CSC 411
Introduction to Machine Learning
Assignment 2
Farzad Niroui
****

1 EM for Mixture of Gaussians

Given that the Gaussian mixture model is:

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

Additional, given the special case in which the covariance matrix $\boldsymbol{\Sigma}_k = \boldsymbol{\Sigma}$, for all k, we can derive the EM equations for maximizing the likelihood function.

E step:

The conditional probability of z given x (responsibility) is as follows:

$$\begin{aligned} \gamma_k = p(z=k|x) &= \frac{p(z=k)p(x|z=k)}{p(x)} \\ &= \frac{p(z=k)p(x|z=k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)} \\ &= \frac{\pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)} \end{aligned} \tag{2}$$

M step:

Taking the log-likelihood of the GMM:

$$ln p(X|\pi, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{n=1}^{N} ln \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}^{(n)}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \tag{3}$$

Given the Gaussian model $\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{\sqrt{(2\pi)^d |\boldsymbol{\Sigma}|}} exp(-\frac{1}{2}(x-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(x-\boldsymbol{\mu}))$ and the property of $\frac{\partial x^T A x}{\partial x} = x^T(A + A^T)$, we can set the derivative of Equation 3 to 0 and compute the

partial derivative with respect to $\boldsymbol{\mu}_k$. We also know that $\boldsymbol{\Sigma} = \boldsymbol{\Sigma}^T$, therefore $\boldsymbol{\Sigma}^{-1} = (\boldsymbol{\Sigma}^{-1})^T$ due to the covariance matrix being symmetrical.

$$
\begin{aligned}
\frac{\partial ln p(X|\pi, \boldsymbol{\mu}, \boldsymbol{\Sigma})}{\partial \boldsymbol{\mu}_k} = 0 &= \sum_{n=1}^{N} \frac{\pi_k}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)} \frac{\partial \mathcal{N}(\mathbf{x}^{(n)}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\partial \boldsymbol{\mu}_k} \\
&= \sum_{n=1}^{N} \frac{\pi_k \mathcal{N}(\mathbf{x}^{(n)}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)} \frac{\partial(-\frac{1}{2}(x^{(n)} - \boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}^{-1}(x^{(n)} - \boldsymbol{\mu}_k))}{\partial \boldsymbol{\mu}_k} \\
&= \sum_{n=1}^{N} \frac{\pi_k \mathcal{N}(\mathbf{x}^{(n)}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)} \frac{(x^{(n)} - \boldsymbol{\mu}_k)^T (\boldsymbol{\Sigma}^{-1} + (\boldsymbol{\Sigma}^{-1})^T)}{2} \\
&= \sum_{n=1}^{N} \frac{\pi_k \mathcal{N}(\mathbf{x}^{(n)}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)} (x^{(n)} - \boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}^{-1} \\
&= \sum_{n=1}^{N} \frac{\pi_k \mathcal{N}(\mathbf{x}^{(n)}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)} \boldsymbol{\Sigma}^{-1}(x^{(n)} - \boldsymbol{\mu}_k) \\
&= \sum_{n=1}^{N} \gamma_k^{(n)} \boldsymbol{\Sigma}^{-1}(x^{(n)} - \boldsymbol{\mu}_k)
\end{aligned}
\tag{4}
$$

Now we can solve for the closed form updates for $\boldsymbol{\mu}_k$. Once again we note that $\boldsymbol{\Sigma}_k = \boldsymbol{\Sigma}$, for all k.

$$
\begin{aligned}
\sum_{n=1}^{N} \gamma_k^{(n)} \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_k) &= \sum_{n=1}^{N} \gamma_k^{(n)} \boldsymbol{\Sigma}^{-1}(x^{(n)}) \\
N_k \boldsymbol{\mu}_k &= \sum_{n=1}^{N} \gamma_k^{(n)} x^{(n)} \\
\boldsymbol{\mu}_k &= \frac{1}{N_k} \sum_{n=1}^{N} \gamma_k^{(n)} x^{(n)}
\end{aligned}
\tag{5}
$$

Where $N_k = \sum_{n=1}^{N} \gamma_k^{(n)}$.

Similarly, we can derive the we can set the partial derivative of Equation 3 to 0 and compute the partial derivative with respect to $\boldsymbol{\Sigma}$. The helpful properties for this derivation are $\frac{d|x|}{dx} = |x|(x^{-1})^T$ and $\frac{dx^{-1}}{dx} = -x^{-1}x^{-1}$

$$\begin{aligned}\frac{\partial \mathcal{N}(\mathbf{x}^{(n)}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\partial \boldsymbol{\Sigma}} &= \frac{-1}{2}\frac{exp(...)(-\boldsymbol{\Sigma}^{-1}(x^{(n)}-\boldsymbol{\mu}_k)(x^{(n)}-\boldsymbol{\mu}_k)^T\boldsymbol{\Sigma}^{-1}))}{\sqrt{(2\pi)^d|\boldsymbol{\Sigma}|}} + \frac{-1}{2}\frac{exp(...)(2\pi)^d|\boldsymbol{\Sigma}|\boldsymbol{\Sigma}^{-1}}{(2\pi)^d|\boldsymbol{\Sigma}|} \\ &= \frac{-1}{2}\mathcal{N}(\mathbf{x}^{(n)}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)[(-\boldsymbol{\Sigma}^{-1}(x^{(n)}-\boldsymbol{\mu}_k)(x^{(n)}-\boldsymbol{\mu}_k)^T\boldsymbol{\Sigma}^{-1})^T + \boldsymbol{\Sigma}^{-1}]\end{aligned} \tag{6}$$

$$\begin{aligned}\frac{\partial lnp(X|\pi, \boldsymbol{\mu}, \boldsymbol{\Sigma})}{\partial \boldsymbol{\Sigma}} = 0 &= \sum_{n=1}^{N}\frac{\pi_k}{\sum_{j=1}^{K}\pi_j\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}\frac{\partial \mathcal{N}(\mathbf{x}^{(n)}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\partial \boldsymbol{\Sigma}} \\ &= \frac{-1}{2}\sum_{n=1}^{N}\frac{\pi_k\mathcal{N}(\mathbf{x}^{(n)}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K}\pi_j\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}[(-\boldsymbol{\Sigma}^{-1}(x^{(n)}-\boldsymbol{\mu}_k)(x^{(n)}-\boldsymbol{\mu}_k)^T\boldsymbol{\Sigma}^{-1})^T + \boldsymbol{\Sigma}^{-1}] \\ &= \frac{-1}{2}\sum_{n=1}^{N}\gamma_k^{(n)}[(-\boldsymbol{\Sigma}^{-1}(x^{(n)}-\boldsymbol{\mu}_k)(x^{(n)}-\boldsymbol{\mu}_k)^T\boldsymbol{\Sigma}^{-1})^T + \boldsymbol{\Sigma}^{-1}] \\ &= \sum_{n=1}^{N}\gamma_k^{(n)}\boldsymbol{\Sigma}^{-1}[(-\boldsymbol{\Sigma}^{-1}(x^{(n)}-\boldsymbol{\mu}_k)(x^{(n)}-\boldsymbol{\mu}_k)^T)^T + 1]\end{aligned} \tag{7}$$

$$\begin{aligned}\sum_{n=1}^{N}\gamma_k^{(n)} &= \sum_{n=1}^{N}\gamma_k^{(n)}[(\boldsymbol{\Sigma}^{-1}(x^{(n)}-\boldsymbol{\mu}_k)(x^{(n)}-\boldsymbol{\mu}_k)^T)^T] \\ \sum_{n=1}^{N}\gamma_k^{(n)} &= \sum_{n=1}^{N}\gamma_k^{(n)}[(x^{(n)}-\boldsymbol{\mu}_k)(x^{(n)}-\boldsymbol{\mu}_k)^T\boldsymbol{\Sigma}^{-1}] \\ \sum_{n=1}^{N}\gamma_k^{(n)}\boldsymbol{\Sigma} &= \sum_{n=1}^{N}\gamma_k^{(n)}(x^{(n)}-\boldsymbol{\mu}_k)(x^{(n)}-\boldsymbol{\mu}_k)^T \\ \boldsymbol{\Sigma} &= \frac{1}{N_k}\sum_{n=1}^{N}\gamma_k^{(n)}(x^{(n)}-\boldsymbol{\mu}_k)(x^{(n)}-\boldsymbol{\mu}_k)^T\end{aligned} \tag{8}$$

Where $N_k = \sum_{n=1}^{N}\gamma_k^{(n)}$.

To derive the equation for the mixing coefficient π_k we repeat the same procedure as before but now we must consider the identity that $\sum_{k=1}^{K}\pi_k = 1$ and use a Lagrange multiplier and

maximize the following expression:

$$
\begin{aligned}
& lnp(X|\pi, \boldsymbol{\mu}, \boldsymbol{\Sigma}) + \lambda(\sum_{k=1}^{K} \pi_k - 1) \\
0 &= \sum_{n=1}^{N} \frac{\mathcal{N}(\mathbf{x}^{(n)}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)} + \lambda \\
\lambda &= -\sum_{n=1}^{N} \frac{\mathcal{N}(\mathbf{x}^{(n)}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)} \\
\pi_k \lambda &= -\sum_{n=1}^{N} \frac{\pi_k \mathcal{N}(\mathbf{x}^{(n)}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)} \\
\sum_{k=1}^{K} \pi_k \lambda &= -\sum_{k=1}^{K} \sum_{n=1}^{N} \gamma_k^{(n)} \\
\lambda &= -N \\
\pi_k \lambda &= -N_k \\
\pi_k(-N) &= -N_k \\
\pi_k &= \frac{N_k}{N}
\end{aligned}
\tag{9}
$$

Where $N_k = \sum_{n=1}^{N} \gamma_k^{(n)}$.

2 Convolutional Neural Networks

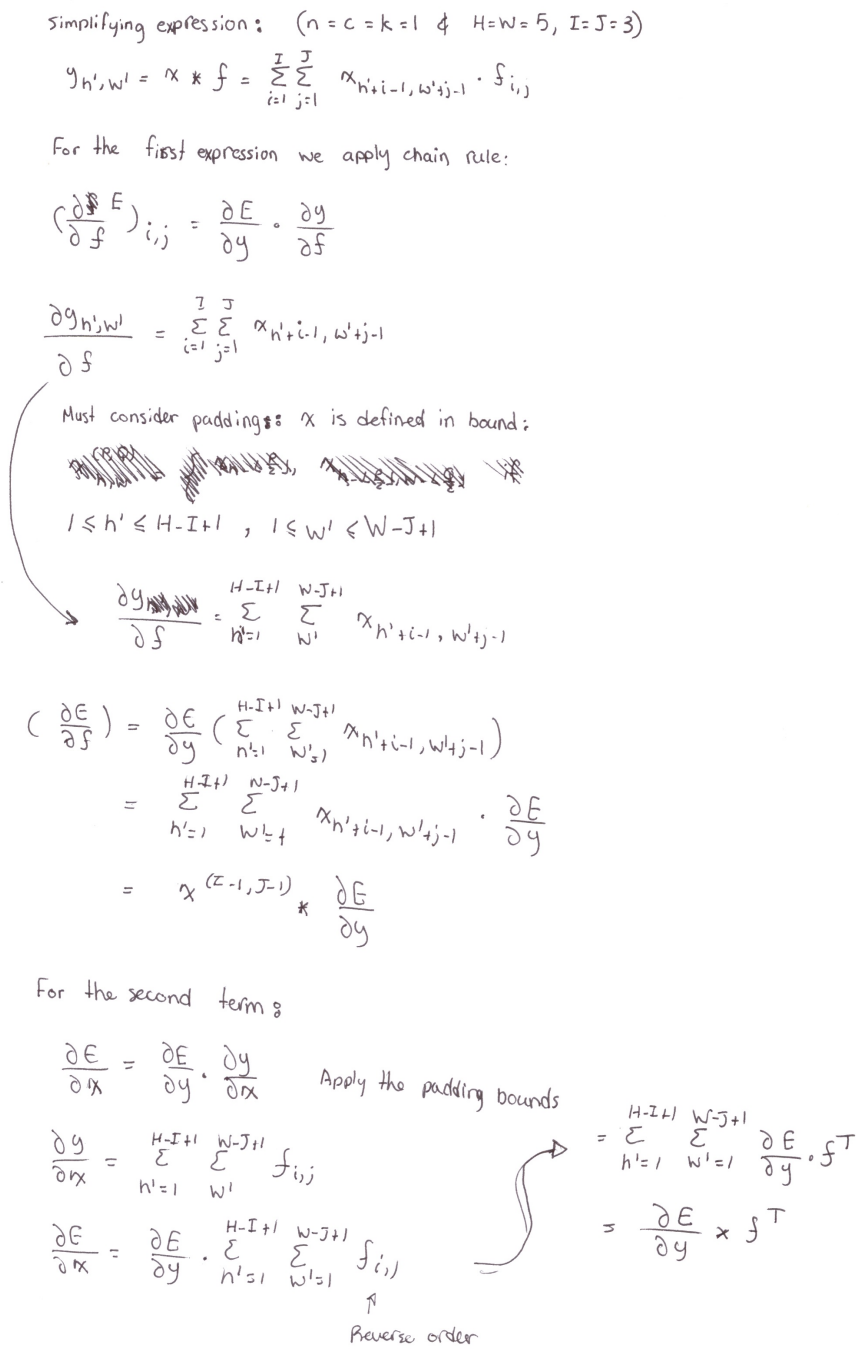

Figure 1:

3 Neural Networks

3.1 Basic generalization

Using the default hyperparameters, both the NN and the CNN were trained. The progression of the accuracy and error for training and validation data in both networks were tracked. Figure 2 shows the accuracy, and Figure 3 shows the error for both NN and CNN. Inspecting both of these figures, we can see similar trends in both networks. The training set is converging to a better accuracy with less error. In NN, the training settles at an accuracy of 0.85625 and error of 0.38740 while the validation set settles at 0.73747 accuracy and 0.95633 error. In CNN, the training set settles at an accuracy of 0.83847 and error of 0.43665 while the validation set settles at 0.79236 accuracy and 0.63323 error.

In the beginning when the networks have not converged the training set and the validation set have similar accuracy and error. When the networks converge, the parameters are chosen based on the training set hence the offset in both the error and the accuracy is a result of over fitting the parameters to the training set.

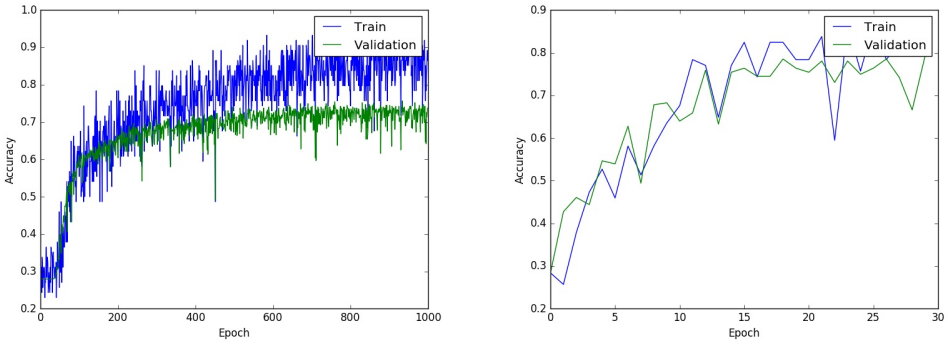

Figure 2: Accuracy of training and validation sets for NN (left) and CNN (right).

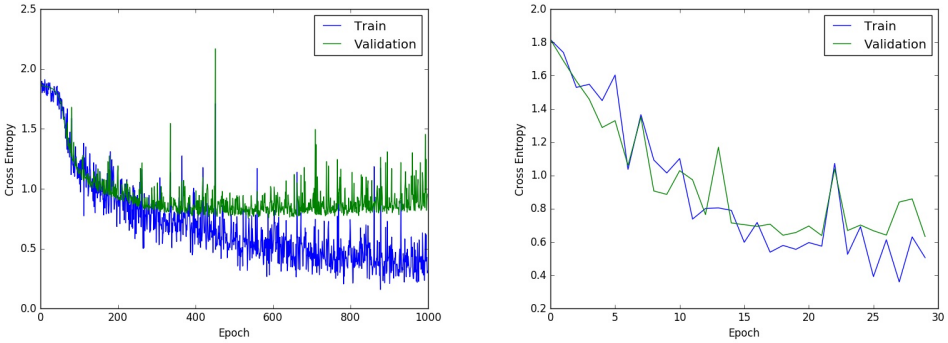

Figure 3: Accuracy of training and validation sets for NN (left) and CNN (right).

3.2 Optimization

Learning Rate

For this section the learning rate ϵ of 0.001, 0.01, 0.05, 0.1, and 1.0 are used. An additional learning rate of 0.005 is used for NN to gain better understanding of the accuracy and error trend. Inspecting the plots of accuracy and errors of the mentioned learning rates, it is easily distinguishable that some values outperform the rest. The plot for accuracy and error plots for the learning rates of 0.001, 0.01, 0.1, and 1.0 can be found in Figure 4 and Figure 5. From these plots it is evident and the learning rate parameter effected the convergence of both networks. For both networks, the learning rates of 0.001, 0.005, and 1.0 can be ignored as they are either too low or too high, causing the network to converge very slowly or not at all. To better understand what is the optimal learning rate, Figure 6 was used to see to what accuracy and error did each rates converge to. For NN, the learning rate of 0.05 resulted in the best accuracy for both the training set and the validation set. Moreover, for this rate, the error was also the lowest for both sets. For CNN, the learning rate of 0.1 resulted in the best accuracy and the lowest error for both the training and validation set.

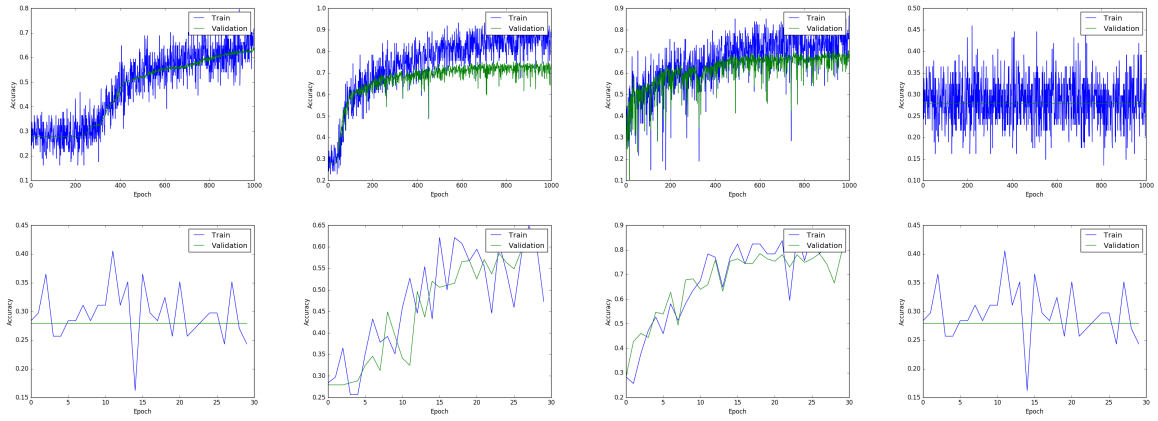

Figure 4: Accuracy of different learning rates(Left to right: 0.001, 0.01, 0.1, 1.0), NN (top) and CNN (bottom).

Momentum

To see the effect of momentum on the convergence rate, the momentum values of 0.3, 0.6, 0.9 are used. The default momentum value of 0.0 is also included to gain more insight on the effect of momentum. Figure 7 and Figure 8 contain the plots of the accuracy and error for all momentums. The purpose of adding momentum to the system is to attenuate the oscillation in the gradient decent. From both figures, it can be seen that momentum value of 0.0 values there is a lot of oscillation, but at values of 0.3 and 0.6 the oscillation has diminished. At 0.9 the momentum starts to interfere with the gradient decent and the system has become unstable. Therefore, momentum helps the network to converge to a more stable accuracy

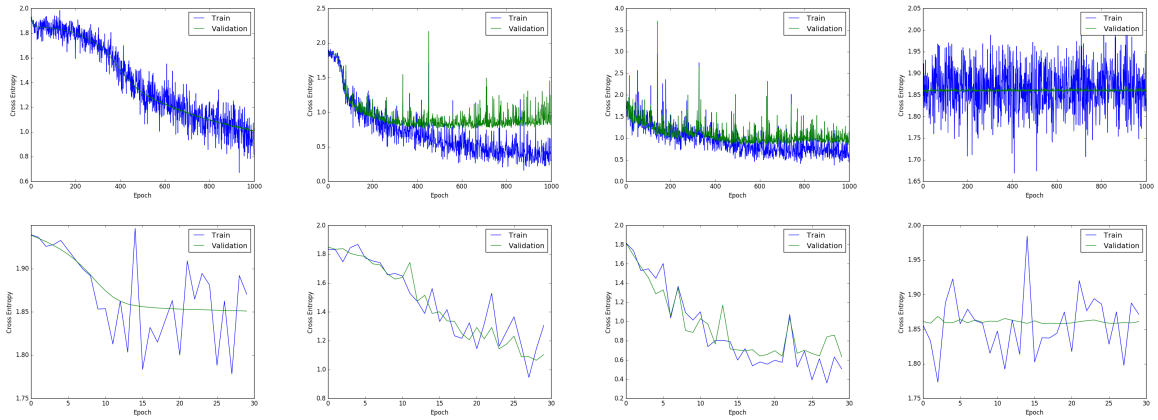

Figure 5: Error of different learning rates(Left to right: 0.001, 0.01, 0.1, 1.0), NN (top) and CNN (bottom).

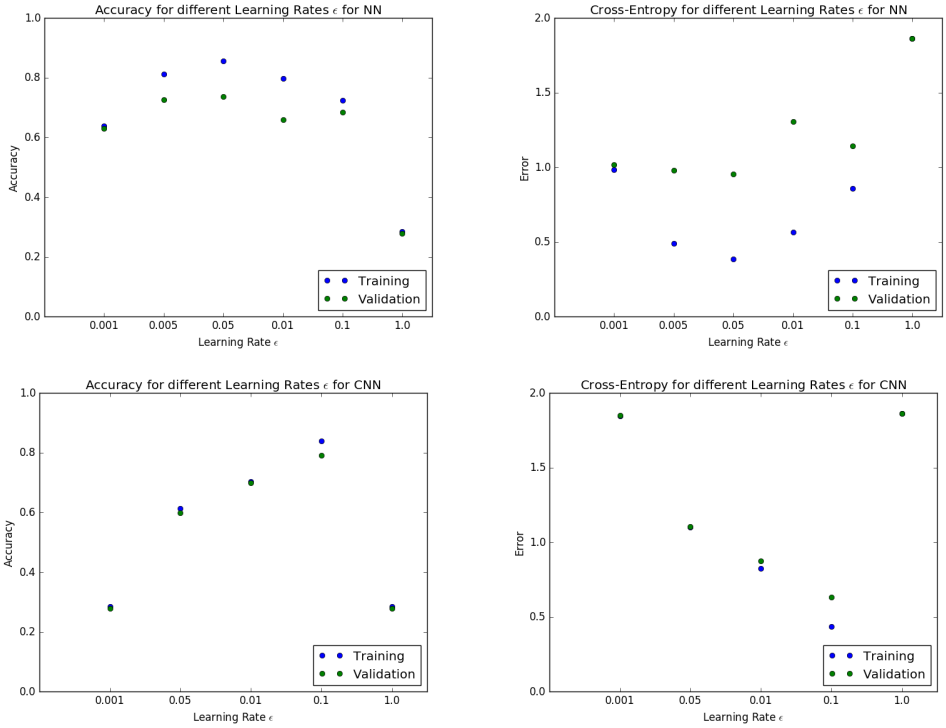

Figure 6: Converged accuracy and error for different learning rates, NN (top) and CNN (bottom).

and error value. From Figure 9, it can be said that both the momentum of 0.6 slightly outperform the value of 0.3 in terms of better accuracy and less error for both NN and CNN.

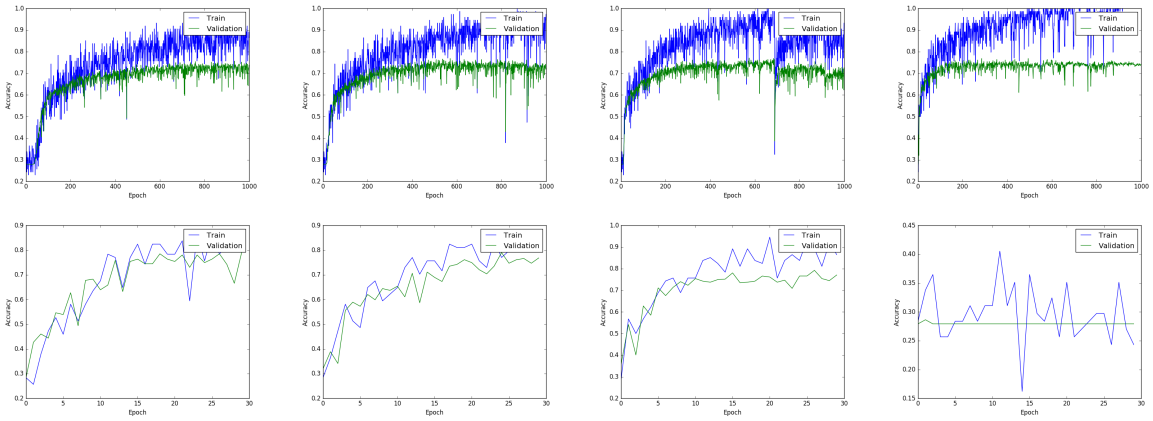

Figure 7: Accuracy of different momentums (Left to right: 0.0, 0.3, 0.6, 0.9), NN (top) and CNN (bottom).

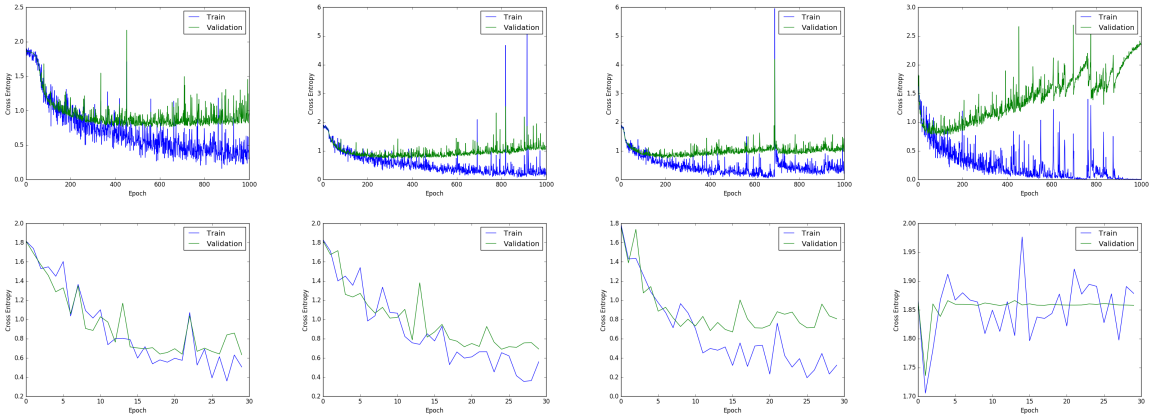

Figure 8: Error of different momentums (Left to right: 0.0, 0.3, 0.6, 0.9), NN (top) and CNN (bottom).

Batch Size

For this section, batch sized of 1, 10, 100, 500, and 1000 are used. Figure 10 and Figure 11 contain the plots of the accuracy and error for all of the different batch sizes. In both NN and CNN it can be seen that batch sizes of 1 and 10 are not sufficient in making the model converge. For batch size of 100, the model converges but now it is over fitting the training set. For 500, the model converges slower than 100 but it has less bias in NN and its less stable in CNN. Batch size of 1000 has even a slower convergence rate than 500 in NN, and for CNN it never converges. To decide if a batch size of 100 or 500 is suitable we inspect Figure 12. In NN, batch size of 100 has better accuracy for both training and validation set, and it has less error for the training set but not the validation set. In CNN, batch size of 100 is superior to 500 in both accuracy and error in both training and validation set. Therefore, for both network the best batch size is 100.

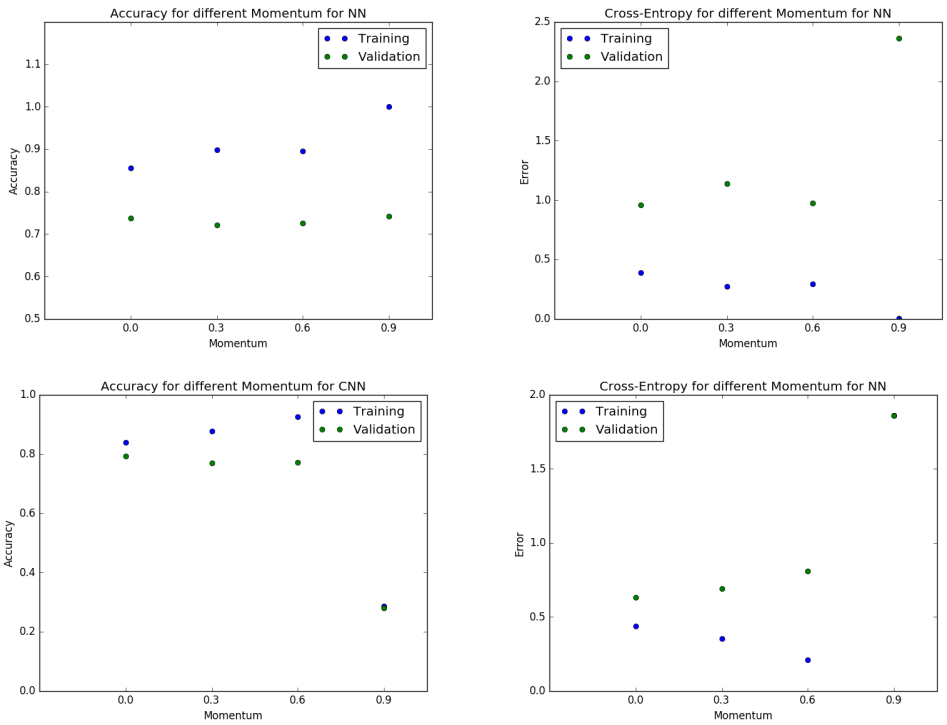

Figure 9: Converged accuracy and error for different momentums, NN (top) and CNN (bottom).

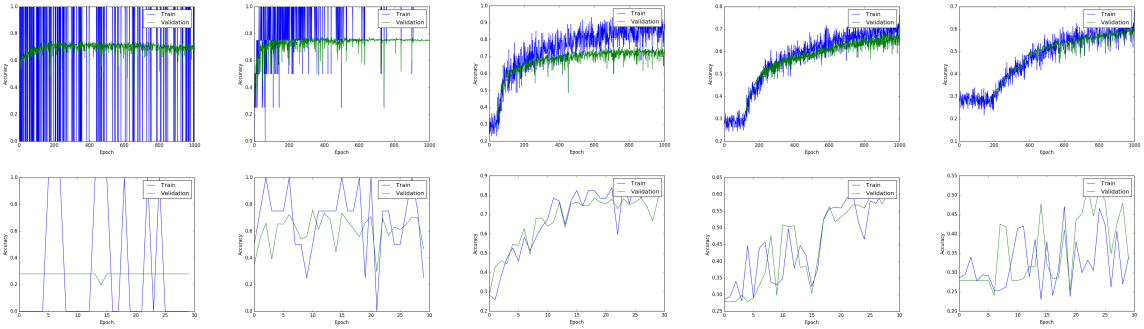

Figure 10: Accuracy of different batch sizes (Left to right: 1, 10, 100, 500, 1000), NN (top) and CNN (bottom).

3.3 Model architecture

3.3.1 NN:

For NN, each layer is tested with hidden units of 2, 50, and 100. In total, 6 different cases are trained with the adjusted epoch of 250 to decrease computation time. The plot of the accuracy and error for different units in the first layer is shown in Figure 13 and the second layer in Figure 14. Furthermore, Figure 15 shows the converged accuracy and error for all 6 difference cases. Inspecting all three figures, it can be concluded that higher number of

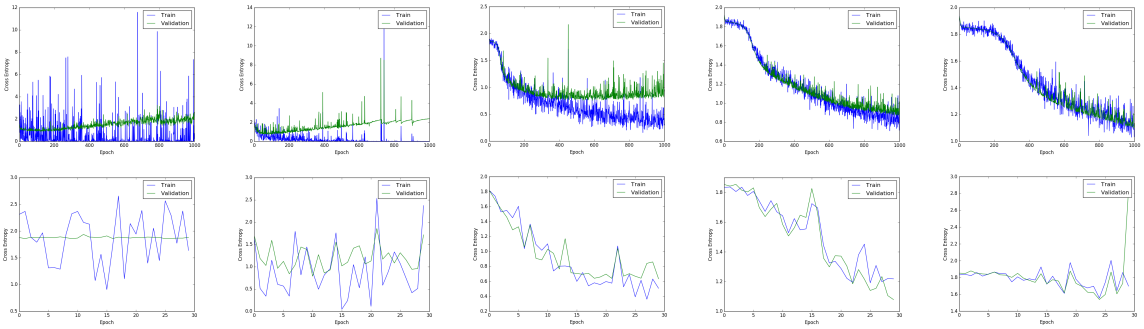

Figure 11: Error of different batch sizes (Left to right: 1, 10, 100, 500, 1000), NN (top) and CNN (bottom).

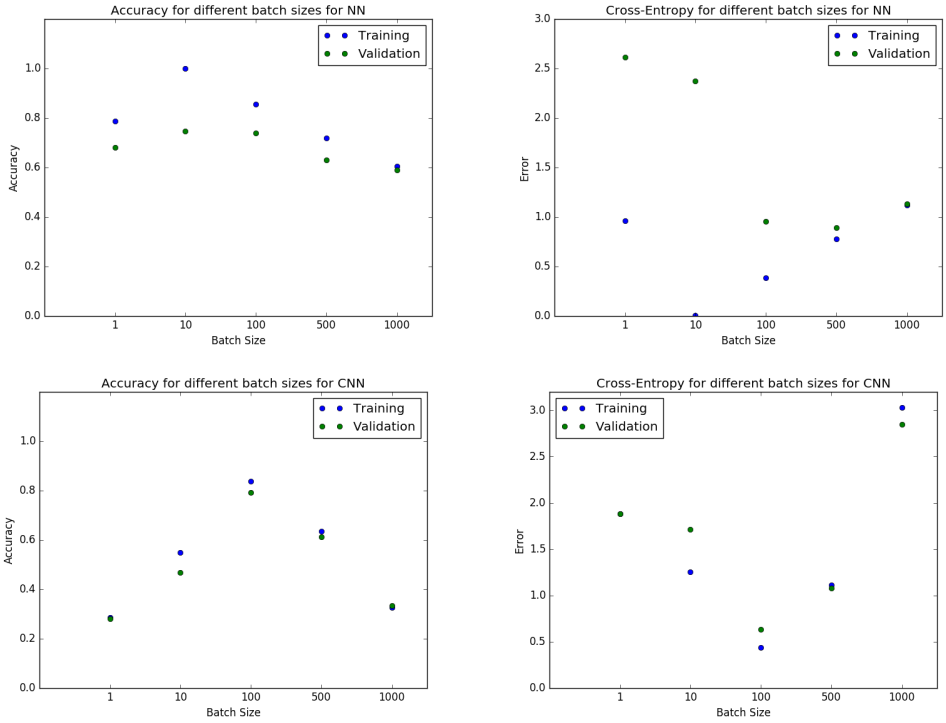

Figure 12: Converged accuracy and error for different batch sizes, NN (top) and CNN (bottom).

units per layer causes the model to over fit the training data. This is evident by the high offset between the accuracy and error of the training set to the validation set at high unit numbers. Higher number of units per layer can lead to better accuracy and error but the training accuracy will converge at a higher percentage while the validation set will converge at a much lower accuracy with higher error. This effect is evident in both layers.

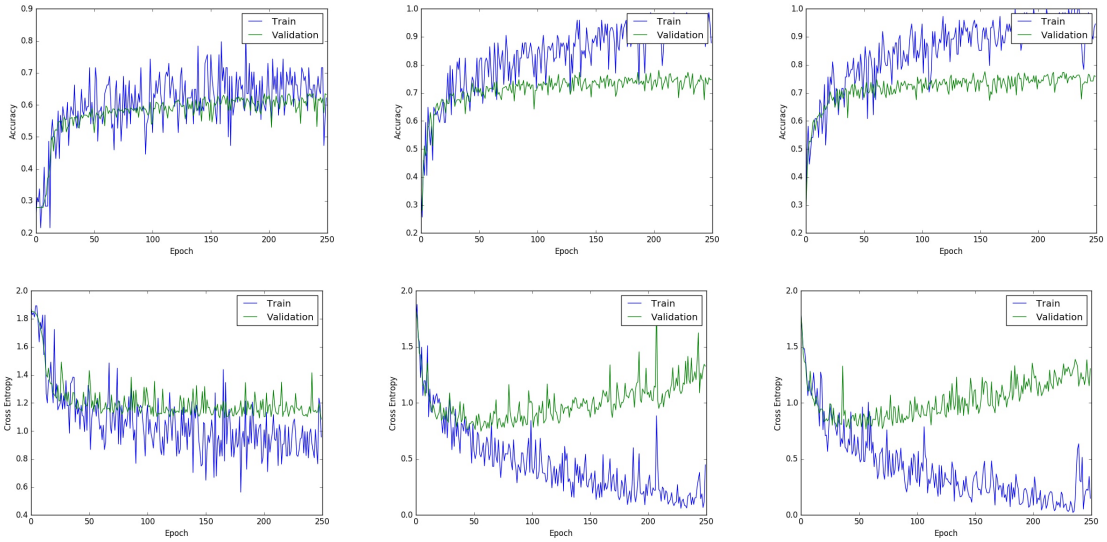

Figure 13: Accuracy (top) and Error (bottom) for different unit sizes in the first layer of NN (Left to right: 2, 50, 100)

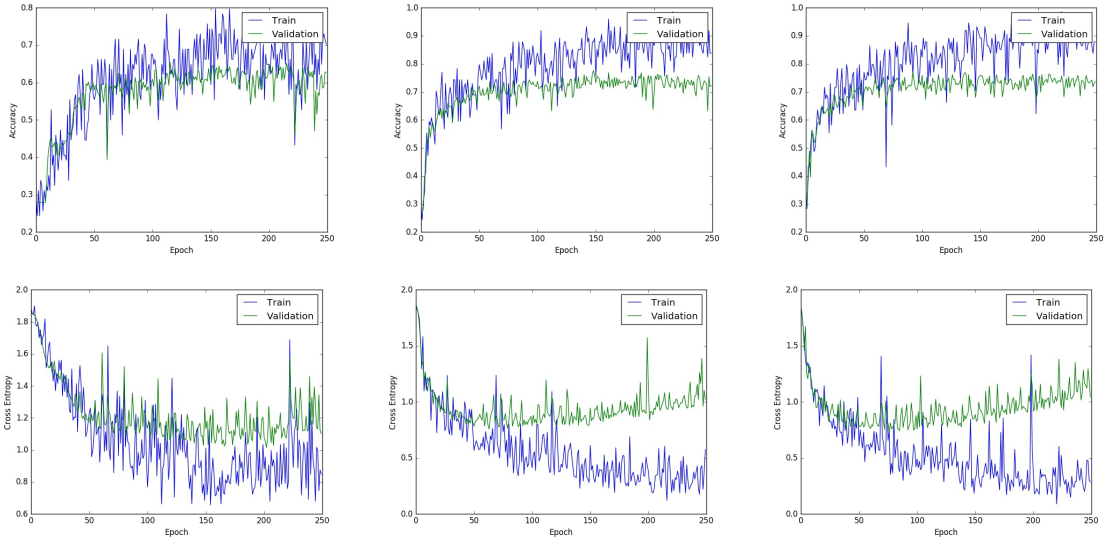

Figure 14: Accuracy (top) and Error (bottom) for different unit sizes in the second layer of NN (Left to right: 2, 50, 100)

3.3.2 CNN:

For CNN, each layer is tested with 2, 15, and 30 filters. All other parameters were kept at default. The plot of the accuracy and error for different number of filters in the first layer is shown in Figure 16 and the second layer in Figure 17. From these two figures, 2 filters did not converge in any layers, 15 filters converged only in the first layer, and 30 filters only converged in the second layer. Furthermore, from 18, 15 filters converged to the best accuracy and lowest error in the first layer, and 30 filters converged was the best in the

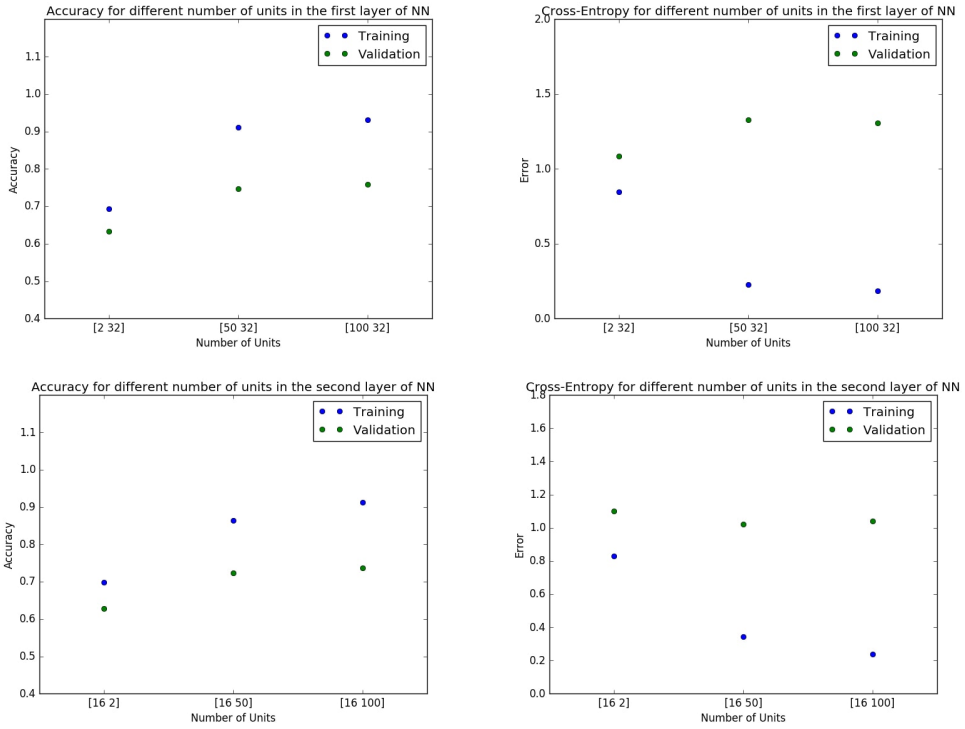

Figure 15: Converged accuracy and error for different number of units in the first (top) and second (bottom) layers of NN.

second layer.

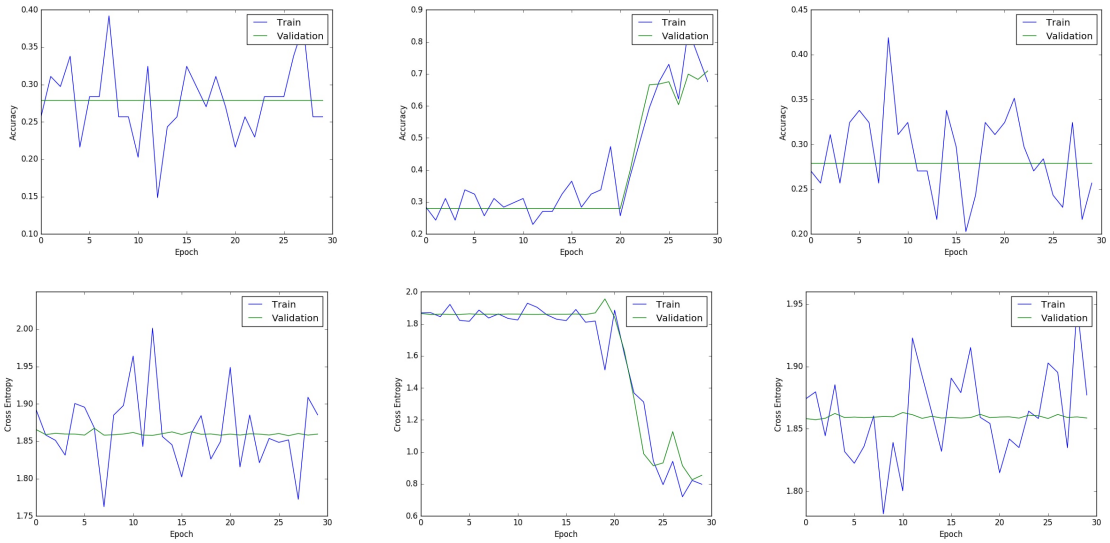

Figure 16: Accuracy (top) and Error (bottom) for different number of filters in the first layer of CNN (Left to right: 2, 15, 30)

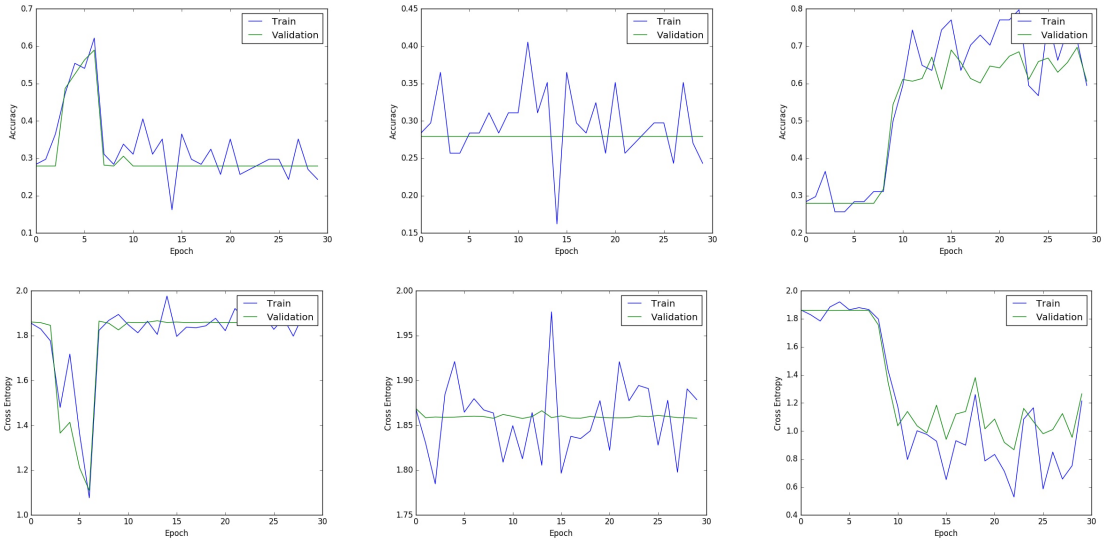

Figure 17: Accuracy (top) and Error (bottom) for different number of filters in the second layer of CNN (Left to right: 2, 15, 30)

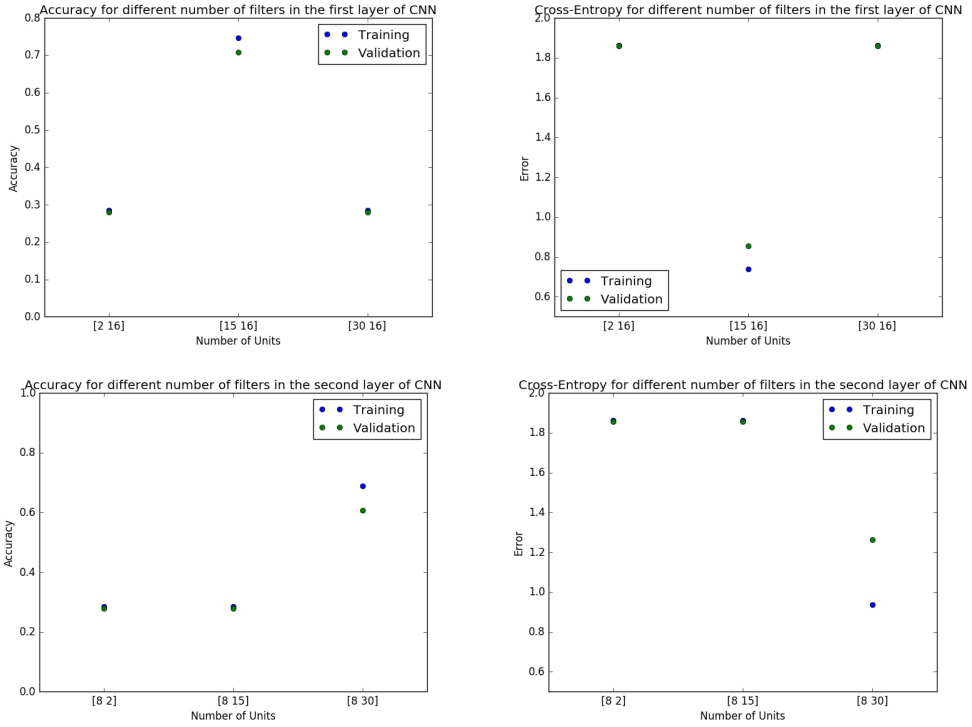

Figure 18: Converged accuracy and error for different number of units in the first (top) and second (bottom) layers of CNN.

3.4 Compare CNNs and fully connected networks

To calculate the number of parameters in each network, the default variables were used. For NN, there are 2304 inputs, 7 outputs, 16 and 32 units.

$$\#Parameters_{nn} = 2304(16) + 16 + 16(32) + 32 + 32(7) + 7 = 37655 \tag{10}$$

For CNN, there are 1 channel, 7 outputs, 8 and 16 filters, and filter size of 5.

$$\#Parameters_{cnn} = 5(5)(1)(8) + 8 + 5(5)(8)(16) + 16 + 16(64)(7) + 7 = 10599 \tag{11}$$

In order to compare the performance of both networks when they have similar number of parameters the number of units in the hidden layers of NN are changed. NN is changed over CNN because increasing the number of parameters in CNN will cause extremely long computation time. The new NN has 5 units in both layers. Upon training both networks, it was found that CNN outperformed NN. For the validation set, CNN got an accuracy of 0.78759 and error of 0.64808, while NN got an accuracy of 0.62768 and error of 1.10018. The better generalization of CNN is likely due to its property of utilizing less parameters for classification. The weights are shared, hence the units can identify a specific feature. The first later filters of the CNN, and first layer weights of NN are shown in Figure 19 and 20 respectively. Comparing these two visualization, it can be seen that for CNN, all the filters have different and specific concentration and densities while the units in NN seem to have random weights with no pattern.

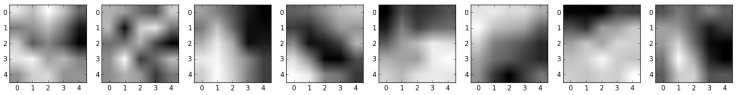

Figure 19: Visualization of the first layer filters of CNN.

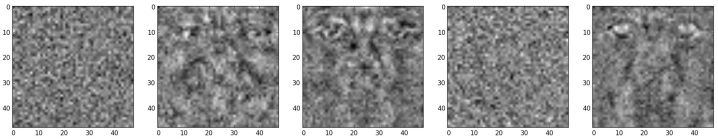

Figure 20: Visualization of the first layer weights of NN.

3.5 Network Uncertainty

For this section the neural network parameters were kept at default, except momentum which was set to 0.3. Three test data are used to do forward propagation and then softmax. The resulting probabilities for all three samples are demonstrated in Figure 21. The probability threshold is set at 0.6 as anything bellow it will not classify with high certainty. In the first plot, the test sample is the label 'Fear' and after softmax, the class with the top score is Fear, but the probability is bellow the threshold. Going with the highest score would result in the correct classification in this case. Similarly, the second sample will be classified correctly if we go with the highest score but this time the probabilities between Anger and Disgust are

fairly close. In the third sample, Neutral has the highest score but the sample label is Sad, therefore the highest score would be wrong.

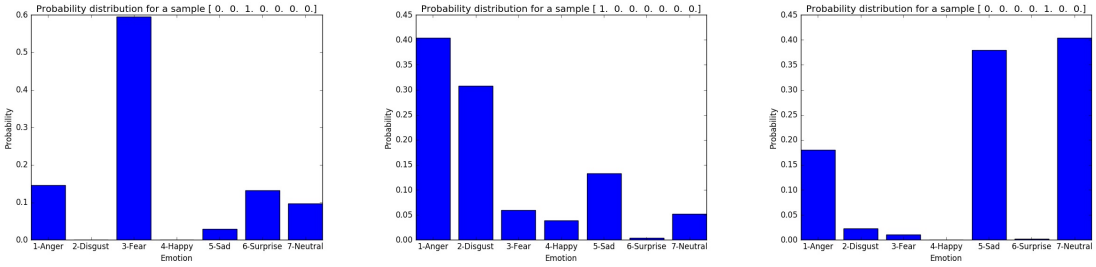

Figure 21: Classification of three different test data.

4 Mixtures of Gaussians (40 points)

4.2 Training

Based on trying different values for randConst it was noted that larger values would result in the mixing coefficient to be initialized closer to the same value. It was decided to set randCost to 1, and because the model converged with the default number of iteration, the number of interaction was not changed. Figure 22 shows the visualization of the mean and variance vectors. Figure 23 shows the log-likelihood plot.

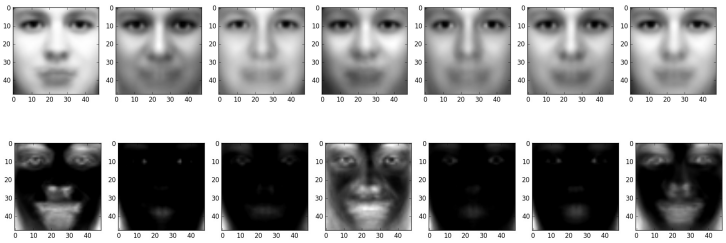

Figure 22: Visualization of mean vector (top) and variance vector (bottom).

4.3 Initializing a mixture of Gaussians with k-means

The MoG model was trained with 7 components using both the original initialization and with k-means. The k-means iterations was set to 5. The log-likelihood using the original initialization and with k-means are shown in Figure 24 and Figure 25 respectively. K-means converges after 1 iteration of EM while the original initialization converged after 2 iterations.

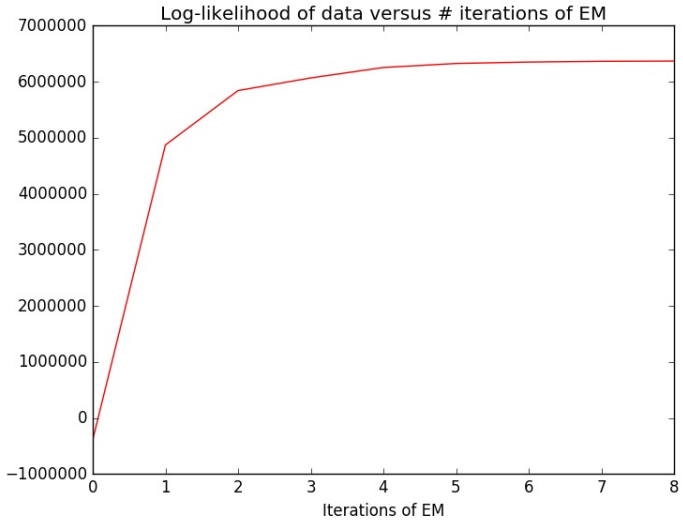

Figure 23: Visualization of mean vector (top) and variance vector (bottom).

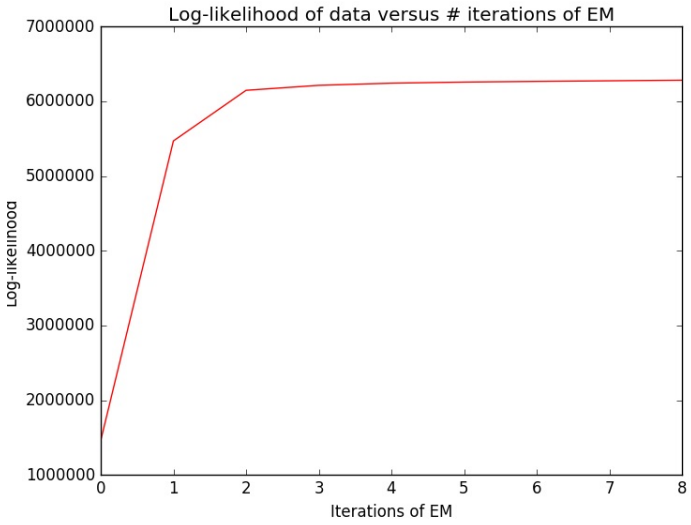

Figure 24: Log-likelihood of the original initialization.

4.4 Classification using MoGs

(a) Running the code for the mixture components of 7, 14, 21, 28, and 35 resulted in the plot shown in Figure 26.

(b) This trend holds in Figure 26. This is due to the fact that with more clusters there is a higher chance of having a Gaussian which the data is from.

(c) The error curve for the test set has always low values and has a very small slope. This can be cause by the data features being distinctly separable, causing more clusters to have

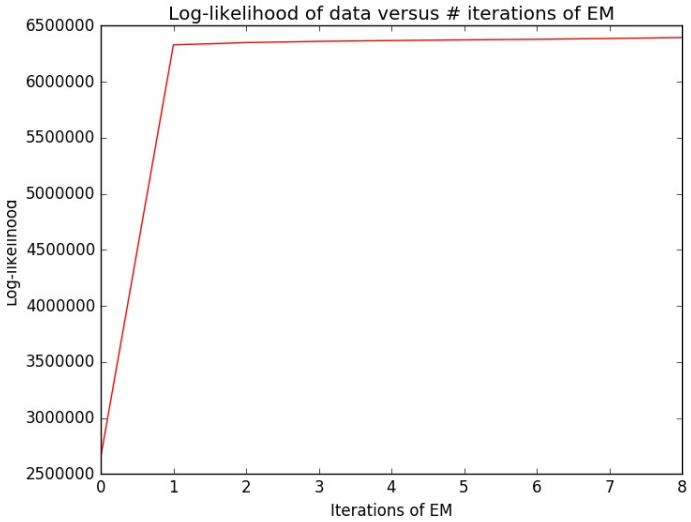

Figure 25: Log-likelihood of k-means.

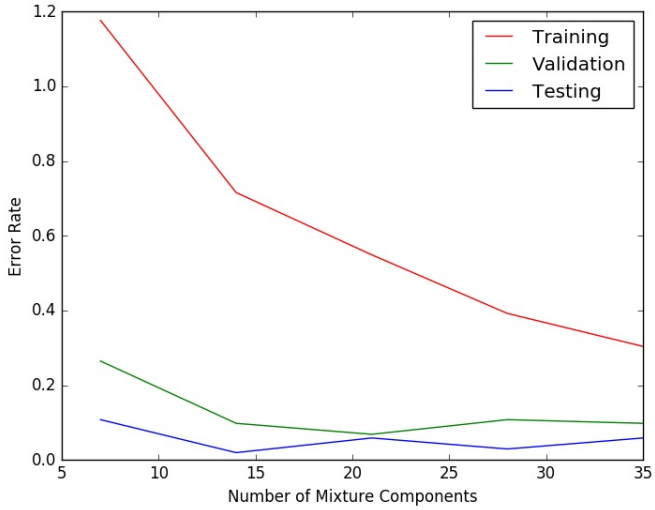

Figure 26: Training, validation and testing error for different number of mixture components.

no impact on the error rate.