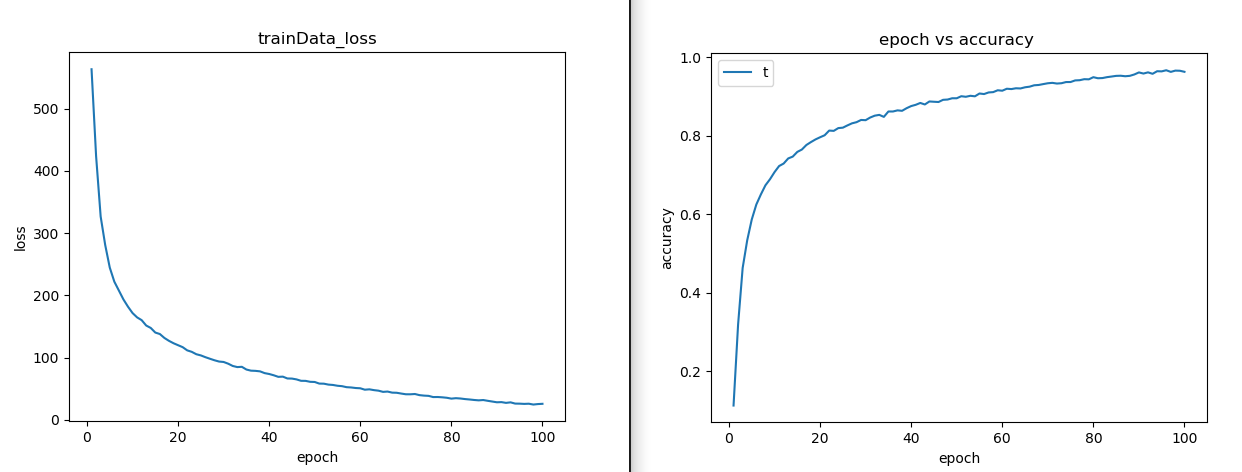


Fig. Epoch vs CrossEntropy loss and Epoch vs Accuracy

Test Accuracy: 80.7 %

7.1.2

Given are the loss and accuracy plots of trainset for 7.1.2

**CNN- MNIST**

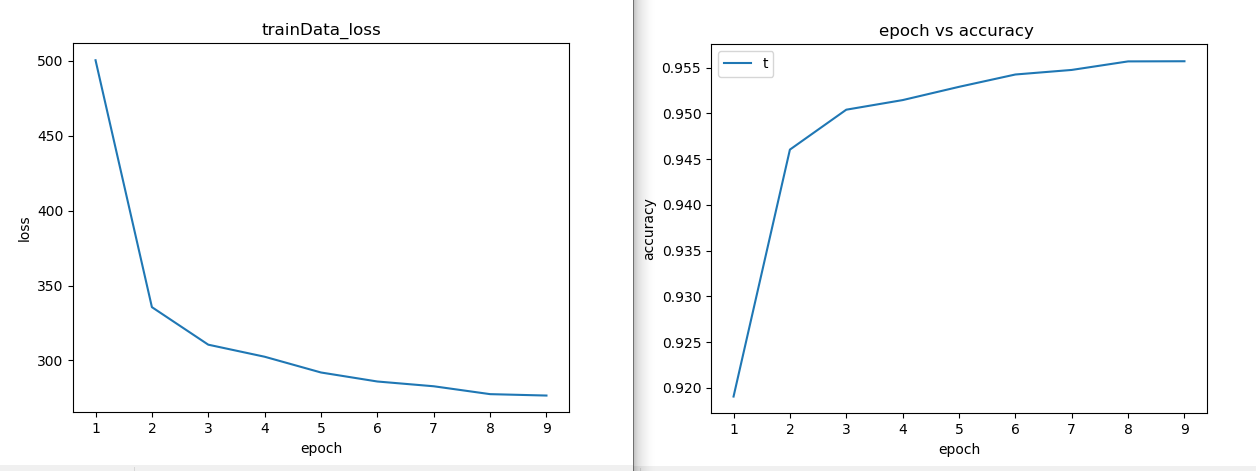
****

Fig. Epoch vs CrossEntropy loss and Epoch vs Accuracy

7.1.3

Given are the loss and accuracy plots of trainset for 7.1.3

Test accuracy: 88.333 %

**CNN – NIST36**

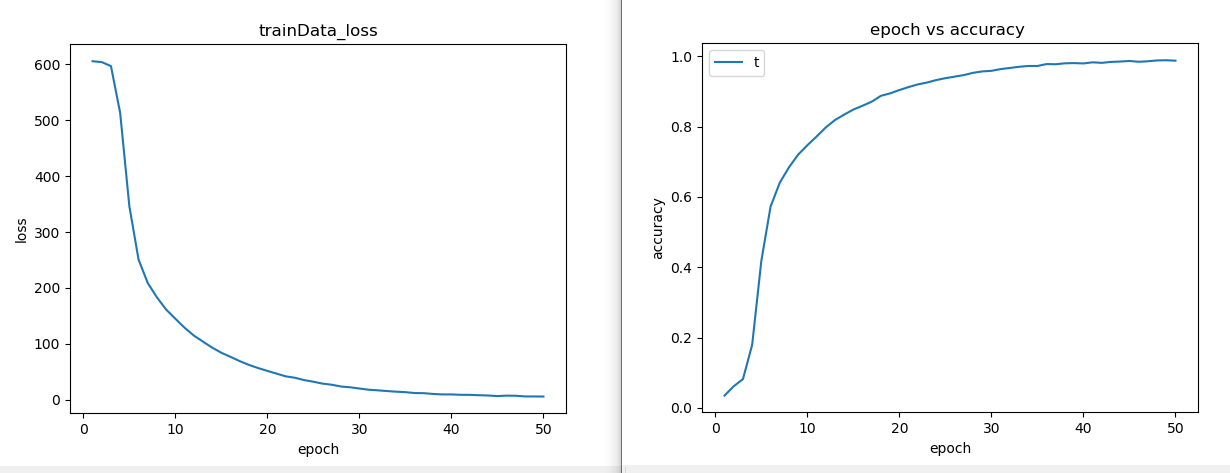
****

Fig. Epoch vs CrossEntropy loss and Epoch vs Accuracy

7.1.4.Given are the loss and accuracy plots of trainset for 7.1.4

**CNN – EMNIST**

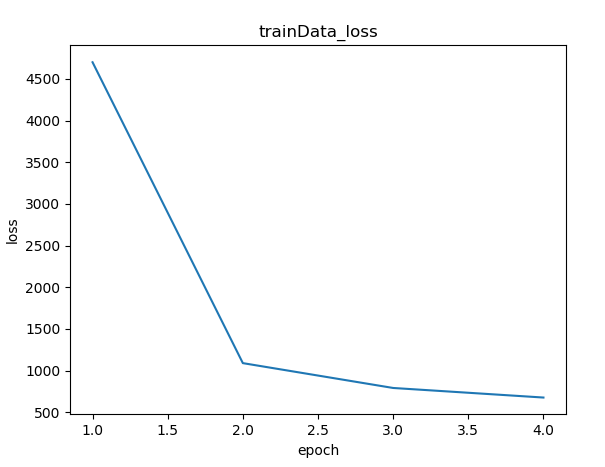


Fig. Epoch vs Accuracy

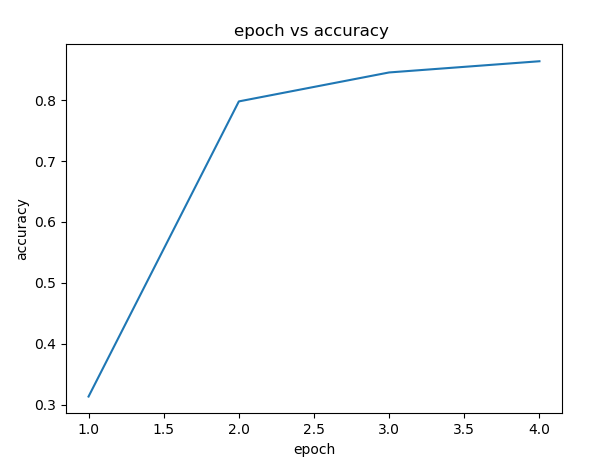
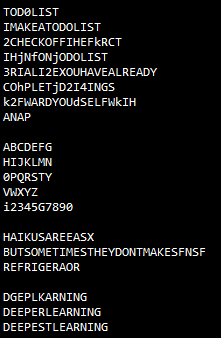


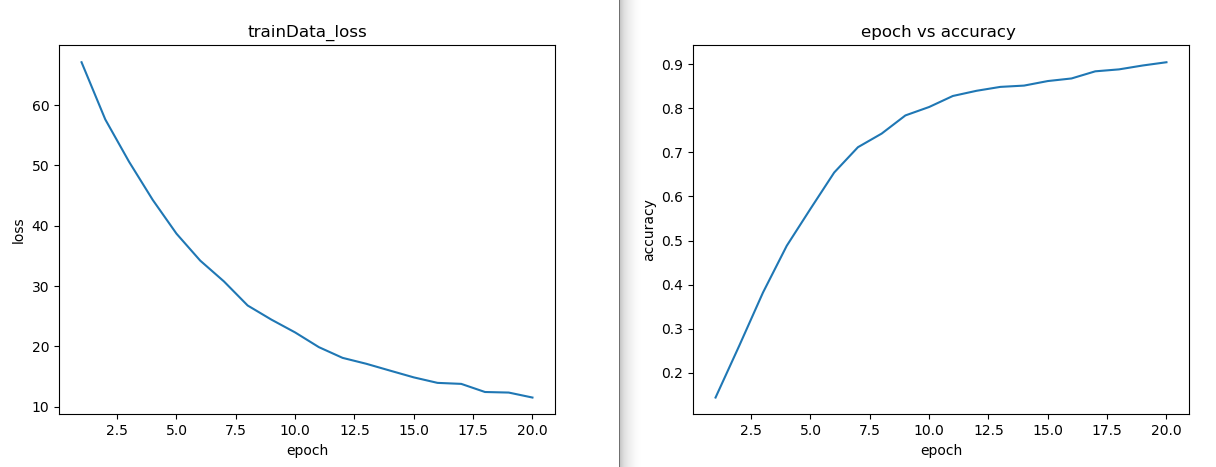
Fig. Epoch vs CrossEntropy loss



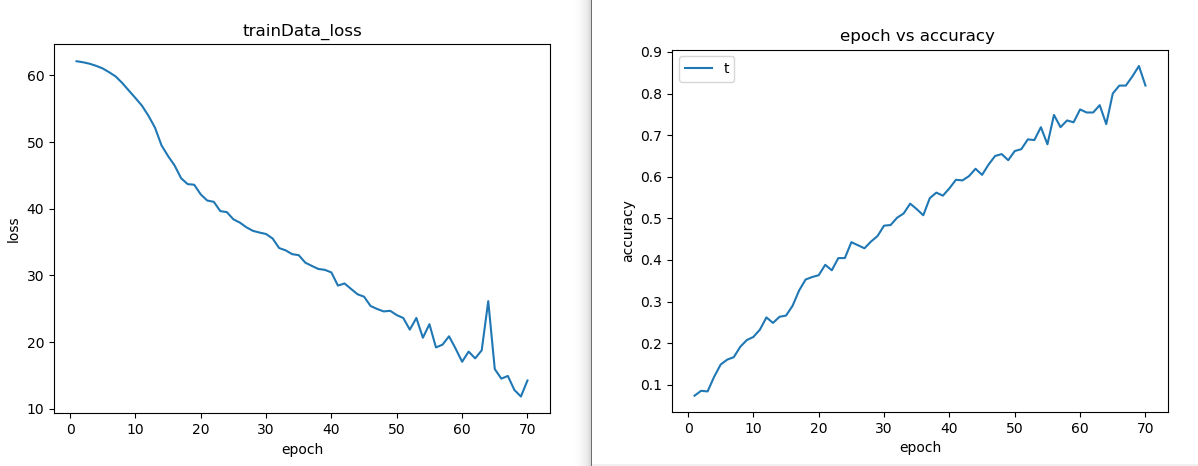
7.2 FINE TUNING:

Given are the loss and accuracy plots of trainset for 7.2 – SuffleNet and Custom made Network.

ShuffleNet- Fine tuned:



CustomNet Self-Made:



From the graphs, it can be seen that the self tuned network takes longer time to converge and has a lot of fluctuation when compared to the Fine tuned shuffleNet which has a steeper loss slope and converges faster than the CustomNet.