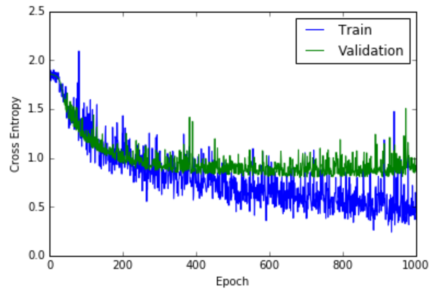
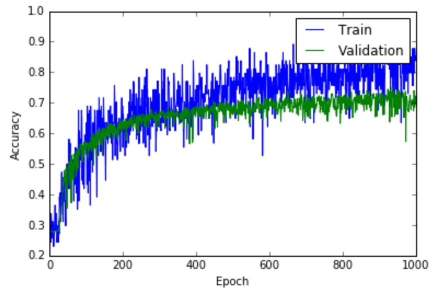
3.1 Basic Generalization

NN

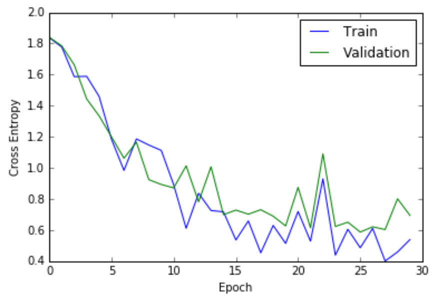
```
CE: Train 0.46786 Validation 0.85853 Test 0.86473
Acc: Train 0.82958 Validation 0.70644 Test 0.70130
Writing to nn_model.npz
Writing to nn_stats.npz
```

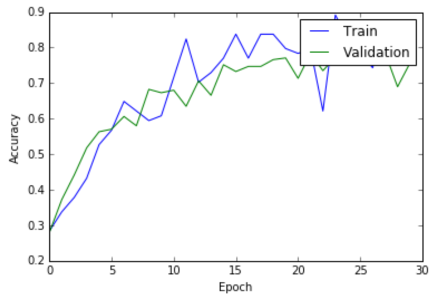




CNN

```
CE: Train 0.43891 Validation 0.69423 Test 0.60291
Acc: Train 0.84143 Validation 0.75656 Test 0.78442
```





The cross entropy is lower on the training set than on the validation set. And the accuracy is higher on the training set than on the validation set. This shows that the network is overfitting, which could be reduced by early stopping. Notice that the convolutional neural network performed better than the fully connected network despite having less epochs to train.

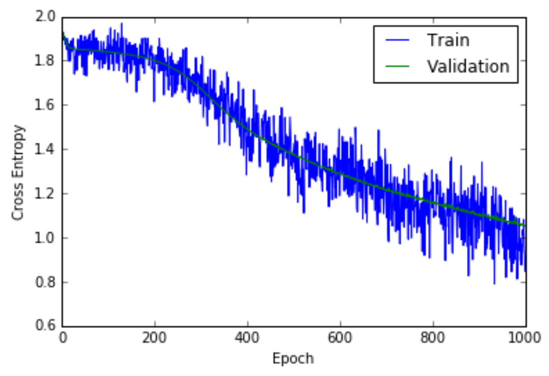
3.2 Optimization

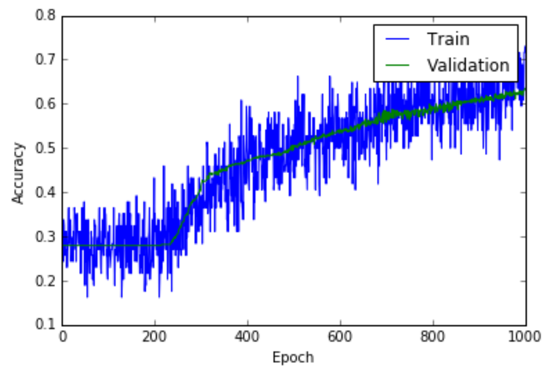
3.2.1 Learning Rates

3.2.1.1 NN Learning Rates

0.001

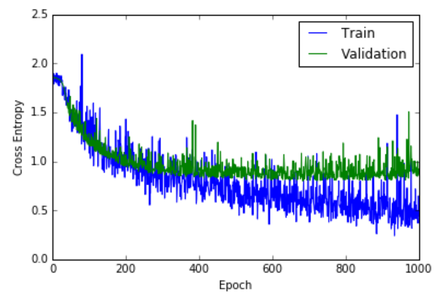
```
CE: Train 1.01750 Validation 1.05466 Test 1.08145
Acc: Train 0.64493 Validation 0.63484 Test 0.60519
Writing to nn_model.npz
Writing to nn_stats.npz
```

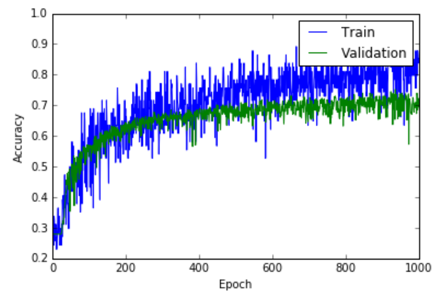




0.01

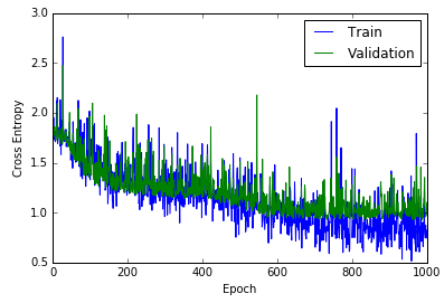
```
CE: Train 0.46786 Validation 0.85853 Test 0.86473
Acc: Train 0.82958 Validation 0.70644 Test 0.70130
Writing to nn_model.npz
Writing to nn_stats.npz
```

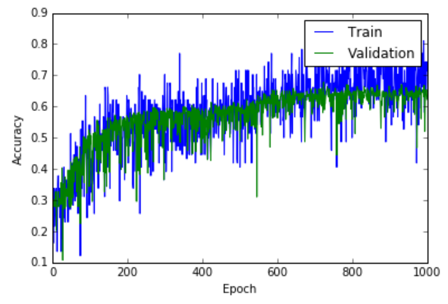




0.1

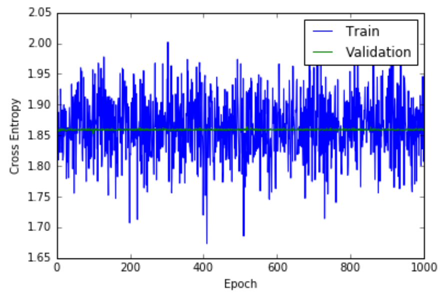
```
CE: Train 0.96453 Validation 1.10079 Test 1.06517
Acc: Train 0.65234 Validation 0.62291 Test 0.61299
Writing to nn_model.npz
Writing to nn_stats.npz
```

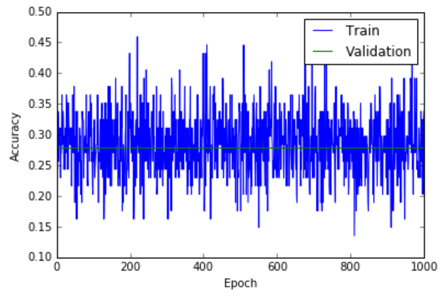




0.5

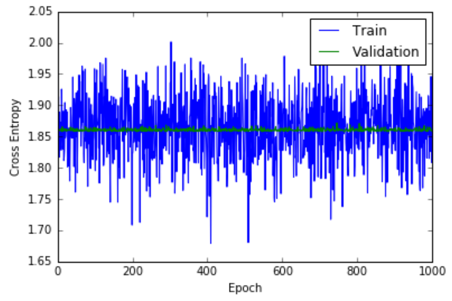
```
CE: Train 1.86119 Validation 1.85773 Test 1.84001
Acc: Train 0.28542 Validation 0.27924 Test 0.31688
Writing to nn_model.npz
Writing to nn_stats.npz
```

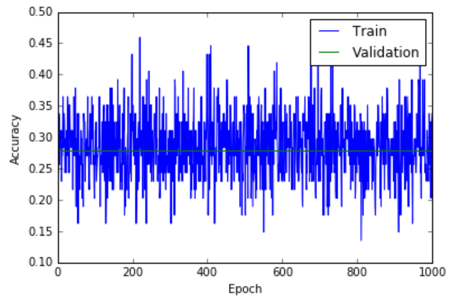




1.0

```
CE: Train 1.86273 Validation 1.85816 Test 1.84281
Acc: Train 0.28542 Validation 0.27924 Test 0.31688
Writing to nn_model.npz
Writing to nn_stats.npz
```

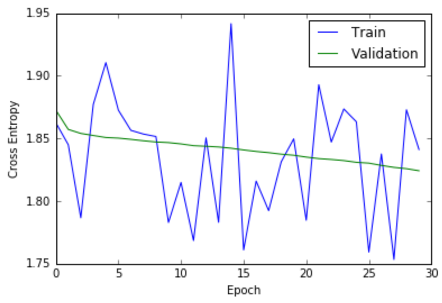


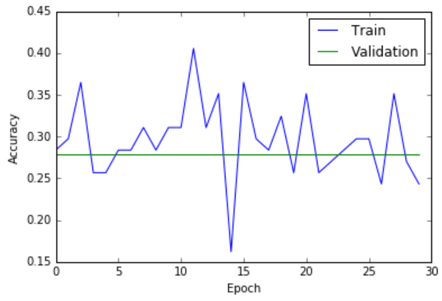


3.2.1.2 Learning Rates CNN

0.001

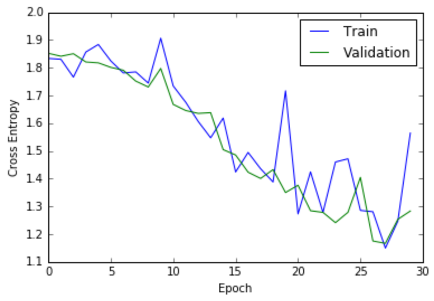
```
CE: Train 1.82144 Validation 1.82422 Test 1.80547
Acc: Train 0.28660 Validation 0.27924 Test 0.31688
```

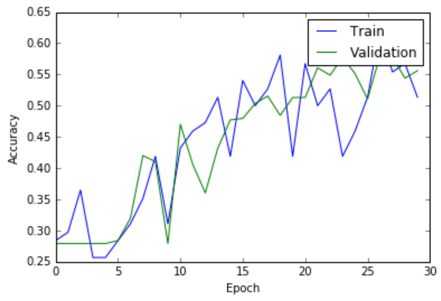




0.01

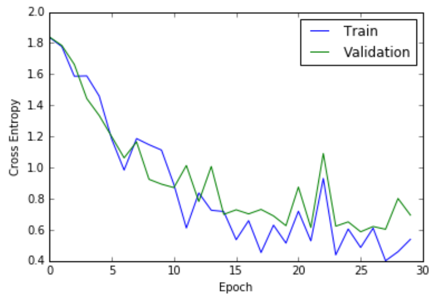
```
CE: Train 1.36903 Validation 1.28352 Test 1.43006
Acc: Train 0.52164 Validation 0.55609 Test 0.47792
Writing to cnn_model.npz
Writing to cnn_stats.npz
```

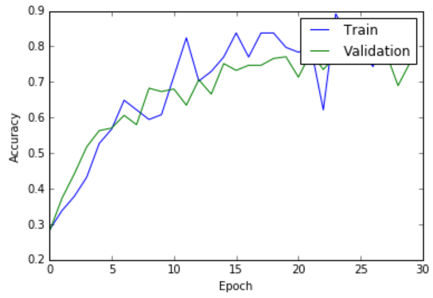




0.1

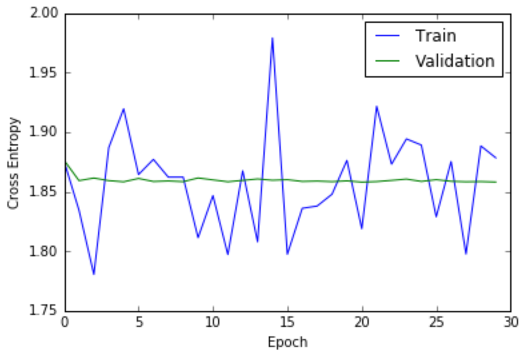
```
CE: Train 0.43891 Validation 0.69423 Test 0.60291
Acc: Train 0.84143 Validation 0.75656 Test 0.78442
```

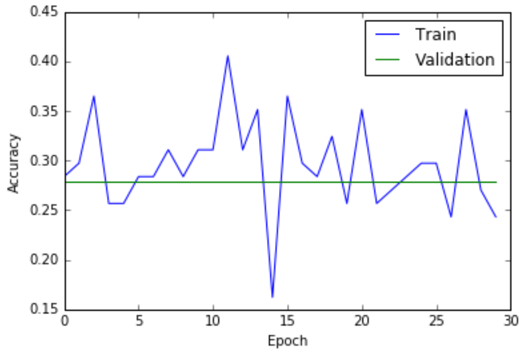




0.5

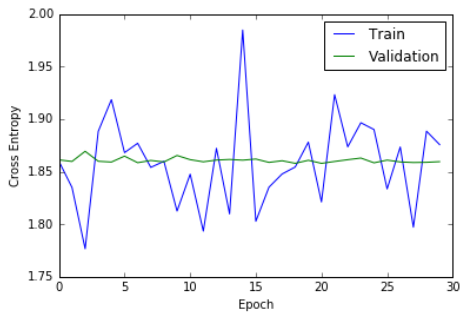
```
CE: Train 1.86125 Validation 1.85807 Test 1.84081
Acc: Train 0.28542 Validation 0.27924 Test 0.31688
```

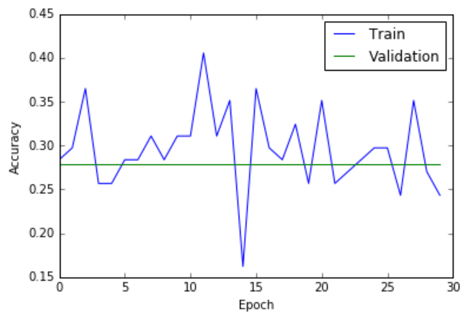




1.0

```
CE: Train 1.86320 Validation 1.85949 Test 1.84551
Acc: Train 0.28542 Validation 0.27924 Test 0.31688
Writing to cnn_model_eps_1.0.npz
Writing to cnn_stats_eps_1.0.npz
```





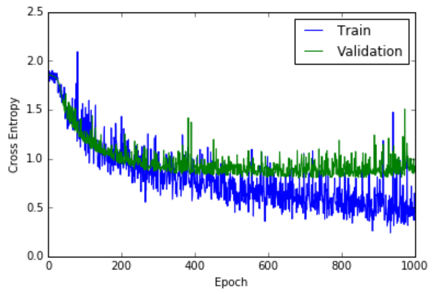
On both networks if the learning rate was greater than 0.5 the network would not converge. For small learning rates such as 0.001 the model was too slow to converge (learn) and thus did not represent the data as well. The optimal learning rates for the fully connected network and convolutional neural network were 0.01 and 0.1 respectively.

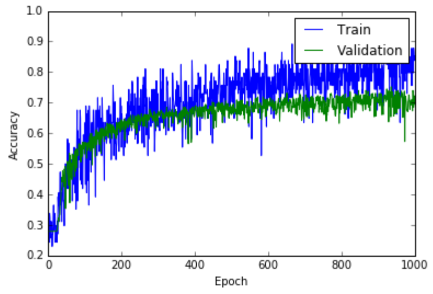
3.2.2 Momentum (with constant learning rate, eps = 0.01)

3.2.2.1 NN Momentum

0.0

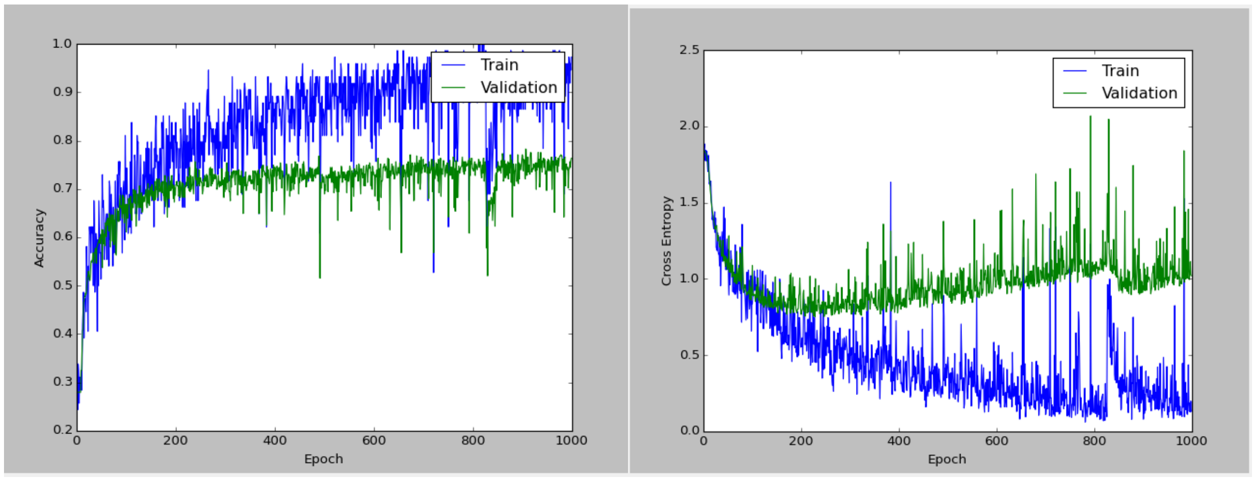
```
CE: Train 0.46786 Validation 0.85853 Test 0.86473
Acc: Train 0.82958 Validation 0.70644 Test 0.70130
Writing to nn_model.npz
Writing to nn_stats.npz
```





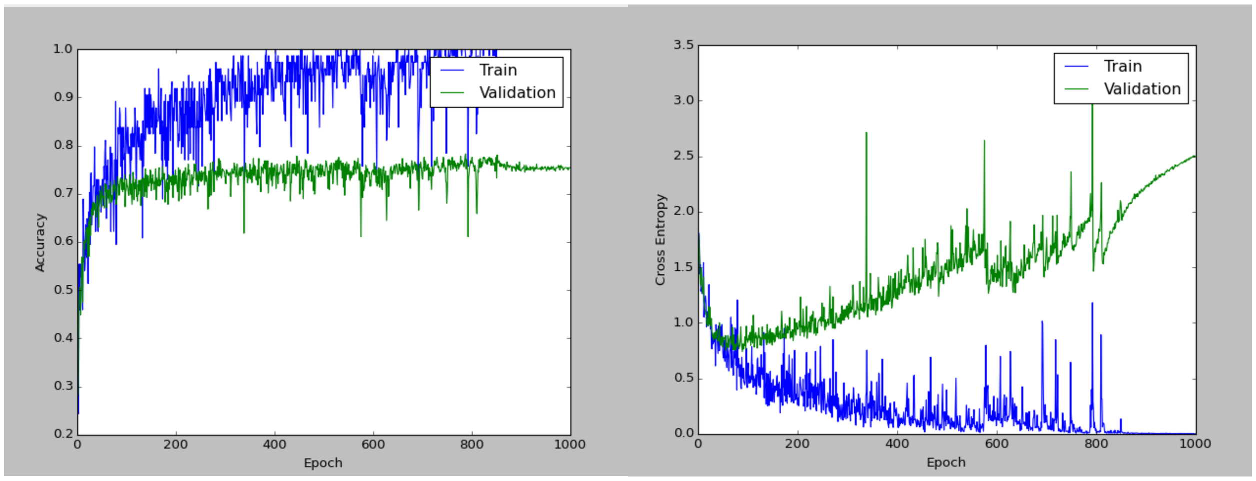
0.5

```
CE: Train 0.13216 Validation 1.02061 Test 0.76593
Acc: Train 0.95969 Validation 0.76372 Test 0.77922
```



0.9

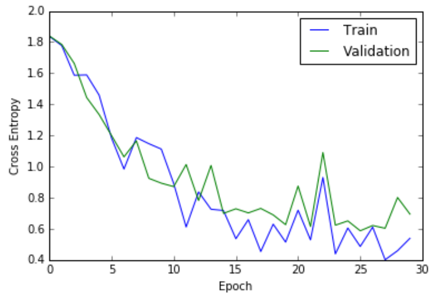
```
CE: Train 0.00209 Validation 2.50053 Test 2.15633
Acc: Train 1.00000 Validation 0.75418 Test 0.76883
```

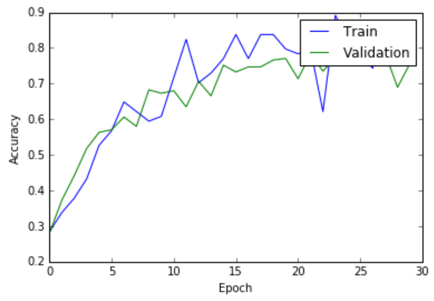


3.2.2.2 CNN Momentum

0.0

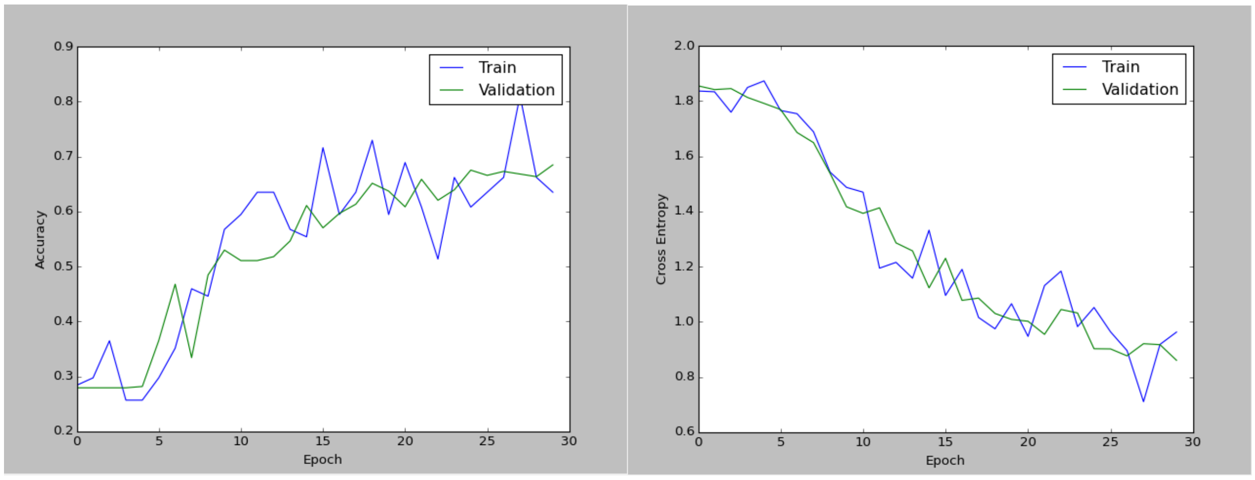
```
CE: Train 0.43891 Validation 0.69423 Test 0.60291
Acc: Train 0.84143 Validation 0.75656 Test 0.78442
```





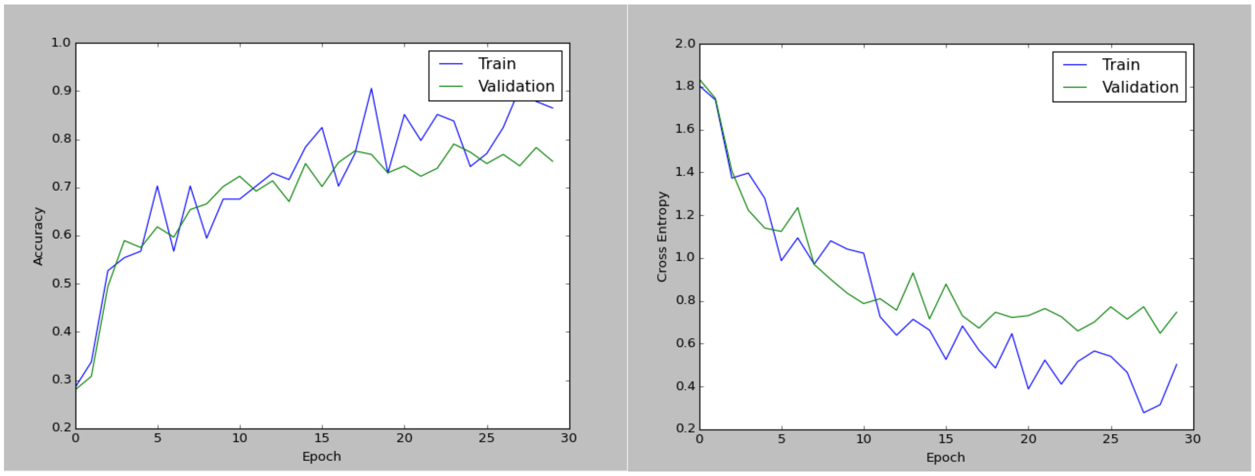
0.5

```
CE: Train 0.84357 Validation 0.86018 Test 0.91850
Acc: Train 0.70628 Validation 0.68496 Test 0.67532
Writing to cnn_model_mom_0.5.npz
Writing to cnn_stats_mom_0.5.npz
```



0.9

```
CE: Train 0.38199 Validation 0.74640 Test 0.69324
Acc: Train 0.86218 Validation 0.75418 Test 0.74805
```



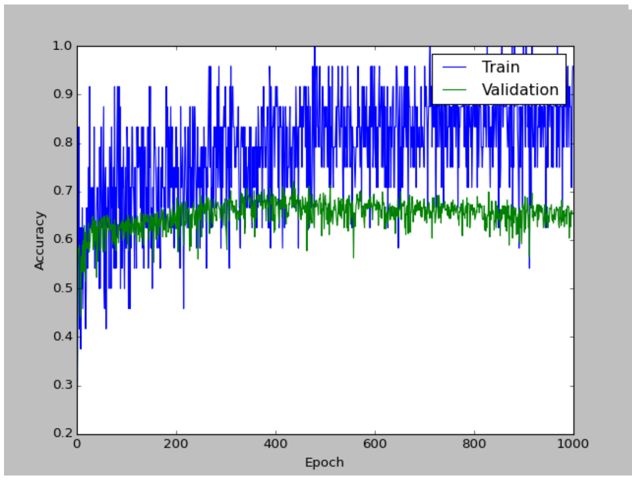
Adding momentum speeds up the convergence rate of both networks. As a result, the training accuracy increases dramatically. That is, the training accuracy for both the fully connected network and convolutional neural network with momentum of 0.9 was 1.0 and 0.86, respectively. However, as can be seen from the plots the training curve starts to deviate from the validation curve which is a sign that the network is starting to over fit. In the above plots the distance from the training curves and validation curves remains relative constant. However, if the number of epochs increased the distance would begin to increase. To prevent this overfitting early stopping could be used. That is, stop training once the distance between the training curve and validation curves begins to increase.

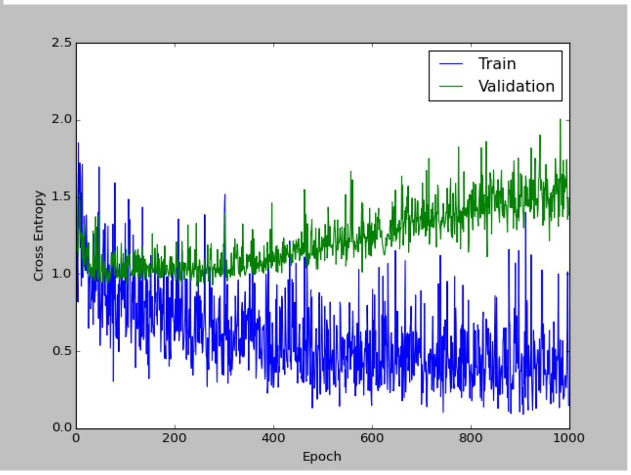
3.2.3 Batch Sizes (with eps=0.01 and momentum=0.9)

3.2.3.1 NN Batch Sizes

25

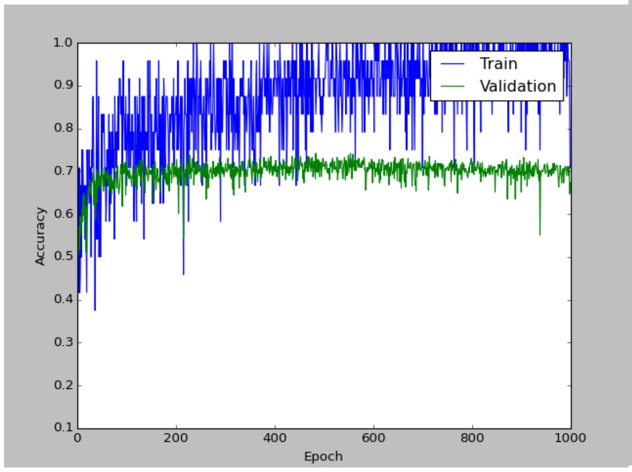
```
CE: Train 0.40507 Validation 1.38286 Test 1.35328
Acc: Train 0.85685 Validation 0.65394 Test 0.64935
```

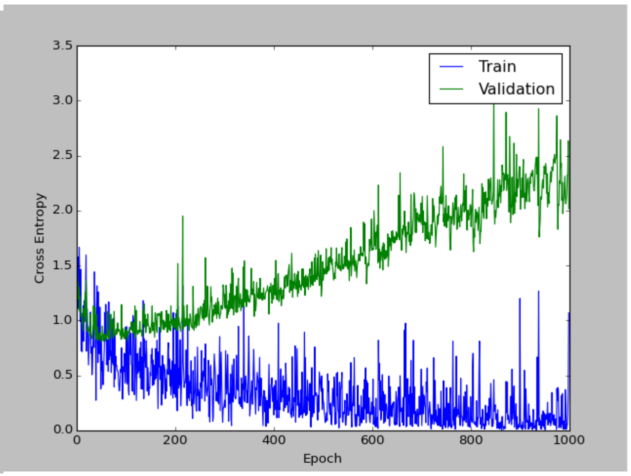




50

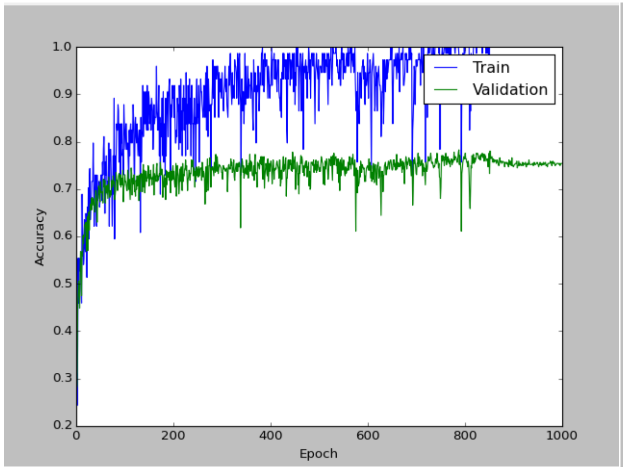
```
CE: Train 0.93656 Validation 2.01657 Test 2.09642
Acc: Train 0.75874 Validation 0.67303 Test 0.65974
```

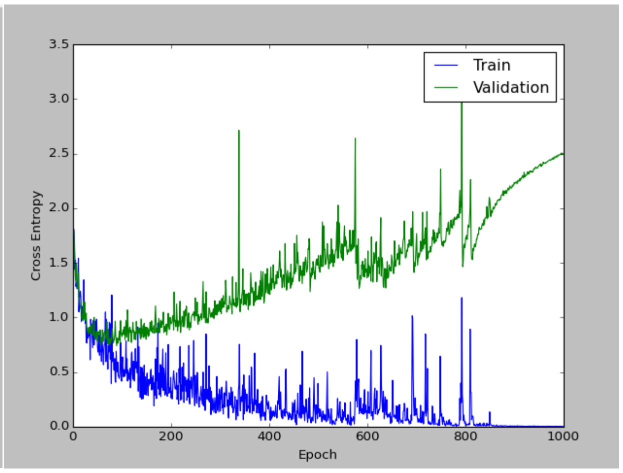




100

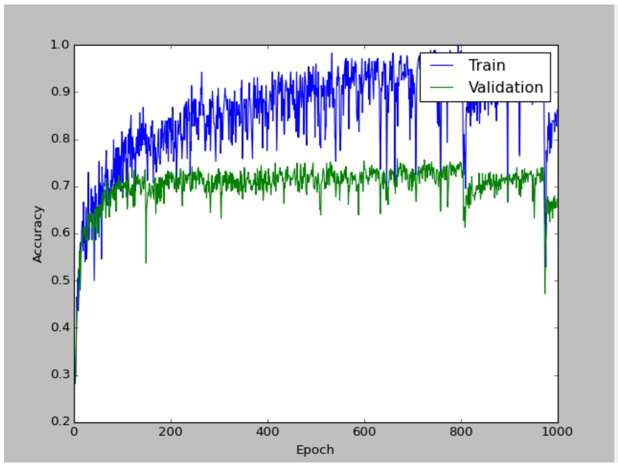
```
CE: Train 0.00209 Validation 2.50053 Test 2.15633
Acc: Train 1.00000 Validation 0.75418 Test 0.76883
```

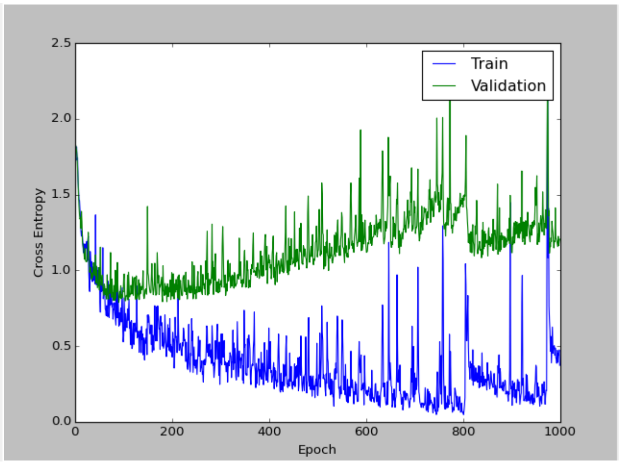




200

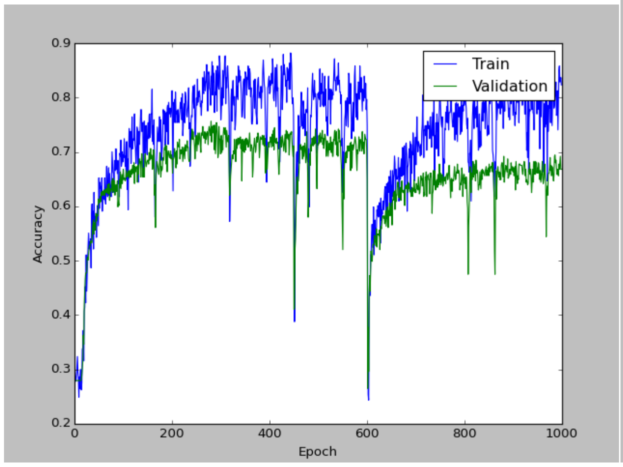
```
CE: Train 0.44070 Validation 1.20554 Test 1.10965
Acc: Train 0.83017 Validation 0.67303 Test 0.65455
```

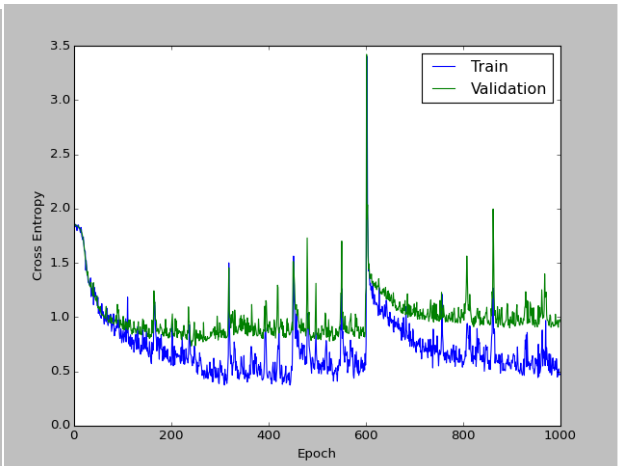




500

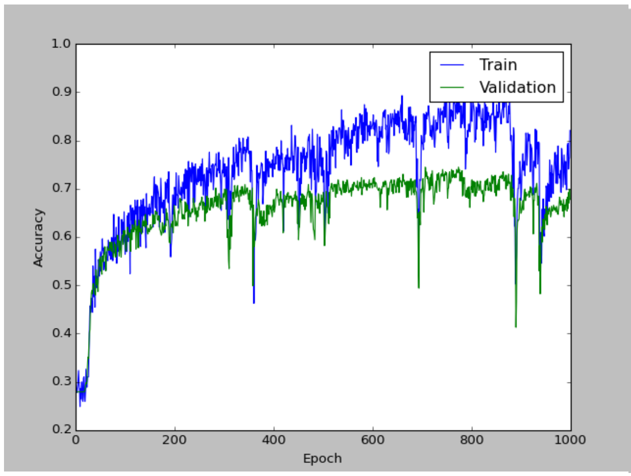
```
CE: Train 0.48548 Validation 0.97581 Test 0.87779
Acc: Train 0.82424 Validation 0.66826 Test 0.69091
```

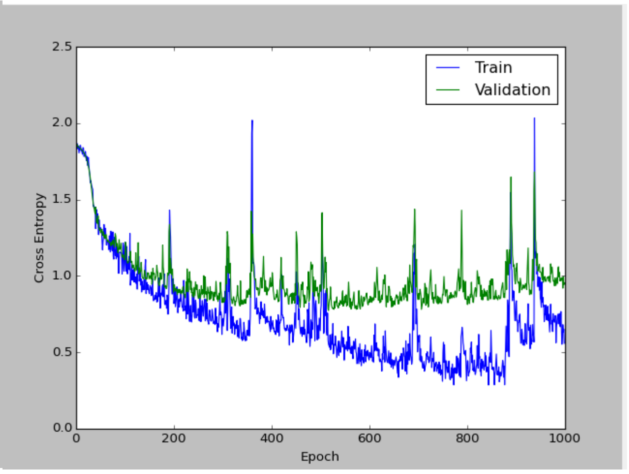




1000

```
CE: Train 0.61027 Validation 0.95330 Test 0.89293
Acc: Train 0.76556 Validation 0.67780 Test 0.67792
Writing to model_nn_batch_1000.npz
Writing to stats_nn_batch_1000.npz
```

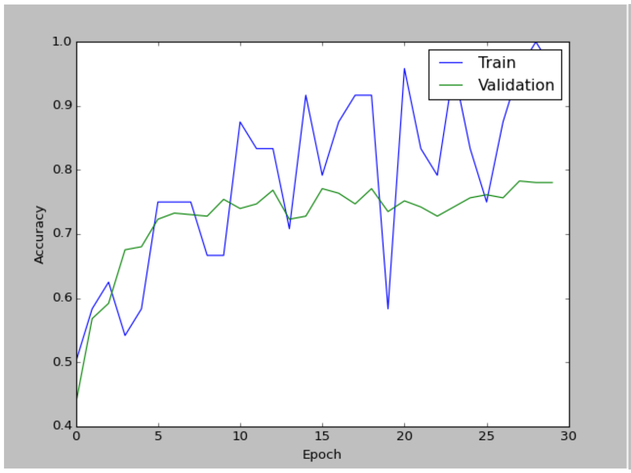


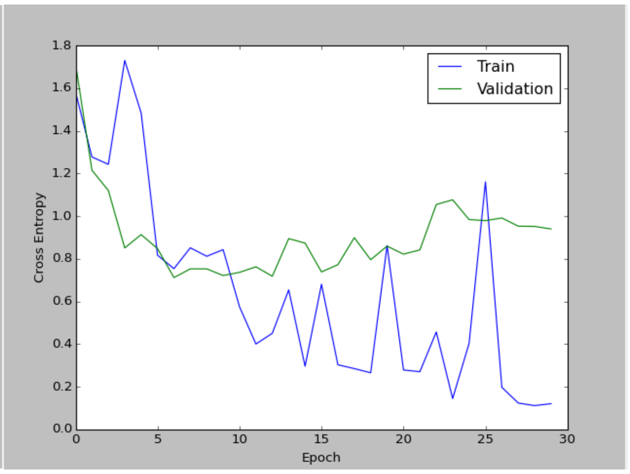


3.2.3.2 CNN Batch Sizes

25

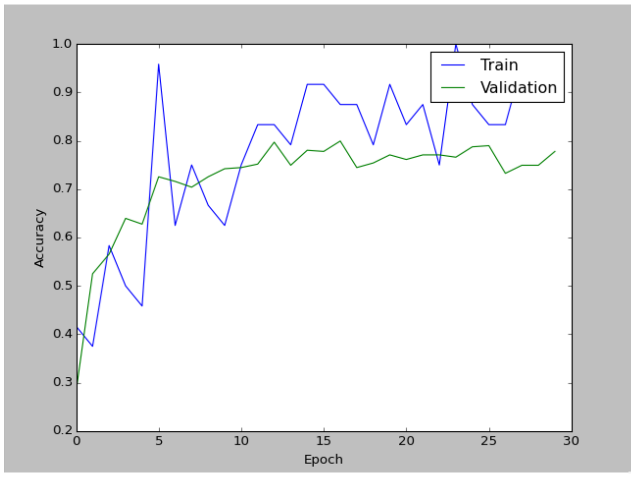
```
CE: Train 0.15341 Validation 0.93988 Test 0.83967
Acc: Train 0.94813 Validation 0.78043 Test 0.77403
Writing to cnn_model_batch_25.npz
Writing to cnn_stats_batch_25.npz
```

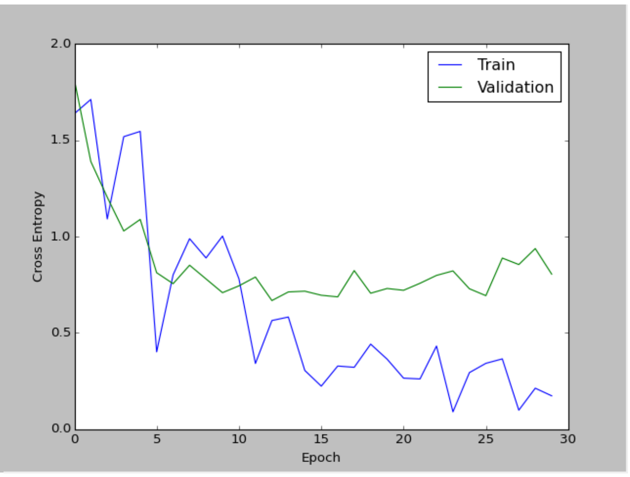




50

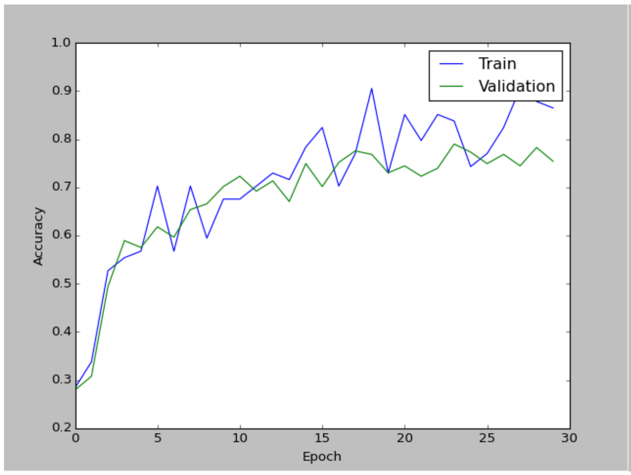
```
CE: Train 0.17446 Validation 0.80477 Test 0.65664
Acc: Train 0.94487 Validation 0.77804 Test 0.77922
Writing to cnn_model_batch_50.npz
Writing to cnn_stats_batch_50.npz
```

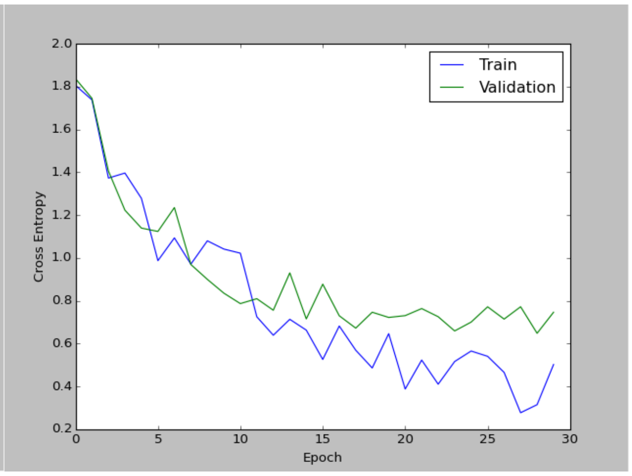




100

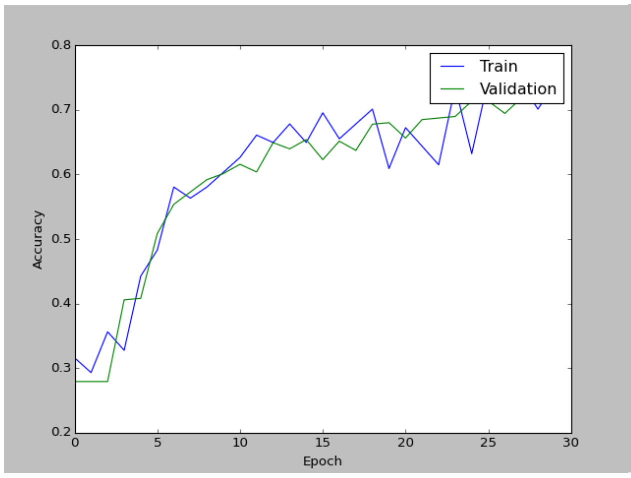
```
CE: Train 0.38199 Validation 0.74640 Test 0.69324
Acc: Train 0.86218 Validation 0.75418 Test 0.74805
Writing to cnn_model_mom_0.9.npz
Writing to cnn_stats_mom_0.9.npz
```

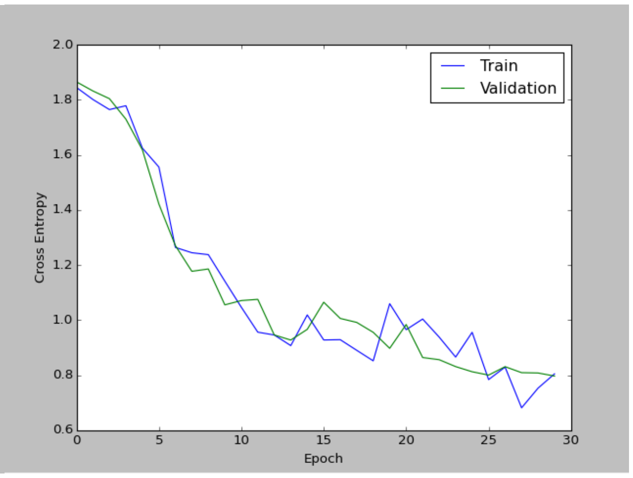




200

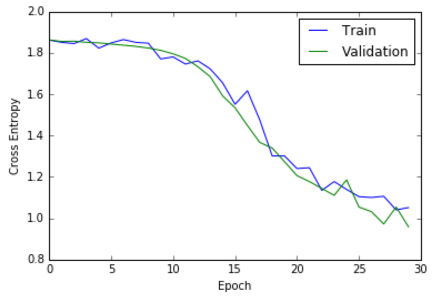
```
CE: Train 0.65776 Validation 0.79676 Test 0.81431
Acc: Train 0.77149 Validation 0.71122 Test 0.69610
Writing to cnn_model_batch_200.npz
Writing to cnn_stats_batch_200.npz
```

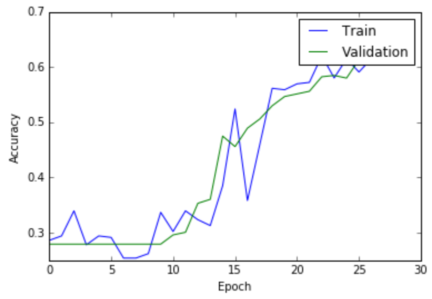




500

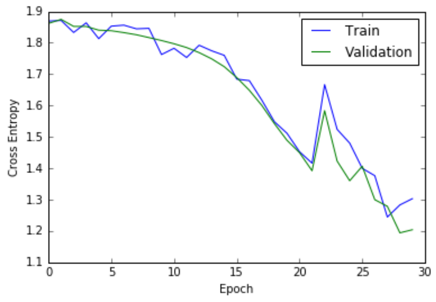
```
CE: Train 0.99859 Validation 0.95808 Test 1.08181
Acc: Train 0.65531 Validation 0.64200 Test 0.59481
Writing to cnn_model_batch_500.npz
Writing to cnn_stats_batch_500.npz
```

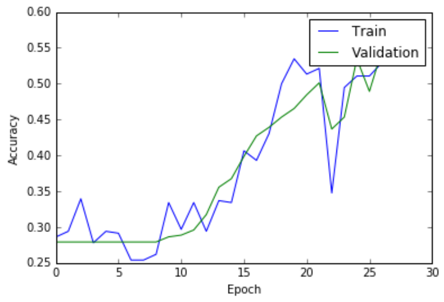




1000

```
CE: Train 1.25815 Validation 1.20414 Test 1.29269
Acc: Train 0.54772 Validation 0.57041 Test 0.49351
Writing to cnn_model_batch_1000.npz
Writing to cnn_stats_batch_1000.npz
```





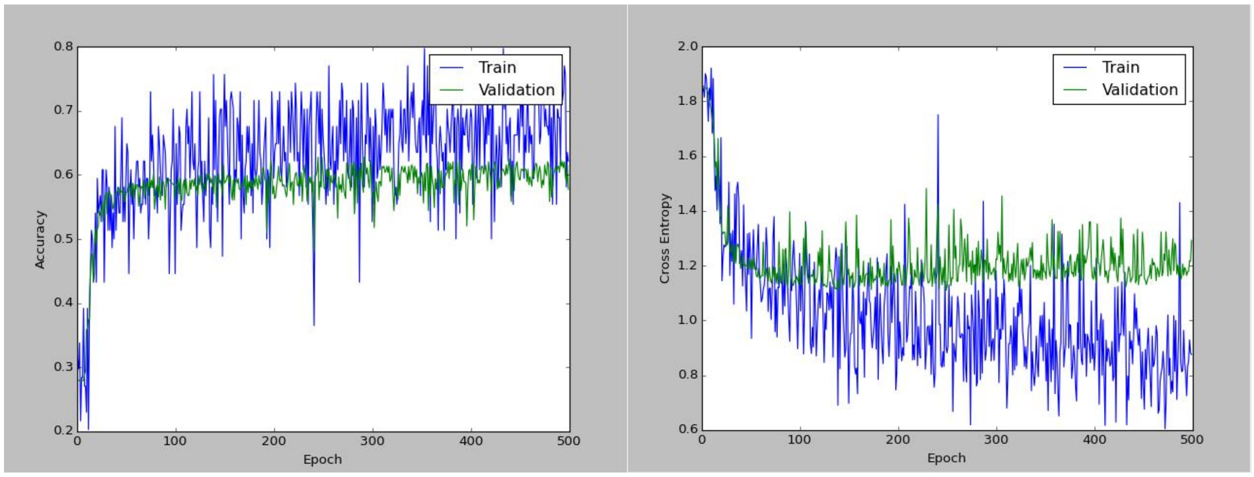
Increasing the batch size will increase the convergence rate. This can be seen by the increasing accuracy on both the training set and validation set as we increase the batch size from 50 to 100 on both networks. However, the accuracy decreases and error increases for batch sizes greater than 100. This occurs because as we get to the optimum we want smaller convergence rates (smaller step sizes) which the larger batch sizes do not allow.

3.3 Model Architecture

3.3.1 NN - Number of Hidden Units

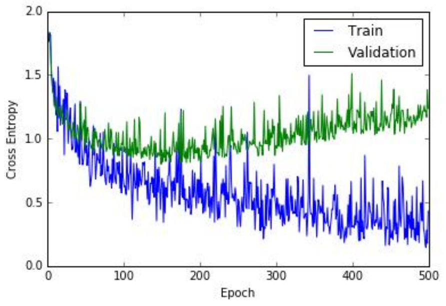
[2,4]

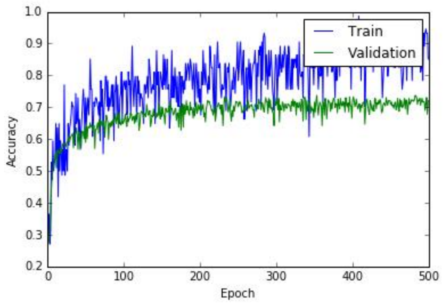
```
CE: Train 0.92486 Validation 1.29221 Test 1.24982
Acc: Train 0.64819 Validation 0.57757 Test 0.57662
Writing to model_nn_hidden_2_4.npz
Writing to stats_nn_hidden_2_4.npz
```



[8,16]

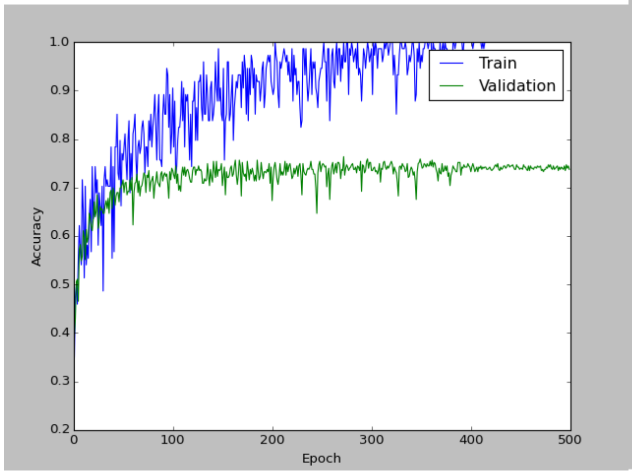
```
CE: Train 0.25131 Validation 1.23540 Test 1.06594
Acc: Train 0.91494 Validation 0.72076 Test 0.72468
Writing to model_nn_hidden_8_16.npz
Writing to stats_nn_hidden_8_16.npz
```

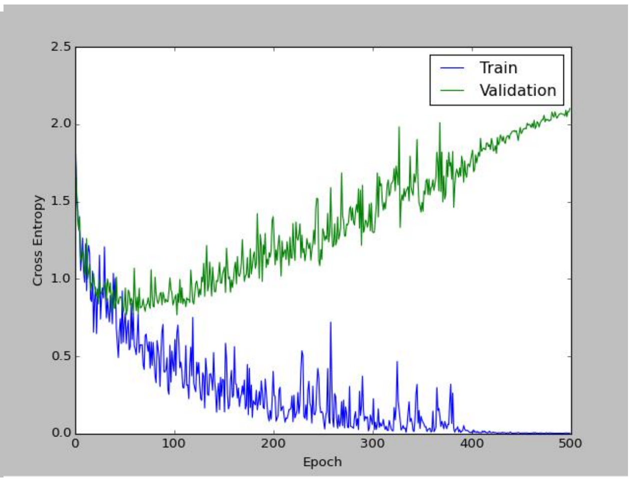




[16,32]

```
CE: Train 0.00200 Validation 2.10195 Test 1.60395
Acc: Train 1.00000 Validation 0.73747 Test 0.76623
Writing to model_nn_hidden_32_64.npz
Writing to stats_nn_hidden_32_64.npz
```

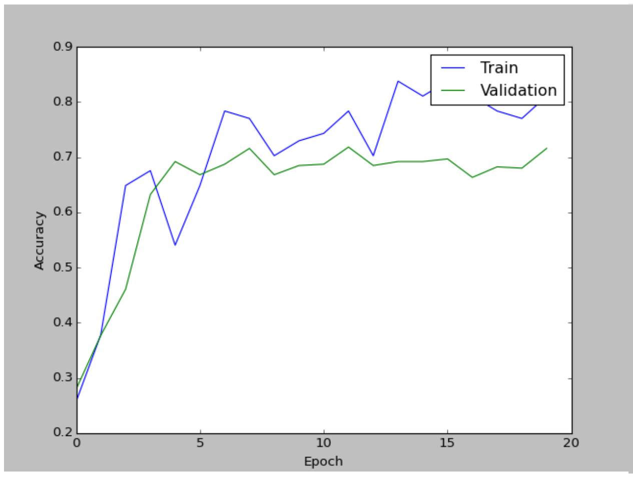


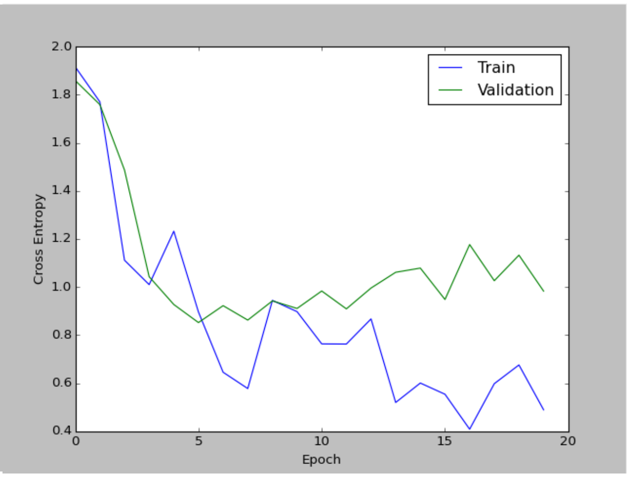


3.3.2 CNN – Number of Filters

[2,4]

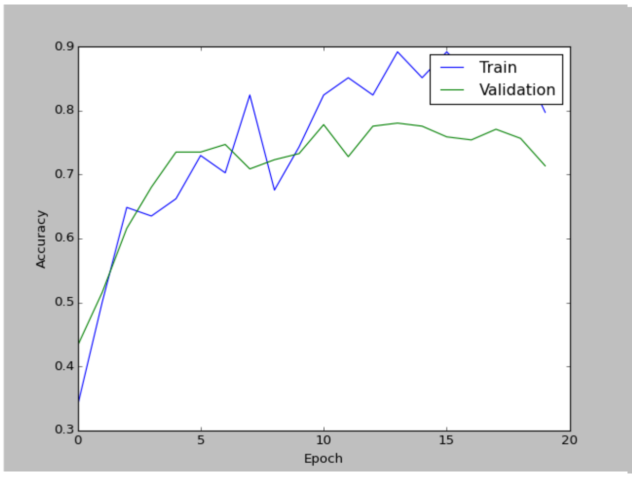
```
CE: Train 0.54889 Validation 0.98303 Test 0.79729
Acc: Train 0.80142 Validation 0.71599 Test 0.75065
Writing to cnn_model_filter_2_4.npz
Writing to cnn_stats_filter_2_4.npz
```

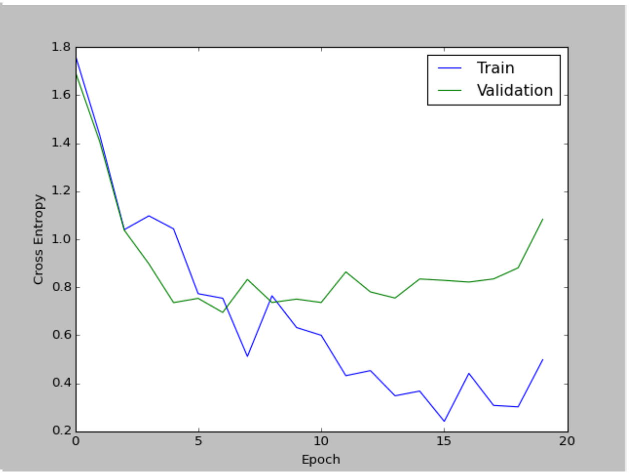




[8,16]

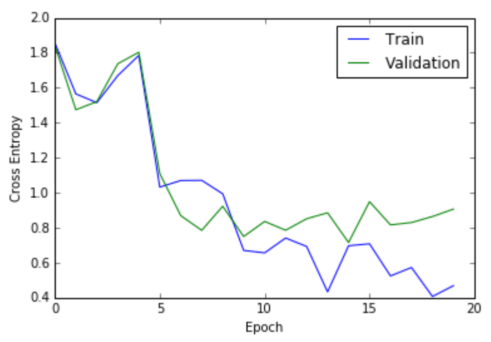
```
CE: Train 0.44815 Validation 1.08300 Test 1.00806
Acc: Train 0.84321 Validation 0.71360 Test 0.70390
Writing to cnn_model_filter_8_16.npz
Writing to cnn_stats_filter_8_16.npz
```

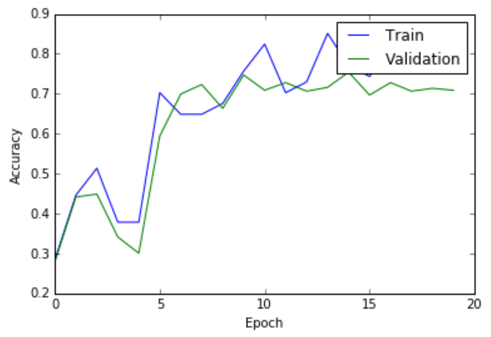




[16,32]

```
CE: Train 0.48787 Validation 0.90653 Test 0.85086
Acc: Train 0.82365 Validation 0.70883 Test 0.71948
Writing to cnn_model_filter_16_32.npz
Writing to cnn_stats_filter_16_32.npz
```





As the hidden layers of the fully connected layer and the number of filters are increased the models performs better since increasing the size of the hidden layers and filters increases the size of the weights and thus the number of parameters which in turn allows for the model to fit the data better. However, increasing the number of parameters slows down our networks and as can be seen from the CNN plots of [8,16] compared to [16,32] it does not improve our performance significantly.

3.4 Compare CNNs and Fully Connected Networks

$$number\ of\ parameters = W_1 + W_2 + W_3 + b_1 + b_2 + b_3$$

Number of Parameters for Fully Connected Network

Hidden Layers = [4,8]

$$W_1 = number\ of\ inputs\ \times hidden\ 1$$

$$W_1 = 2304\ \times 4 = 9216$$

$$b_1 = 4$$

$$W_2 = hidden\ 1 \times hidden\ 2$$

$$W_2 = 4 \times 8 = 32$$

$$b_2 = 8$$

$$W_3 = hidden\ 2 \times outputs$$

$$W_3 = 8 \times 7 = 56$$

$$b_3 = 7$$

$$number\ of\ parameters = 9216 + 4 + 32 + 8 + 56 + 7 = 9323$$

Number of Parameters for Convolutional Neural Network

Filter Size = [8,16]

$$I = J = size\ of\ filters, C = input\ channels, K = number\ of\ filters$$

$$W_1 = (I \times J \times C + b_1) \times K_1$$

$$W_1 = (5 \times 5 \times 1 + 1) \times 8 = 208$$

$$W_2 = (I\ \times J \times C + b_2) \times K_2$$

$$W_2 = (5 \times 5 \times 8 + 1) \times 16 = 3216$$

$$W_3 = (C \times 64 + b_3) \times K_3$$

$$W_3 = (16 \times 64 + 1) \times 7 = 7175$$

$$number\ of\ parameters = 10599$$

100 epochs for both networks

```
CE: Train 0.07449 Validation 1.19625 Test 1.00637
Acc: Train 0.97362 Validation 0.77088 Test 0.78701
Writing to cnn_model_batch_500.npz
Writing to cnn_stats_batch_500.npz
```

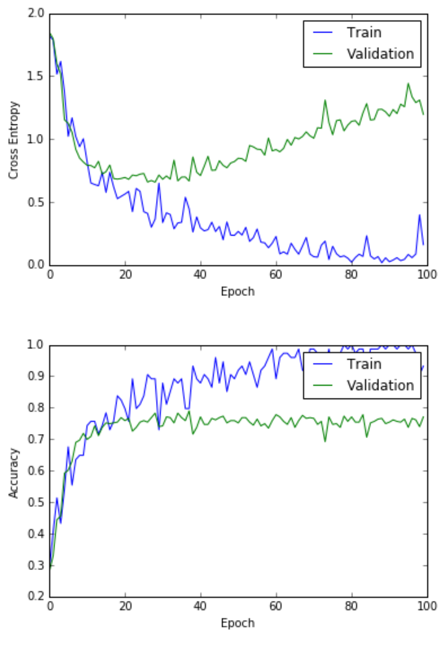


Figure 1: Convolutional Neural Network

```
CE: Train 0.85075 Validation 1.06144 Test 0.96110
Acc: Train 0.69235 Validation 0.63723 Test 0.63896
Writing to nn_model_epochs_100.npz
Writing to nn_model_epochs_100.npz
```

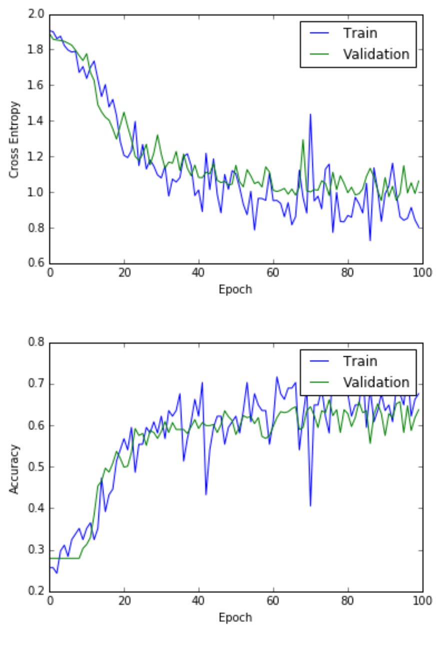
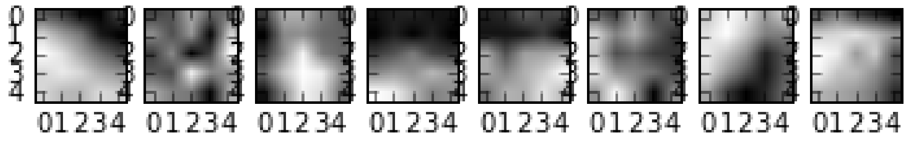


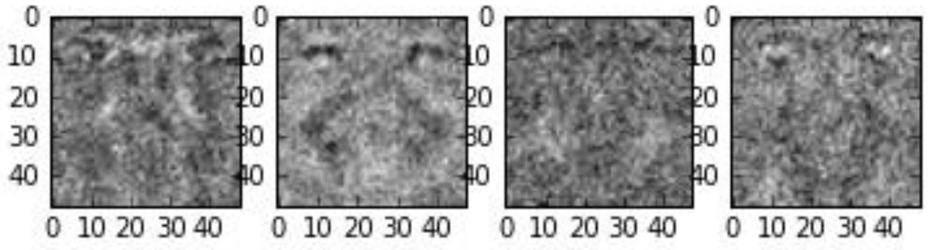
Figure 2: Fully Connected Neural Network

With similar number of parameters, the convoluted neural network outperforms the fully connected neural network. This is a result of the architecture of both models. Regular neural nets do not scale well with larger images as the inputs are fully connected to the first layer of weights. As a result, a lot of the parameters are wasted in the first layer. Meanwhile, convolutional neural nets loosey-compress the image by using filters of a certain size. This reduces the number of parameters used in the first layer significantly. It is also import to realize that we do not want a large number of parameters for the first layer (first pass) because we want the first pass to be the most general. Regular fully connected neural nets do not allow for this generalization and thus do not perform as well as convolutional neural nets because they over fit the data early on.

First layer of filters for CNN



First layer of weights of fully connected network



3.5 Network Uncertainty

We will set the threshold to be less than 50%.

The network will be uncertain when the probabilities of the outputs do not vary by much. This will happen when the training and validation accuracies are low. It is possible for the classifier to be correct but it is less likely.

4 Mixture of Gaussians

4.2 Training

After testing the code with different values of momentum.

For smaller values of randConst it seems to converge in less iterations than for larger values. Most of the time the model converges at around 20 iterations for different values of randConst. This is why I set the number of iterations to be 20 and the randConst to be 1.5. The model seems to converge at a local optimum of around 6 million log likelihood.

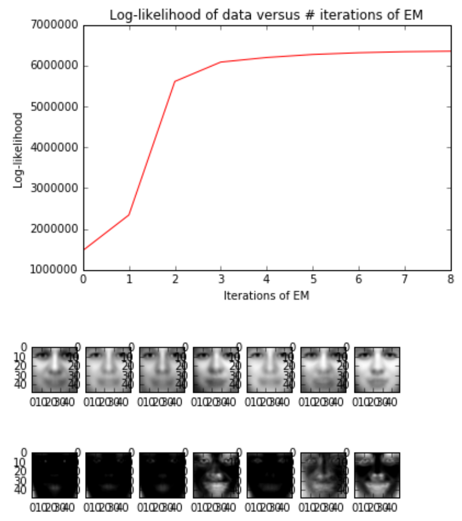


Figure 3: Mixture of Gaussians, iters = 10, randConst = 1000

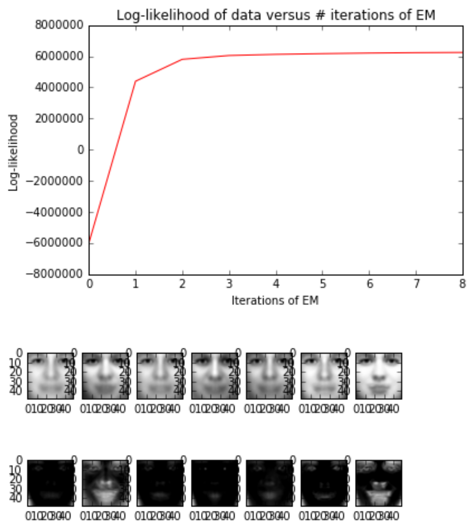


Figure 4: Mixture of Gaussians , iters = 10, randConst = 0.5

The above plots show faster convergence as randConst decreases.

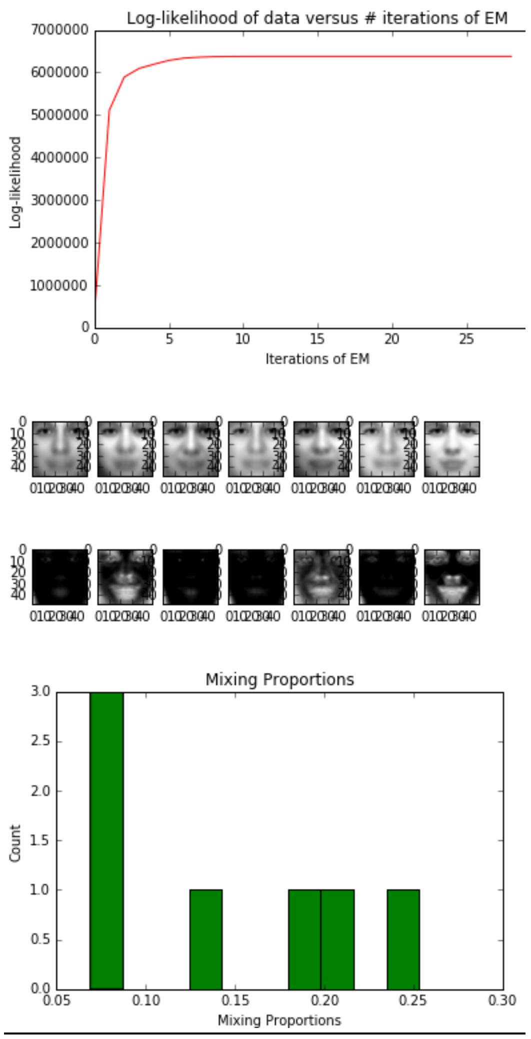


Figure 5: Mixture of Gaussians, iters = 30, randConst = 1.25

When randConst was set too low (0.1) the log likelihood would give a nan error.

Chose a randConst = 1.25 for faster convergence and since it produced a high log likelihood.

4.3 Initializing a mixture of Gaussians with k-means

Figure 5 above shows the training curve of a mixture of gaussian model not initialized with kmeans. Figure 6 shows a mixture of gaussian models initialized with kmeans. The steeper slope of the training curve of the model initialized with kmeans shows that initialization with k-means leads to a faster learning rate. This is because instead of randomly initializing the means we initialize them near the means of the data. As a result, of initializing the means in a more sensible location the learning rate not only speeds up but the risk of getting stuck at a local optimum is eliminated.

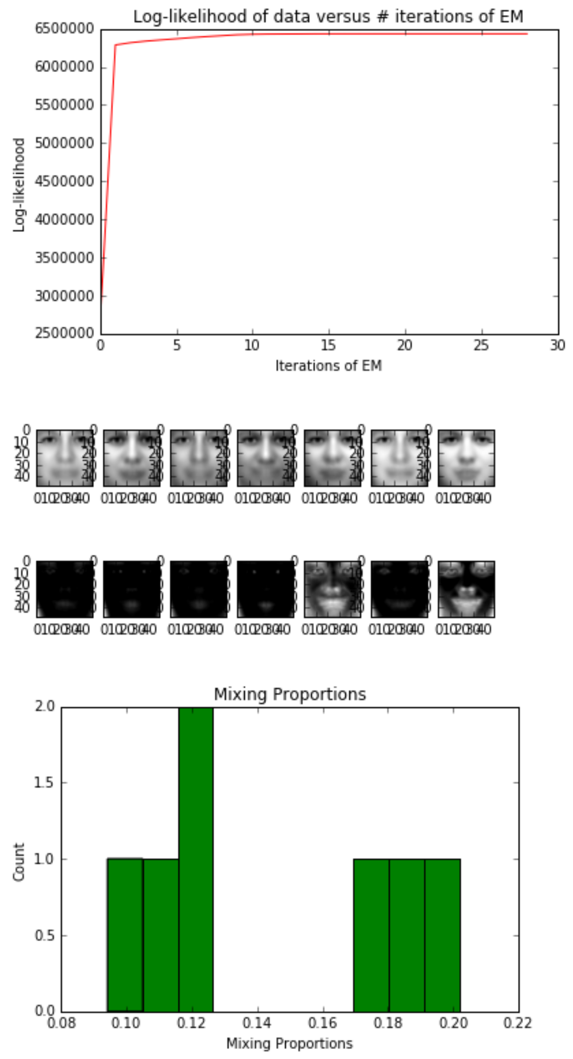
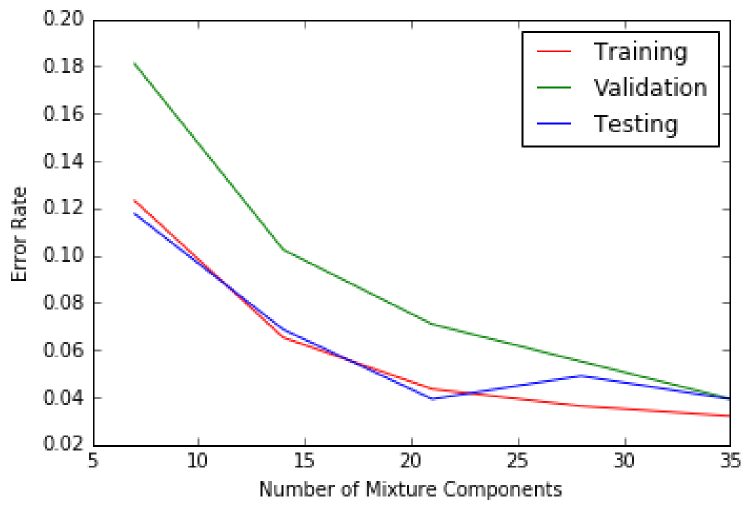


Figure 6: Mixture of Gaussians Initialized with Kmeans, iters = 30, randConst = 1.25

<u>4.4 Classification using MOGs</u>



a) As the number of clusters increases we are essentially fitting the data better by having more groups to classify the data in. Thus, low number of clusters will under fit the data.

b) The testing curve follows the training curve, that is the error rate steadily decreases until the number of mixture components is higher than 20, afterwards the error begins rising again. This is because the model is beginning to over fit.