1 EM for Mixture of Gaussians

Using the model $p(\mathbf{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \Sigma)$ with data $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}^{(n)}\}$ the log-likelihood is

$$L = \ln p(\mathbf{X}|\pi_k, \boldsymbol{\mu}_k, \Sigma) = \ln \prod_{n=1}^{N} p(\mathbf{x}^{(n)}) = \sum_{n=1}^{N} \ln p(\mathbf{x}^{(n)}) = \sum_{n=1}^{N} \ln \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}^{(n)}|\boldsymbol{\mu}_k, \Sigma)$$

where $\mathcal{N}(\mathbf{x}^{(n)}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}) = (2\pi)^{-d/2}|\boldsymbol{\Sigma}|^{-1/2} e^{-\frac{1}{2}(\mathbf{x}^{(n)}-\boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}^{-1}(\mathbf{x}^{(n)}-\boldsymbol{\mu}_k)}$

- We calculate the derivative wrt $\boldsymbol{\mu}_k$

$$\frac{\partial L}{\partial \boldsymbol{\mu}_k} = \sum_{n=1}^{N} \frac{\pi_k \mathcal{N}(\mathbf{x}^{(n)}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma})}{\sum_j \pi_j \mathcal{N}(\mathbf{x}^{(n)}|\boldsymbol{\mu}_j, \boldsymbol{\Sigma})} \boldsymbol{\Sigma}^{-1}(\mathbf{x}^{(n)} - \boldsymbol{\mu}_k) = \sum_{n=1}^{N} \gamma(z_k^{(n)}) \boldsymbol{\Sigma}^{-1}(\mathbf{x}^{(n)} - \boldsymbol{\mu}_k)$$

  we can solve for the EM $\boldsymbol{\mu}_k$

$$\frac{\partial L}{\partial \boldsymbol{\mu}_k} = 0 \iff \boxed{\boldsymbol{\mu}_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_k^{(n)}) \mathbf{x}^{(n)}}$$

  where we have labeled $N_k = \sum_{n=1}^{N} \gamma(z_k^{(n)})$

- For π_k consider the lagrangian $\mathcal{L} = L + \lambda \left(\sum_{k=1}^{K} \pi_k - 1 \right)$ so that

$$\frac{\partial \mathcal{L}}{\partial \pi_k} = \sum_{n=1}^{N} \frac{\gamma(z_k^{(n)})}{\pi_k} + \lambda = 0 \iff N_k + \lambda \pi_k = 0$$

  now sum $\sum_k$ to see $\lambda = -N$ now sub this again to get : $\boxed{\pi_k = \frac{N_k}{N}}$

- We calculate the derivative wrt Σ

$$\frac{\partial L}{\partial \boldsymbol{\Sigma}} = \frac{1}{2} \sum_{n=1}^{N} \frac{\sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}^{(n)}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma})}{\sum_j \pi_j \mathcal{N}(\mathbf{x}^{(n)}|\mu_j, \boldsymbol{\Sigma})} [(\mathbf{x}^{(n)} - \boldsymbol{\mu}_k)(\boldsymbol{\Sigma}^{-1})^2 (\mathbf{x}^{(n)} - \boldsymbol{\mu}_k)^T - \boldsymbol{\Sigma}^{-1}]$$

  $\frac{\partial}{\partial \boldsymbol{\Sigma}} \mathcal{N}(\mathbf{x}^{(n)}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}) = \mathcal{N}(\mathbf{x}^{(n)}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma})[(\mathbf{x}^{(n)} - \boldsymbol{\mu}_k)(\boldsymbol{\Sigma}^{-1})^2 (\mathbf{x}^{(n)} - \boldsymbol{\mu}_k)^T - \boldsymbol{\Sigma}^{-1}]$ expand in terms of γ

$$\frac{\partial L}{\partial \Sigma} = \frac{1}{2} \sum_{n=1}^{N} \sum_{k=1}^{K} \gamma(z_k^{(n)}) [(\mathbf{x}^{(n)} - \boldsymbol{\mu}_k)(\boldsymbol{\Sigma}^{-1})^2 (\mathbf{x}^{(n)} - \boldsymbol{\mu}_k)^T - \boldsymbol{\Sigma}^{-1}]$$

  and use properties of traces :

$$\frac{\partial L}{\partial \boldsymbol{\Sigma}} = \mathbf{0} \iff \mathrm{Tr}((\boldsymbol{\Sigma}^{-1})^2 \mathbf{S} - N \boldsymbol{\Sigma}^{-1}) = 0$$

  where $\mathbf{S} = \sum_{n=1}^{N} \sum_{k=1}^{K} \gamma(z_k^{(n)})(\mathbf{x}^{(n)} - \boldsymbol{\mu}_k)(\mathbf{x}^{(n)} - \boldsymbol{\mu}_k)^T$ so that

$$\frac{\partial L}{\partial \Sigma} = \mathbf{0} \iff (\boldsymbol{\Sigma}^{-1})^2 \mathbf{S} = N \boldsymbol{\Sigma}^{-1} \iff \boxed{\boldsymbol{\Sigma} = \frac{1}{N} \sum_{n=1}^{N} \sum_{k=1}^{K} \gamma(z_k^{(n)})(\mathbf{x}^{(n)} - \boldsymbol{\mu}_k)(\mathbf{x}^{(n)} - \boldsymbol{\mu}_k)^T}$$

2 Convolutional Neural Networks

Let $x \in \mathbb{R}^{N\times H\times W\times C}$ be N images, and $f \in \mathbb{R}^{I\times J\times C\times K}$ be the convolutional filters.

H, W are the height and width of the image; I, J are the height and width of the filters; C is the depth of the image (a.k.a. channels); K is the number of filters.

Padding is an operation that adds zeros to the edges of an image to form a larger image. Formally, the padding operator $\texttt{pad}$ is defined as: $x^{(P,Q)} = \texttt{pad}(x, P, Q) \in \mathbb{R}^{N\times(H+P)\times(W+Q)\times C}$

$$x^{(P,Q)}_{h,w} = x_{h-\lfloor\frac{P}{2}\rfloor, w-\lfloor\frac{Q}{2}\rfloor} \cdot \mathbb{I}\left(\text{if } h \in \left[\lfloor\frac{P}{2}\rfloor + 1, \lfloor\frac{P}{2}\rfloor + H\right], w \in \left[\lfloor\frac{Q}{2}\rfloor + 1, \lfloor\frac{Q}{2}\rfloor + W\right]\right)$$

$$y_{p,q} = x * f = \sum_{i=1}^{I}\sum_{j=1}^{J} x_{p+i-1,q+j-1} \cdot f_{i,j}$$

Define the filter transpose as: $f^{\top}_{i,j} = f_{I-i+1,J-j+1,}$ and forward propagation equation $y_{pq} = x^{(I-1,J-1)} * f$

Using chain rule

$$\boxed{\left(\frac{\partial E}{\partial f}\right)_{ij} = \sum_{p,q}\frac{\partial E}{\partial y_{pq}}\frac{\partial y_{pq}}{\partial f_{ij}} = \sum_{p,q} x^{(I-1,J-1)}_{p+i-1,\ q+j-1}\frac{\partial E}{\partial y_{pq}} = x^{(I-1,J-1)}_{h,w} * \left(\frac{\partial E}{\partial y}\right)_{h,w}}$$

since

$$\frac{\partial y_{pq}}{\partial f_{ij}} = \frac{\partial}{\partial f_{ij}}\sum_{i',j'} x^{(I-1,J-1)}_{p+i'-1,q+j;-1} f_{i',j'} = x^{(I-1,J-1)}_{p+i-1,q+j-1}$$

let padded $x^{(I-1,J-1)}_{h'w'} = \tilde{x}_{h'w'}$ and $\alpha = \lfloor\frac{I-1}{2}\rfloor, \quad \beta = \lfloor\frac{J-1}{2}\rfloor$

$$\frac{\partial E}{\partial x_{hw}} = \sum_{p,q,h',w'}\frac{\partial E}{\partial y_{pq}}\frac{\partial y_{pq}}{\partial \tilde{x}_{h'w'}}\frac{\partial \tilde{x}_{h'w'}}{\partial x_{hw}} = \sum_{p,q,h',w'}\frac{\partial E}{\partial y_{pq}} f_{h'-p+1,w'-q+1}\cdot\delta^{h',w'}_{h-\alpha,w-\beta} = \sum_{p,q}\frac{\partial E}{\partial y_{pq}} f_{h-\alpha-p+1,w-\beta-q+1}$$

since

$$\frac{\partial y_{pq}}{\partial \tilde{x}_{h'w'}} = \frac{\partial}{\partial \tilde{x}_{h'w'}}\sum_{i',j'}\tilde{x}_{p+i'-1,q+j;-1} f_{i',j'} = f_{h'-p+1,w'-q+1}$$

then manipulating the sum we find :

$$\left(\frac{\partial E}{\partial x}\right)_{hw} = \sum_{p',q'}\left(\frac{\partial E}{\partial y}\right)^{(I-1,J-1)}_{p',q'} f_{h-p'+1,w-q'+1} = \sum_{\tilde{p},\tilde{q}}\left(\frac{\partial E}{\partial y}\right)^{(I-1,J-1)}_{h+\tilde{p}-1,w-\tilde{q}-1} f_{I-\tilde{p}+1,J-\tilde{q}+1}$$

which is precisely

$$\boxed{\left(\frac{\partial E}{\partial x}\right)_{hw} = \left(\frac{\partial E}{\partial y}\right)^{(I-1,J-1)}_{hw} * f^{T}}$$

3 Neural Networks

3.1 Basic generalization [5 points]

Training a regular NN and a CNN with the default set of hyperparameters :

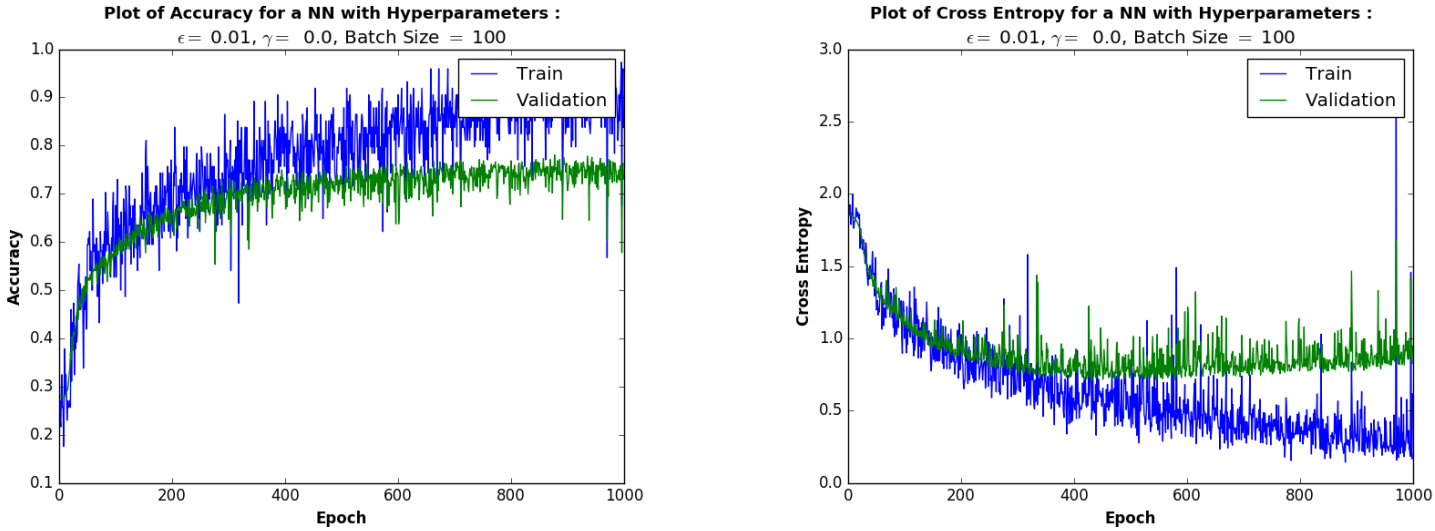


Figure 1: Plots for NN with default parameters

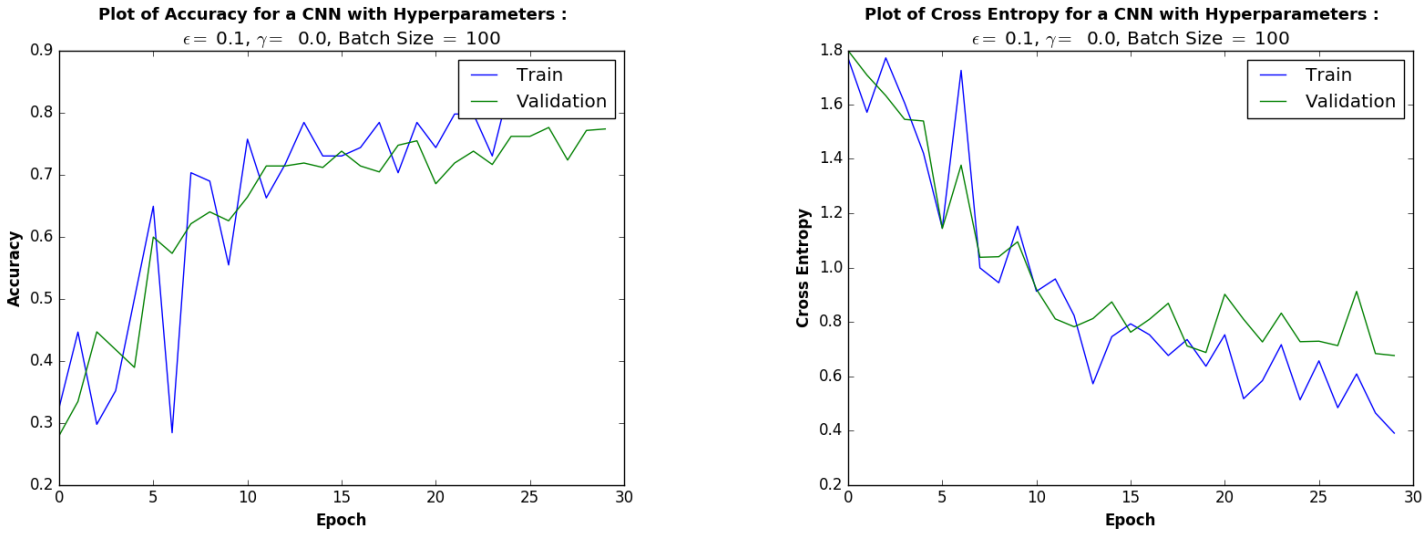


Figure 2: Plots for CNN with default parameters

Question : Examine the statistics and plots of training error and validation error (generalization). How does the network's performance differ on the training set versus the validation set during learning?

During the earlier Epochs the performance on both sets is very similar but as more epochs go by the model performs better on the training data then it does on the validation set (as expected). Meaning the model has worse generalization as you increasing fine tune the model to the training set. The CNN has better generalization then the NN as in its accuracy plot the validation and training curves are closer and reach a higher accuracy.

3.2 Optimization [10 points]

The Relevant plots comprises the next 6 pages (5-10) :

Question : Try different values of the learning rate ϵ ("eps"). Try 5 different settings of ϵ from 0.001 to 1.0. What happens to the convergence properties of the algorithm (looking at both cross-entropy and percent-correct)?

For epsilon = $\epsilon \in \{0.001, 0.01, 0.05, 0.2, 0.1\}$
I found For NN the best convergence tended to the default 0.01 and 0.05. For ϵ smaller than the defaults convergence took longer and accuracy was worse while for larger epsilon the convergence was poor : more noise and accuracy decreased
I found For CNN the best convergence tended to the default value 0.01. For smallest and largest ϵ learning failed and accuracy straightlined. In general starting at the default and increasing epsilon quickens convergence up to a point where convergence fails and flat lines prematurely. While decreasing epsilon from the default value slows convergence up the point where no learning will occur.

Question : Try 3 values of momentum from 0.0 to 0.9. How does momentum affect convergence rate?

I used momentum = $\gamma = 0.1, 0.3, 0.7$
For Both NN and CNN I observed increasing momentum increases the rate at which the accuracy rises and convergence occurs.

Question : Try 5 different mini-batch sizes, from 1 to 1000. How does mini-batch size affect convergence?

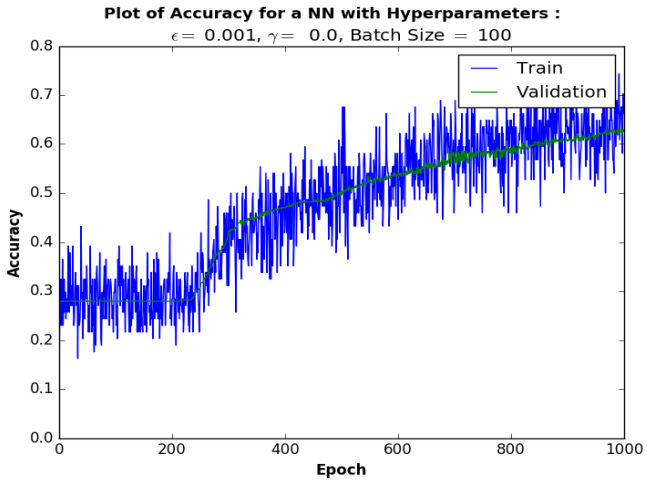
I used Batch Sizes $10, 50, 100, 500$
For Both NN and CNN decreasing the the batch size causes quicker and noisier convergence and increasing the batch size causes slower convergence. The optimal Batch Size seems to be near the default 100.

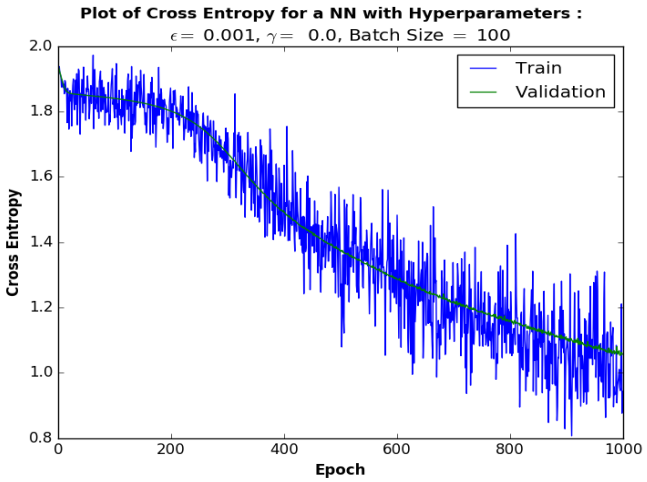
Question : How would you choose the best value of these parameters?

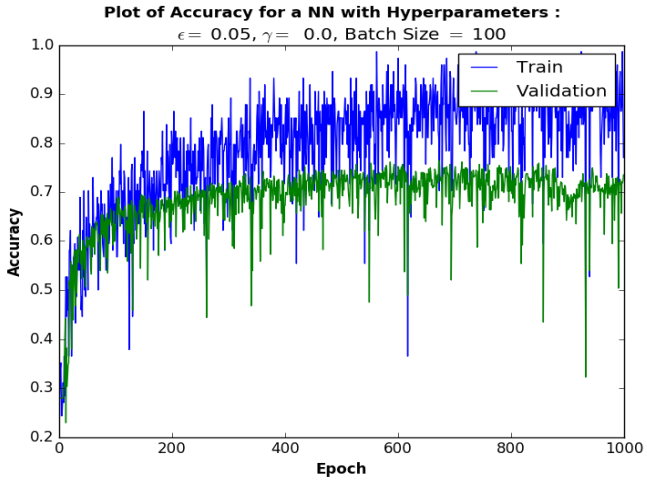
I would choose the best values of these parameters by considering the parameters which lead to the highest accuracy rate and best generalization, with reasonable convergence time and level of noise.

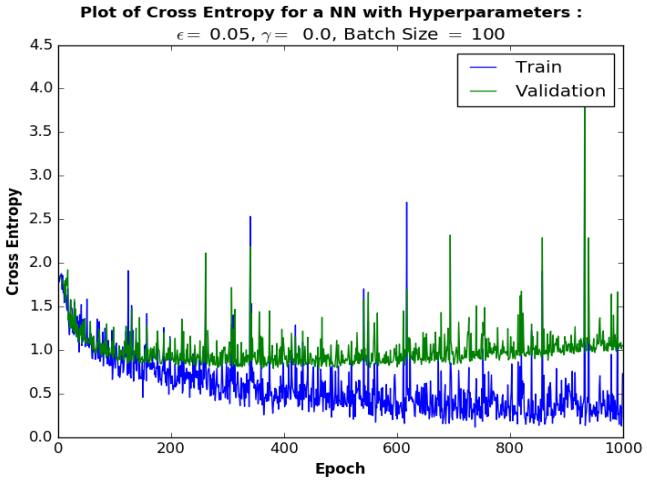
PLOTS ON NEXT 6 PAGES

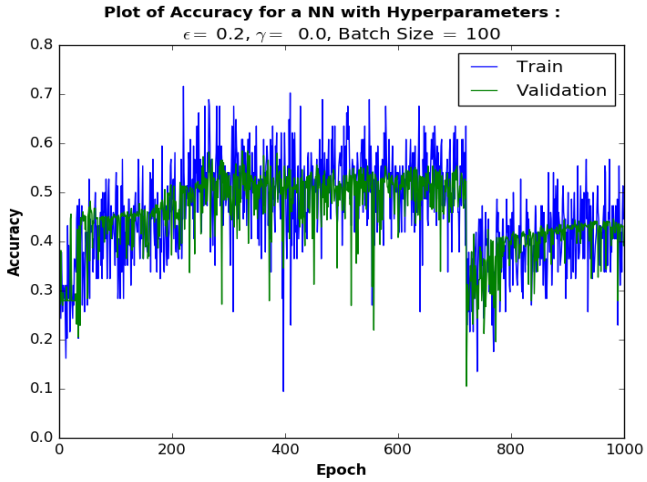
Plots for NN with $\epsilon = 0.001, 0.05, 0.2, 0.5$

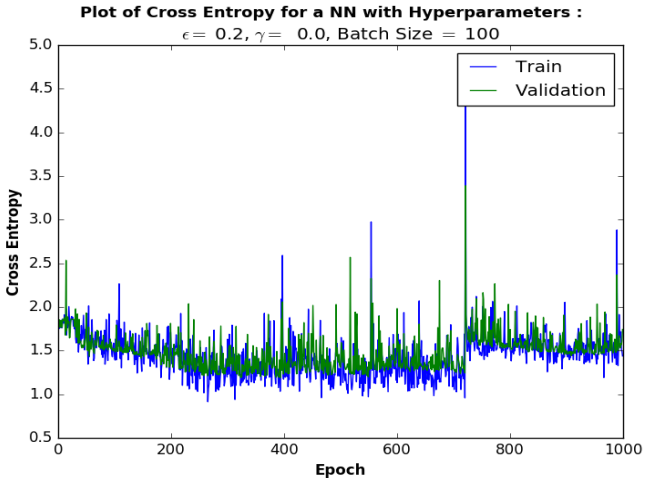




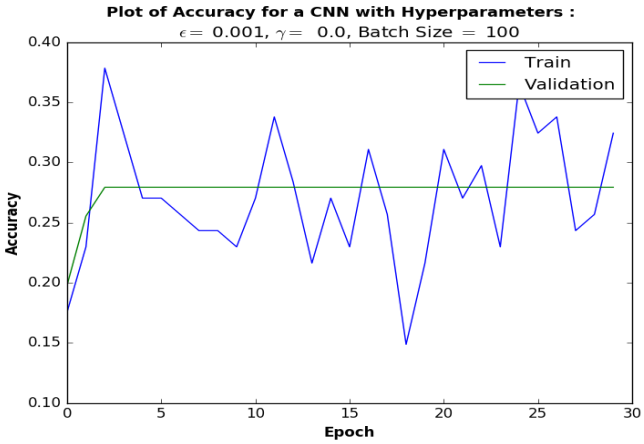


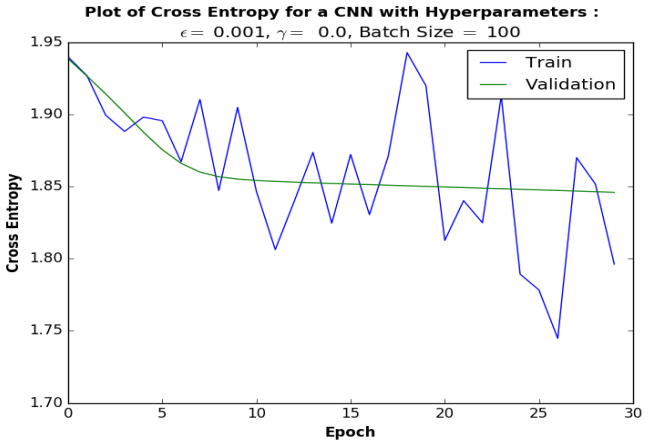


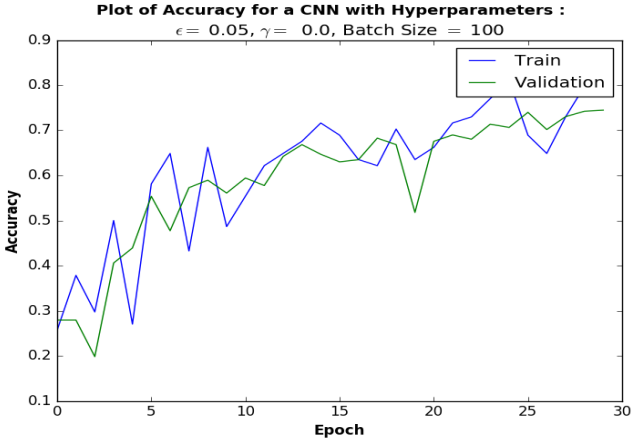


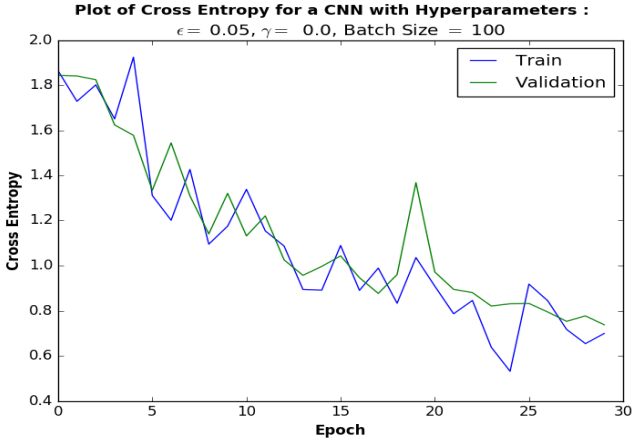


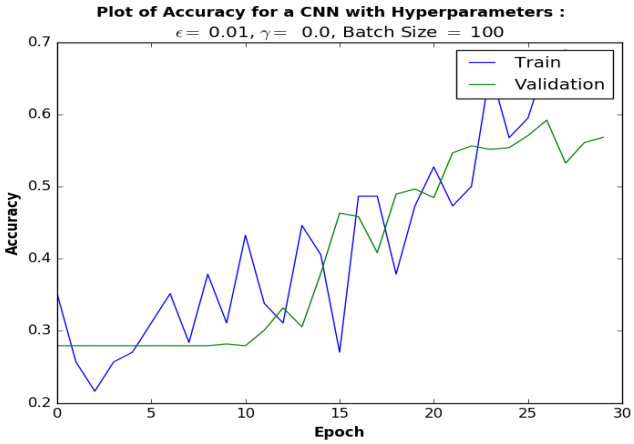
Plots for CNN with $\epsilon = 0.001, 0.05, 0.01, 0.5$

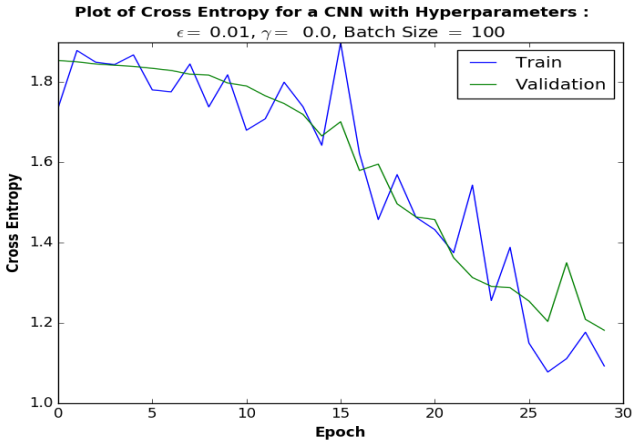


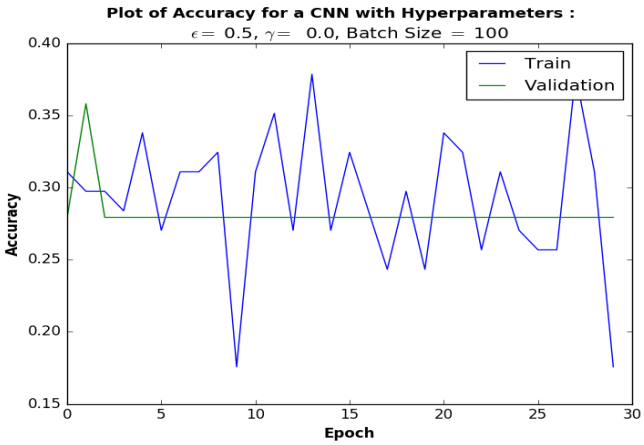


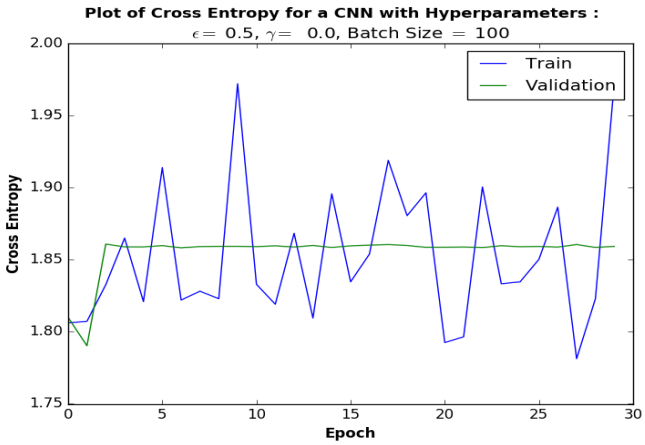




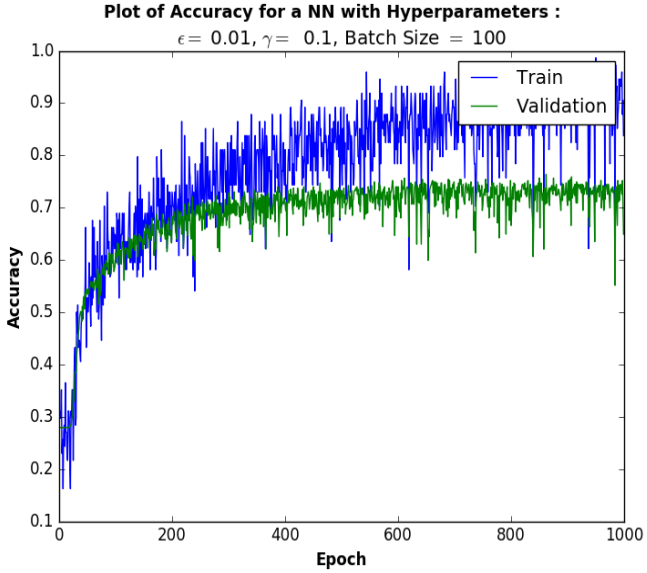


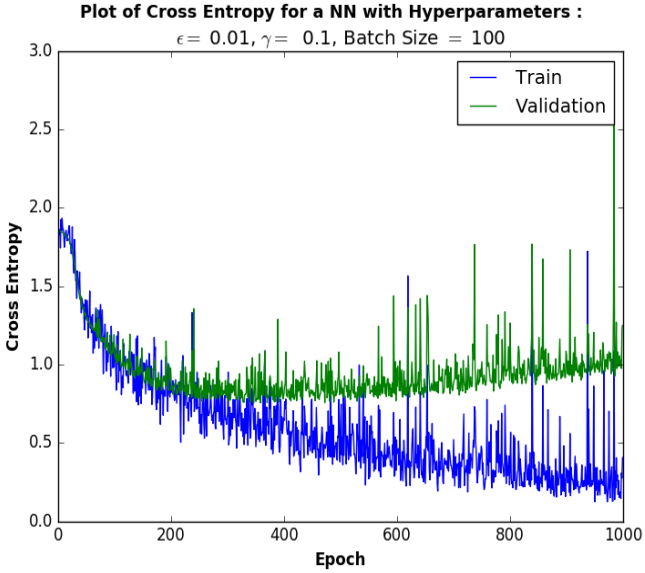


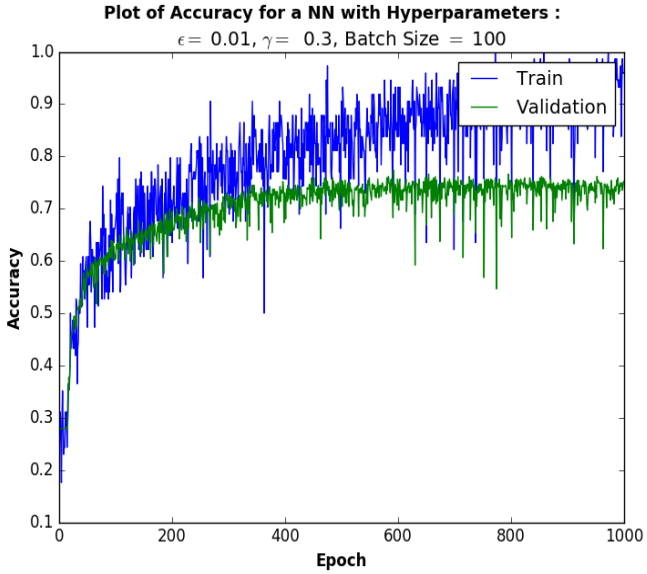


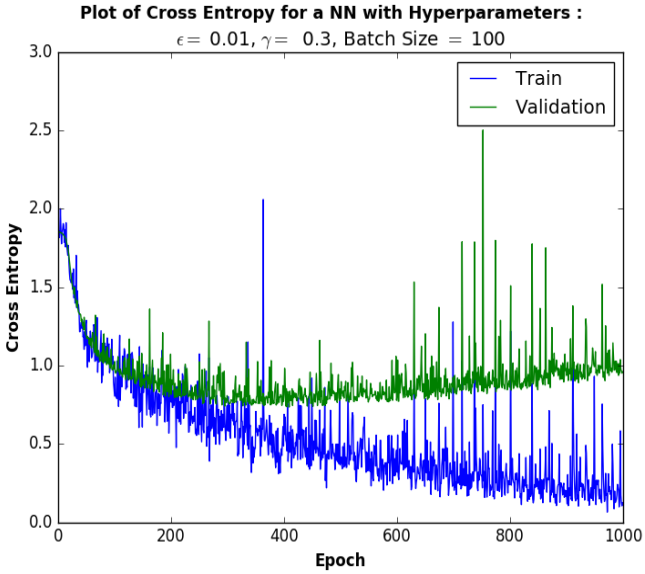


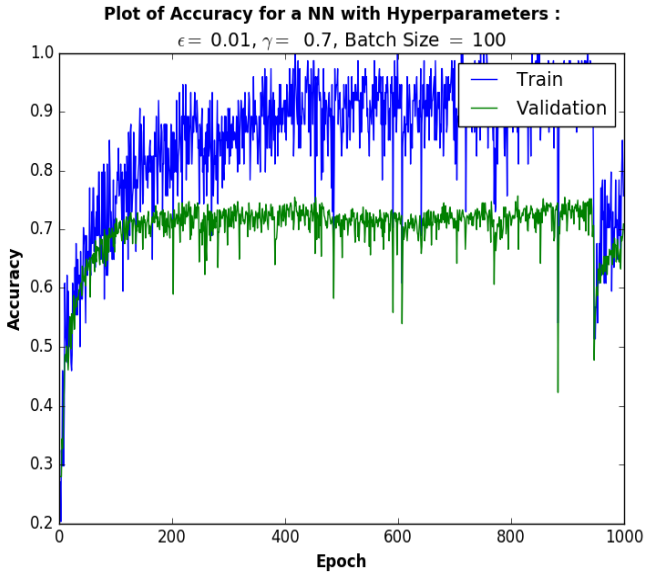
Plots for NN with $\gamma = 0.1, 0.3, 0.7$

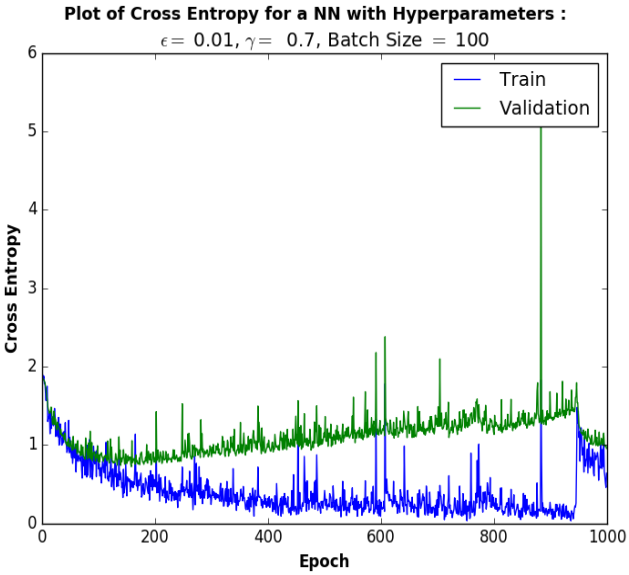




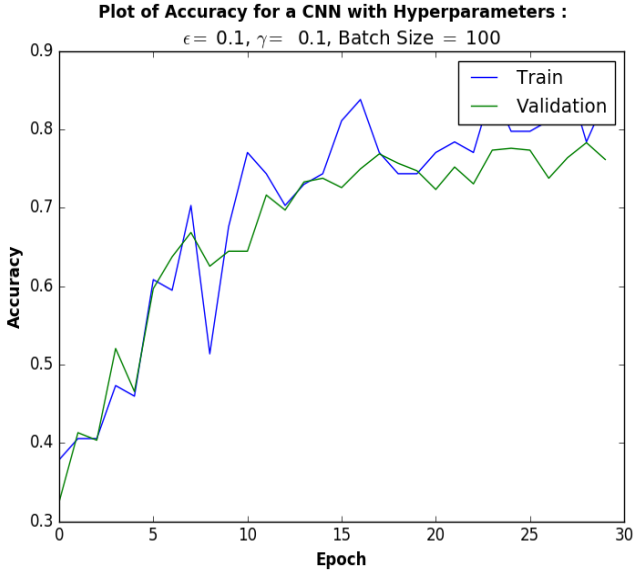


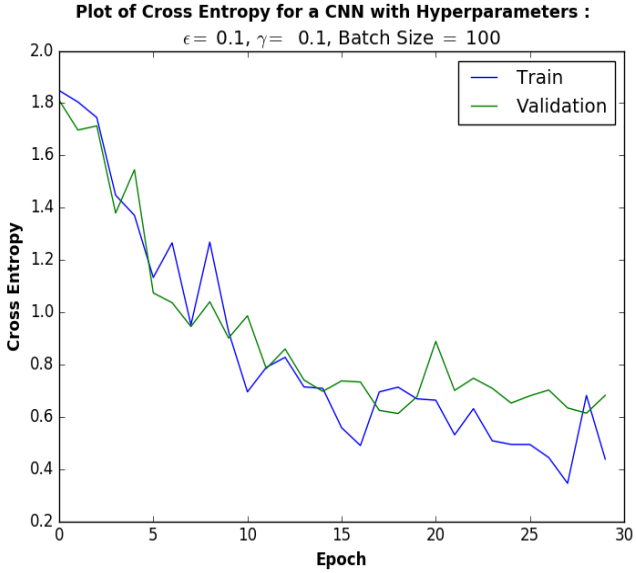


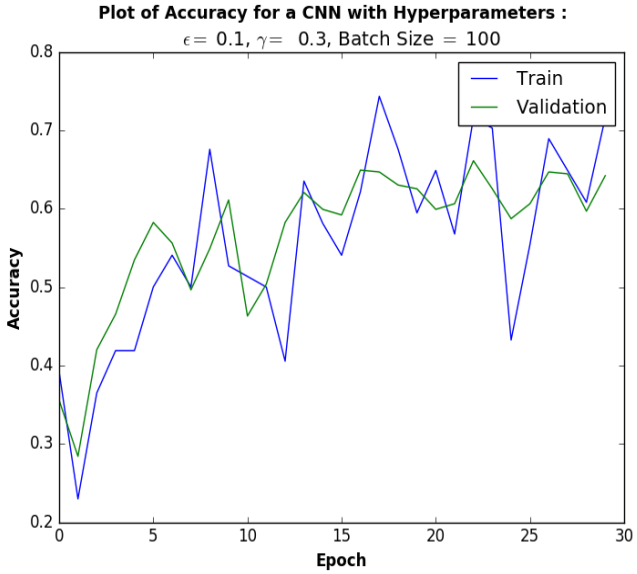


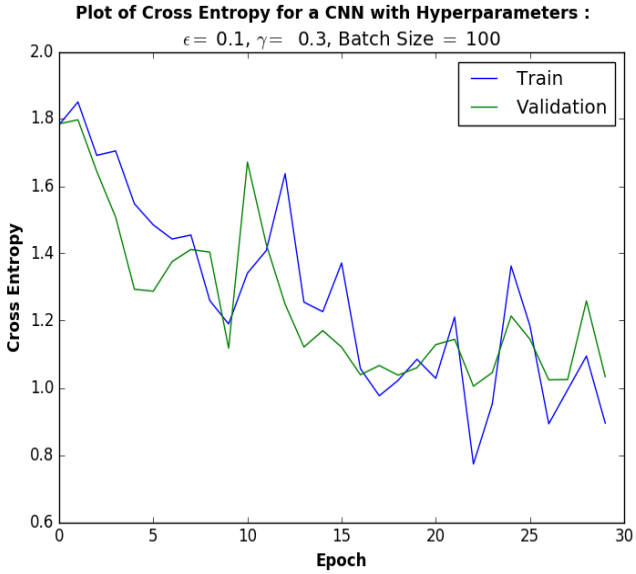


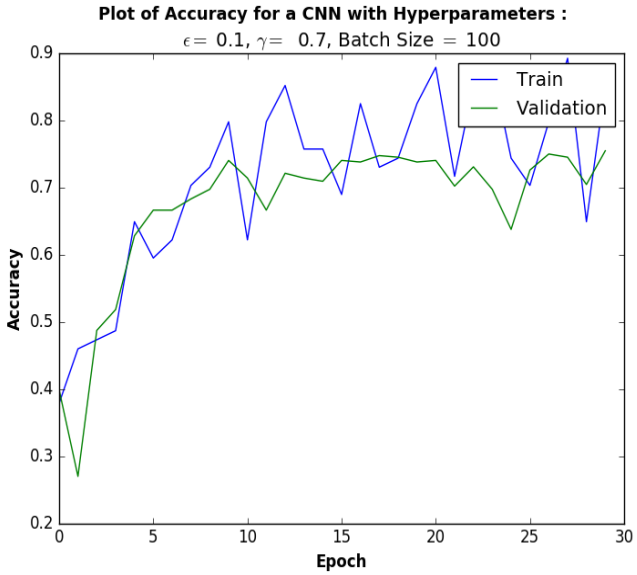
Plots for CNN with $\gamma = 0.1, 0.3, 0.7$

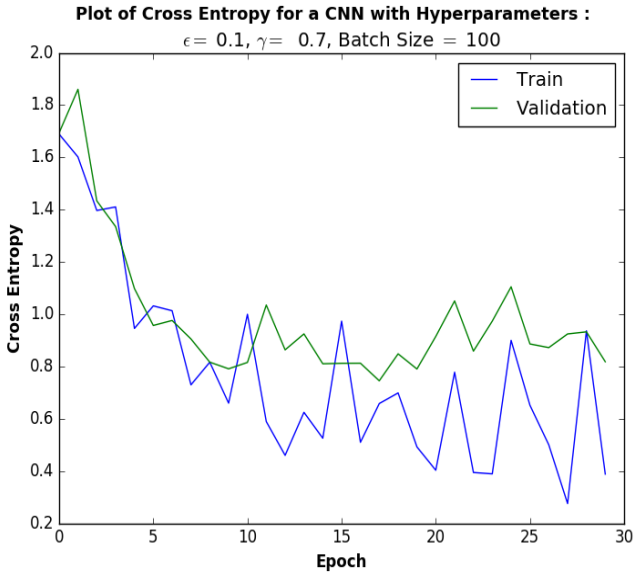




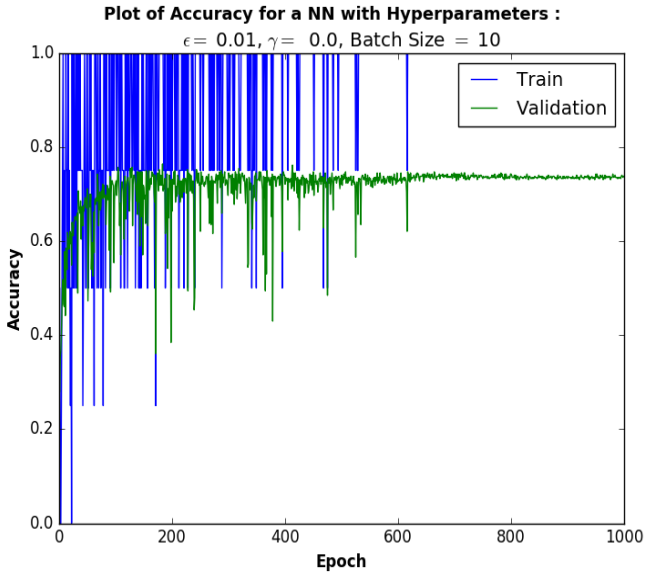


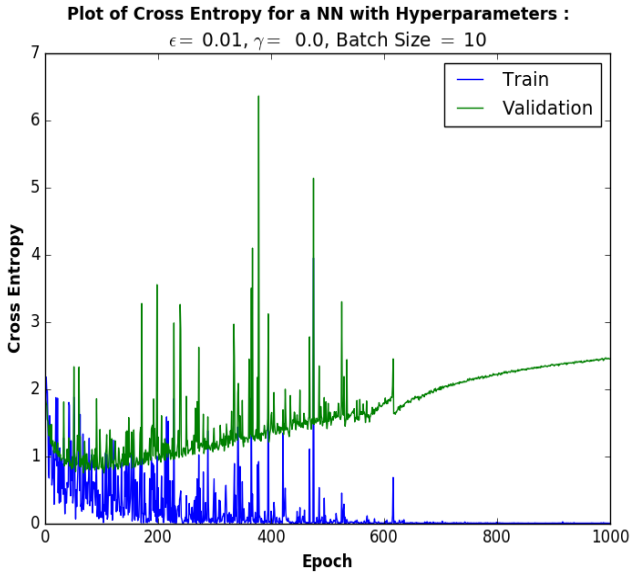


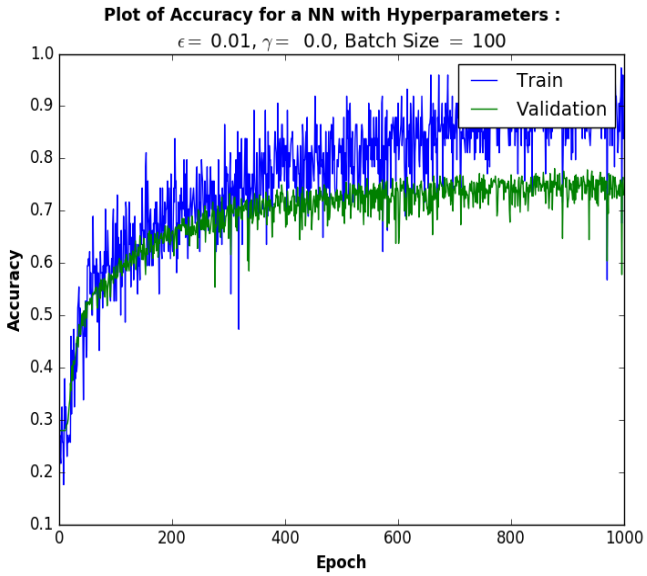


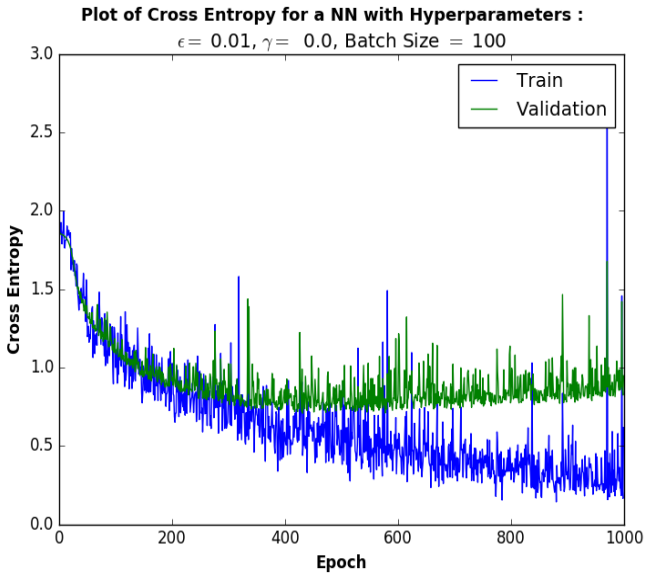


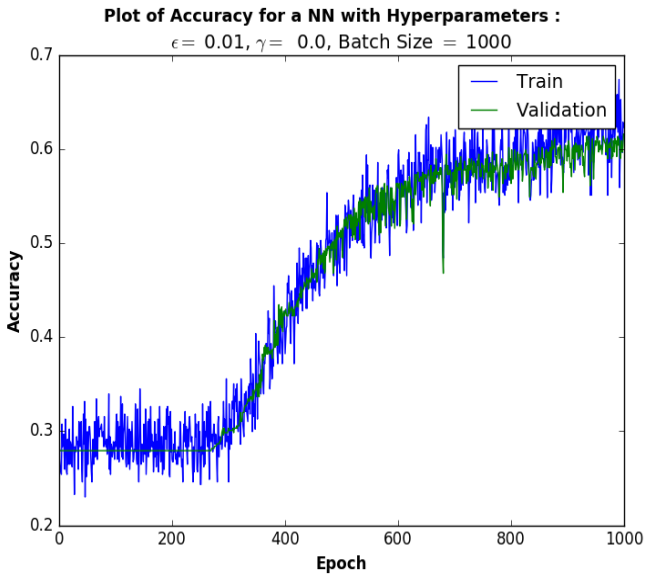
Plots for NN with Batch Size $= 10, 100, 1000$

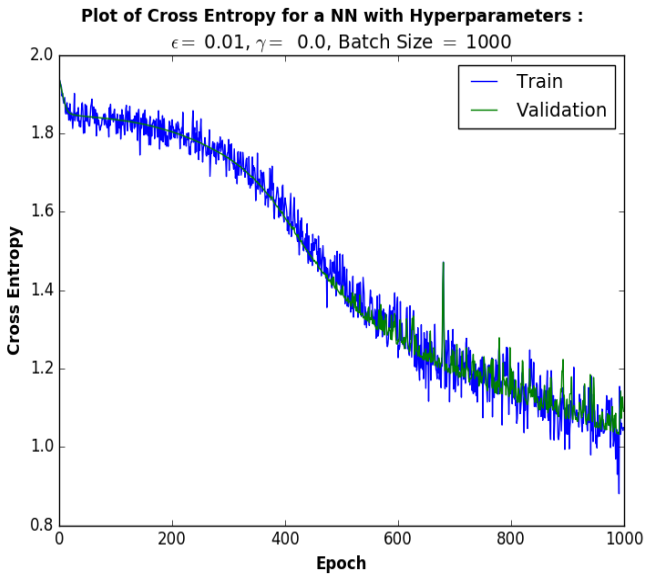




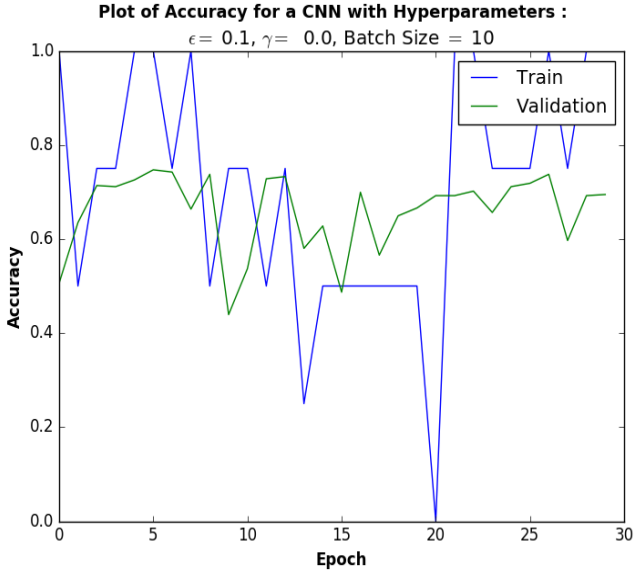


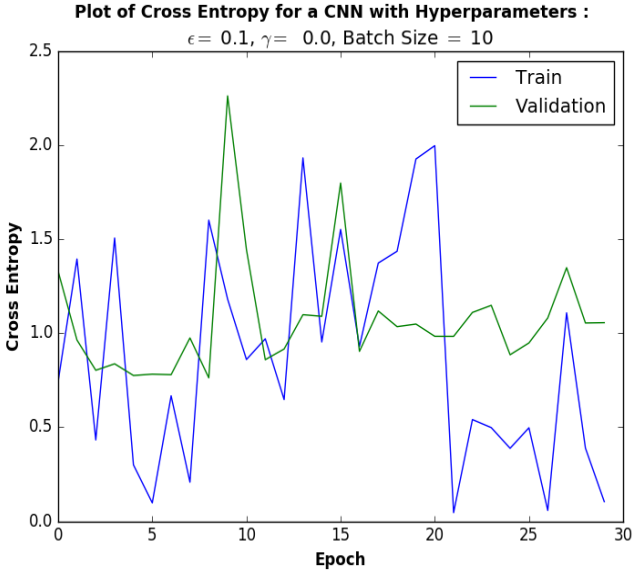


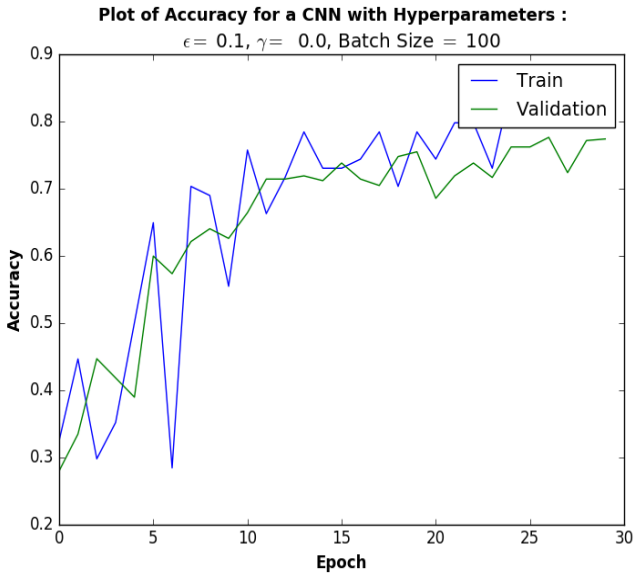


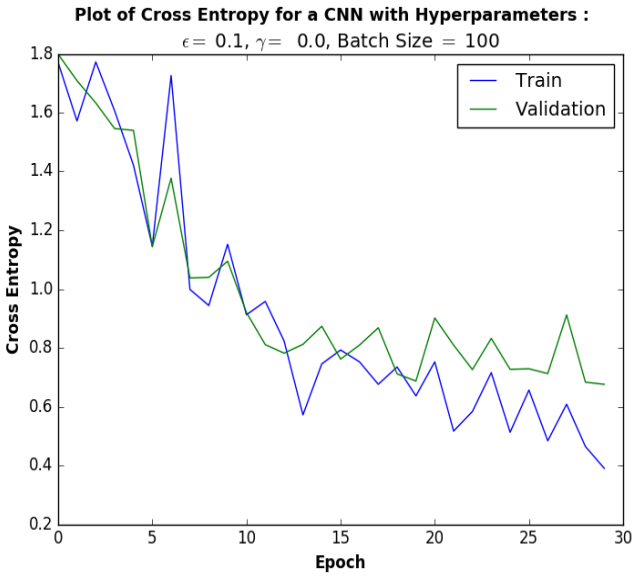


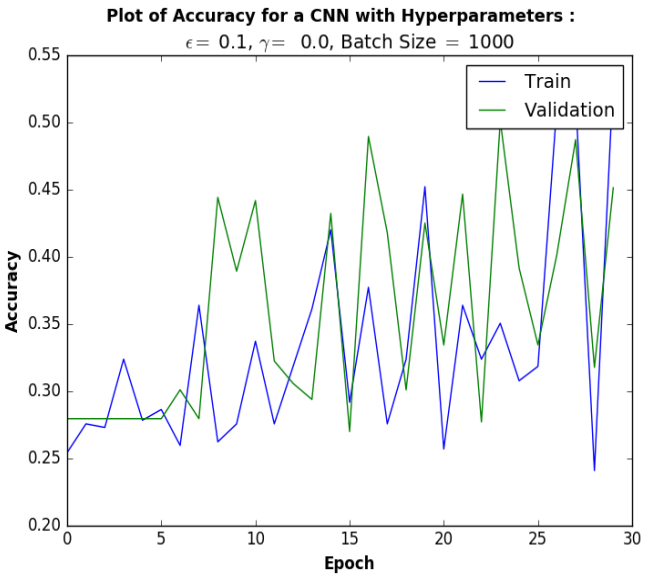
Plots for CNN with Batch Size $= 10, 100, 1000,$

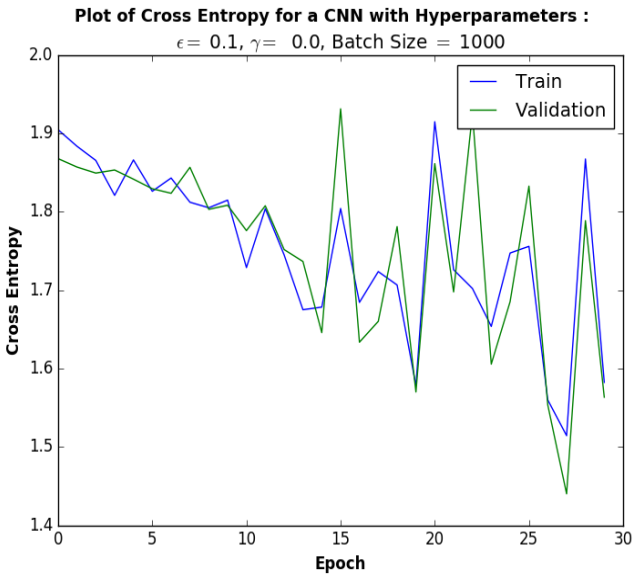










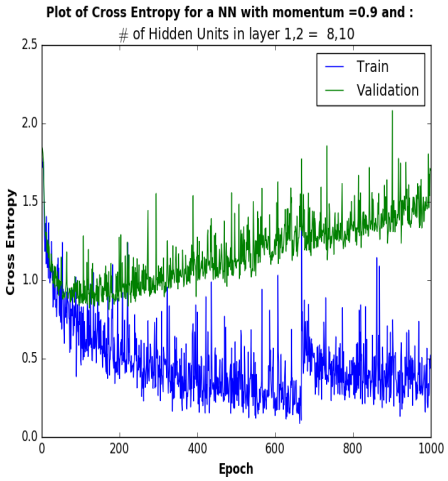


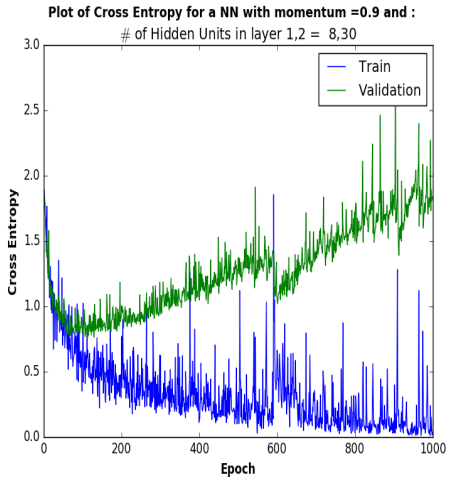
3.3 Model architecture for NN

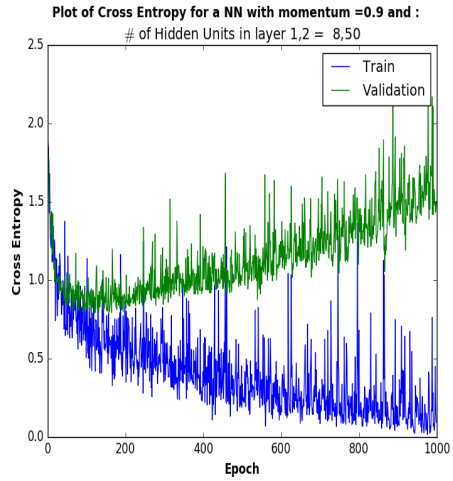
Question : Comment on the effect of changing Hidden Unit/ Filter numbers on the convergence properties, and the generalization of the network.

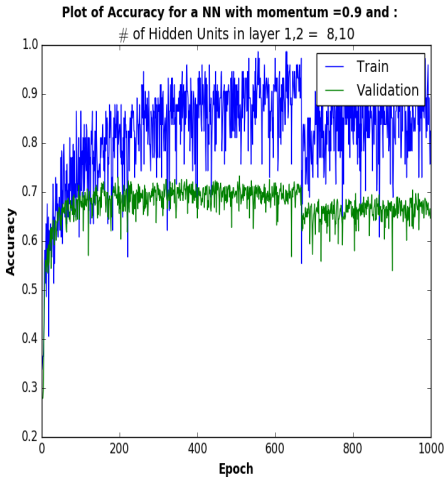
After fixing momentum =0.9 for the NN I tried $\{8, 24, 12\}$ and $\{10, 30, 50\}$ for the number of Hidden units in the first and second layers respectively. Increasing the number of Hidden Units seemed to slightly quicken convergence. As well as lead to slightly worse generalization as the model performs increasing better on the training set for increasing number of filters but reaches a threshold for the validation set where cross entropy starts to increase and accuracy stagnates

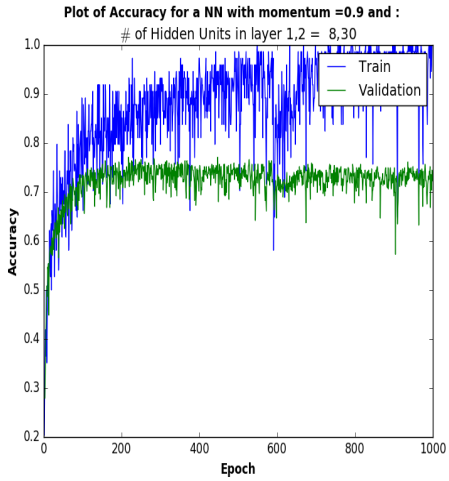
The Plots for 8 Hidden Units in the first Layer and {10,30,50} in the Second:

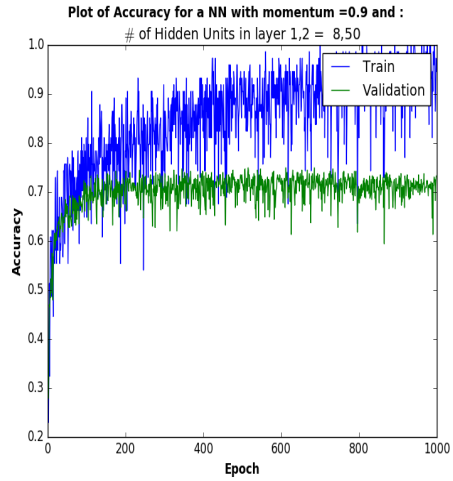




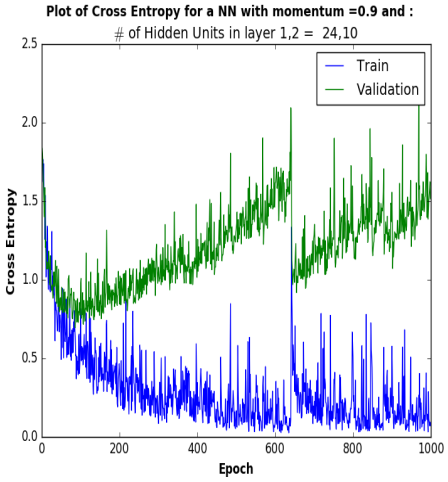


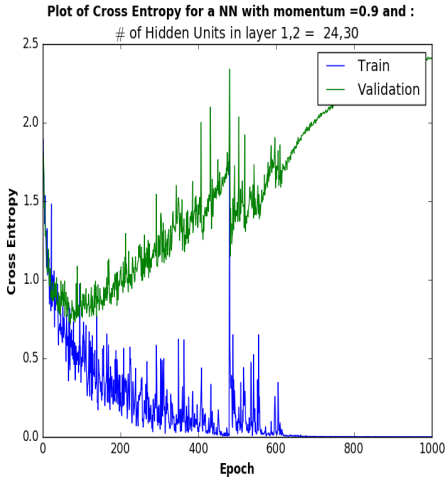


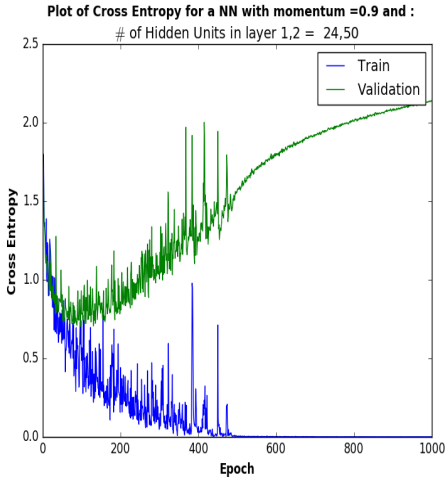


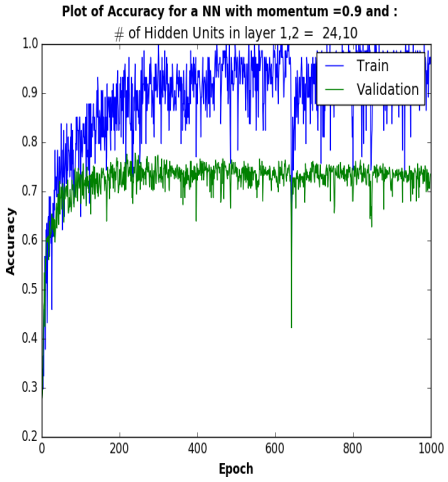


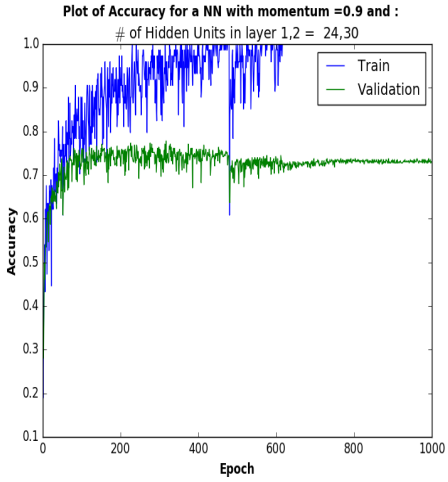
The Plots for 24 Hidden Units in the first Layer and {10,30,50} in the Second:

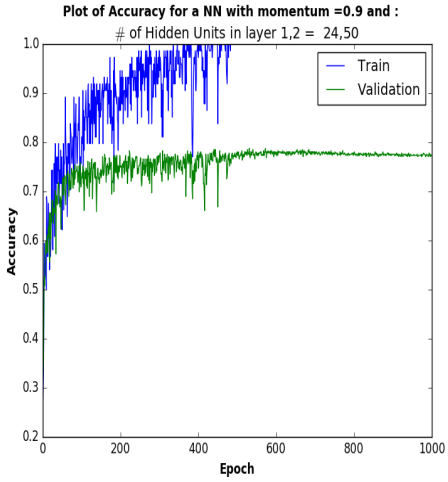




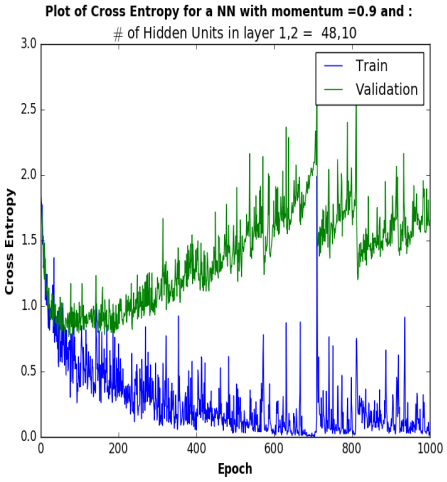


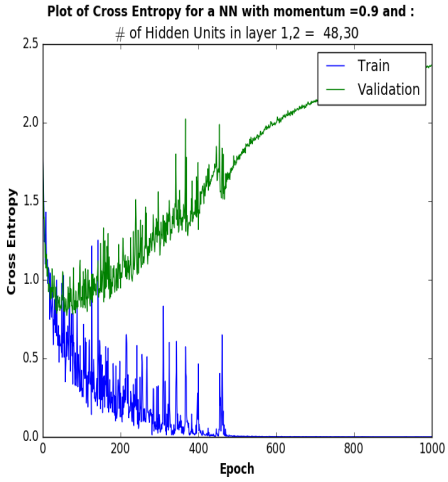


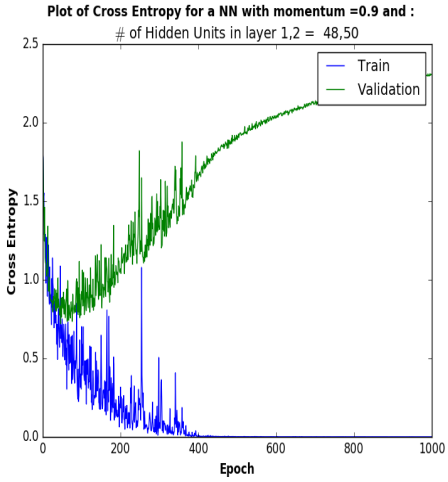


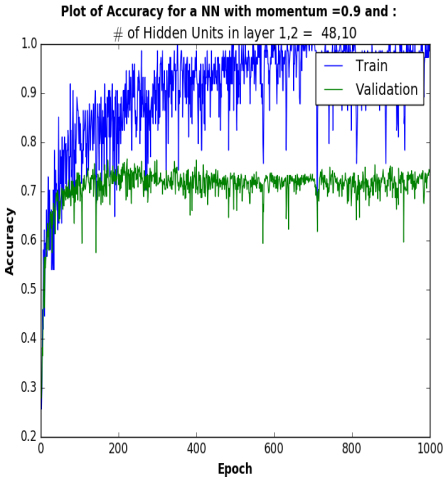


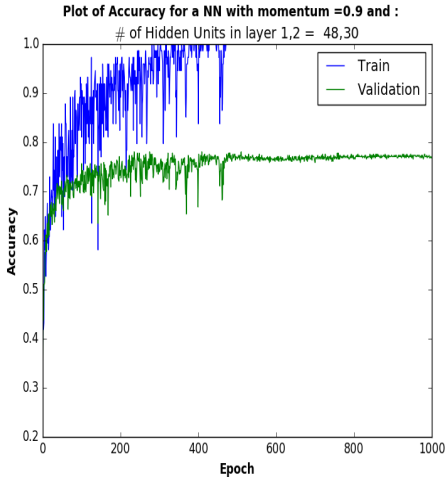
The Plots for 48 Hidden Units in the first Layer and {10,30,50} in the Second:

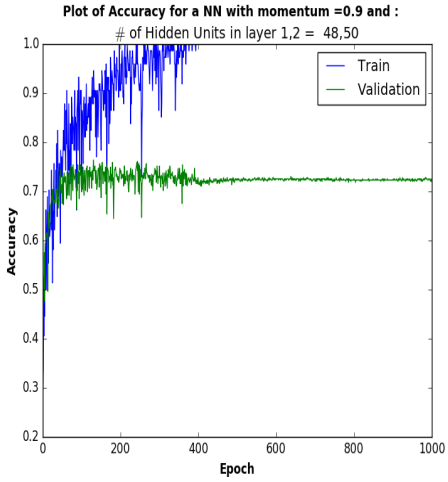










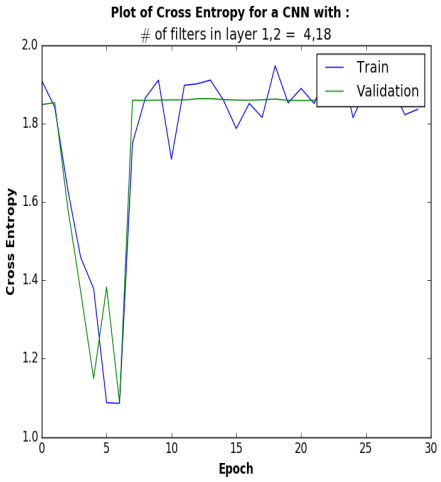


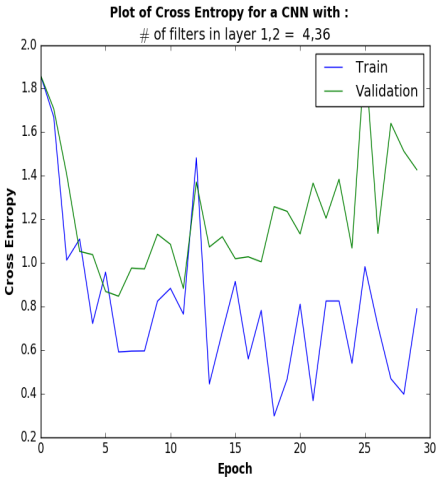
Model architecture for CNN Question : Comment on the effect of changing Filter numbers on the convergence properties, and the generalization of the network.

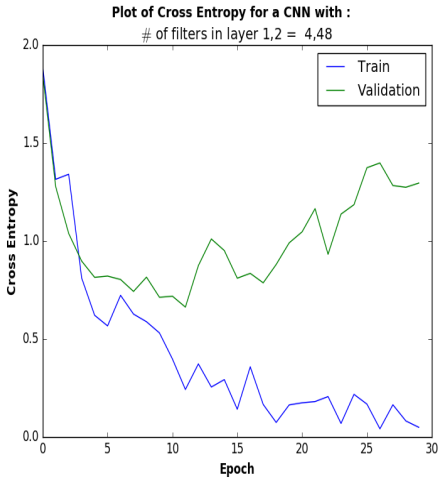
After fixing momentum = 0.9 for the CNN I tried $\{4, 12, 24\}$ and $\{18, 36, 48\}$ for the number of filters in the first and second layers respectively.
Increasing the number of filters seemed to quicken convergence and make the model generalize better as the green and blue accuracy curves got closer together for more filters. Increasing filter number also increased the accuracy of the model for the performance of the CNN on both sets.

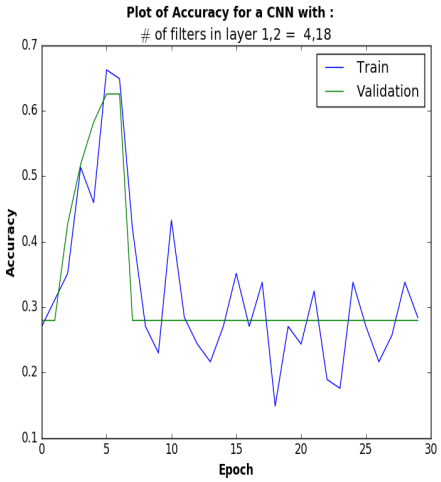
The Relevant plots are in the Next 3 pages

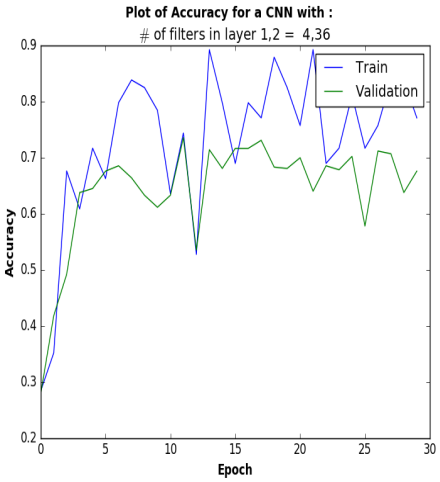
The Plots for 4 Hidden Units in the first Layer and $\{18, 36, 48\}$ in the Second:

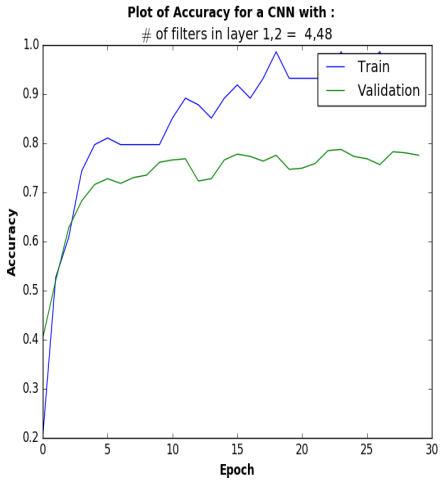




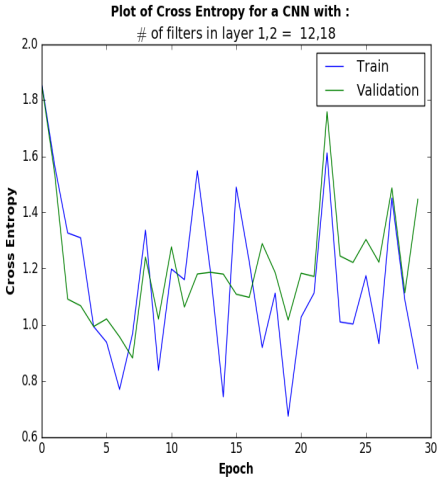


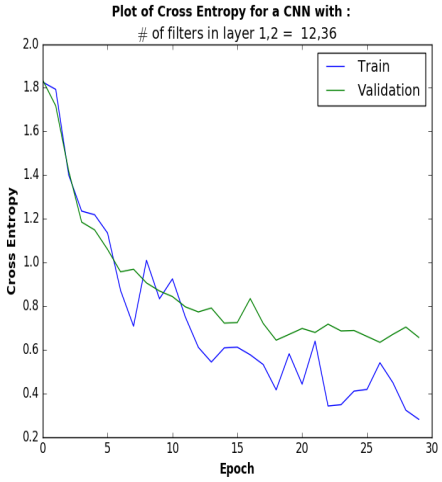


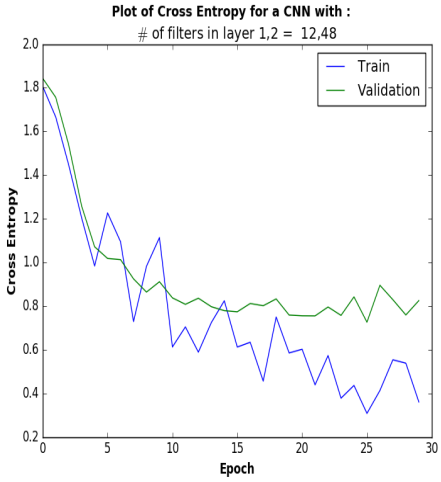


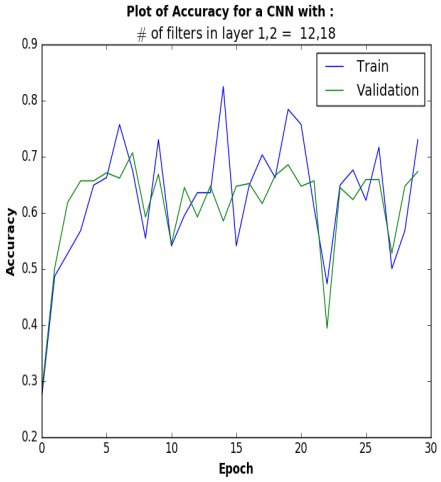


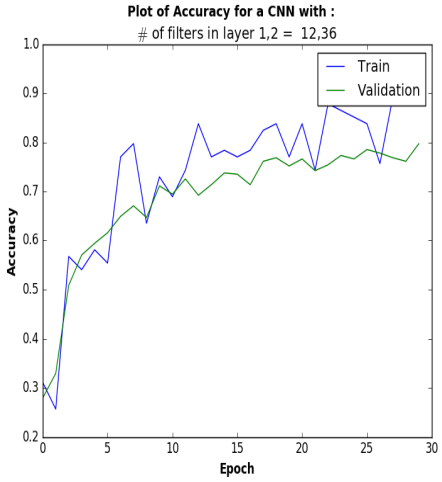
The Plots for 12 Hidden Units in the first Layer and $\{18, 36, 48\}$ in the Second:

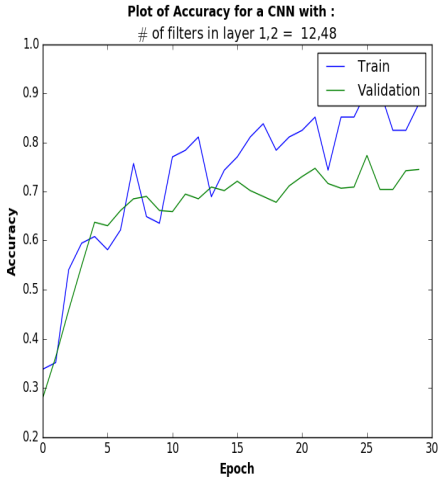




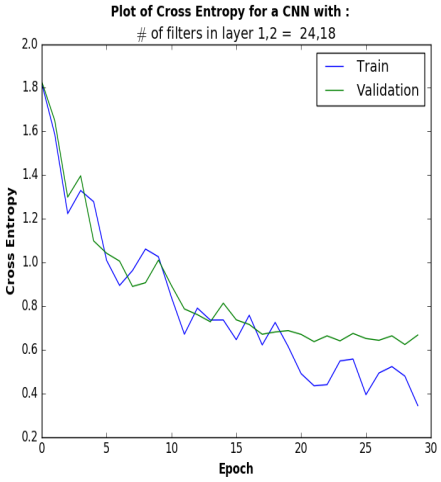


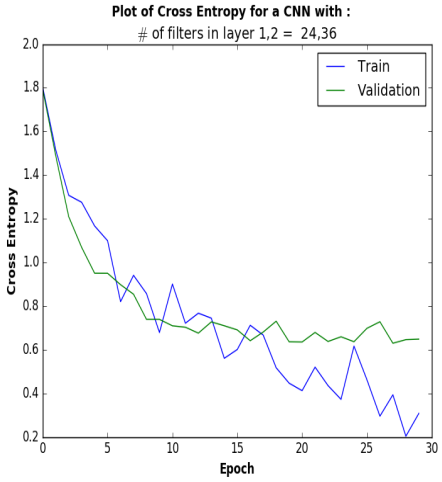


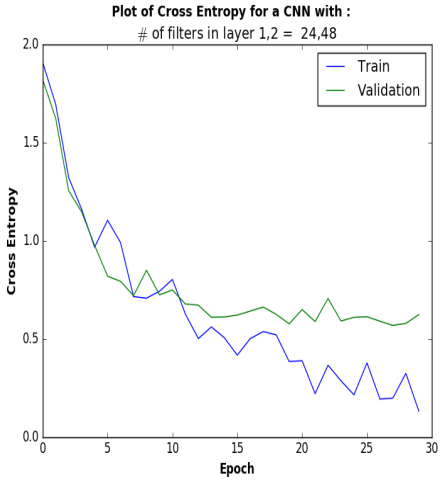


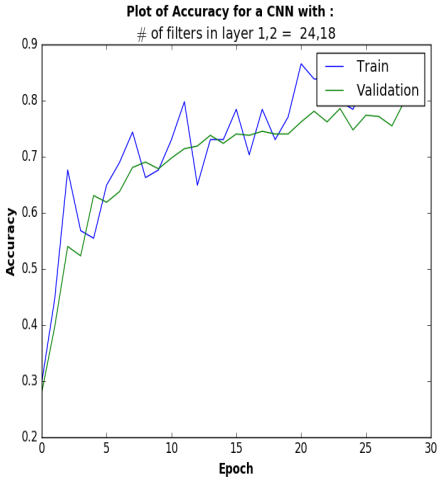


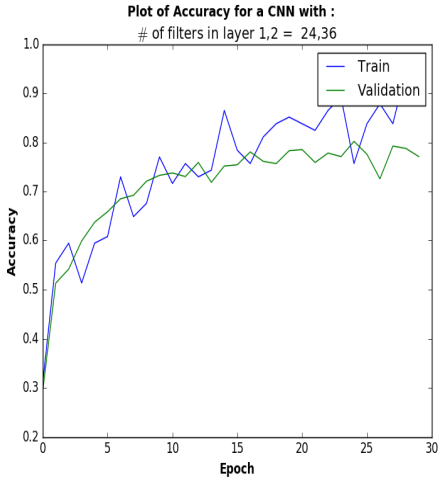
The Plots for 48 Hidden Units in the first Layer and $\{18, 36, 48\}$ in the Second:

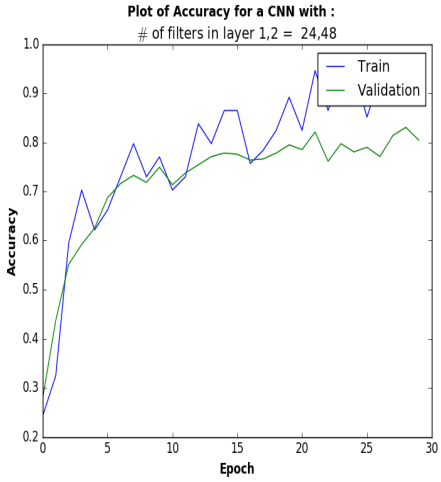












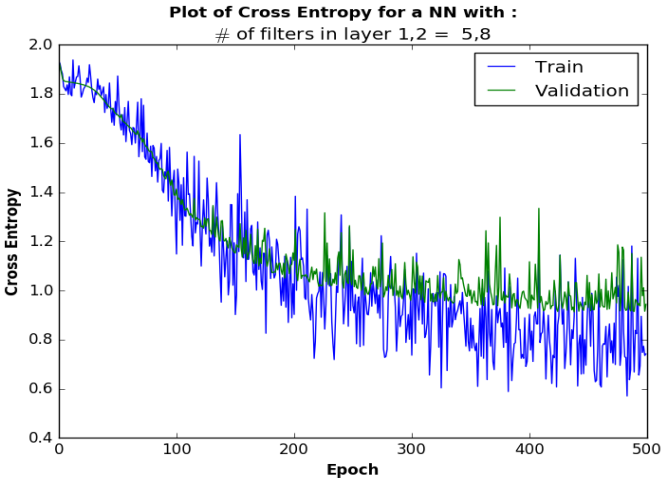
3.4 Compare CNNs and fully connected networks [10 points]

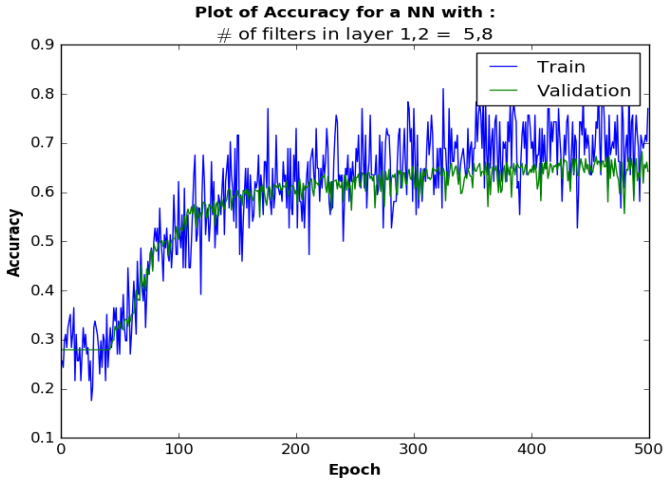
Calculate the number of parameters (including biases) in each type of network.

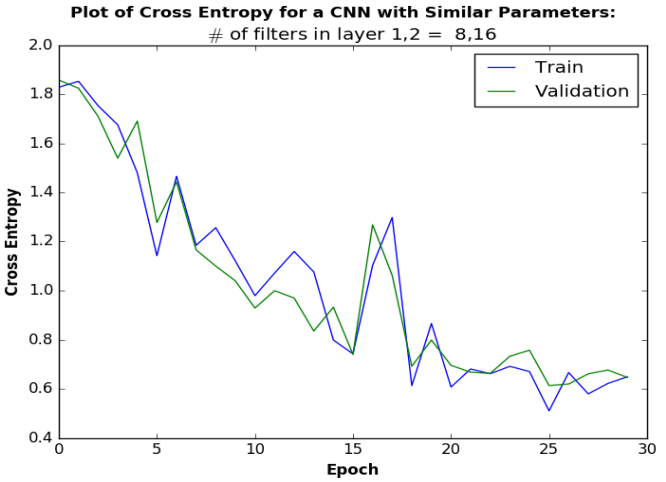
For the NN the number of parameters is the sum of the dimensions of the weights and biases $\sum_i \dim(\mathbf{W}_i) + \dim(\mathbf{b}_i) = 2304 \times 16 + 16 \times 32 + 32 \times 7 + 16 + 32 + 7 = 37,655$ **.**
And For the CNN the number of parameters is the sum $\sum_i \dim(\mathbf{W}_i) + \dim(\mathbf{b}_i) = 8 \times 25 + 8 \times 16 \times 25 + 16 \times 64 \times 7 + 8 + 16 + 7 = 10,599$

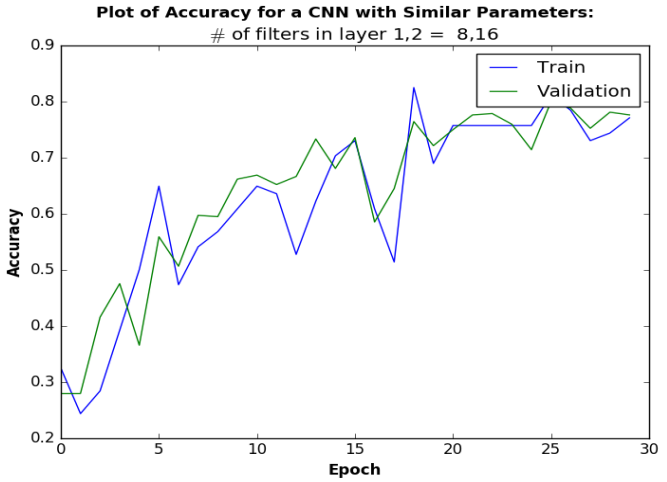
Compare the performance of a conv-net and a regular network for the similar number of parameters. Which one leads to better generalization and why?

Keeping the parameter number in CNN fixed to the default and changing the Hidden units to 5,8 in first and second layer of the NN gives a similar number of parameters for both ($\sim 11K$)**. The error plots are given and we can see that the CNN generalizes better then the NN as it reaches an accuracy of 0.8 and NN barely breaks 0.6 for the validation curve. The validation and training curves are also more close to each for CNN. This is expected as the CNN is able to better learn important features in the training set by focusing in on parts of the image. So for similar parameter numbers it will do this better than the NN which looks at the image as a whole and in this case where there are not many hidden units in the layers : discerning important features in the images is more difficult.**



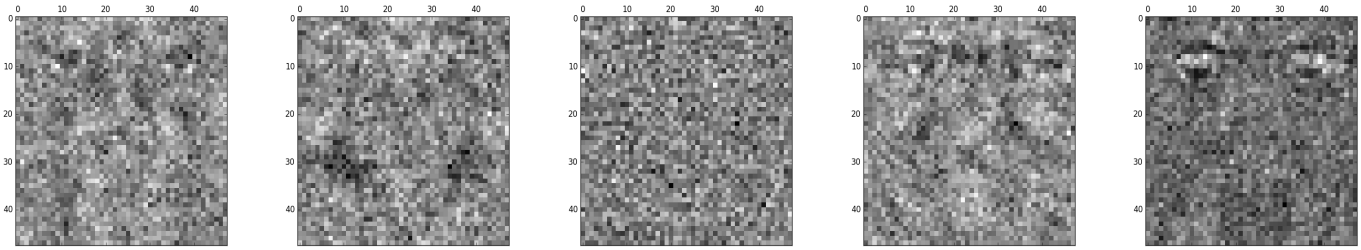




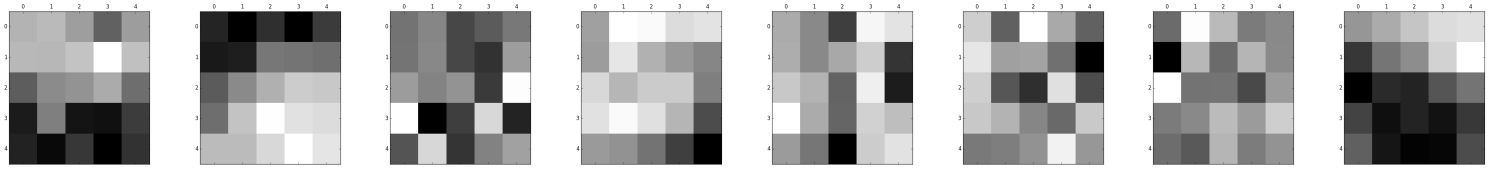


Question : Plot the first layer filters of the CNN, and the first layer weights of the NN.

The Plots of the first Layer Weights for the NN



The Plots of the first Layer Weights for the CNN



Question : Briefly comment on the visualization.

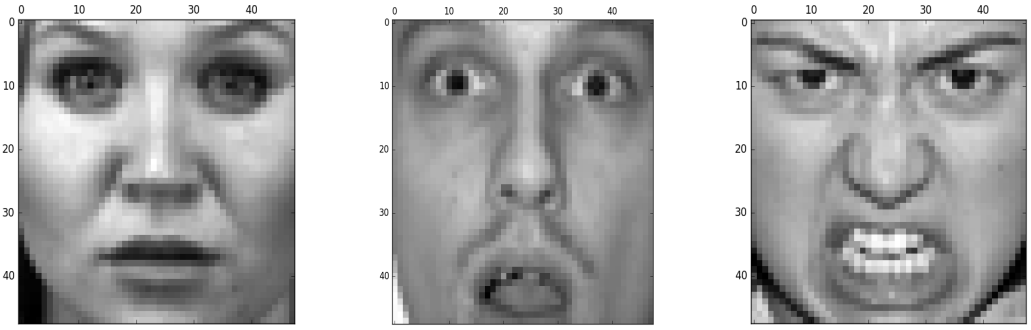
The weights visualization for NN will highlight features it deems important such as eyes and noses and mouths some of which are slightly visible.

The weights visualization for CNN will highlight edges as changes from black to white.

3.5 Network Uncertainty [5 points]

Plot some examples where the NN is not confident of the classification output (the top score is below some threshold),

Here are a few examples of where the NN is not confident. The threshold I used was having the prediction probability for each class to be near equal class probability or (> 0.1) So that the classifier to completely not confident of the classifier. It took a few different models to get examples of this threshold :



Question : Comment on them. Will the classifier be correct if it outputs the top scoring class anyways?

The classifier classified the images as Anger, Surprise and Neutral (from left to right). Which is only correct for the middle image.

4 Mixtures of Gaussians [40 points]

4.1 Code: I read and understood the code.

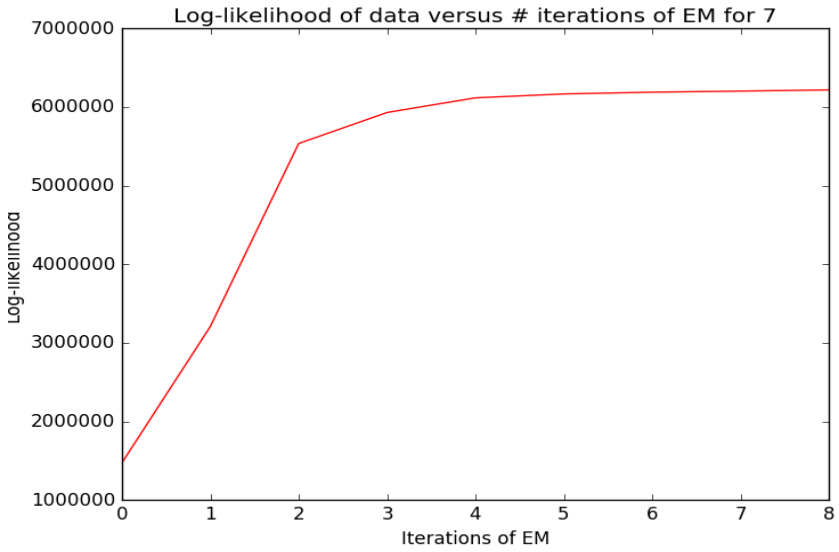
4.2 Training [10 points]

Questions : Choose a good model, Show mean and variance images. Show the mixing proportions. Provide the curve of log-likelihood.

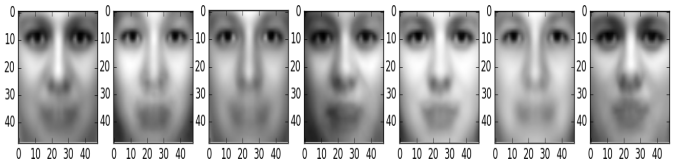
For 7 clusters in the Gaussian mixture model, and the minimum variance be 0.01. I choose to test models with `randConst` to 0, 1, 10, 100. I went with `randConst = 100` since this leads to initializing π_k as a uniform distribution. I got the following mixing coefficients

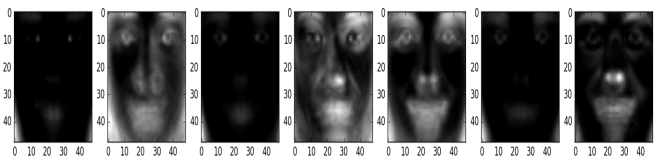
```
[[ 0.28037112] [ 0.07348266] [ 0.21544974] [ 0.04593954] [ 0.11920476] [ 0.22372302] [ 0.04182916]]
```

Here is the plot of log-likelihood:



Here are the means and variance vector images :

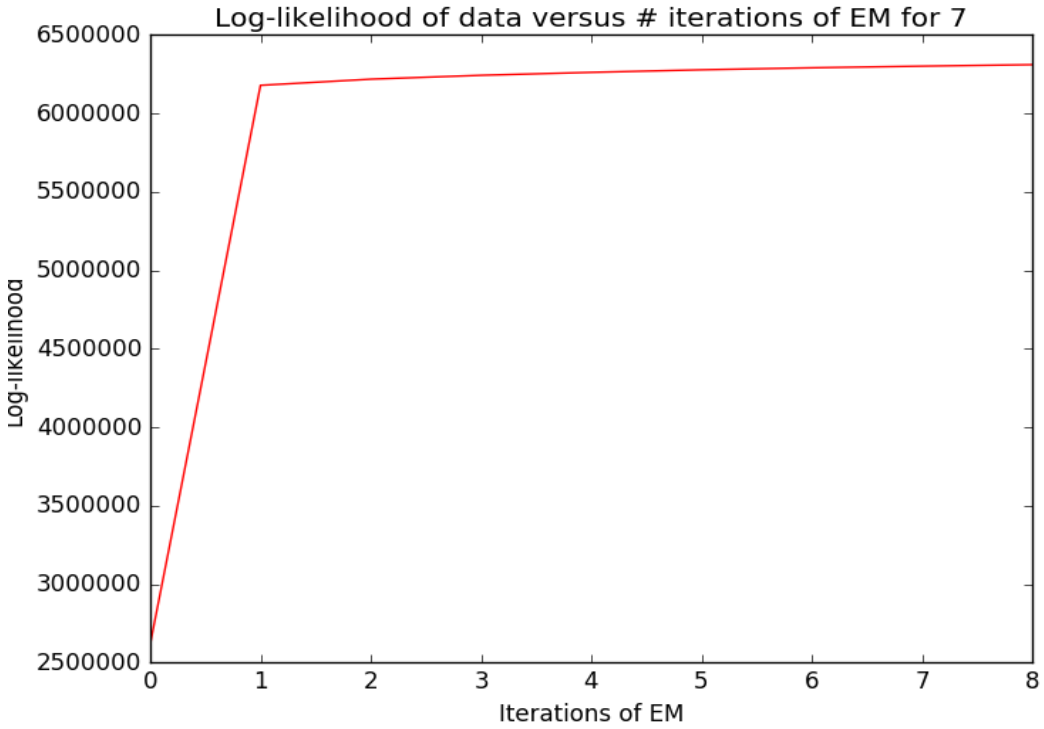




4.3 Initializing a mixture of Gaussians with k-means [10 points]

Question : Show the log-likelihood curves and comment on the speed of convergence resulting from the two methods.

Here is the log-likelihood plot

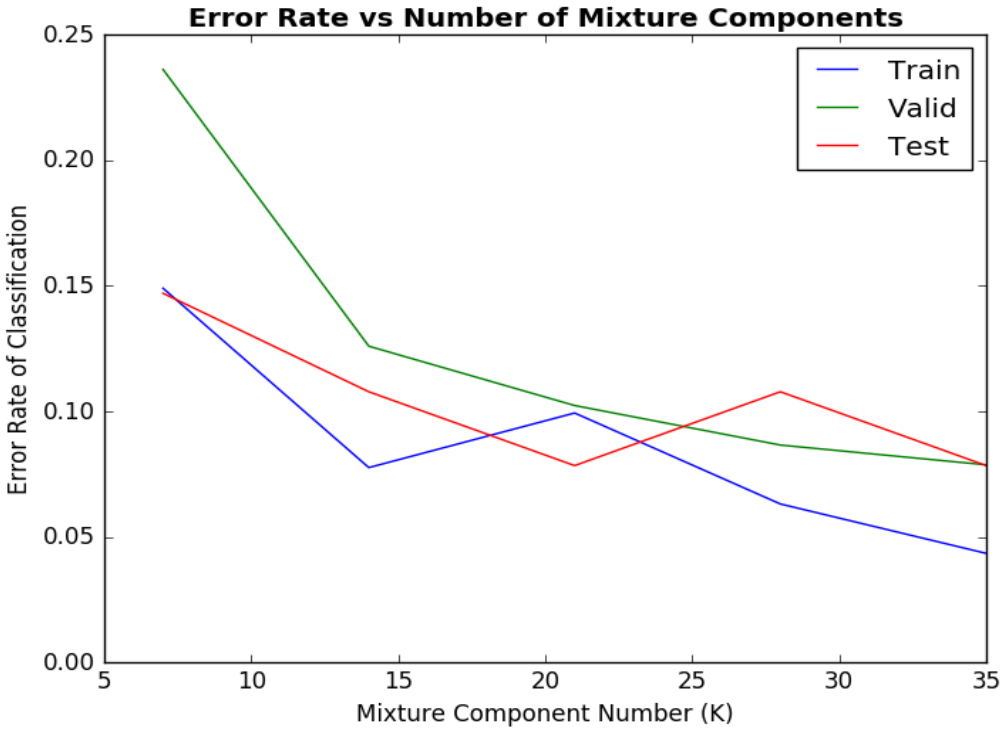


Clearly with the kmeans initialization the EM algorithm converges quicker than the first plot in 4.2. Here it takes 2 iterations to converge while with the other initialization it takes about 4 iterations to converge.

4.4 Classification using MoGs [20 points]

(a) (10 points) **Plot the error rate of classification for training, validation and testing sets.**

The error plot is shown :



(b) **[5 points] Question : You should find that the error rates on the training sets generally decrease as the number of clusters increases. Explain why.**

Clearly as you increase the number of mixture components you are adding complexity to the model and fitting the training set closer and closer so that the error rates on the training set generally decrease as number of clusters increases.

(c) **[5 points] Question : Examine the error rate curve for the test set and discuss its properties. Explain the trends that you observe.**

The red test curve decreases slightly from 0.15 to 0.08. This could mean that the model has slightly better generalization and accuracy for more mixture components. From 21-28 mixture components there is an increase in the error rate before it goes back down at 35 mixture components. Thus the optimal model is 21 mixture components for smallest error rate on the testing set. Also 35 mixture components is almost as good.