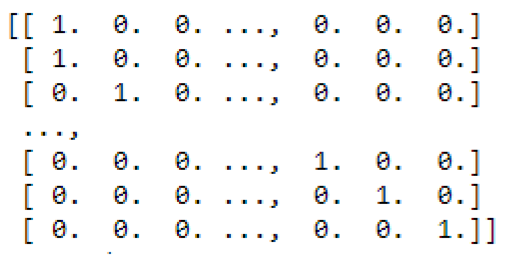
# Neural Networks (Assignment 3):

Given the data set contains features extracted from the utterances of four human speakers saying the characters \a" to \z" several times. There are 26 classes – one for each characters. The test set contains the same features extracted from the utterances of a fifth speaker. From the original data set, we only use the 300 features for our classification task. The main objective here is to finding a suitable neural network to aptly classify the above dataset with the minimum possible test and train errors and with the least misclassification rate.

## Tasks:

a): The given class range of range 26 has been applied with one – hot encoding as shown below to enable multi - class classification. The standard scalar() from sklearn has been used to apply one – hot encoding of the class labels.

One – hot encoding of the class labels



```
[[ 1.  0.  0. ...,  0.  0.  0.]
 [ 1.  0.  0. ...,  0.  0.  0.]
 [ 0.  1.  0. ...,  0.  0.  0.]
 ...,
 [ 0.  0.  0. ...,  1.  0.  0.]
 [ 0.  0.  0. ...,  0.  1.  0.]
 [ 0.  0.  0. ...,  0.  0.  1.]]
```

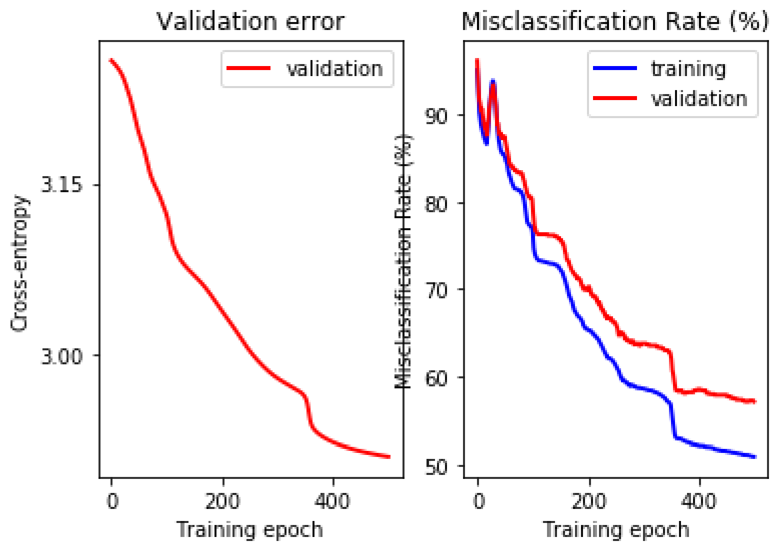
In addition the given training data was divided into : training $(\tau, 70\%)$ , validation $(\nu, 15\%)$ and early stopping $(\varepsilon, 15\%)$ and each was standardized with mean and variance of set $\tau$.

b): In the next step the performance comparison of the standard classification algorithms such as Stochastic Gradient Descent, RMSProp and Adamoptimizer was analysed. A simple network of one hidden layer with 20 neurons has been assumed. The hidden layer was applied with sigmoidal activation function by default. Also the training and evaluation has been performed using tensorflow libraries in mini batches of batch size 40 in order to ensure quicker convergence. The training has been covered for over 500 epochs for each run, and the mean misclassification rate on $\nu$ and the number of epochs on each run over 10 runs has also been calculated. Also, the learning rate has been assumed as per the best performances of the algorithm involved. The evaluation results from different algorithms are shown hereinafter:

## SGD:

SGD with learning rate of 0.001 was found to be very slow in convergence. Hence a learning rate of 0.1 was assumed for SGD and the performances are shown in fig.1

```
Result over a single run
Stopped at: 499
```



```
Results over 10 independent runs
No. epoch over 0 th run: 499
No. epoch over 1 th run: 499
No. epoch over 2 th run: 499
No. epoch over 3 th run: 499
No. epoch over 4 th run: 134
No. epoch over 5 th run: 499
No. epoch over 6 th run: 499
No. epoch over 7 th run: 499
No. epoch over 8 th run: 159
No. epoch over 9 th run: 499
Mean error on V: 73.0982894897
```

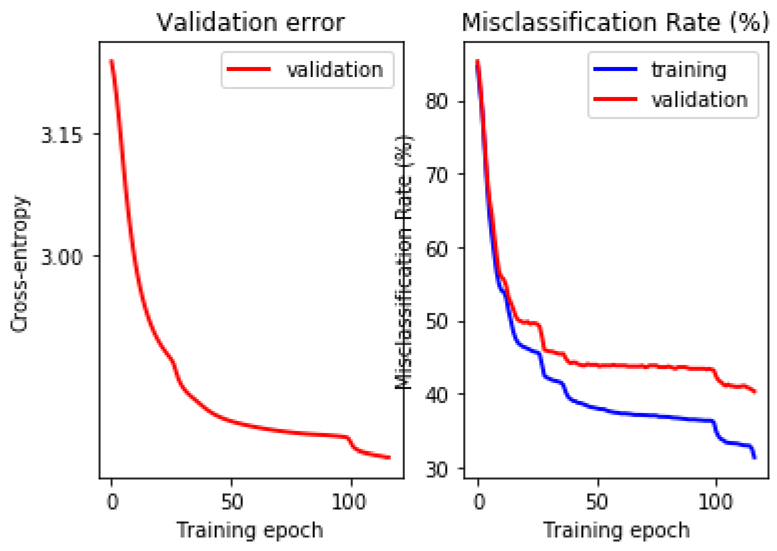
Fig – 1: SGD with learning rate 0.1

It was found that the mean misclassification rate on $v$ was around 73% and the algorithm tends to take more run towards convergence.

**AdamOptimizer:**

The Adam optimizer was run with learning rate 0.001 as choosing a higher value tended to make the algorithm somewhat unstable in performance. The performances are as follows:

```
Result over a single run
Stopped at: 116
```



```
Results over 10 independent runs
No. epoch over 0 th run: 167
No. epoch over 1 th run: 212
No. epoch over 2 th run: 296
No. epoch over 3 th run: 278
No. epoch over 4 th run: 248
No. epoch over 5 th run: 282
No. epoch over 6 th run: 240
No. epoch over 7 th run: 287
No. epoch over 8 th run: 260
No. epoch over 9 th run: 192
Mean % misclassification V: 41.7628196716
```

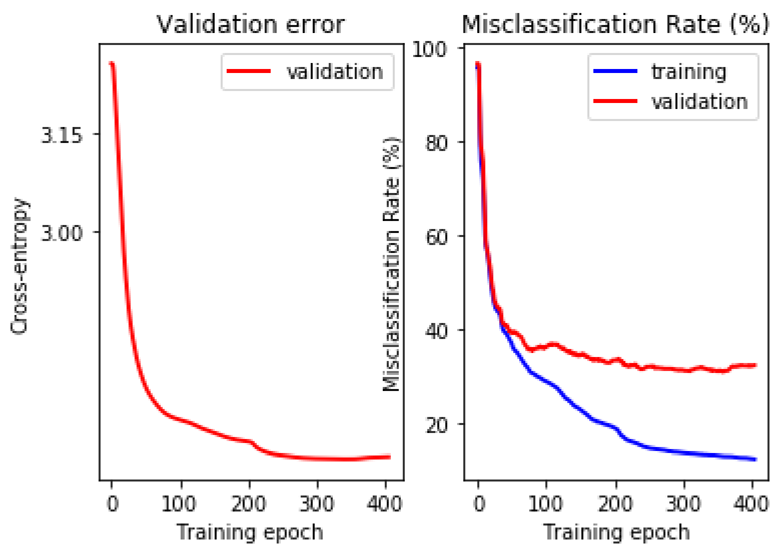
Fig – 2: Adam with learning rate 0.001

The results were satisfactory with mean misclassification rate on $v$ at around 41.76% and with average number of epochs for convergence at each run at around 250.

**RMSProp:**

Similarly the RMSProp was also analysed with a learning rate of 0.001 for a stable performance. The results could be seen hereinafter:

```
Result over a single run
Stopped at: 404
```



```
Results over 10 independent runs
No. epoch over 0 th run: 228
No. epoch over 1 th run: 273
No. epoch over 2 th run: 190
No. epoch over 3 th run: 187
No. epoch over 4 th run: 183
No. epoch over 5 th run: 183
No. epoch over 6 th run: 214
No. epoch over 7 th run: 178
No. epoch over 8 th run: 182
No. epoch over 9 th run: 330
Mean % misclassification V: 41.217950058
```

Fig – 3: RMSprop with learning rate 0.001

The performances stood at 41.26% of mean misclassification rate on $v$ over 10 runs and an average number of epochs slightly higher than 200.

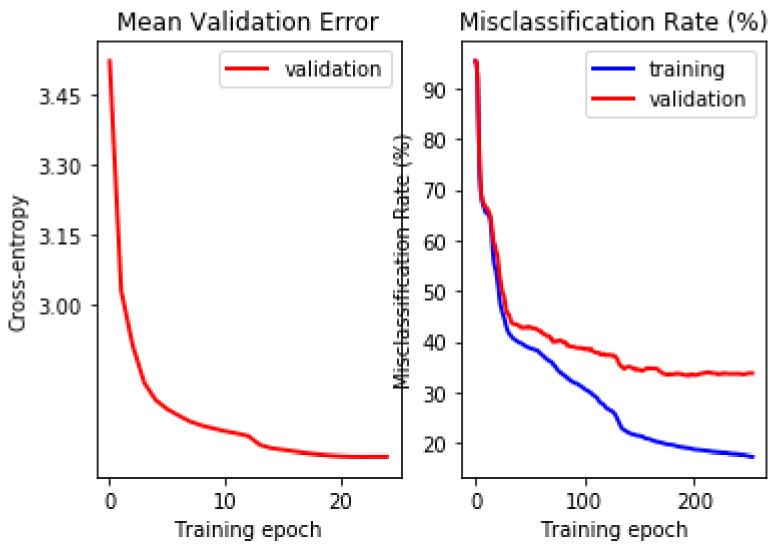
From the above performances the SGD was discarded because of costlier computations involved and higher % misclassification. The performances of RMSProp and Adam was comparable to each other. However RMSProp performed slightly better than Adam providing a relatively lesser % misclassification over multiple tests. Therefore RMSProp with learning rate 0.001 was chosen for the final network.

c): The test for the better activation has been performed. The optimizer assumed was RMSProp as could be noted from previous section. The results are summarized below:

**Sigmoid:**

When the hidden layer was activated with sigmoidal function the following results were seen:

```
Stopped at: 253
```



```
No. epoch over 0 th run: 152
No. epoch over 1 th run: 132
No. epoch over 2 th run: 222
No. epoch over 3 th run: 158
No. epoch over 4 th run: 196
No. epoch over 5 th run: 291
No. epoch over 6 th run: 177
No. epoch over 7 th run: 288
No. epoch over 8 th run: 279
No. epoch over 9 th run: 281
Mean error on V: 38.1944454193
```

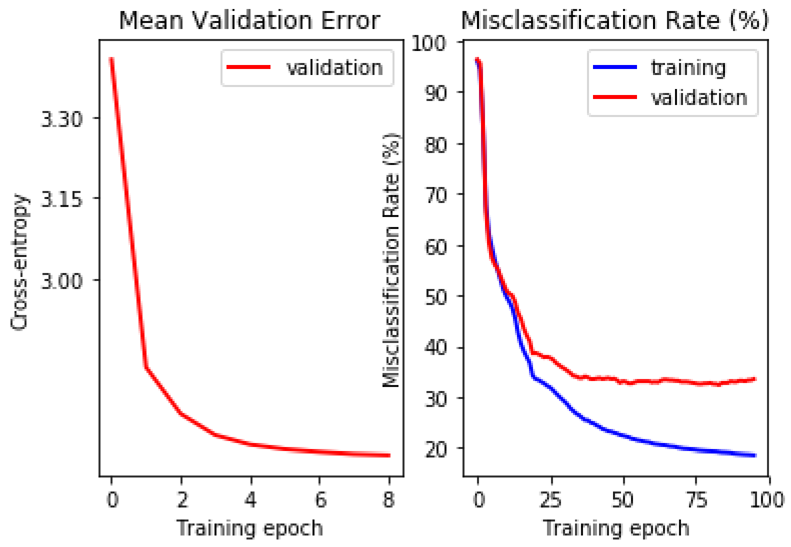
Fig – 4: Sigmoidal with RMSProp

The mean % misclassification observed was 38.19% and the average number of iterations towards convergence stood at just over 200.

**Tanh:**

Similarly for the tanh activation function the mean % misclassification observed was around 35.17% with average no. of epochs of about 110 as shown in fig. 5

```
Stopped at: 95
```



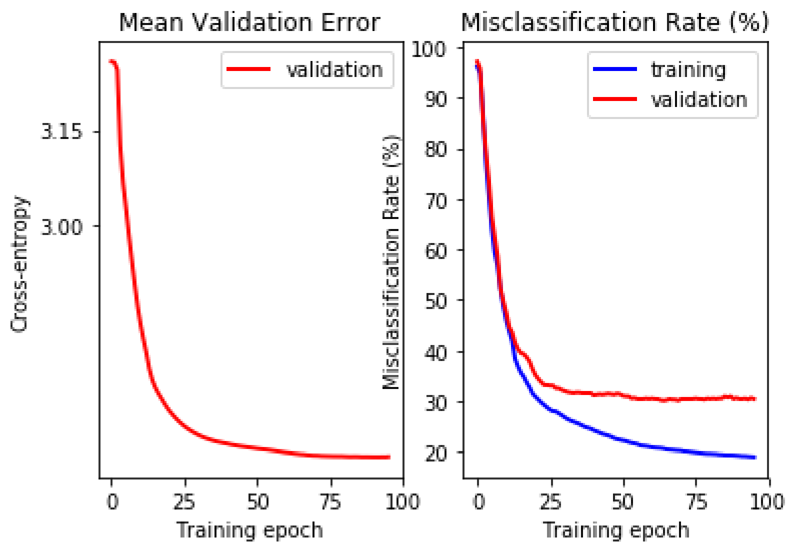
```
No. epoch over 0 th run: 92
No. epoch over 1 th run: 130
No. epoch over 2 th run: 104
No. epoch over 3 th run: 125
No. epoch over 4 th run: 112
No. epoch over 5 th run: 92
No. epoch over 6 th run: 123
No. epoch over 7 th run: 108
No. epoch over 8 th run: 113
No. epoch over 9 th run: 102
Mean error on V: 35.1709396362
```

Fig – 5: Tanh with RMSProp

**Relu:**

When tested the network with Relu activation function optimized by RMSProp, the below performance could be seen:

```
Result over a single run
Stopped at: 95
```



```
Results over 10 independent runs
No. epoch over 0 th run: 92
No. epoch over 1 th run: 86
No. epoch over 2 th run: 91
No. epoch over 3 th run: 102
No. epoch over 4 th run: 144
No. epoch over 5 th run: 68
No. epoch over 6 th run: 66
No. epoch over 7 th run: 85
No. epoch over 8 th run: 77
No. epoch over 9 th run: 54
Mean % misclassification on V: 34.4230764389
```

Fig – 6: Relu with RMSProp

Mean misclassification %: 34.42 and Avg. no. of epochs over 10 runs: around 90.

It could be seen that the performances of Relu and Tanh were better as compared to sigmoid in terms of misclassification % as well as the no. of epochs for convergence. The Relu, however, has outperformed tanh in no. epochs in which it takes only fewer steps than tanh towards convergence. Therefore we would pick out Relu for the final network shown in E).

d): The selections for best architectures were chosen among network with: 1 hidden layer – 20 neurons; 1 hidden layer – 150 neurons; 2 hidden layers – 20 neurons; 3 hidden layers – 20 neurons. And the performance of each network are summarized below. The performances over a single run has been summarized below:

```
Stopped at: 73
```

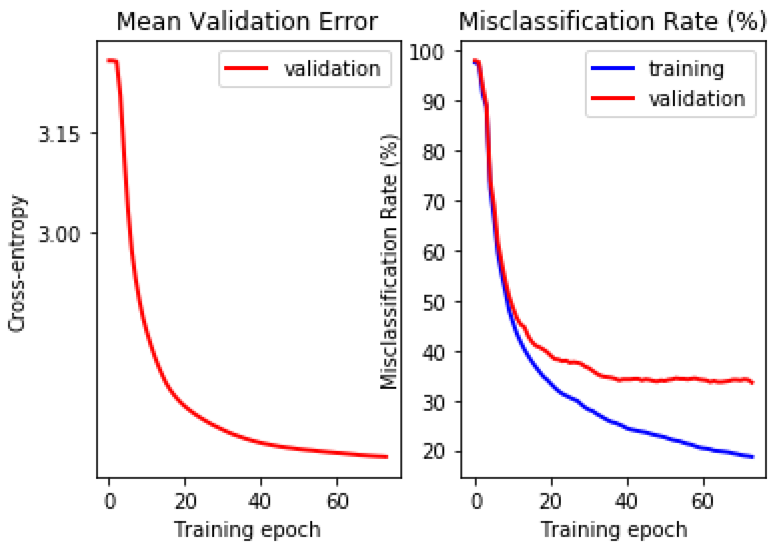


Fig – 7: One hidden layer – 20 Neurons

```
Stopped at: 38
```

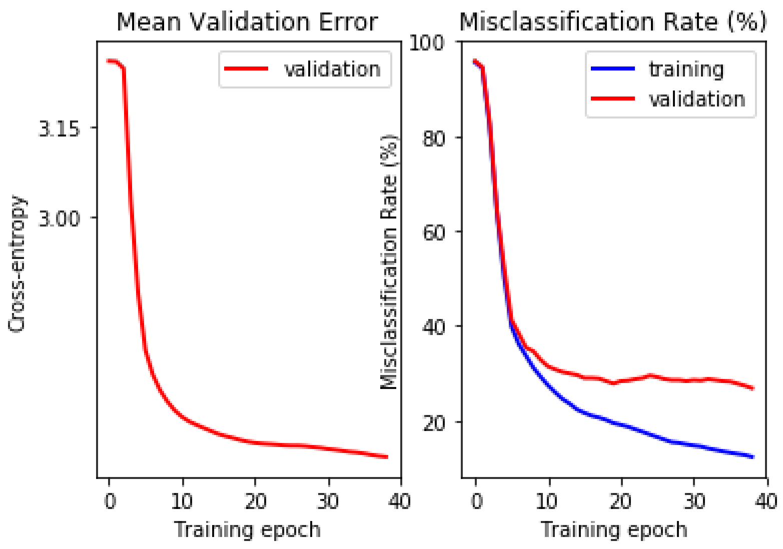


Fig – 8: One hidden layer – 150 Neurons

```
Stopped at: 104
```

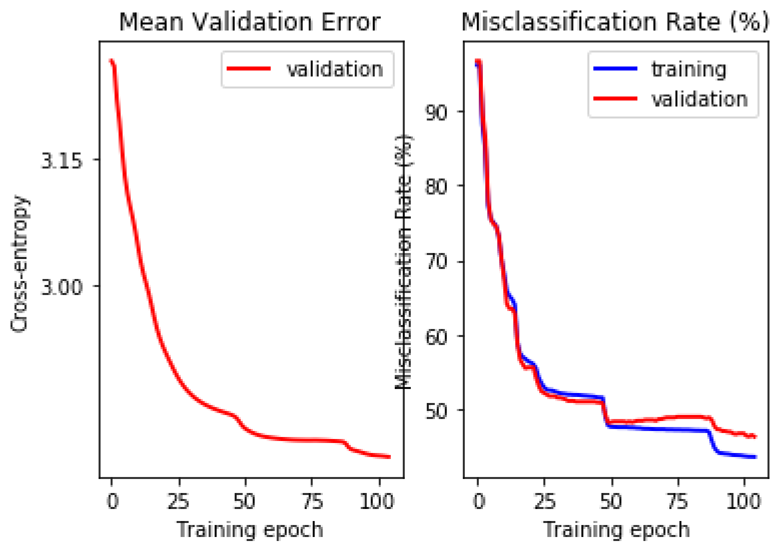


Fig – 9: three hidden layers – 20 Neurons

```
Stopped at: 120
```

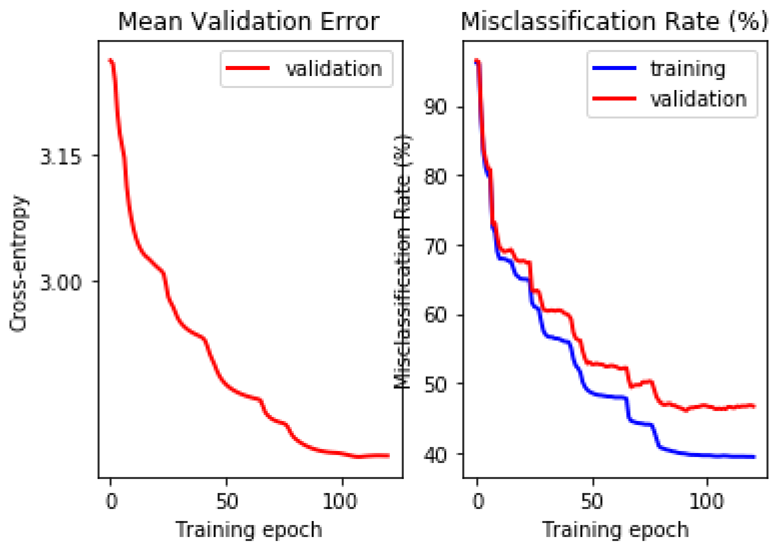


Fig – 10: two hidden layers – 20 Neurons

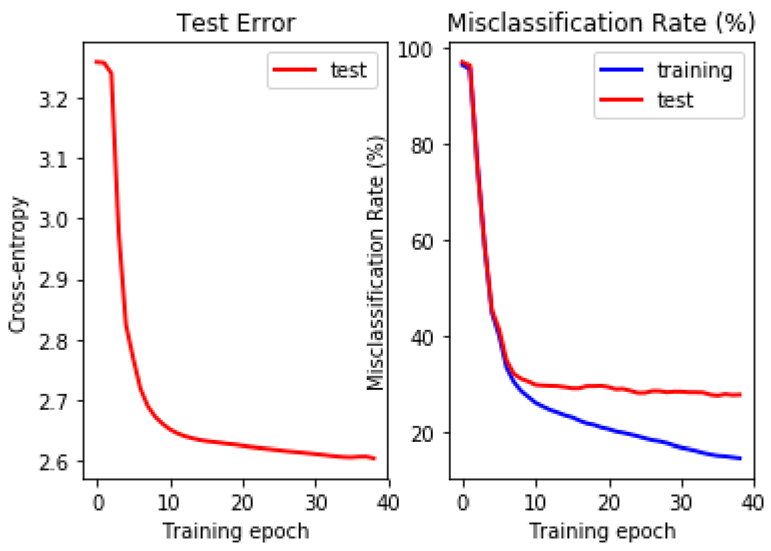
From fig.8 and fig.9 it can be concluded that network with single hidden layer performed better. However the larger amount of neurons, quicker was the convergence – network with 150 neurons took 38 iterations to converge as compared to network with 20 neurons which required about 73 iterations to converge. Henceforth the architecture with one hidden layer 150 neurons was selected for the final network.

e): From the above tests the final network is about to be realized with the below hyperparameters:

- RMSProp algo with learning rate 0.001
- Relu activation function for the hidden layer
- One hidden layer with 150 neurons.

The performance analysis is as follows:

```
Stopped at: 38
```



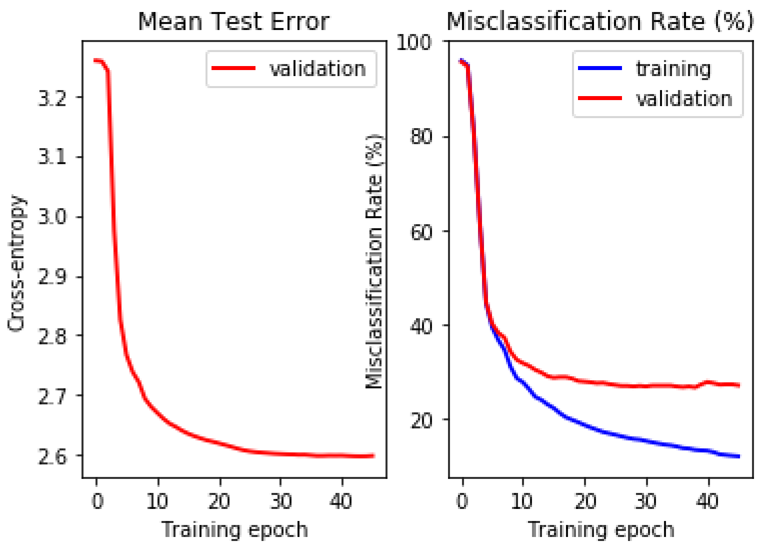
```
No. epoch over 0 th run: 35
No. epoch over 1 th run: 38
No. epoch over 2 th run: 36
No. epoch over 3 th run: 41
No. epoch over 4 th run: 42
No. epoch over 5 th run: 44
No. epoch over 6 th run: 34
No. epoch over 7 th run: 30
No. epoch over 8 th run: 30
No. epoch over 9 th run: 39
Mean % misclassification on test data: 29.0506732941
```

Fig – 11: Best Network

The above analysis shows the network performed better in terms of least possible % misclassification on test data as well as in terms of quicker convergence. The mean error observed on the test set has been found to be around 29.05%

f): Drop out has been implemented for the final network where the hidden layer nodes are dropped by a probability $p$ during training phase. The probability chosen for the below network has been 0.75. The performance summary is as follows:

```
Stopped at: 45
```



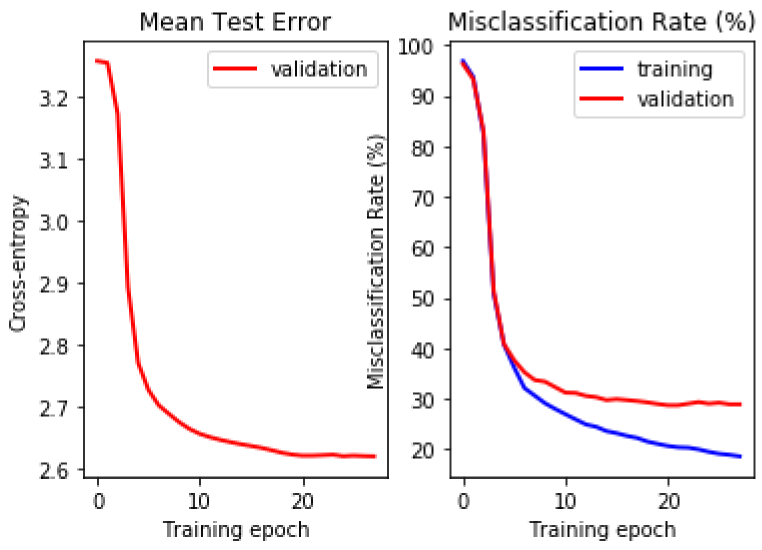
```
Results over 10 independent runs
No. epoch over 0 th run: 38
No. epoch over 1 th run: 34
No. epoch over 2 th run: 36
No. epoch over 3 th run: 46
No. epoch over 4 th run: 33
No. epoch over 5 th run: 45
No. epoch over 6 th run: 35
No. epoch over 7 th run: 31
No. epoch over 8 th run: 36
No. epoch over 9 th run: 32
Mean % misclassification V: 28.4092374802
```

Fig – 12: NN with dropout

There could be observed a considerable improvement in the performance of the network when drop out is applied without affecting the convergence rate. Mean % misclassification at test error stood at around 28.40%. Hence the application of drop out to NN tends to decrease to solve the overfitting problem to some extent by reducing the error in the test data.

g): The same network when used for dropping out with 'crelu' activation function showed a further marginal increase in the overall performance of the network.

```
Stopped at: 27
```



```
Results over 10 independent runs
No. epoch over 0 th run: 29
No. epoch over 1 th run: 27
No. epoch over 2 th run: 37
No. epoch over 3 th run: 33
No. epoch over 4 th run: 26
No. epoch over 5 th run: 26
No. epoch over 6 th run: 25
No. epoch over 7 th run: 24
No. epoch over 8 th run: 24
No. epoch over 9 th run: 22
Mean % misclassification V: 26.452854538
```

It can therefore be inferred that the application of drop out increases the overall performance of the network thereby producing a relatively lower misclassification error while also avoiding overfitting of data.