University of Toronto
CSC 2515 Introduction to Machine Learning
Assignment 2
Due: November 14, 2016 (10:00AM EST)

Name: John de Vera
Student Number: 996393353

Assignment 2

Part 1 - EM for Mixture of Gaussians

$$p(x) = \sum_{k=1}^{k} \pi_k \mathcal{N}(x|\mu_k \Sigma_k) = \sum_{k=1}^{k} p(z_k) p(x|z_k) \qquad (1)$$

where $$\Sigma_k = \Sigma, \qquad \pi_k = p(z), \qquad \mathcal{N}(x|\mu_k \Sigma_k) = p(x|z_k) \qquad (2)$$

where responsibility $\gamma(z_k)$is defined as the responsibility of the variable $x^{(n)}$ using Bayes' Rule:

$$\gamma(z_k) = p(z_k = k|x) = \frac{p(z_k=k)p(x|z_k=k)}{p(x)} = \frac{p(z_k=k)p(x|z_k=k)}{\sum_{j=1}^{k} p(z_j=j)p(x|z_j=j)} \qquad (3)$$

Performing the Maximum Likelihood Estimation, on p(x) from (1), conditioned on π, μ, Σ

Log likelihood:

$$l(\pi, \mu, \Sigma) = p(x|\pi, \mu, \Sigma) = \sum_{n=1}^{N} \ln p\left(x^{(n)}\middle|\pi, \mu, \Sigma\right) = \sum_{n=1}^{N} \ln \sum_{k=1}^{k} p\left(x^{(n)}\middle|z^{(n)};\ \mu, \Sigma\right) p(z^{(n)}|\pi) \qquad (4)$$

We can replace the sum of k because it is only relevant for z = 1. Therefore:

$$l(\pi, \mu, \Sigma) = \sum_{n=1}^{N} \ln p\left(x^{(n)}, z^{(n)}\middle|\pi, \mu, \Sigma\right) = \sum_{n=1}^{N} \ln p\left(x^{(n)}\middle|z^{(n)};\ \mu, \Sigma\right) + \ln p(z^{(n)}|\ \pi)$$

From equation 1**, take derivative, w.r.t. μ** and set to 0

$$\frac{\partial \ln p(X|\ \pi, \mu, \Sigma)}{\partial \mu_k} = 0 = -\sum_{n=1}^{N} \gamma(z_k)\, \Sigma(x^{(n)} - \mu)$$

multiply both side by Σ^{-1}

$$0 = -\sum_{n=1}^{N} \gamma(z_k)\, x^{(n)} + \sum_{n=1}^{N} \gamma(z_k)\, \mu$$

$$\mu_k = \frac{1}{N} \sum_{n=1}^{N} \gamma(z_k)\, x^{(n)}$$

$$N_k = number\ of\ points\ assigned\ to\ cluster\ k = \sum_{n=1}^{N} \gamma(z_k)$$

From equation 1, **take derivative (chain rule), w.r.t. $\boldsymbol{\Sigma_k = \Sigma}$** and set equal to 0

$$\frac{\partial \ln p(X|\ \pi, \mu, \Sigma)}{\partial \Sigma} = 0 = \frac{\partial \ln p(X|\ \pi, \mu, \Sigma)}{\partial \mathcal{N}} \frac{\partial \mathcal{N}}{\partial \Sigma}$$

$$= \frac{\partial (ln\pi_k + ln\mathcal{N}(x|\mu_k \Sigma))}{\partial \mathcal{N}} \frac{\partial \mathcal{N}}{\partial \Sigma}$$

$$= \frac{1}{\sum_{j=1}^{k} \pi_k \mathcal{N}(x|\mu_k \Sigma)} \frac{\partial \mathcal{N}}{\partial \Sigma} \ (first\ term\ denominator\ \forall\ k)$$

$$\frac{\partial \mathcal{N}}{\partial \Sigma} = \left(-\frac{1}{2}\right)\left((2\pi)^{\frac{-D}{2}}\right)\left(|\Sigma|^{\frac{-1}{2}}\right)\left(|\Sigma|^{\frac{-3}{2}}\right)^T \exp\left(\left(-\frac{1}{2}\right)\left(x^{(n)} - \mu\right)^T \Sigma^{-1}\left(x^{(n)} - \mu\right)\right) +$$
$$\left((2\pi)^{\frac{-D}{2}}\right)\left(|\Sigma|^{\frac{-1}{2}}\right) \exp\left(\left(-\frac{1}{2}\right)\left(x^{(n)} - \mu\right)^T \Sigma^{-1}\left(x^{(n)} - \mu\right)\right)\left(\left(\frac{1}{2}\right)\left(x^{(n)} - \mu\right)^T \Sigma^{-2}\left(x^{(n)} - \mu\right)\right)$$

$$= \left((2\pi)^{\frac{-D}{2}}\right)\left(|\Sigma|^{\frac{-1}{2}}\right) \exp\left(\left(-\frac{1}{2}\right)\left(x^{(n)} - \mu\right)^T \Sigma^{-1}\left(x^{(n)} - \mu\right)\right)\left(\left(-\frac{1}{2}\right)\left(|\Sigma|^{\frac{-3}{2}^T} + \Sigma^{-2}\left(x^{(n)} - \mu\right)^T\left(x^{(n)} - \mu\right)\right)\right)$$

$$= \mathcal{N}(x|\mu_k \Sigma)\left(\left(-\frac{1}{2}\right)\left(|\Sigma|^{\frac{-3}{2}^T} + \Sigma^{-2}\left(x^{(n)} - \mu\right)^T\left(x^{(n)} - \mu\right)\right)\right)$$

$$\frac{\partial \ln p(X|\ \pi, \mu, \Sigma)}{\partial \Sigma} = \sum_{n=1}^{N} \frac{\pi_k \mathcal{N}(x|\mu_k \Sigma)}{\sum_{j=1}^{k} \pi_j \mathcal{N}(x|\mu_j \Sigma)}\left(\left(-\frac{1}{2}\right)\left(|\Sigma|^{\frac{-3}{2}^T} + \Sigma^{-2}\left(x^{(n)} - \mu\right)^T\left(x^{(n)} - \mu\right)\right)\right) = 0$$

$$= \sum_{n=1}^{N} \gamma(z_k)\left(\left(-\frac{1}{2}\right)\left(|\Sigma|^{\frac{-3}{2}^T} + \Sigma^{-2}\left(x^{(n)} - \mu\right)^T\left(x^{(n)} - \mu\right)\right)\right)$$

therefore, rearranging for Σ:

$$\Sigma = \frac{1}{N}\sum_{k=1}^{k}\sum_{n=1}^{N} \gamma(z_k)\left(x^{(n)} - \mu_k\right)^T\left(x^{(n)} - \mu_k\right)$$

Intuitively, summing over k makes sense since μ_k is involved in determining the covariance for all the clusters. Finally, from equation 1, take the derivative w.r.t. π_k

$$\frac{\partial \ln p(X|\ \pi, \mu, \Sigma)}{\partial \Sigma} = 0 = \frac{\partial \ln p(X|\ \pi, \mu, \Sigma)}{\partial \mathcal{N}} \frac{\partial \mathcal{N}}{\partial \Sigma}$$

$$= \frac{\partial (ln\pi_k + ln\mathcal{N}(x|\mu_k \Sigma))}{\partial \pi_k} \ where \ \sum_{k=1}^{K} \pi_k = 1 \ and \ 0 \leq \pi_k \leq 1$$

$$\pi_k = \frac{N_k}{N}$$

Part 2 – Convolutional Neural Net

Let f = filter (height I, width J, C depth, K filters)
Let x = image (height H, width W, C depth, N Images)
Padding = number of zeros on image for height (P) and width (Q)

$$define\ filter\ transpose\ as: f_{i,j,k,c} \rightarrow {f_{i,j,k,c}}^T = f_{I-i+1,J-j+1,c,k}$$

$$f = \begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix} \rightarrow \begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix}^T = \begin{bmatrix} f_{33} & f_{32} & f_{31} \\ f_{23} & f_{22} & f_{21} \\ f_{13} & f_{12} & f_{11} \end{bmatrix}$$

$$define\ padding\ of\ image\ as: x^{(P,Q)} = pad(x,P,Q) \in \mathbb{R}^{N \times (H+P) \times (W+Q) \times C}$$

$$x = \begin{bmatrix} x_{11} & x_{12} & x_{13} & x_{14} & x_{15} \\ x_{21} & x_{22} & x_{23} & x_{24} & x_{25} \\ x_{31} & x_{32} & x_{33} & x_{34} & x_{35} \\ x_{41} & x_{42} & x_{43} & x_{44} & x_{45} \\ x_{51} & x_{52} & x_{53} & x_{54} & x_{55} \end{bmatrix} \rightarrow x^{(P,Q)} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & x_{11} & x_{12} & x_{13} & x_{14} & x_{15} & 0 \\ 0 & x_{21} & x_{22} & x_{23} & x_{24} & x_{25} & 0 \\ 0 & x_{31} & x_{32} & x_{33} & x_{34} & x_{35} & 0 \\ 0 & x_{41} & x_{42} & x_{43} & x_{44} & x_{45} & 0 \\ 0 & x_{51} & x_{52} & x_{53} & x_{54} & x_{55} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$convoluted\ layer = y = x^{(P,Q)} * f$$

where the convoluted layer is the same size as the original image

$$y_{11} = x^{(P,Q)} * f = \begin{bmatrix} 0\times f_{11} & 0\times f_{12} & 0\times f_{13} & 0 & 0 & 0 & 0 \\ 0\times f_{21} & x_{11}\times f_{22} & x_{12}\times f_{23} & x_{13} & x_{14} & x_{15} & 0 \\ 0\times f_{31} & x_{21}\times f_{32} & x_{22}\times f_{33} & x_{23} & x_{24} & x_{25} & 0 \\ 0 & x_{31} & x_{32} & x_{33} & x_{34} & x_{35} & 0 \\ 0 & x_{41} & x_{42} & x_{43} & x_{44} & x_{45} & 0 \\ 0 & x_{51} & x_{52} & x_{53} & x_{54} & x_{55} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} \boldsymbol{y_{11}} & y_{12} & y_{13} & y_{14} & y_{15} \\ y_{21} & y_{22} & y_{23} & y_{24} & y_{25} \\ y_{31} & y_{32} & y_{33} & y_{34} & y_{35} \\ y_{41} & y_{42} & y_{43} & y_{44} & y_{45} \\ y_{51} & y_{52} & y_{53} & y_{54} & y_{55} \end{bmatrix}$$

$$y_{11} = x_{11}\times f_{22} + x_{12}\times f_{23} + x_{21}\times f_{32} + x_{22}\times f_{33} + 0 + 0 + 0 + 0 + 0$$

$$\frac{\partial y_{11}}{\partial x_{11}} = f_{22}$$

calculate $y_{i,j}$ by moving filter 1 step across all columns and down all rows

$$\frac{\partial E}{\partial f_{c,i,j,k}} = x^{(I,J)}_{c,h,w,n} * \frac{\partial E}{\partial y_{h,w,n,k}} \quad , \quad x^{(I,J)}_{c,h,w,n} = \frac{\partial y}{\partial f} \quad (chain\ rule)$$

$$\frac{\partial E}{\partial x_{n,h,w,c}} = \frac{\partial E^{(I,J)}}{\partial y_{n,h,w,k}} * f^T, \quad where\ f^T\ is\ \frac{\partial y_{ij}}{\partial x_{ij}}\ (chain\ rule)$$

substituting $\frac{\partial y_{11}}{\partial x_{11}} = f_{22}$

$$\frac{\partial E}{\partial x_{11}} = \frac{\partial E}{\partial y_{11}} * \frac{\partial y_{11}}{\partial x_{11}} + \frac{\partial E}{\partial y_{12}} * \frac{\partial y_{12}}{\partial x_{11}} + \cdots = \frac{\partial E}{\partial y_{11}} * f_{22} + \frac{\partial E}{\partial y_{12}} * f_{21} + \cdots$$

Part 3 – Neural Networks

3.1 Basic Generalization

Table 3.1.1 - CE and Accuracy for Training and Validation Sets under default hyper-parameters for NN/CNN

	Hyper-Parameters				Results					
Default	**EPS**	**MOMENTUM**	**EPOCH**	**BATCH SIZE**	**Train Acc**	**Valid Acc**	**Test Acc**	**Train CE**	**Valid CE**	**Test CE**
NN	0.01	0.0	1000	100	0.887	0.735	0.745	0.316	0.947	0.849
CNN	0.1	0.0	30	100	0.781	0.739	0.725	0.589	0.844	0.810

For NN (figure 3.1.1), the training set performs better than the validation set in both accuracy and cross entropy, which is expected. The validation accuracy flattens out at 70% at around 400 epochs and this corresponds to the split in CE between Train and Validation, indicating overfitting and the weights growing too large due to many epochs. At this point, the weights are operating in the non-linear region as they grow too large. For CNN (figure 3.1.2), the image appears less “colourful” because it is only run with a default 30epochs (~1/3NN). You can see an improvement in accuracy with validation closely matching training.

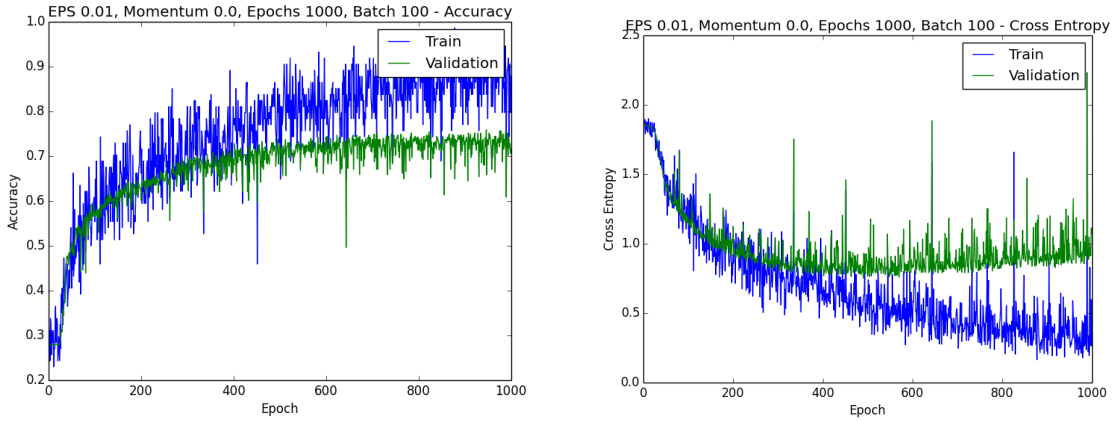


Figure 3.1.1 – CE and Accuracy for Training and Validation Sets under default hyper-parameters for NN

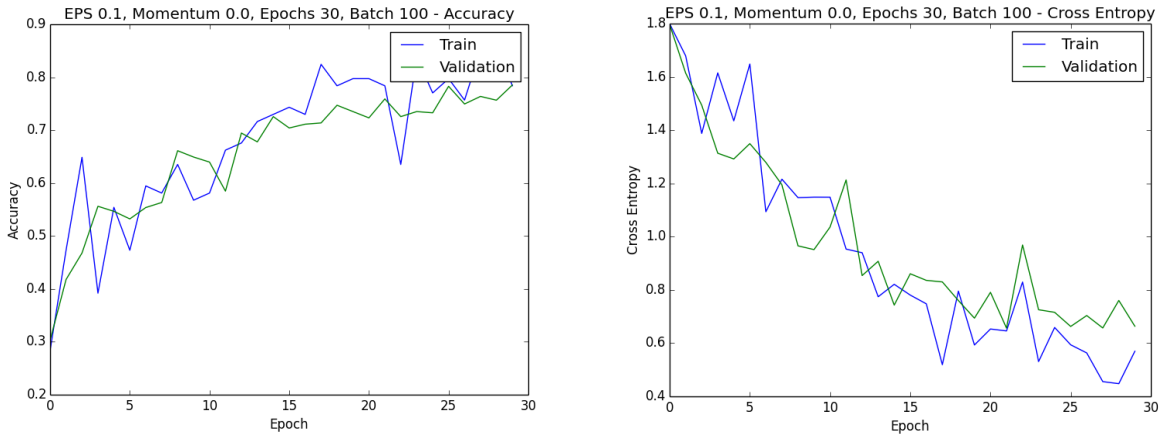


Figure 3.1.2 – CE and Accuracy for Training and Validation Sets under default hyper-parameters for CNN

3.2 Optimization

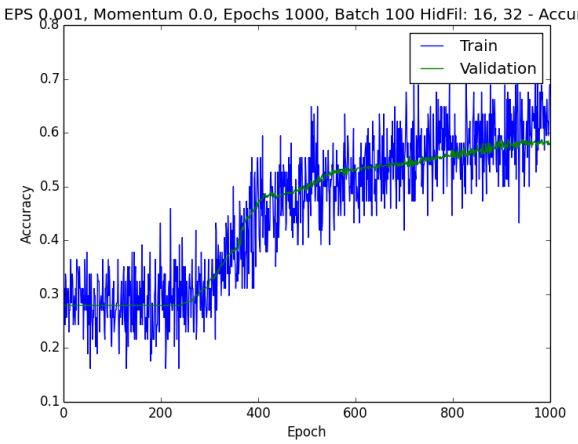
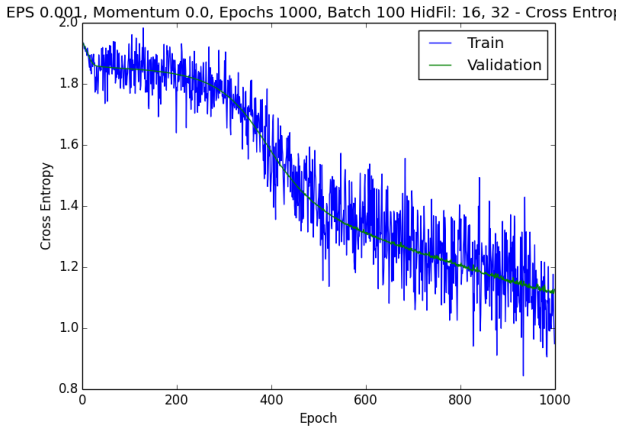
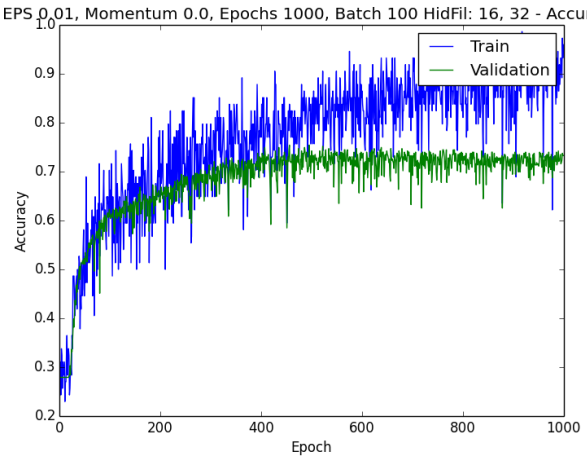
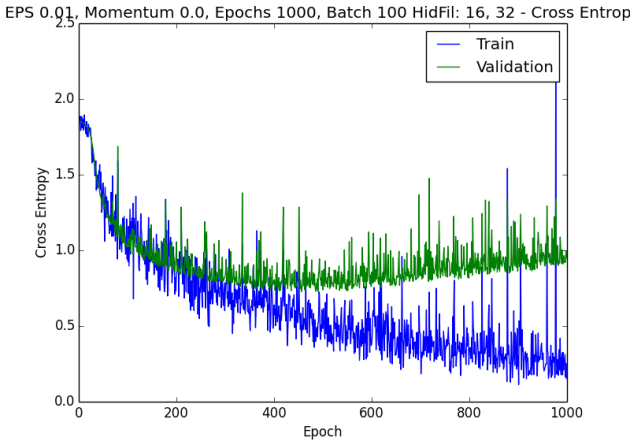
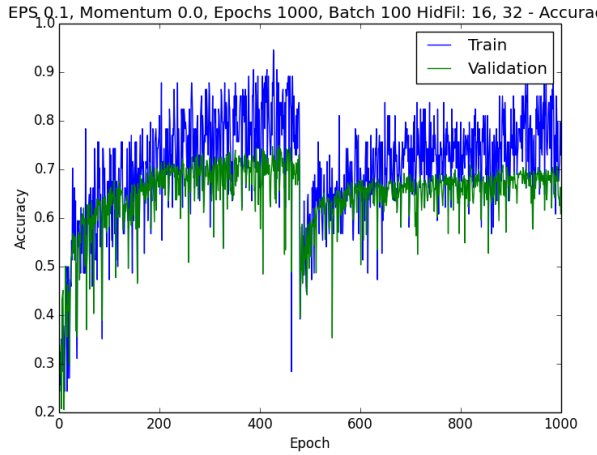
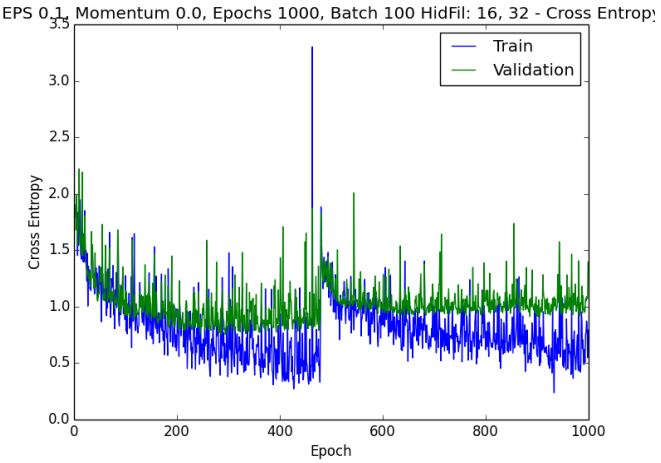
Table 3.2.1 - CE and Accuracy for Training and Validation Sets under default hyper-parameters for NN

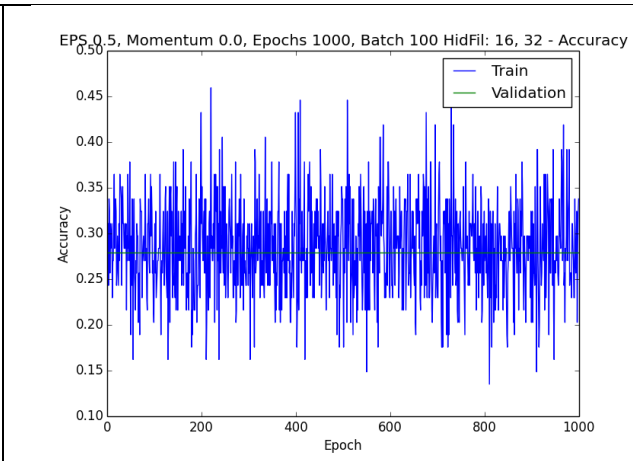
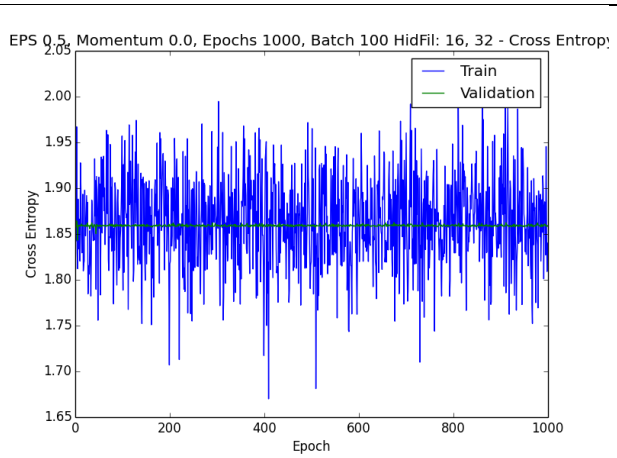
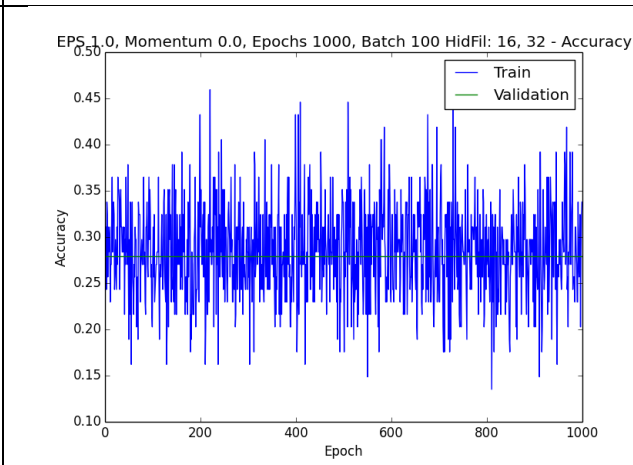
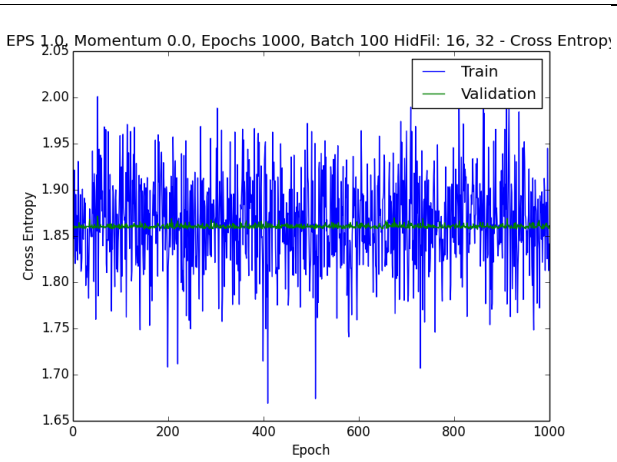
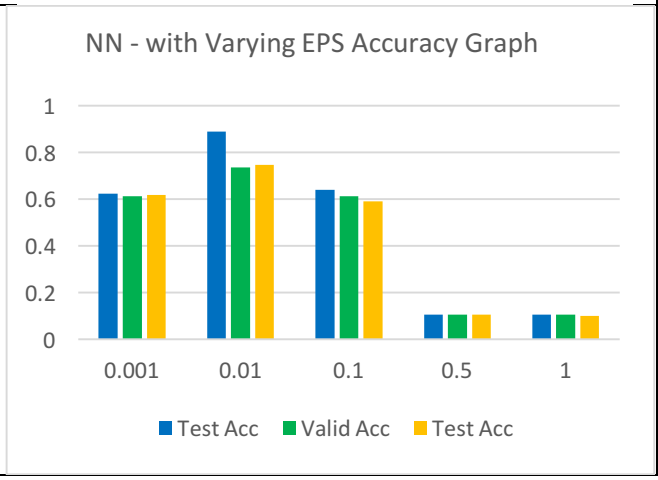
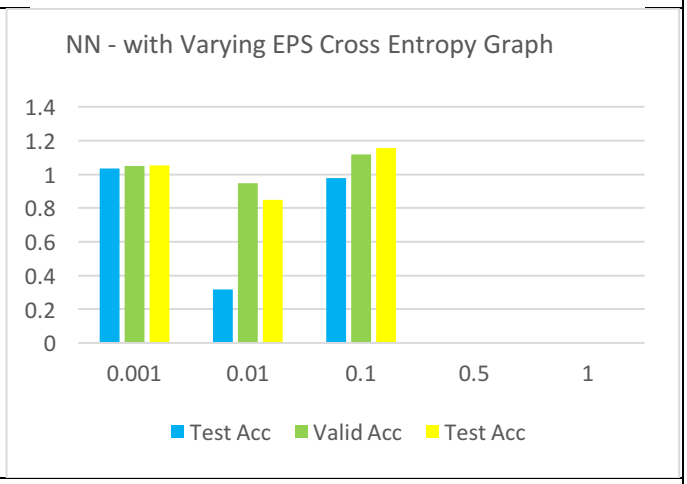
	Hyper-Parameters				Results					
Focus Param	**EPS**	**MOMENTUM**	**EPOCH**	**BATCH SIZE**	**Train Acc**	**Valid Acc**	**Test Acc**	**Train CE**	**Valid CE**	**Test CE**
EPS		0.0	1000	100	0.624	0.613	0.618	1.035	1.049	1.055
	0.01	**0.0**	**1000**	**100**	0.887	0.735	0.745	0.316	0.947	0.849
	0.1	0.0	1000	100	0.638	0.611	0.590	0.976	1.116	1.156
	0.5	0.0	1000	100	0.105	0.105	0.105	NaN	NaN	NaN
	1.0	0.0	1000	100	0.105	0.105	0.099	NaN	NaN	NaN
MOM.	**0.01**	**0.0**	**1000**	**100**	0.887	0.735	0.745	0.316	0.947	0.849
	0.01	0.45	1000	100	0.963	0.718	0.751	0.122	1.146	0.952
	0.01	0.9	1000	100	1.000	0.716	0.735	0.001	2.695	2.012
BATCH	0.01	0.45	1000	1000	0.723	0.670	0.660	0.761	0.864	0.887
	0.01	0.45	1000	500	0.806	0.728	0.740	0.545	0.831	0.731
	0.01	0.45	1000	100	0.930	0.721	0.743	0.197	1.064	0.810
	0.01	0.45	1000	10	1.000	0.742	0.771	0.000	3.317	2.551
	0.01	0.45	1000	1	0.286	0.279	0.317	1.864	1.863	1.842

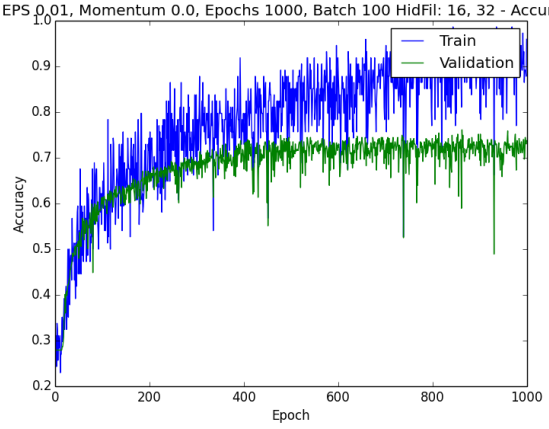
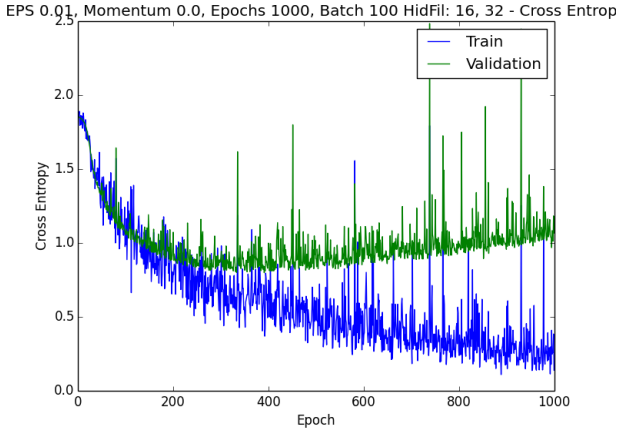
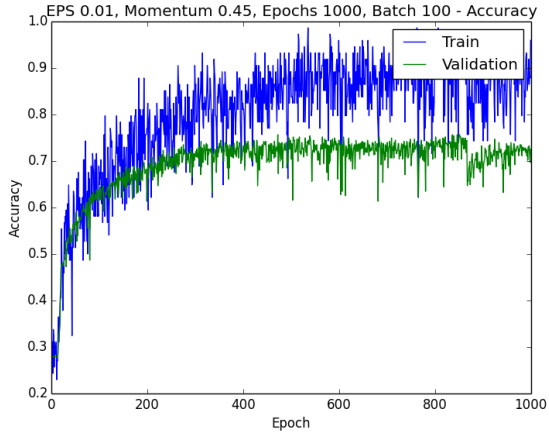
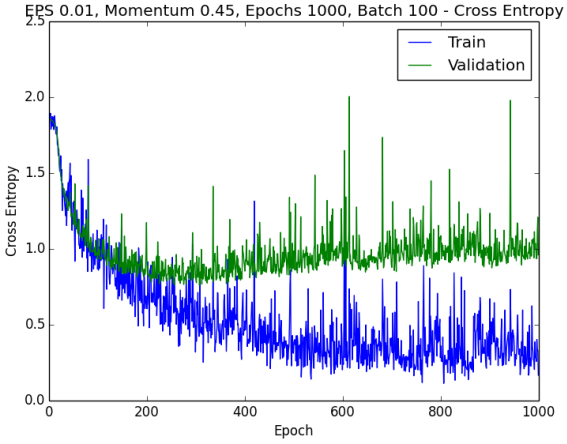
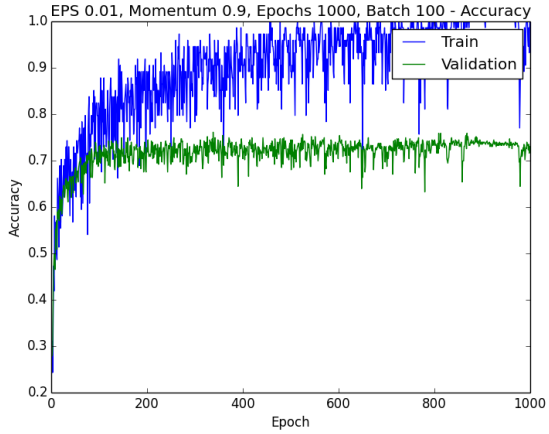
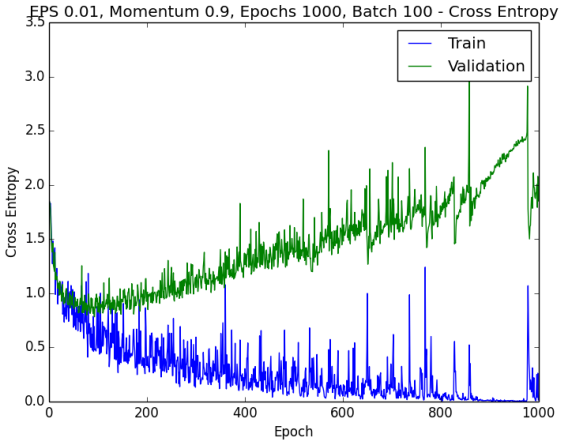
Table 3.2.2 - CE and Accuracy for Training and Validation Sets under default hyper-parameters for CNN

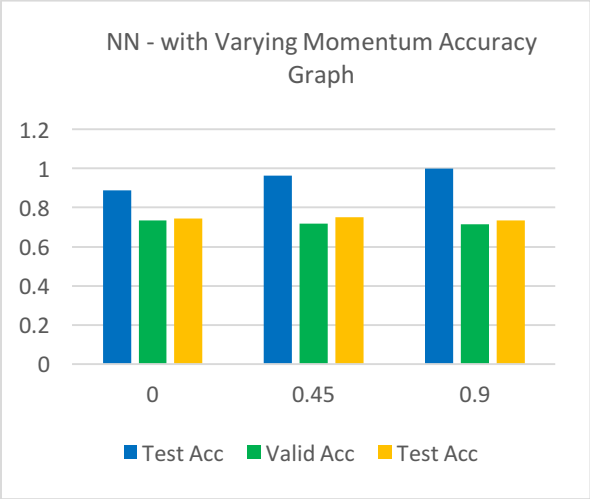
	Hyper-Parameters				Results					
Focus Param	**EPS**	**MOMENTUM**	**EPOCH**	**BATCH SIZE**	**Train Acc**	**Valid Acc**	**Test Acc**	**Train CE**	**Valid CE**	**Test CE**
EPS	0.001	0.0	30	100	0.321	0.281	0.272	1.834	1.844	1.810
	0.01	**0.0**	**30**	**100**	0.285	0.279	0.317	1.842	1.841	1.828
	0.1	0.0	30	100	0.811	0.818	0.792	0.584	0.682	0.751
	0.5	0.0	30	100	0.304	0.281	0.267	1.872	1.858	1.877
	1.0	0.0	30	100	0.307	0.283	0.261	1.851	1.864	1.911
MOM.	**0.01**	**0.0**	**30**	**100**	0.285	0.279	0.317	1.842	1.841	1.828
	0.01	0.45	30	100	0.712	0.688	0.651	1.012	1.081	1.156
	0.01	0.9	30	100	0.298	0.281	0.270	1.855	1.864	1.891
BATCH	0.01	0	30	1000	0.344	0.342	0.337	1.851	1.782	1.895
	0.01	0	30	500	0.632	0.638	0.622	1.214	1.189	1.226
	0.01	**0**	**30**	**100**	0.285	0.279	0.317	1.842	1.841	1.828
	0.01	0	30	10	0.200	0.200	0.400	0.981	1.292	1.561
	0.01	0	30	1	0.300	0.280	0.200	2.115	1.910	2.151

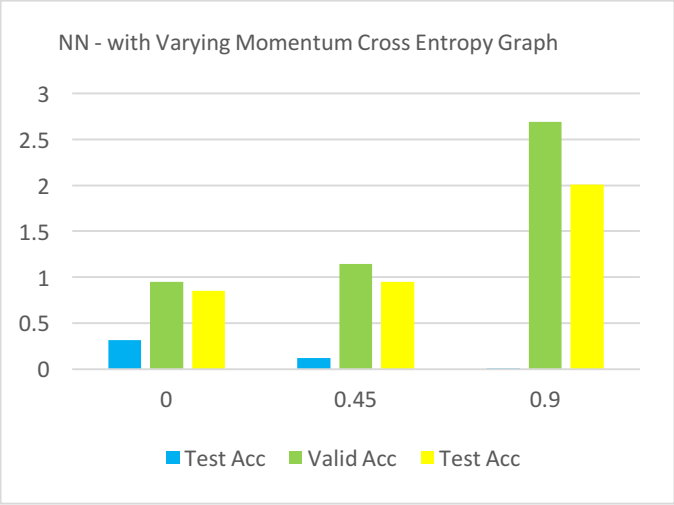
See next pages for plots. There will be a brief comment about the data at the end of each EPS, MOM and BATCH for both NN and CNN. As well bar graphs are included to show trends within the hyperparameter.

	NN	
EPS	**Accuracy**	**Cross Entropy**
0.001		
0.01		
0.1		

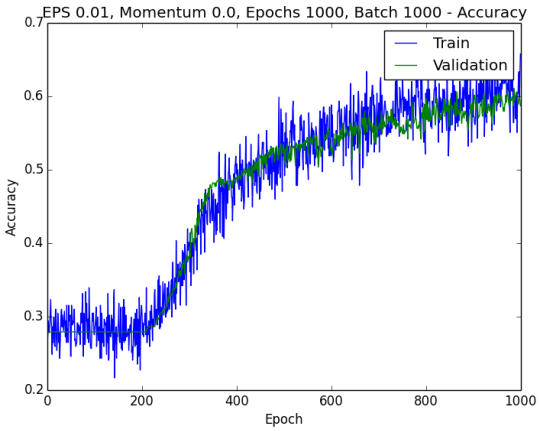
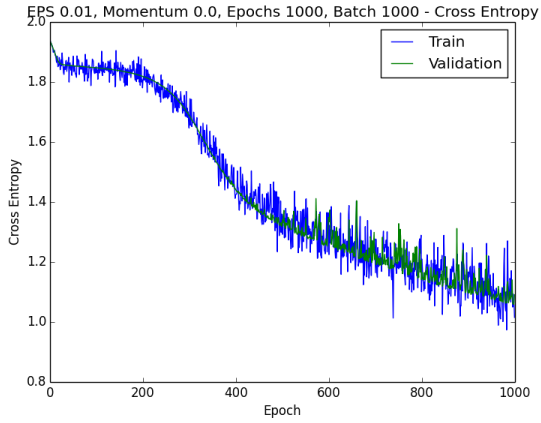
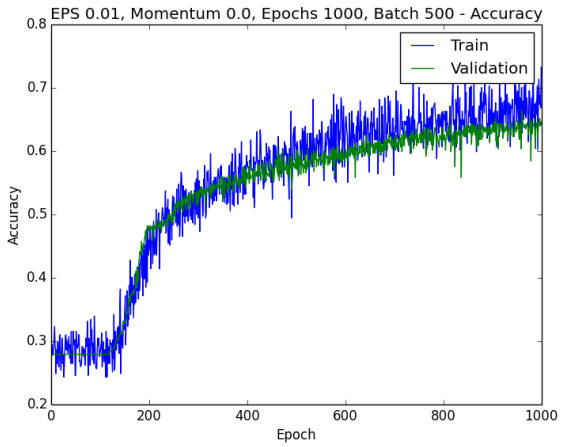
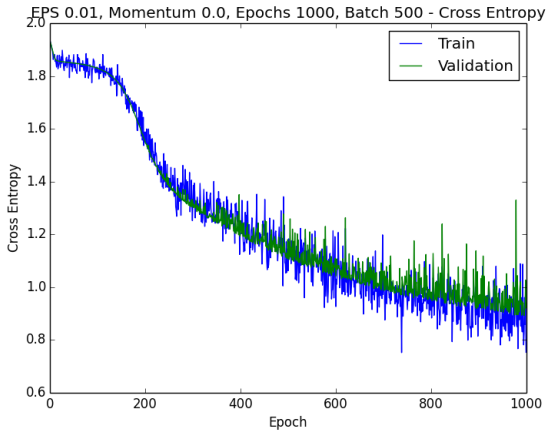
0.5		
1.0		
		
	As learning rate changes, you see an increase in the accuracy of the data. Training is always better since it learns on it, but the best learning rate is at 0.01. With low EPS the convergence is slow. With high EPS its non-generalizable since learning rate jumps around.	

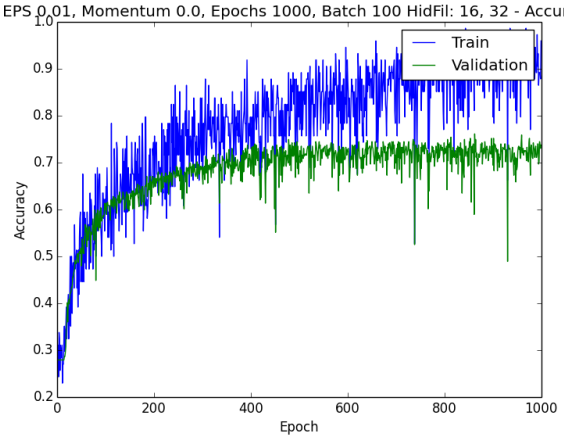
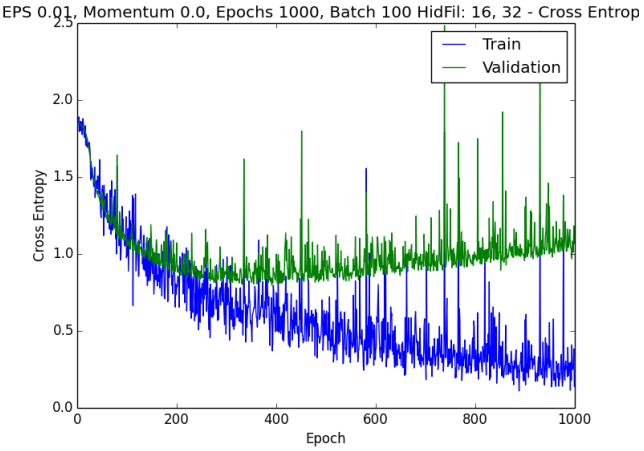
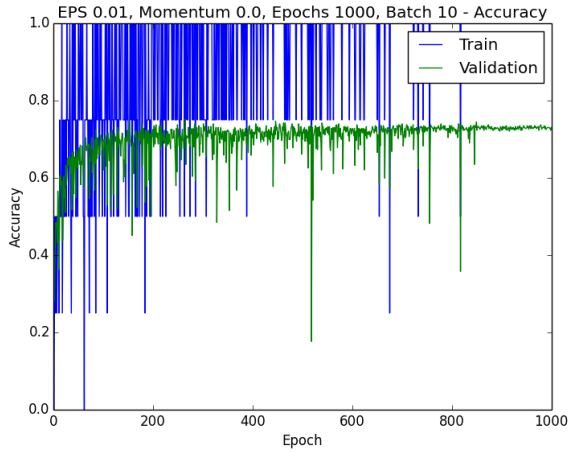
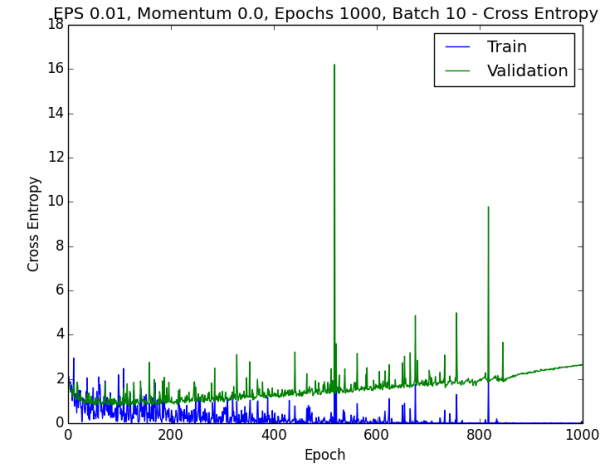
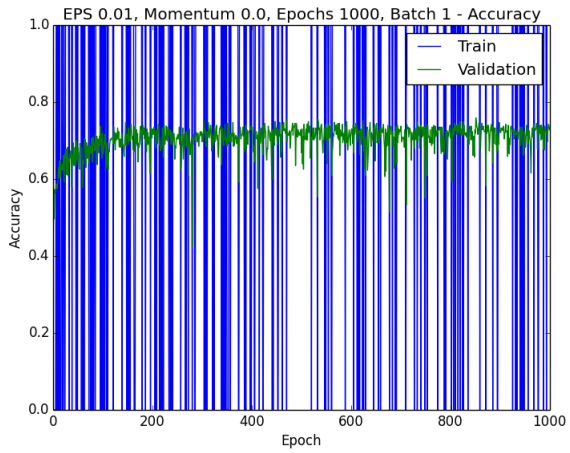
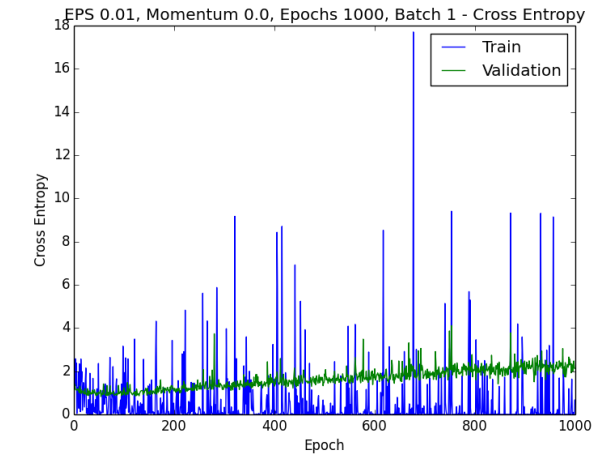
MOM	Accuracy	Cross Entropy
0.0		
0.45		
0.9		

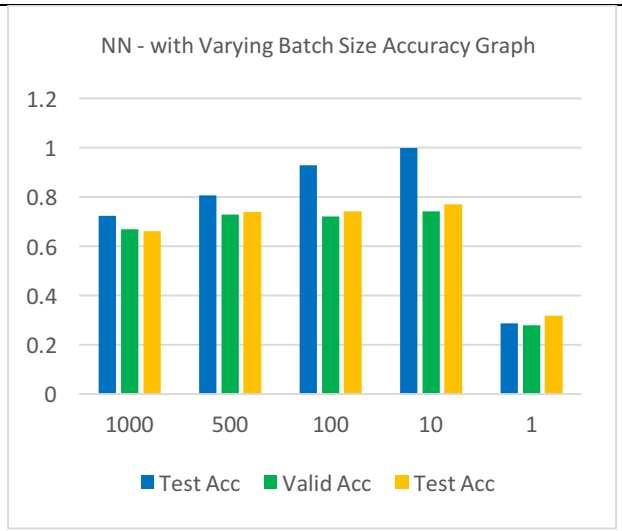


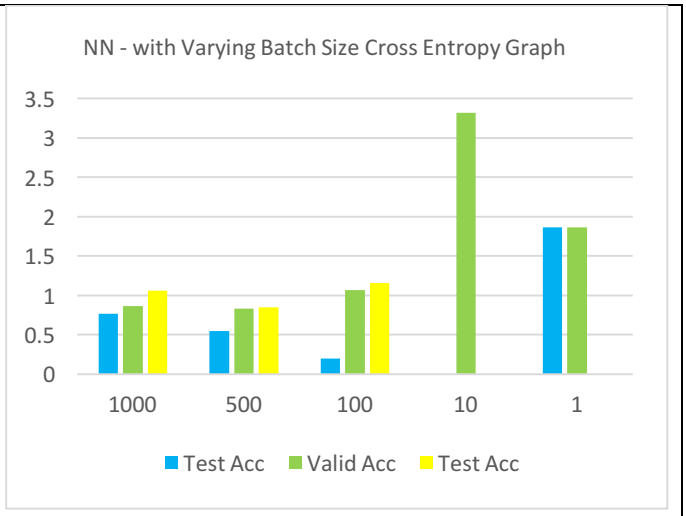


As momentum increases, the rate of convergence increases. Validation accuracy stays roughly the same, but the training accuracy goes close to 100% because as it learns and retains prior knowledge, so it models the training data very well. This can also be seen with cross entropy close to 0. The trade-off for modelling the training data well is that the model does not generalize well and so you see overfitting with the validation data. Momentum is learning based on prior values and not as affected by outliers, and. For validation accuracy is typically at 70% but cross entropy is going higher as momentum increases.

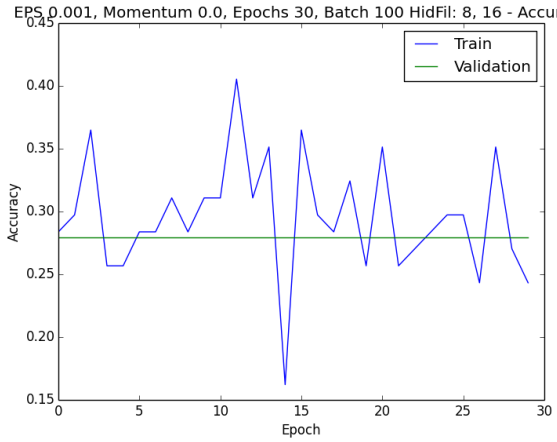
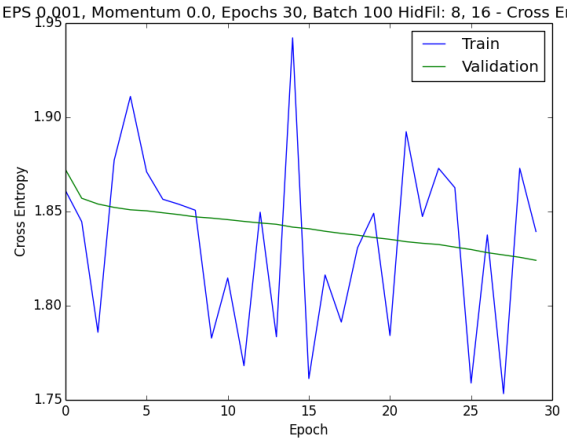
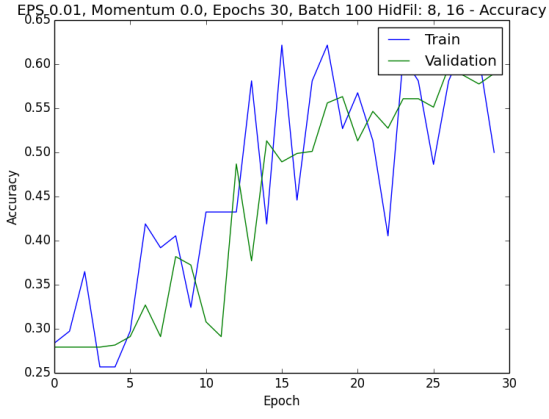
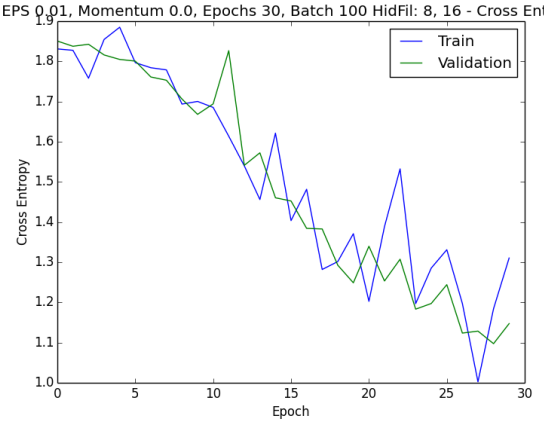
BAT	Accuracy	Cross Entropy
1000		
500		

100	 EPS 0.01, Momentum 0.0, Epochs 1000, Batch 100 HidFil: 16, 32 - Accuracy Train Validation Accuracy Epoch	 EPS 0.01, Momentum 0.0, Epochs 1000, Batch 100 HidFil: 16, 32 - Cross Entrop Train Validation Cross Entropy Epoch
10	 EPS 0.01, Momentum 0.0, Epochs 1000, Batch 10 - Accuracy Train Validation Accuracy Epoch	 EPS 0.01, Momentum 0.0, Epochs 1000, Batch 10 - Cross Entropy Train Validation Cross Entropy Epoch
1	 EPS 0.01, Momentum 0.0, Epochs 1000, Batch 1 - Accuracy Train Validation Accuracy Epoch	 EPS 0.01, Momentum 0.0, Epochs 1000, Batch 1 - Cross Entropy Train Validation Cross Entropy Epoch

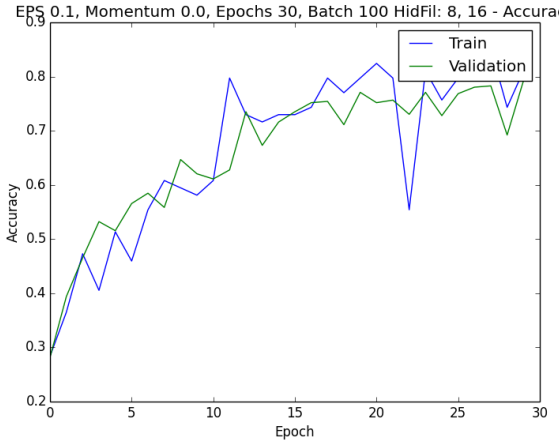


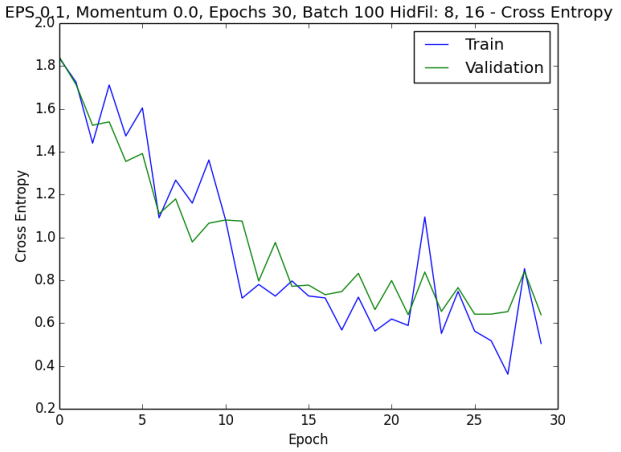


Smaller batch sizes are similar to learning fewer things very well, as opposed to generalizing over larging amount os training data. You can see that the smaller batch sizees have a higher rate of convergence, but are less generalizeable as the validation data does not follow the trainining data well. This is expected because with small batch sizes, the model overfits individual images very well in training as if it examines each image very closely.

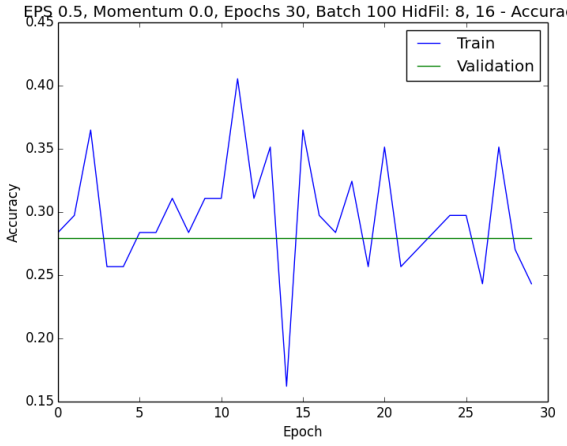
	CNN	
EPS	**Accuracy**	**Cross Entropy**
0.001		
0.01		

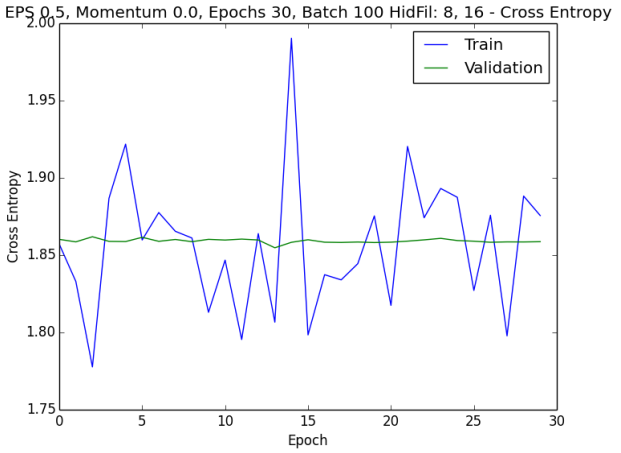
0.1


EPS 0.1, Momentum 0.0, Epochs 30, Batch 100 HidFil: 8, 16 - Accuracy
Train
Validation
Accuracy
Epoch

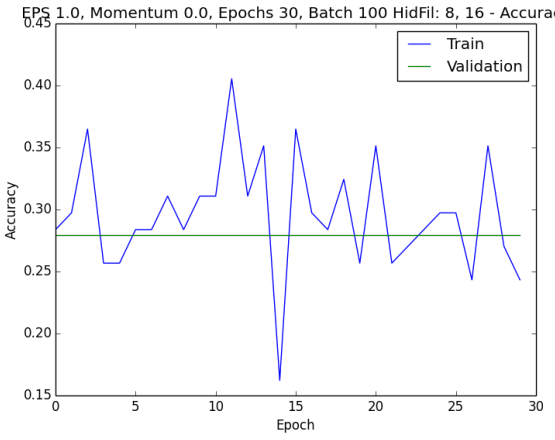

EPS 0.1, Momentum 0.0, Epochs 30, Batch 100 HidFil: 8, 16 - Cross Entropy
Train
Validation
Cross Entropy
Epoch

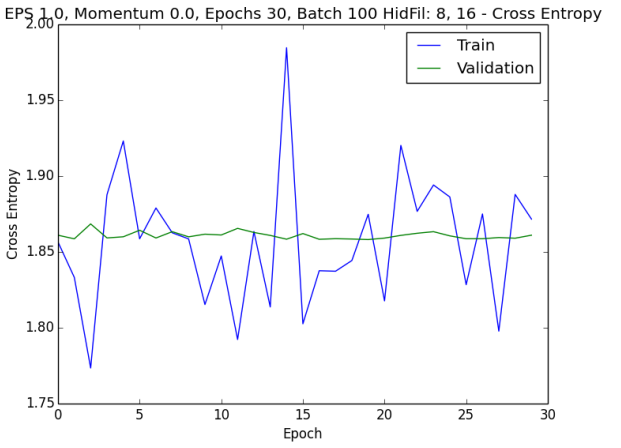
0.5

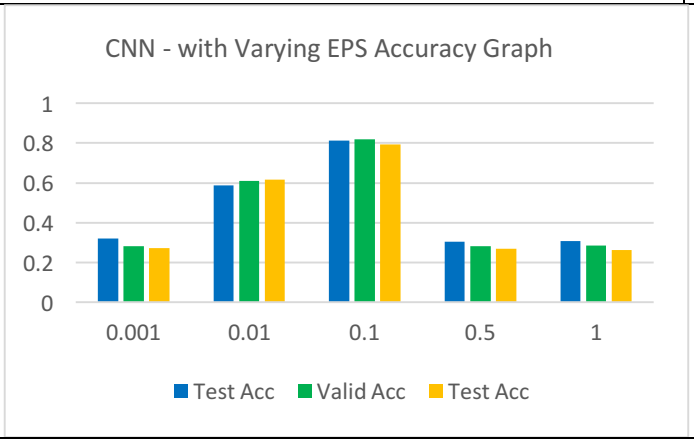

EPS 0.5, Momentum 0.0, Epochs 30, Batch 100 HidFil: 8, 16 - Accuracy
Train
Validation
Accuracy
Epoch

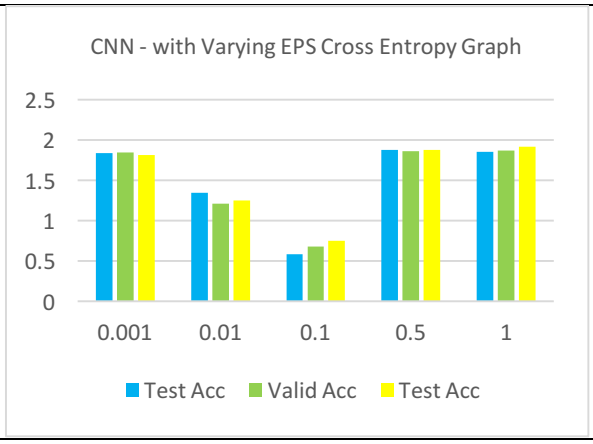

EPS 0.5, Momentum 0.0, Epochs 30, Batch 100 HidFil: 8, 16 - Cross Entropy
Train
Validation
Cross Entropy
Epoch

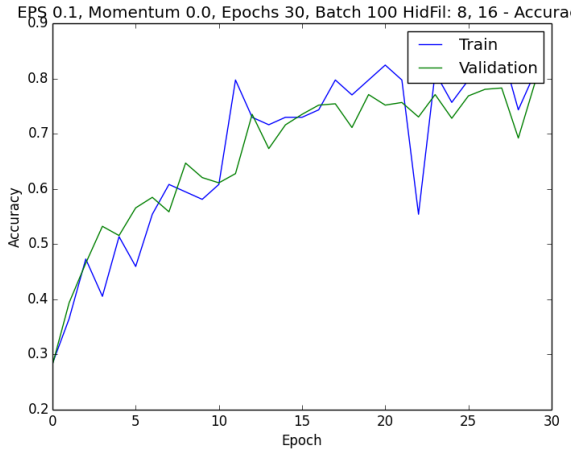
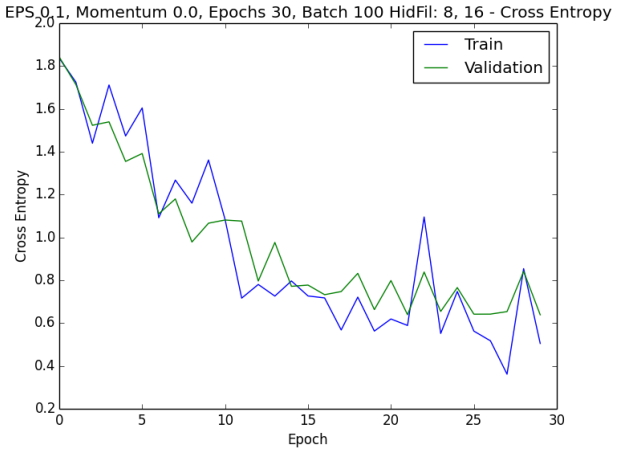
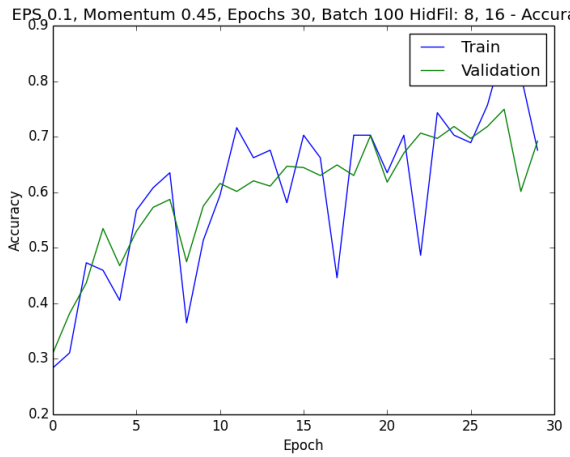
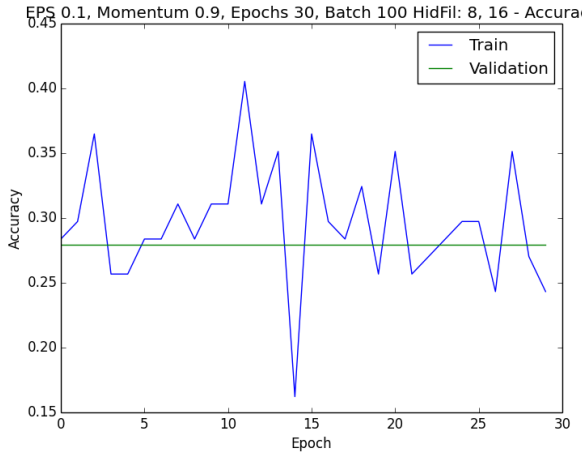
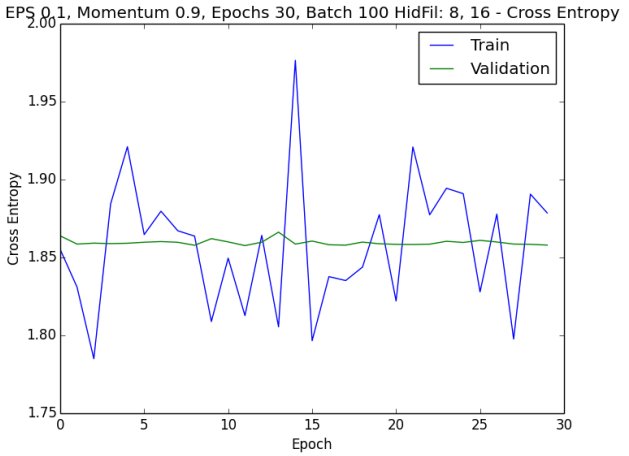
1.0

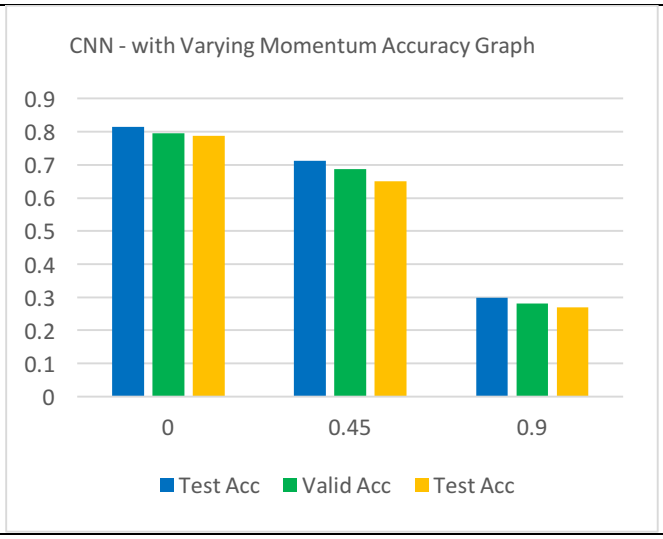

EPS 1.0, Momentum 0.0, Epochs 30, Batch 100 HidFil: 8, 16 - Accuracy
Train
Validation
Accuracy
Epoch

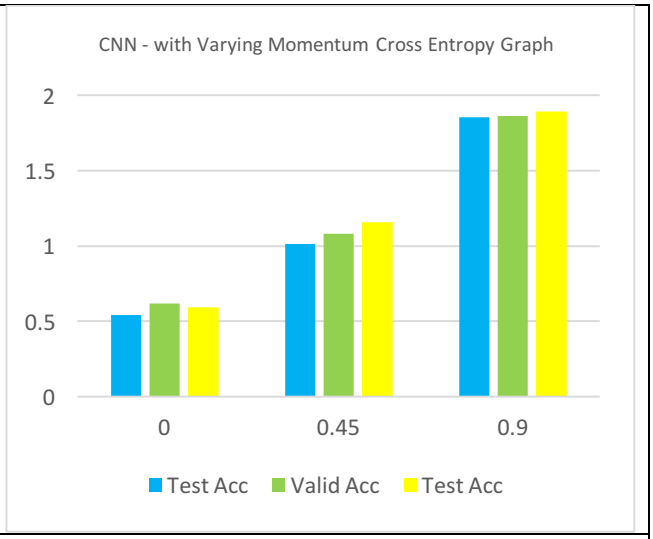

EPS 1.0, Momentum 0.0, Epochs 30, Batch 100 HidFil: 8, 16 - Cross Entropy
Train
Validation
Cross Entropy
Epoch


CNN - with Varying EPS Accuracy Graph
0.001
0.01
0.1
0.5
1
Test Acc
Valid Acc
Test Acc

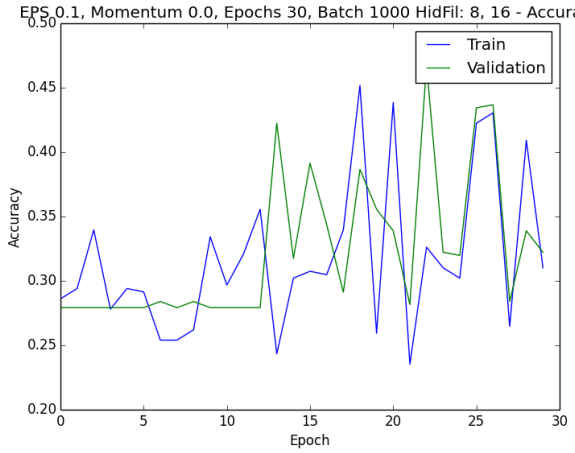
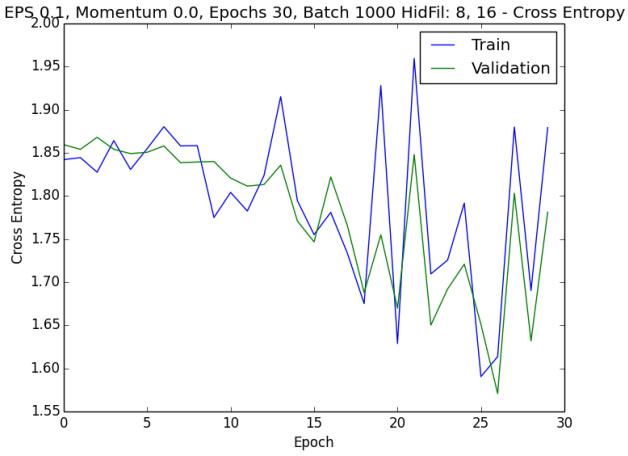
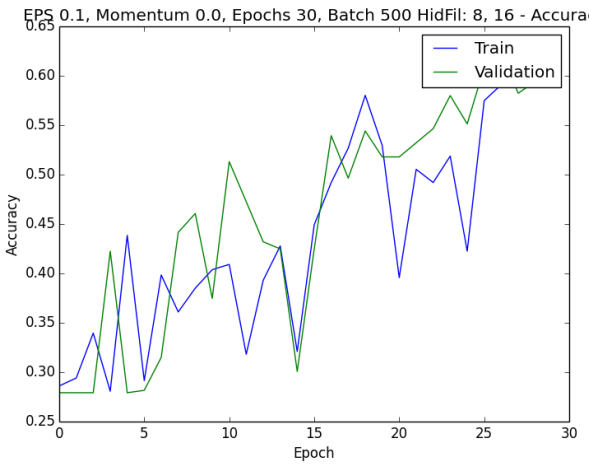
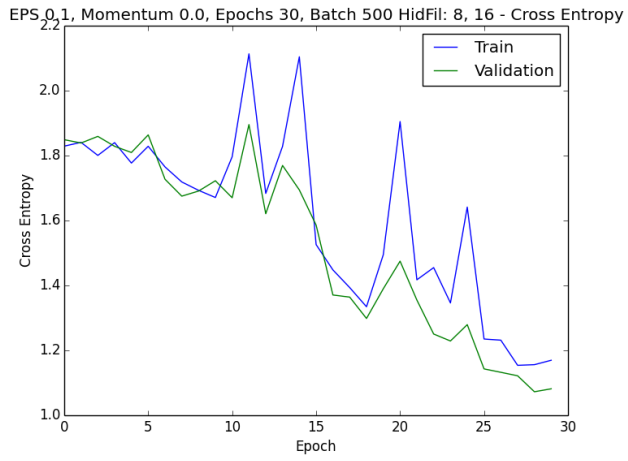

CNN - with Varying EPS Cross Entropy Graph
0.001
0.01
0.1
0.5
1
Test Acc
Valid Acc
Test Acc

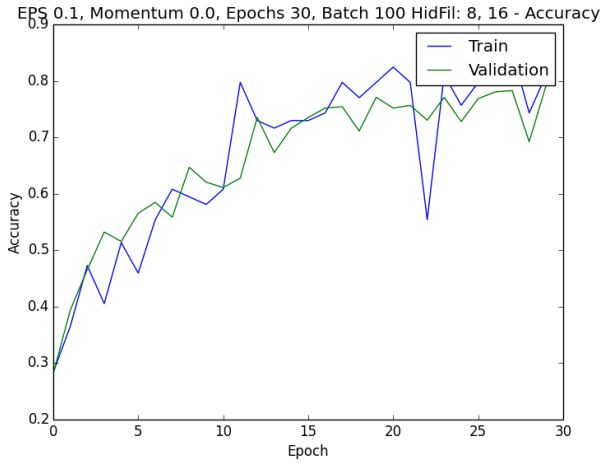
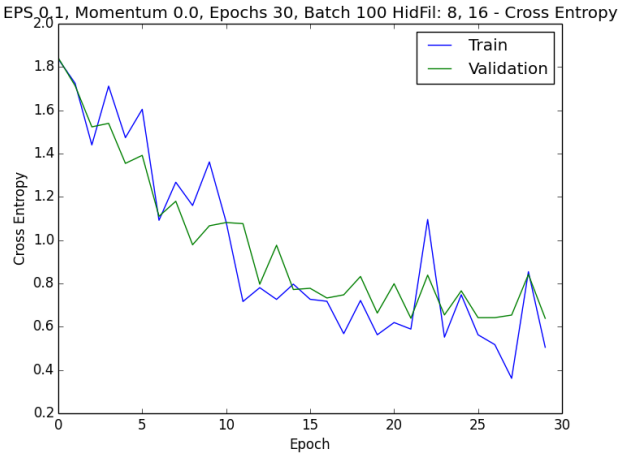
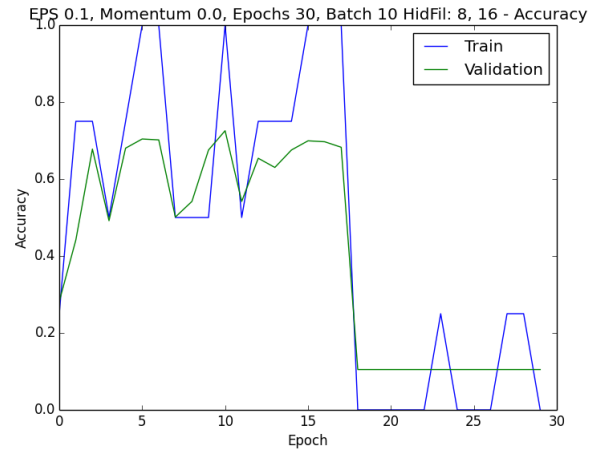
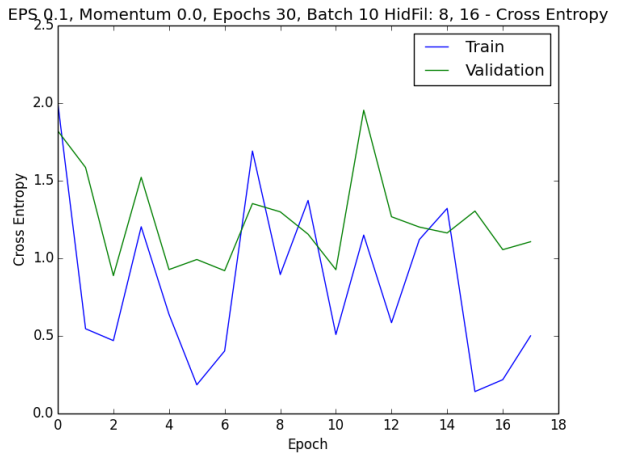
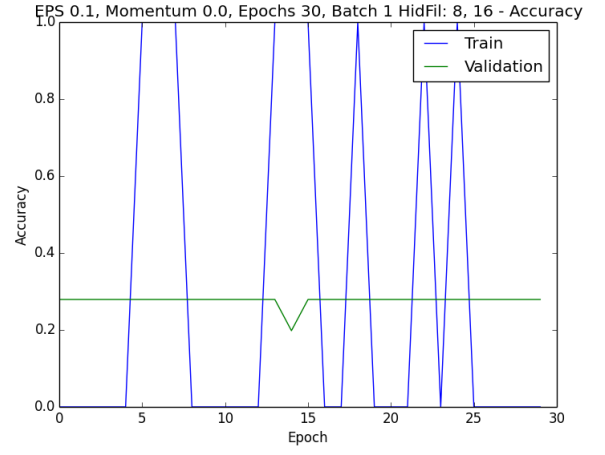
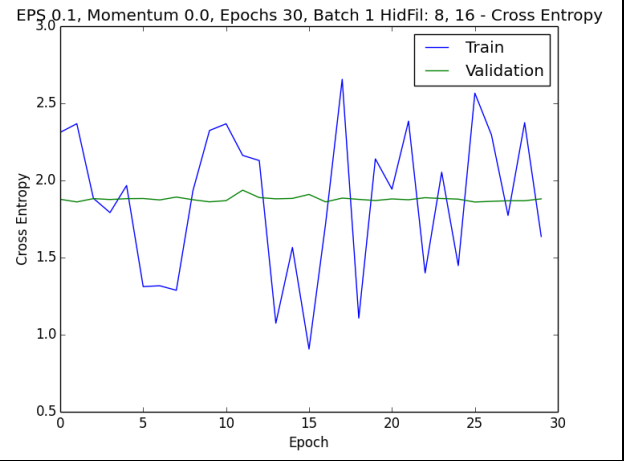
	Only 0.01 and 0.1 learning rates cause convergence. 0.1 is better because validation more closely follows training. Possibly, if 0.01 had more time to learn it would converge, but 30 is not enough, espcially since there are a lot of parameters in CNN. For the EPS values that don't have convengence, 0.001 does not have enough time to learn. 0.5 and 1 is learning too much that it would never generalize, even with enough epochs.	
MOM	**Accuracy**	**Cross Entropy**
0.0		
0.45		
0.9		

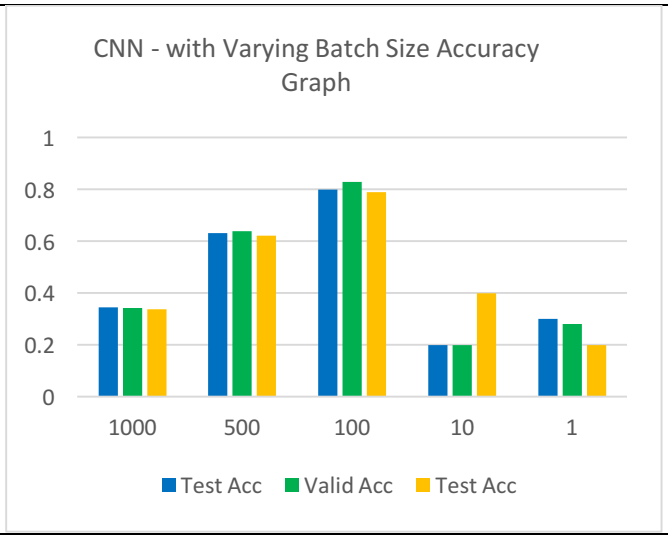
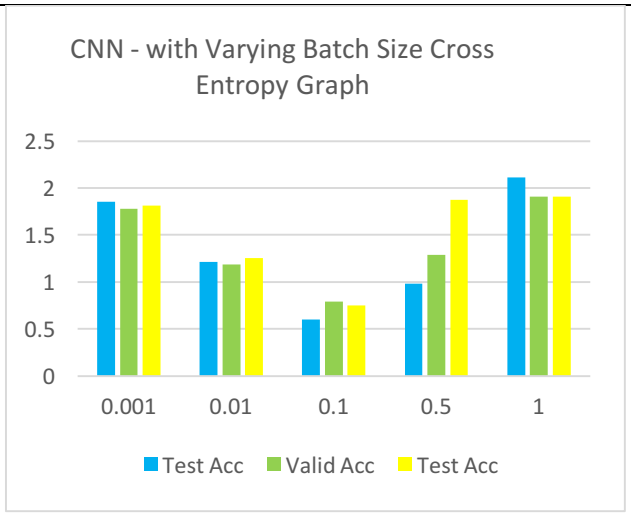




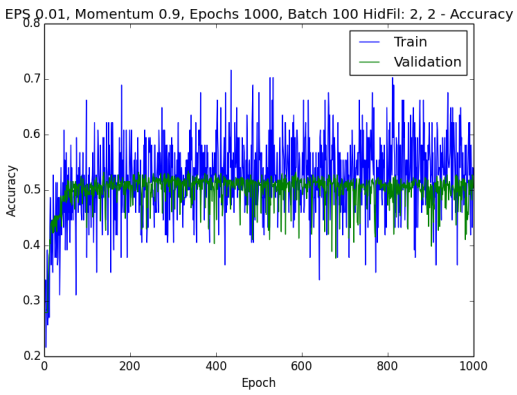
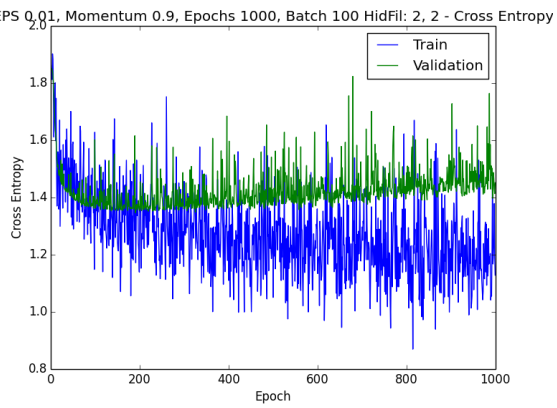
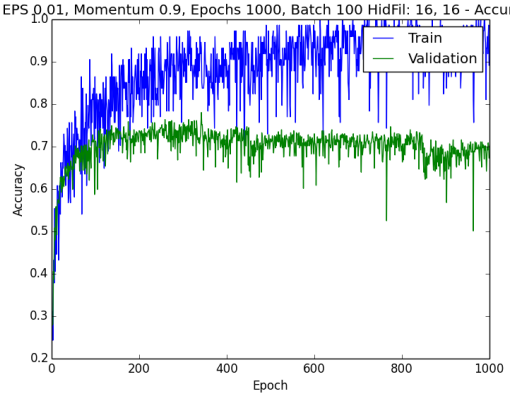
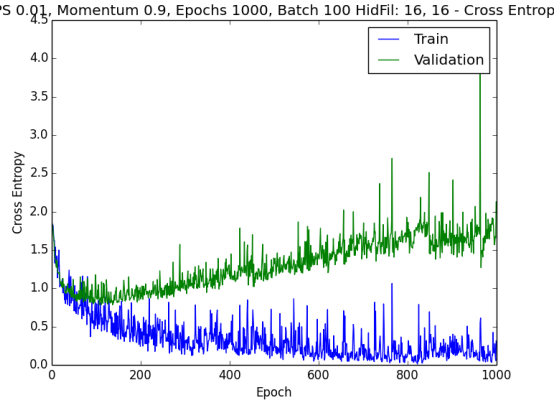
The momentum between 0.0 and 0.45 converges just a little faster as you increase momentum. It is a little hard to see but you can intuitively think this because momentum allows for past training to aid in the ability to learn faster. For 0.9 the validation accuracy is a flat line because it cannot generalize, especially with a high learning rate (0.1) and high momentum (0.9).

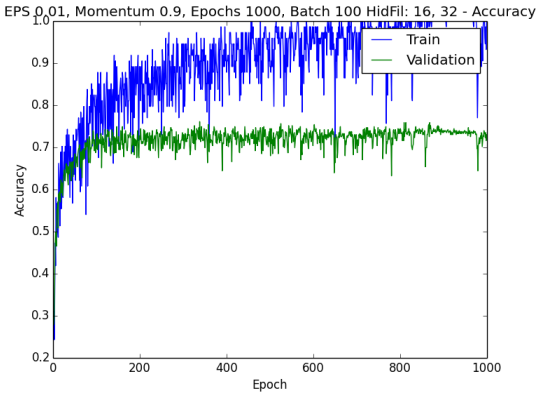
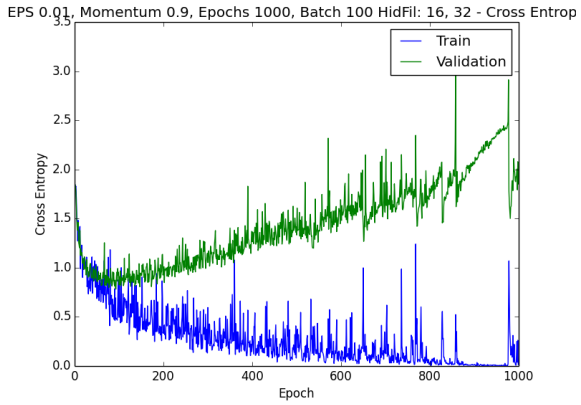
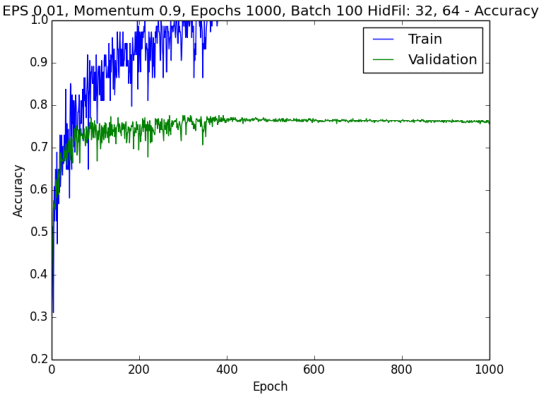
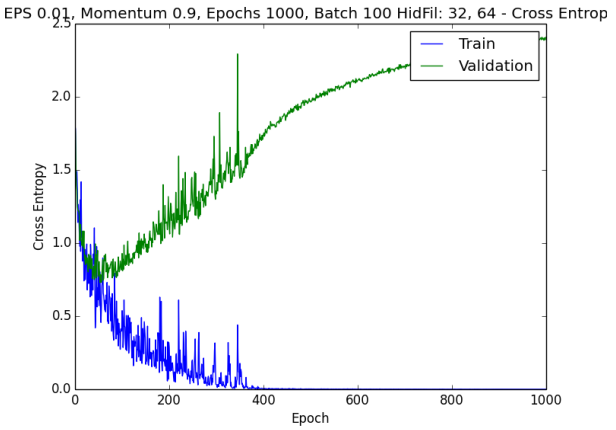
BAT	Accuracy	Cross Entropy
1000		
500		

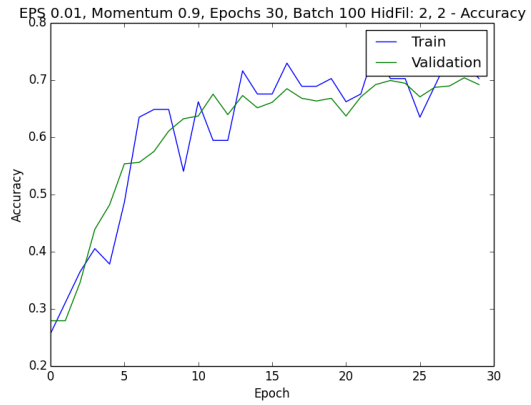
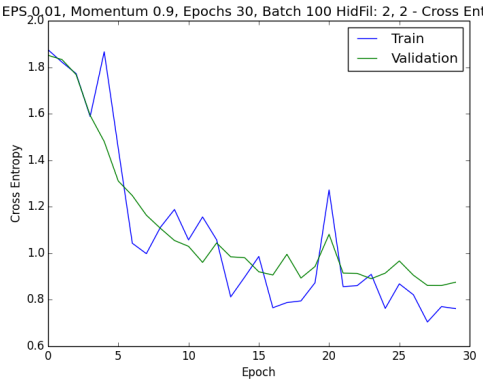
100	 EPS 0.1, Momentum 0.0, Epochs 30, Batch 100 HidFil: 8, 16 - Accuracy Train Validation Accuracy Epoch	 EPS 0.1, Momentum 0.0, Epochs 30, Batch 100 HidFil: 8, 16 - Cross Entropy Train Validation Cross Entropy Epoch
10	 EPS 0.1, Momentum 0.0, Epochs 30, Batch 10 HidFil: 8, 16 - Accuracy Train Validation Accuracy Epoch	 EPS 0.1, Momentum 0.0, Epochs 30, Batch 10 HidFil: 8, 16 - Cross Entropy Train Validation Cross Entropy Epoch
1	 EPS 0.1, Momentum 0.0, Epochs 30, Batch 1 HidFil: 8, 16 - Accuracy Train Validation Accuracy Epoch	 EPS 0.1, Momentum 0.0, Epochs 30, Batch 1 HidFil: 8, 16 - Cross Entropy Train Validation Cross Entropy Epoch

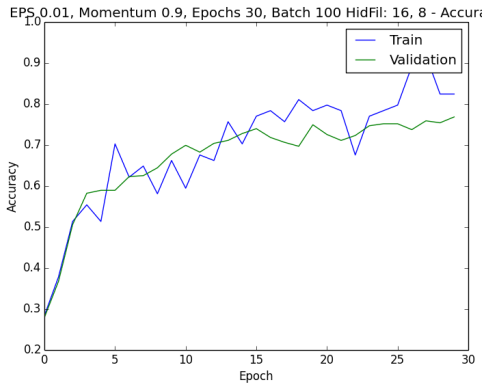
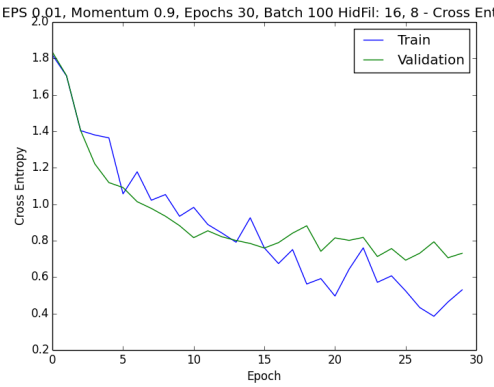
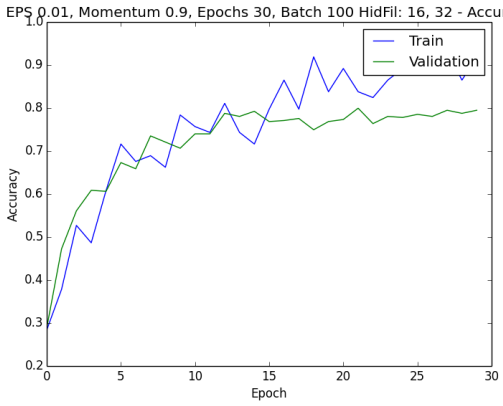
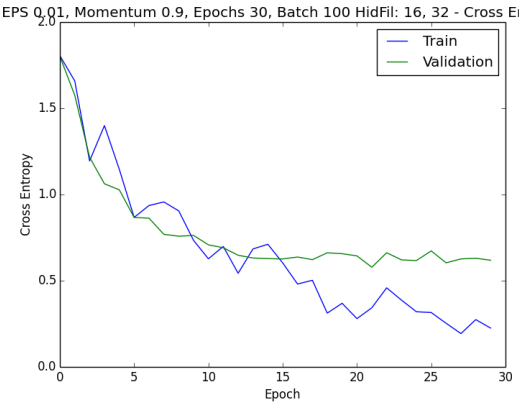
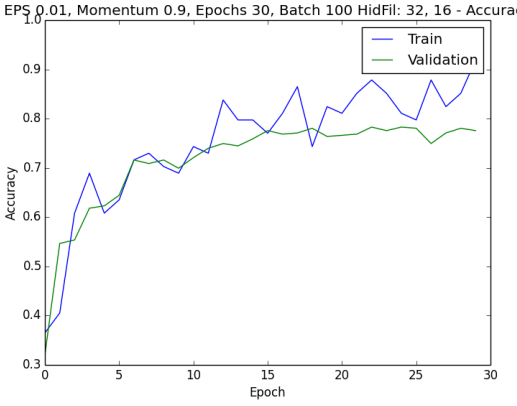
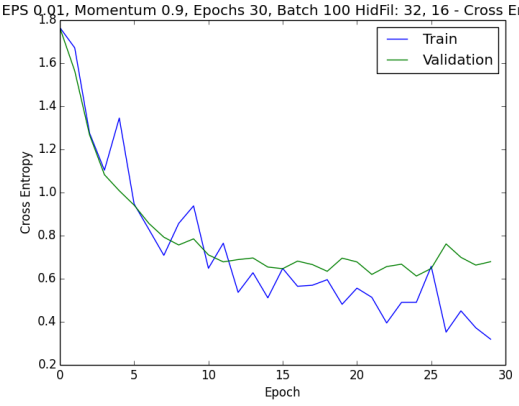
		
	CNNs have poor performance for high and low batch sizes. Feeding too much data with a high batch size causes a diffused amount of data that it cannot generalize on it. Intuitively, with more epochs, it would converge with more epochs. For low batch sizes, you are training to too much detail such that your training data does not generalize well and is very confident on its guesses.	

3.3 Model Architecture

Num Hidden	NN	
	Accuracy	**Cross Entropy**
2,2 = 4626 params		
16,16 = 37232 params		

16,32 = 37600 params		
32,64 = 76224		
	You can see that as the with more hidden units, validation accuracy goes up and there is faster convergence. Looking at the models with hidden units of 16,16 and 16,32, you will see that an increase in hidden units in the second term leads to negligibly higher number of parameters, and the validation cross entropy for both end at around 2.0. The model still learns the training data well, but it becomes less generalizable and more uncertain of its predictions as seen in their cross-entropy graphs.	

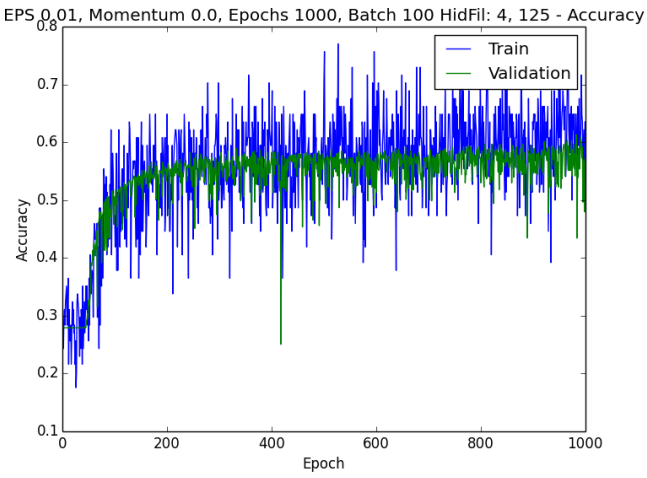
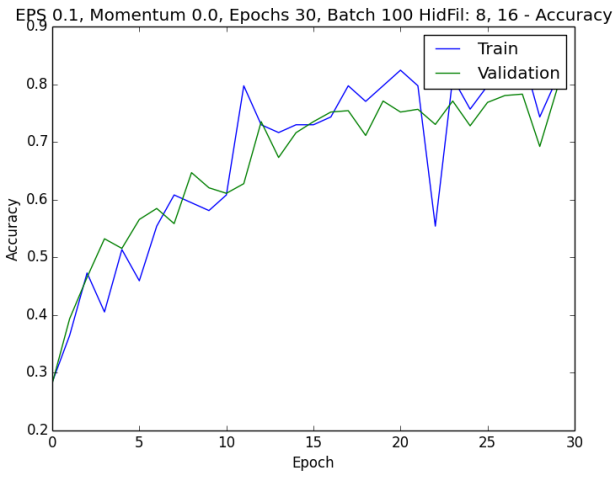
Num Filters	**CNN**	
	Accuracy	**Cross Entropy**
2,2 = 1057 params		

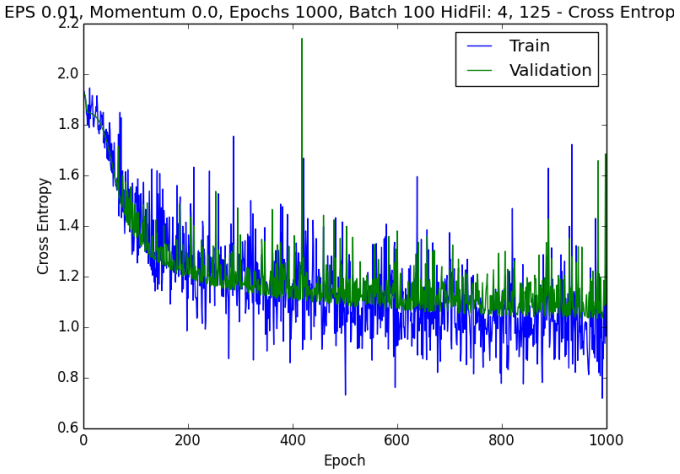
16,8 = 7215 params		
16,32 = 27591 params		
32, 16 = 20823 params		
	You can see that with more filters in the second layer, the more parameters there are. As well the validation and training cross entropy are closely related, making this model more generalizable. The learning rate for CNN was changed from 0.1 to 0.01. Previously, a high learning rate of 0.1 had graphs that showed no learning where the validation accuracy was a flat line. Here the learning rate is slow enough that the validation gets better at learning and converges just about 70% accuracy for all combination of number of filters.	

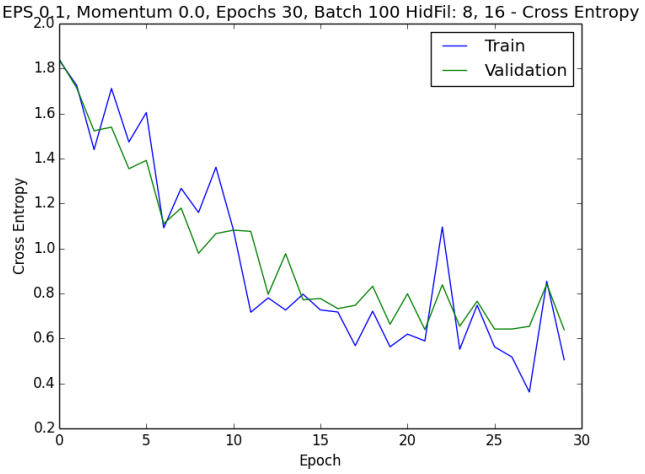
3.4 Compare CNNs and fully connected networks

NN Parameters	CNN Parameters
Input = 2304	Input Image: Height = 5, Width = 5, Depth 1
Hidden Units 1 = 16	Filters Layer 1 = 8
Hidden Units 2 = 32	Filters Layer 2 = 16
Output = 7	Output Image = 8x8, Output = 7
Bias at H1 = 16	Bias at Filter 1 = 8
Bias at H2 = 32	Bias at Filter 2 = 16
Bias at Output = 7	Bias at Output = 7, Output Image 8x8
Total NN Parameters (w/biases) = (Input x HiddenUnits1) + (HiddenUnits1 x HiddenUnits2) + (HiddenUnits2 x Output) + (Bias at H1) + (Bias at H2) + (Bias at H3) = (2304 x 16)+(16 x 32)+(32x7)+(16)+(32)+(7) = **37655**	**Total CNN Parameters (w/biases)** = [(Input Image x FiltersL1) + Bias at Filter 1] + [(Input Image X FiltersL2) + Bias at Filter 2] + (Output Image X FiltersL2 x Output) + Bias at Output = (5x5x1x8+8) + (5x5*8*16+16) + (8x8x16x7+7) = **10599**

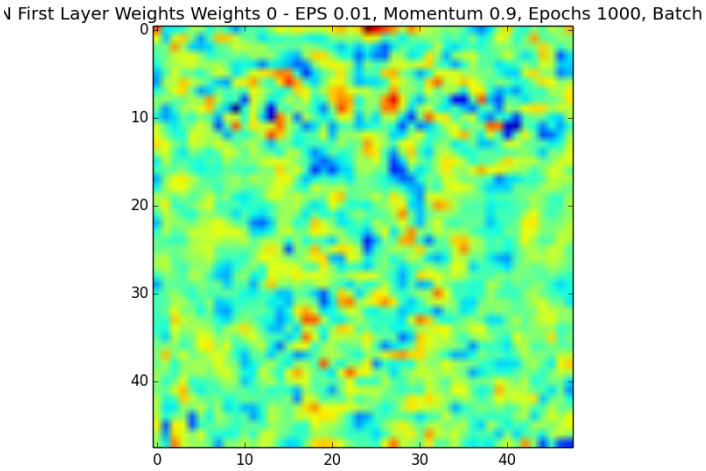
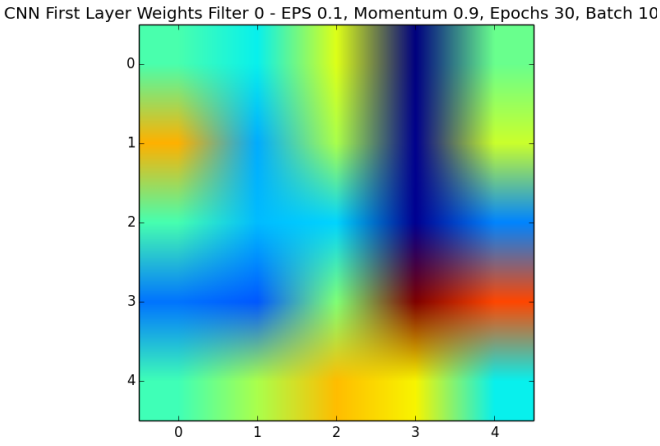
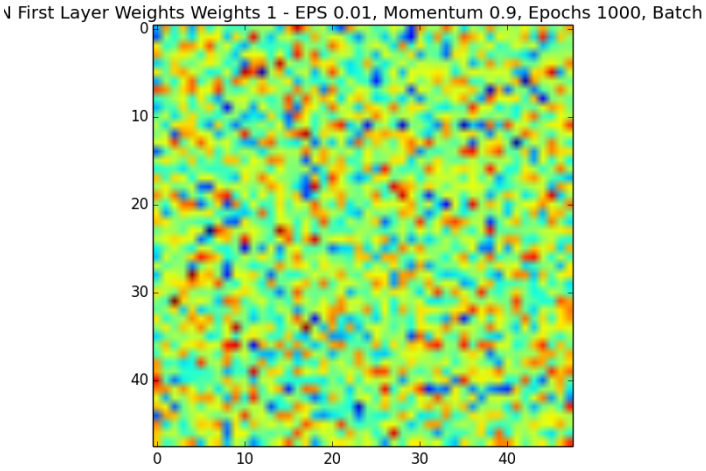
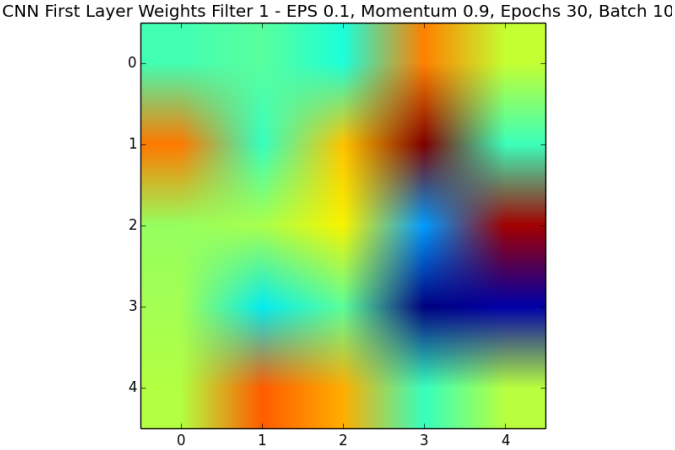
In NN, modify Hidden Units 1 and 2 to match Total Parameters of CNN. This will allow us to compare NN and CNN based on a roughly equal number of parameters between the two.
In NN, try using Hidden Units 1,2 = (4, 125)
In CNN, keep Filters 1,2 = (8,16)
= Total NN Parameters = 10602
~ Total CNN Parameters = 10599

MODIFIED NN (4, 125) **10603 Parameters**	**DEFAULT CNN (8,16)** **10599 Parameters**
	

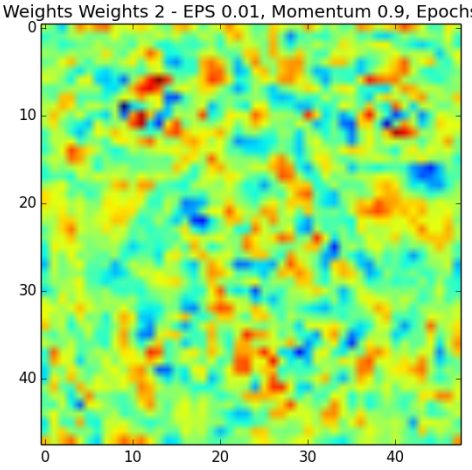




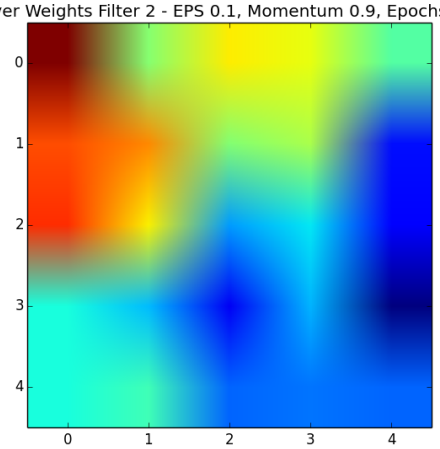
In terms of performance, CNN is more accurate than NN, given roughly the same number of parameters. CNN has better generalization because the validation cross entropy is lower valued and closely follows that of the training data. This shows that a new validation point is generalized or predicted well, similar to how good its training data was. The first layer weights and filters produced show what areas that are being detected with a high value as shown in red.

First Layer Weights	First Layer Filters
	
	

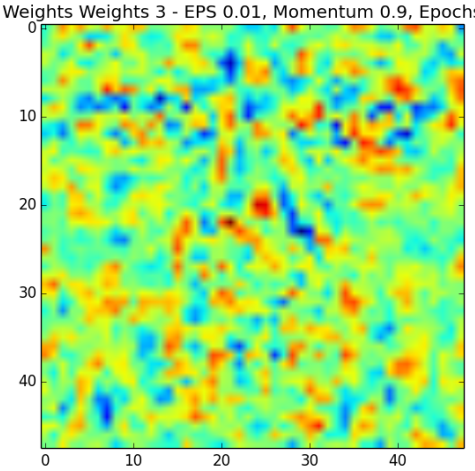
N First Layer Weights Weights 2 - EPS 0.01, Momentum 0.9, Epochs 1000, Batch



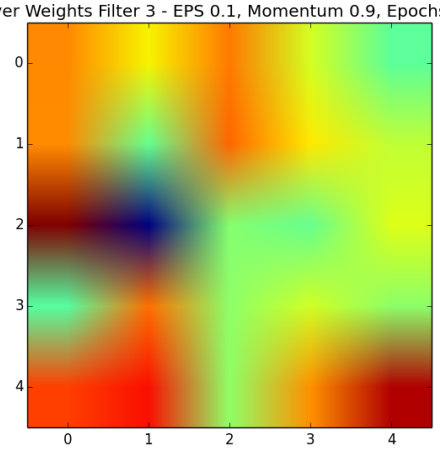
CNN First Layer Weights Filter 2 - EPS 0.1, Momentum 0.9, Epochs 30, Batch 10



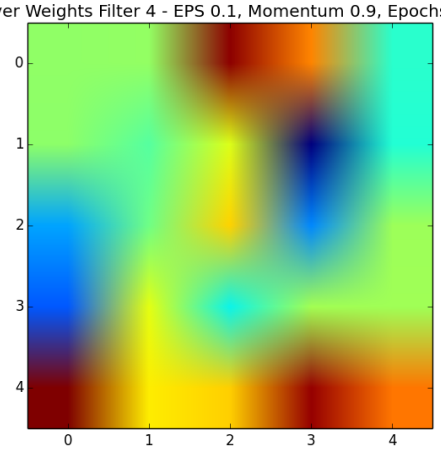
N First Layer Weights Weights 3 - EPS 0.01, Momentum 0.9, Epochs 1000, Batch



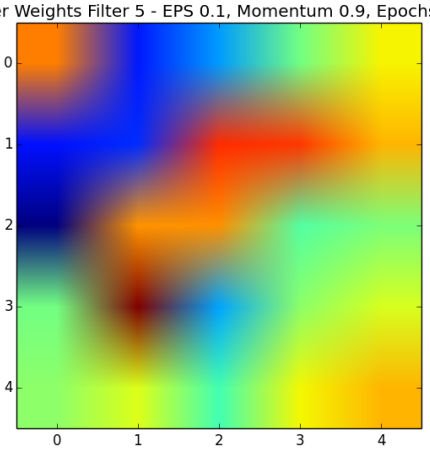
CNN First Layer Weights Filter 3 - EPS 0.1, Momentum 0.9, Epochs 30, Batch 10



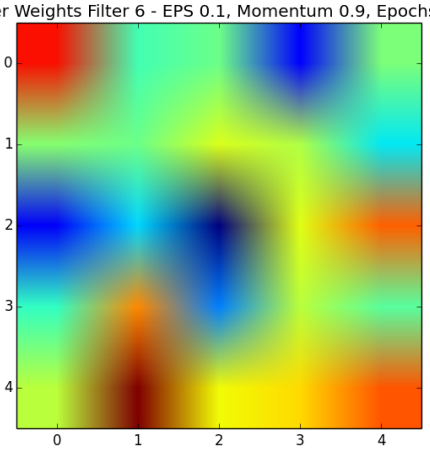
CNN First Layer Weights Filter 4 - EPS 0.1, Momentum 0.9, Epochs 30, Batch 10



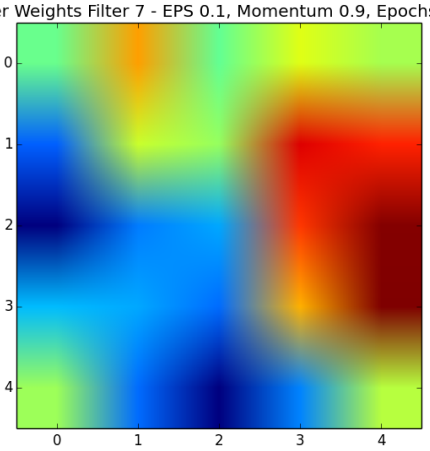
CNN First Layer Weights Filter 5 - EPS 0.1, Momentum 0.9, Epochs 30, Batch 10



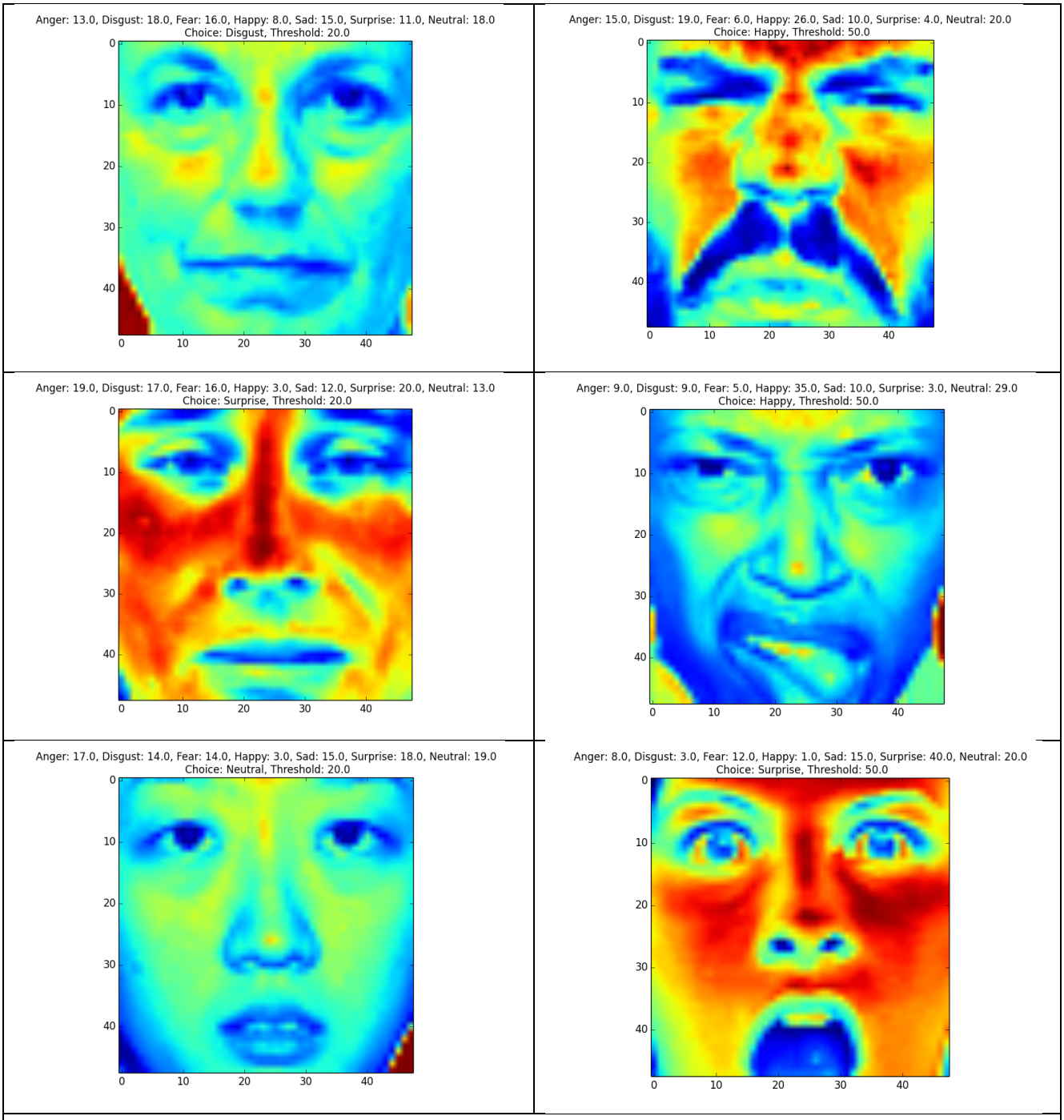
CNN First Layer Weights Filter 6 - EPS 0.1, Momentum 0.9, Epochs 30, Batch 10



CNN First Layer Weights Filter 7 - EPS 0.1, Momentum 0.9, Epochs 30, Batch 10



3.5 Network Uncertainty

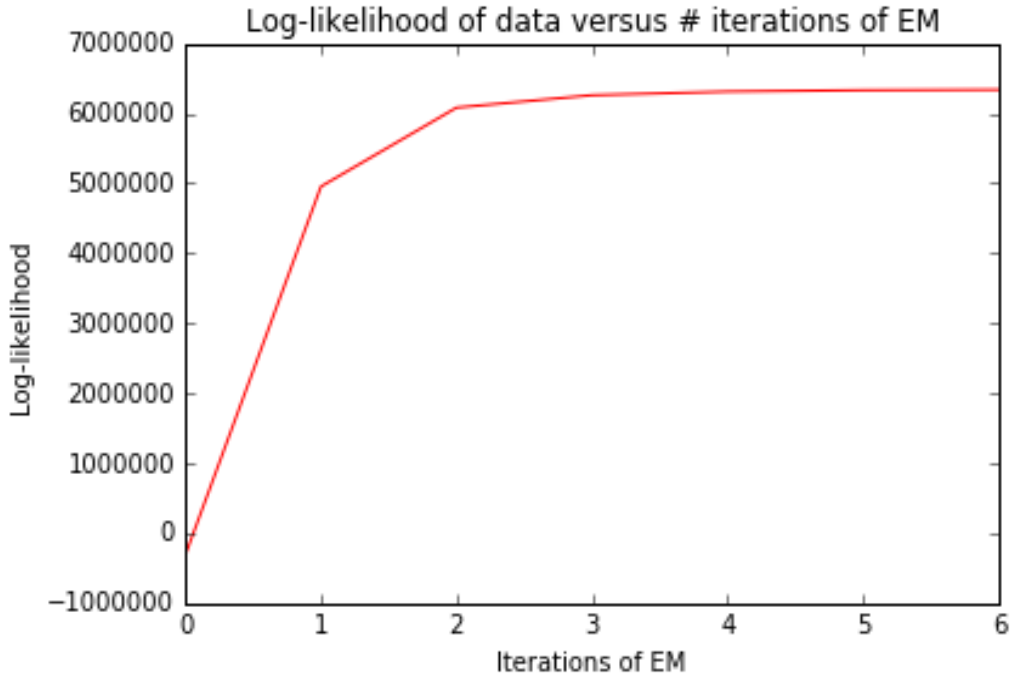


Using NN, this is a collection of faces that the confidence of classification is less than a chosen threshold of 50% for each of the classes. This 50% was chosen because it shows that none of these faces generalize well and can never have a slight advantage at predicting the correct class. The odds will always be against them for all classes (none will be 51% or greater).

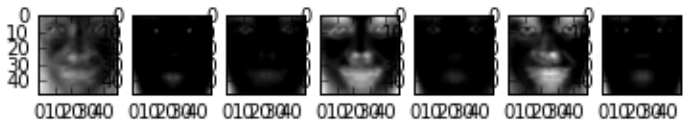
Part 4 Mixture of Gaussians

4.2 Training

4.2 (Random Initialization)

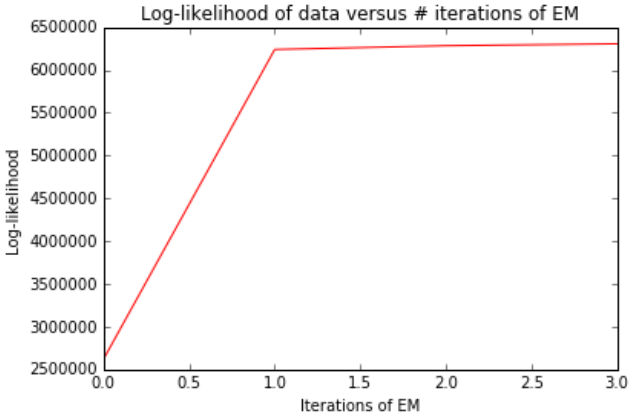
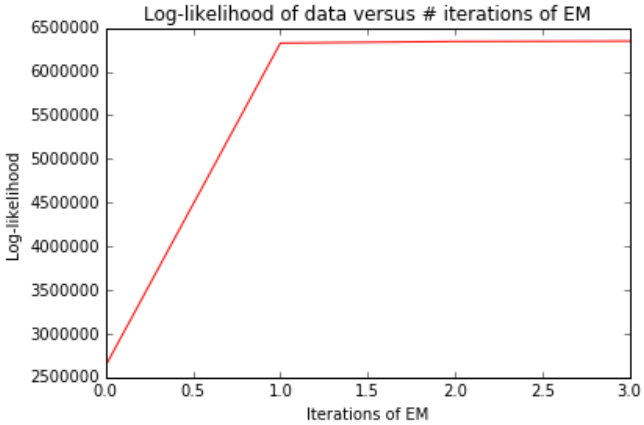
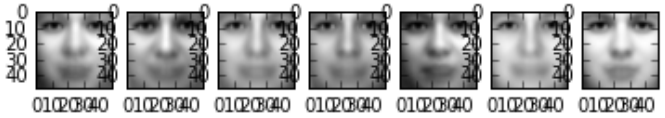
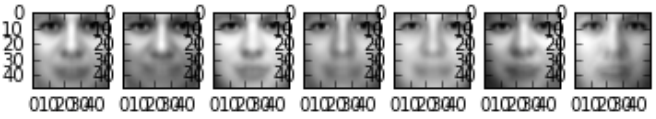
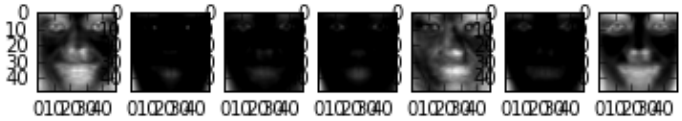
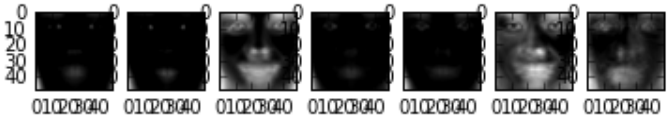


Mean – the mean vector represents a collection of faces, which show the average of all the classes, individually with 7 different expressions. This is a generative face as it is centered at the cluster of each class.



Variance – shows the variation in the class. For example, a happy face would look the same across all images classified as happy, which means low variance and high probablility of proper classification (dark). The white colour is shows high variance and thus lower change of proper classification.

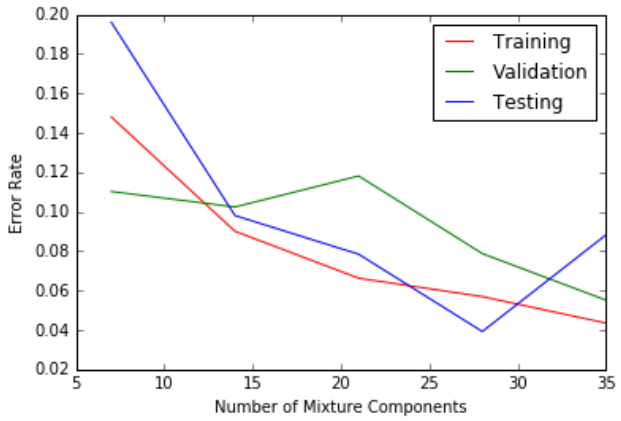
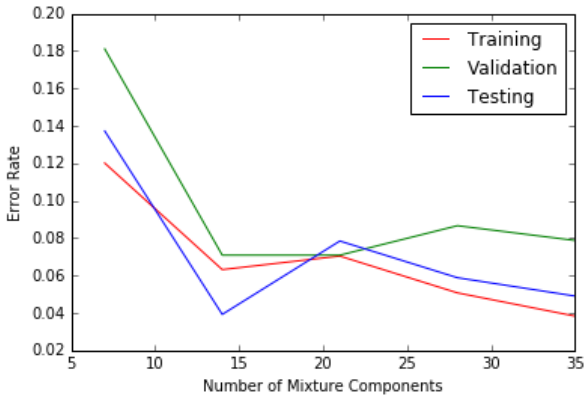
4.3 Initializing a mixture of Gaussians with K-means

First Run	Second Run
	
	
	

Comparing the the speed of convergence in 4.2 and 4.3 you can see that K means converges faster, at 1 iteration as oppsed to 3-4 iterations as seen with random initialization in 4.2.
You can see that in the two runs of K means shown above, there are slight differences in the Log-Likelihood, mu and variance. This shows that different runs converge to a local optima, but not the same global optima.

4.4 Classification using MoGs

A)

Random Initialization	K Mean Initialization
	

B) As can be seen the error rates on the training sets generally decrease as the number of clusters increases because there are more Gaussians to better model the data. Specifically, we started with 7, then 14, 21, 28, 35. You will see that as the number of Gaussians increases (closer to the number of data points), the error rate and variance will decrease and the class probability will increase, centered around mean.

C) For both types of initialization, you can see that as the number of Gaussians increases, test data is more accurate (lower error). As more Gaussians are used to approximate the training space, you can expect the model to generalize reasonably well for testing and fit very well the training data. As the number of Gaussians increases towards the number of data points, over fitting will occur as each data point will have a Gaussian centered around it, with low variance and high probability.