# Competitive Learning Classifier for LFW

Authors : James Mnatzaganian & Qutaiba Saleh
Date : 04/25/15

A preprocessed version of the LFW dataset was used. This dataset was obtained from http://www.openu.ac.il/home/hassner/projects/frontalize/. This new dataset utilizes frontalized images. Additionally, those images were converted to grayscale and then resized (many sizes were tried, with a final size of 10x10 being utilized). 800 training samples were used (400 male and 400 female) and 200 testing samples were used (100 male and 100 female).

The network model is shown in Figure 1. This network consists of two competitive learning networks (referred to as gender networks), one for each label (male and female). The training is performed by only showing the male network images of males and the female network images of females. In this manner, each gender network will learn a representation of a specific gender. There are 100 inputs to each network (one for each pixel). The network utilizes the cost function shown in Equation 1. The weights are updated using Equation 2. For testing, each gender network is shown the same image and computes an output. The output represents the distance of the input image to the learned representation of the network. Those distances are optimized during training and refer to the Euclidean distance between the weights and the inputs. The two gender network outputs are compared, and the gender network with the lowest output is chosen as the network's prediction, i.e. the network with the shortest distance.

The network was able to support more advanced configurations than the one shown in Figure 1. As many outputs as desired may be chosen for each gender network. In this case, training and classification occurs the same as before; however, classification now has to consider which output belongs to which gender network. When there are more than one output per gender network, boosting is used. Boosting was implemented using a sliding window, where units too active were forced to become active less frequently and units too inactive were forced to become more active.

All software was written in Python. All of the code may be accessed by going to the following GitHub repo: https://github.com/jwmqms/lfw_gender. The code executed extremely fast (less than 10 ms per training epoch), so parallelizing it was unnecessary. All details about the code are explained in the repo. Additionally, there is a full API available for the code: http://techtorials.me/lfw_gender/index.html.

The hardware will be written in VHDL and a fully digital approach will be utilized. The optimizations outlined, below, were done to allow for the most simplistic hardware design possible, without sacrificing overall classification accuracy.

To determine what type of network configuration would be acceptable, a number of experiments were performed, using a simple grid search parameter optimization approach. The first experiment involved varying the size of the image with each gender network consisting of a single output. These results for training and testing are shown in Figure 2 and Figure 3, respectively. From those images, it is apparent that as long as the image size is between 10x10 and 30x30, the accuracy is relatively the same. Given this, the smallest image size was chosen as the preferred size, as this reduces the overall hardware complexity, as well as training and testing time.

The next experiment involved varying the number of outputs (clusters) for each gender network. The results for training and testing are shown in Figure 4 and Figure 5, respectively. It is shown that with only a single output, the results are very good and are converged upon after a very short number of epochs. It is likely that upon tuning other parameters, in particular those relating to boost, that more clusters would results in a better accuracy. Given that only one cluster produces good results and results in a much simpler hardware design (boosting does not have to be implemented and the classifier becomes a simple comparator between two values), a single output for the gender networks was converged upon. The weights of each of the networks for both the male networks and female networks are shown in Figure 6 - Figure 15. It is also observed that as the number of clusters increases, the time to converge upon suitable weights

increases. With too many clusters, finding suitable weights for all clusters can prove to be a difficult task, typically requiring many training epochs.

The final experiment involved varying the learning rate. The results for training and testing are shown in Figure 16 and Figure 17, respectively. Once the learning rate was small, enough, it was found that it did not make too much difference in the accuracy. To limit the amount of precision required in hardware, the learning rate of 0.01 was chosen. With this learning rate, after only 2 epochs a training and test accuracy of 68.9% and 69.1%, respectively is obtained. This is shown in Figure 18. A very few number of epochs proved to be sufficient, as a relatively high learning rate was chosen. This proves to be quite advantageous, as the training can be done very efficiently.

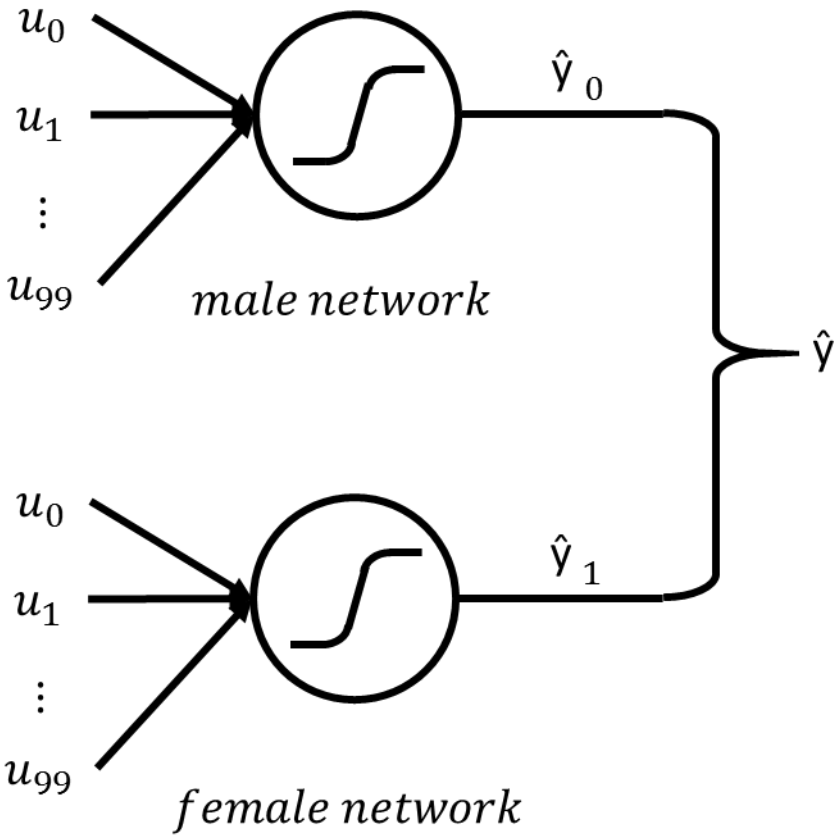


**Figure 1**: Competitive learning network structure

**Equation 1**: Competitive learning cost function

$$J(\boldsymbol{w}) = \frac{1}{2}\sum_{p=1}^{m}\sum_{i=1}^{M}\hat{y}_i^{(p)}\left|\boldsymbol{w}_i - \boldsymbol{u}^{(p)}\right|^2$$

**Equation 2**: Competitive learning weight update equation

$$\Delta w_{i,j} = \alpha\hat{y}^{(p)}(u_j^{(p)} - w_{i,j})$$

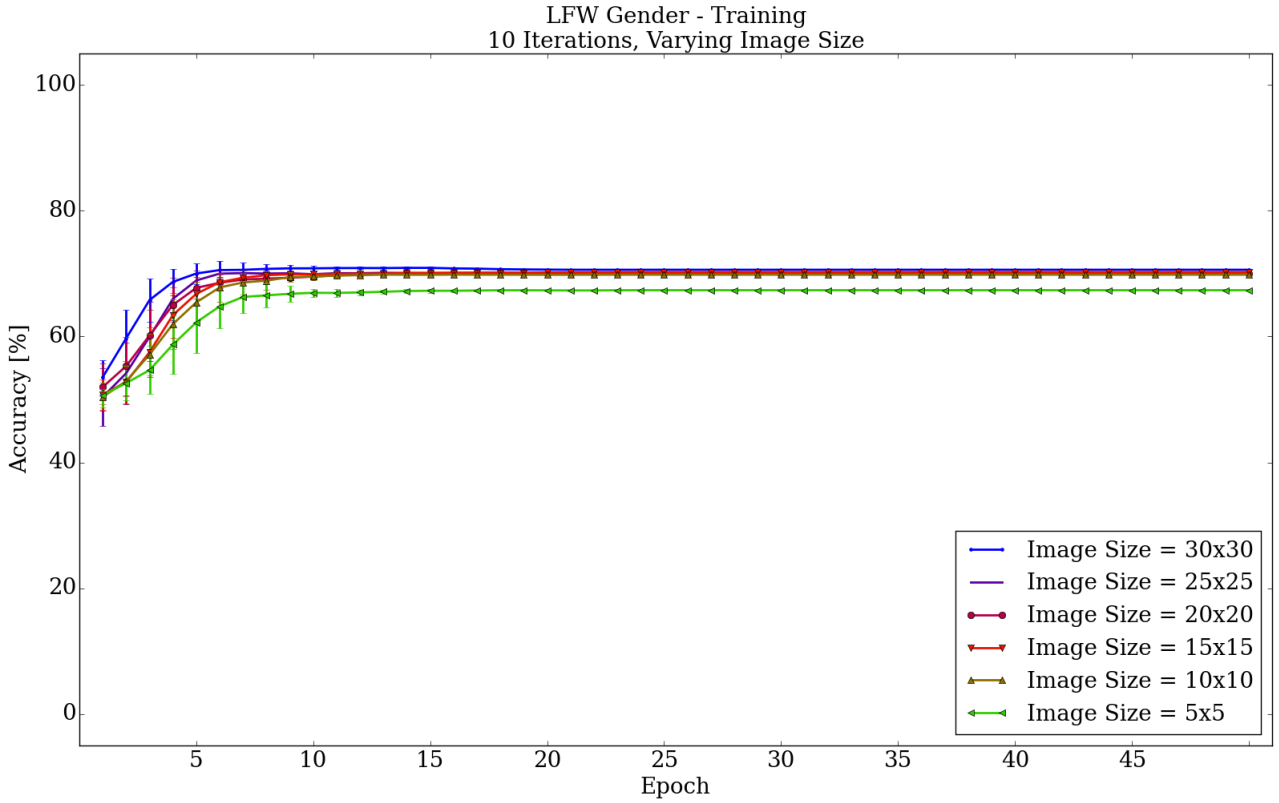


**Figure 2**: Training results for varying the image size

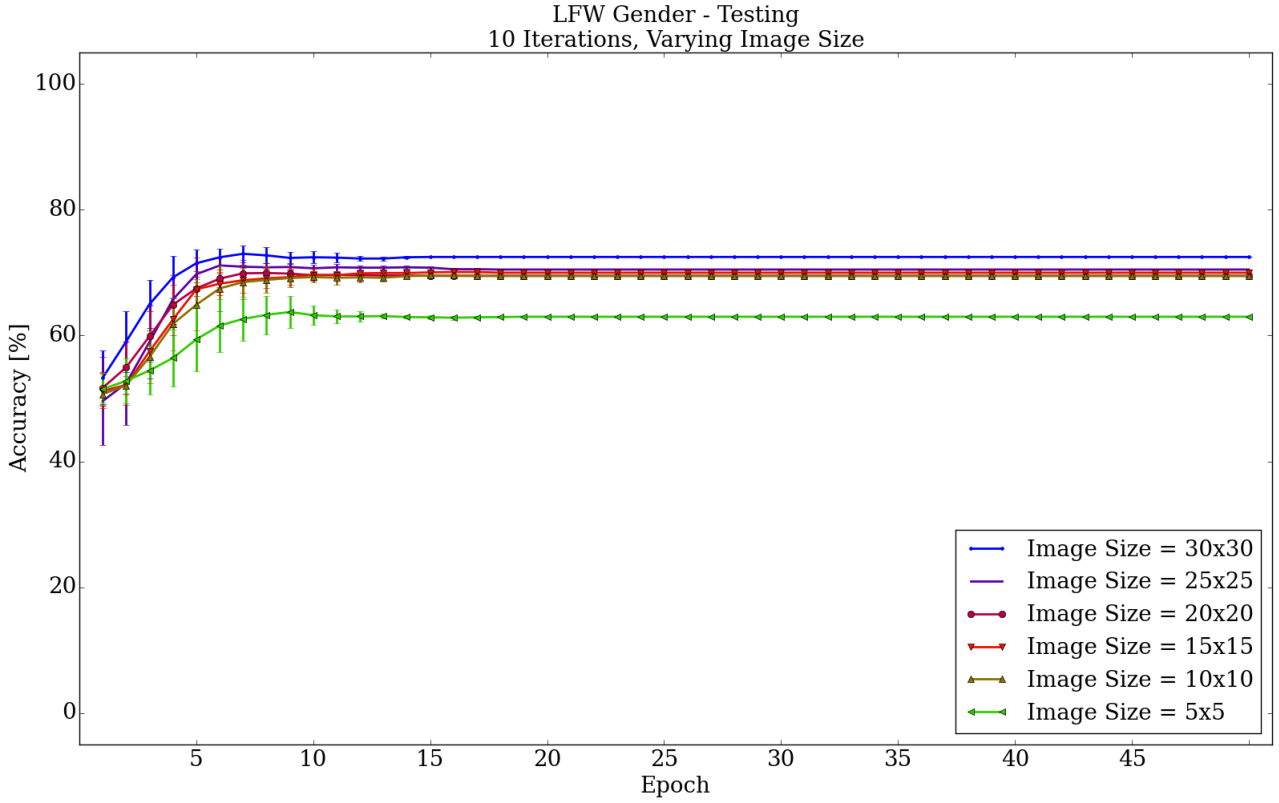


**Figure 3**: Testing results for varying the image size

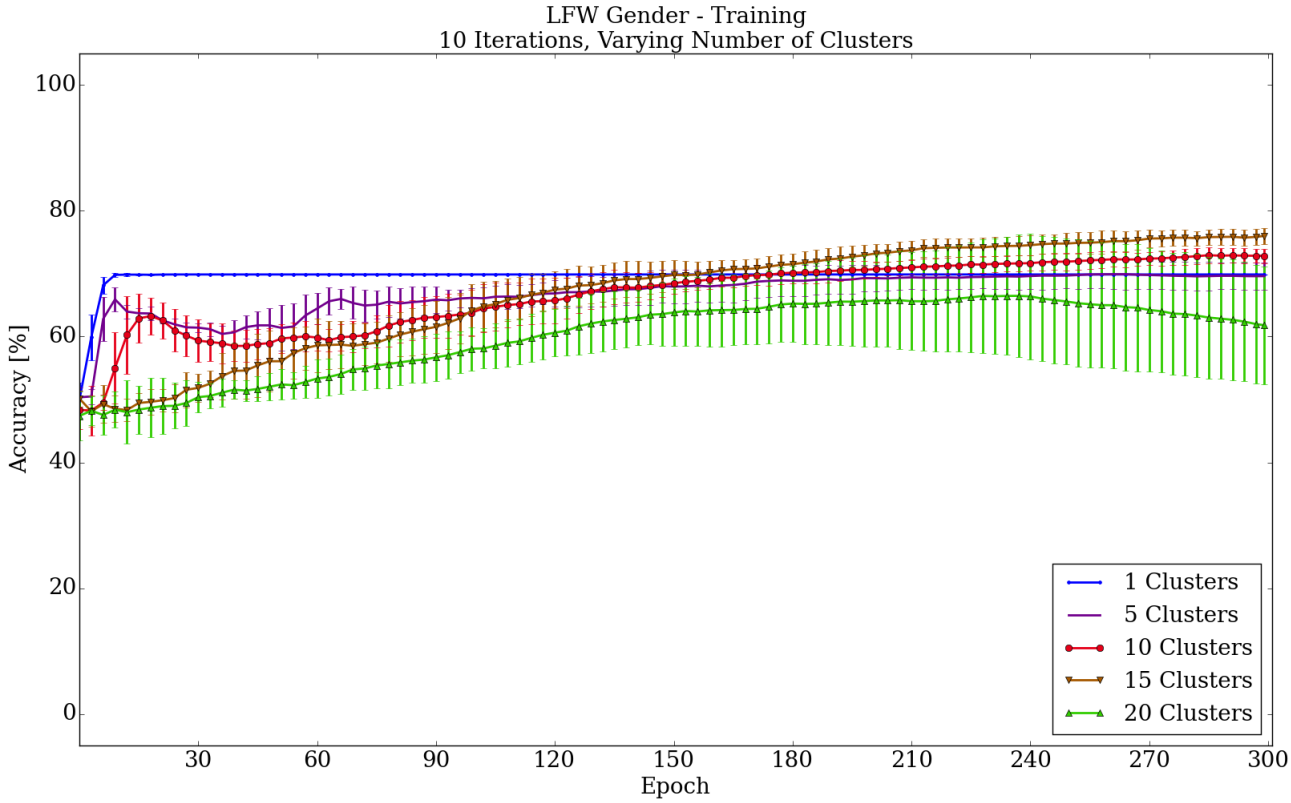


**Figure 4**: Training results for varying the number of clusters

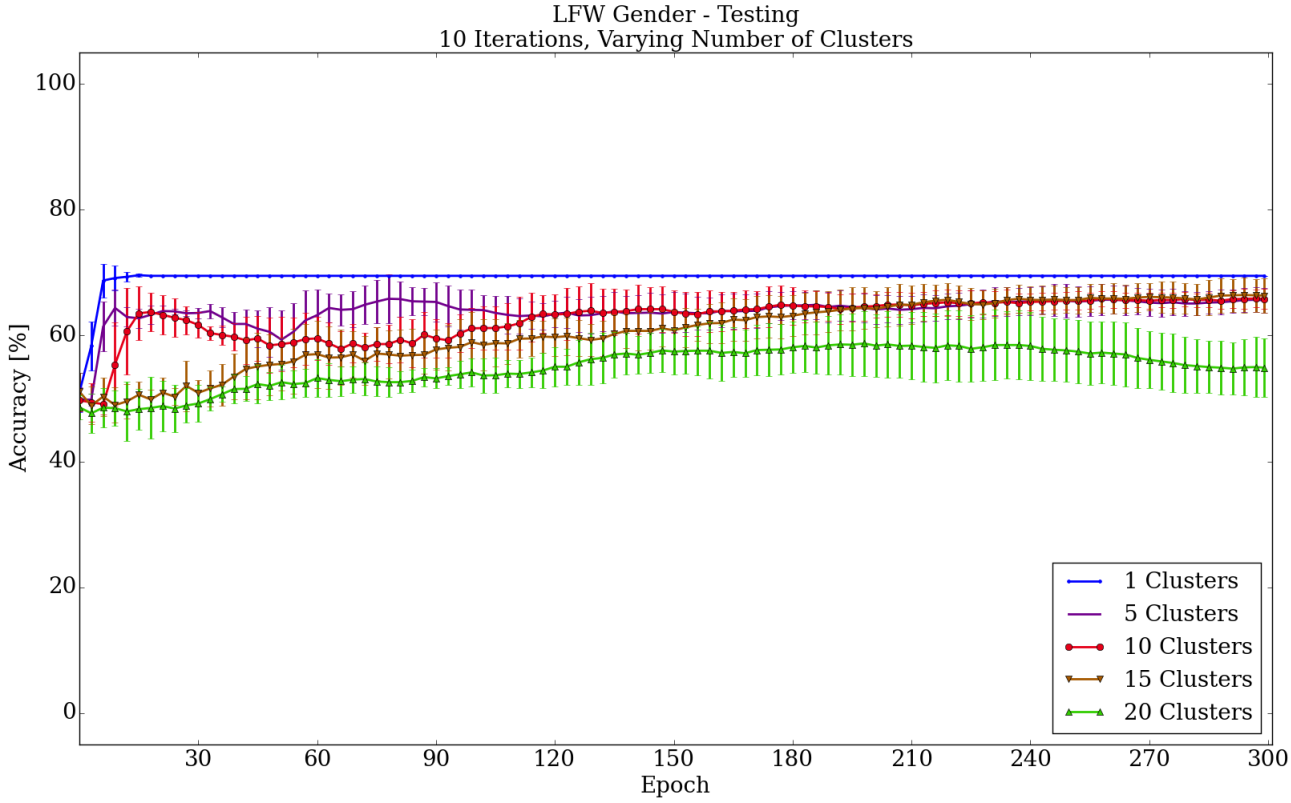


**Figure 5**: Testing results for varying the number of clusters

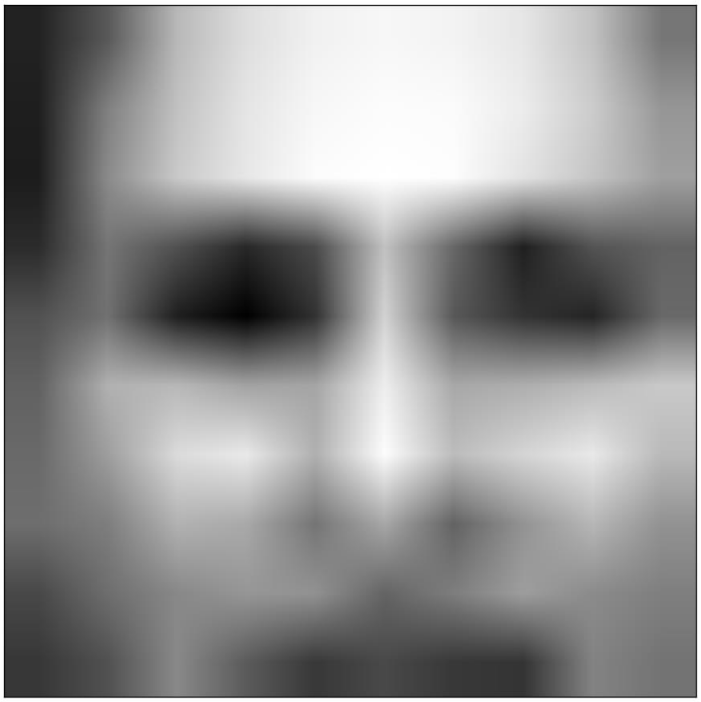

**Figure 6**: Cluster weights for male network with one cluster

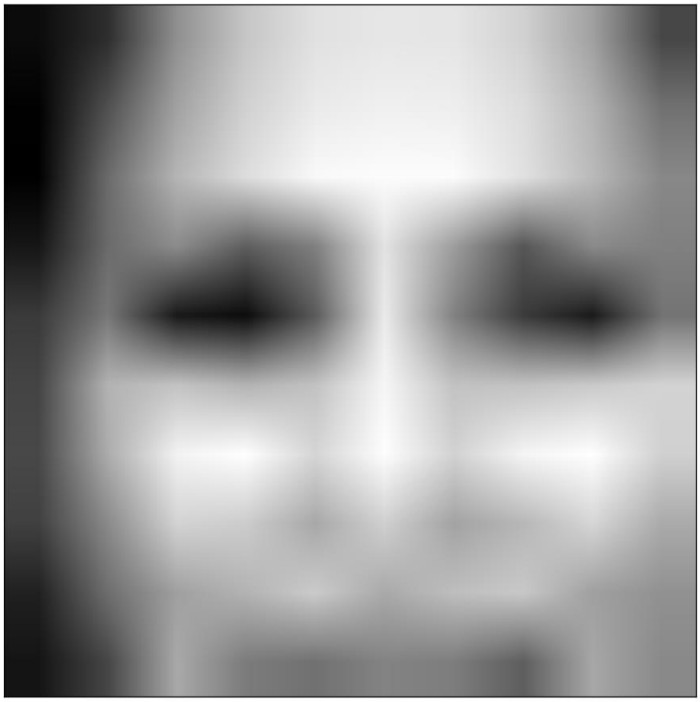

**Figure 7**: Cluster weights for female network with one cluster

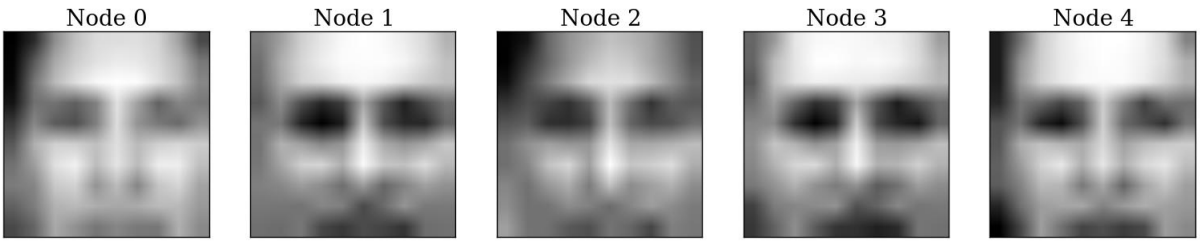


**Figure 8**: Cluster weights for male network with five clusters

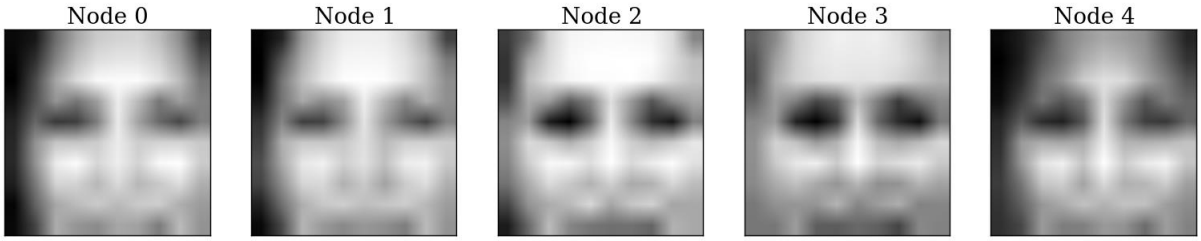


**Figure 9**: Cluster weights for female network with five clusters

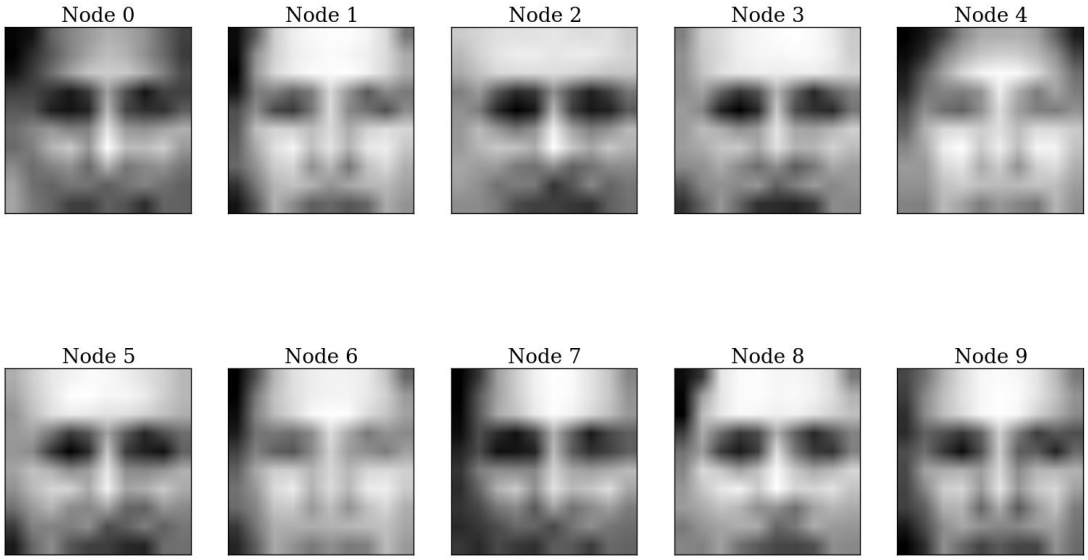


**Figure 10**: Cluster weights for male network with 10 clusters

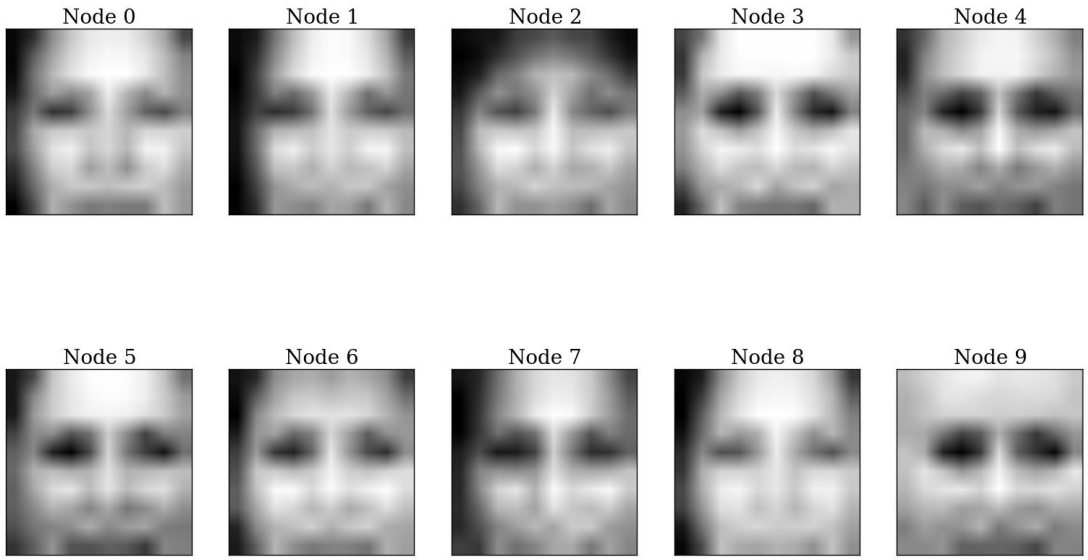


**Figure 11**: Cluster weights for female network with 10 clusters

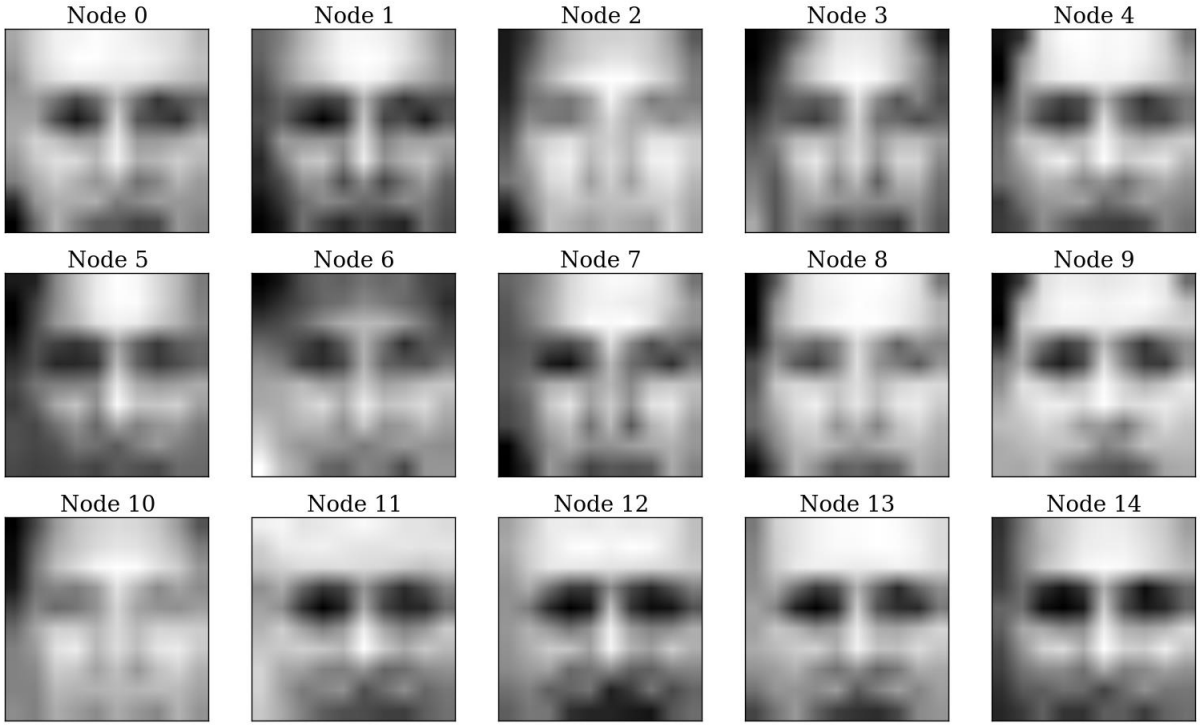


**Figure 12**: Cluster weights for male network with 15 clusters

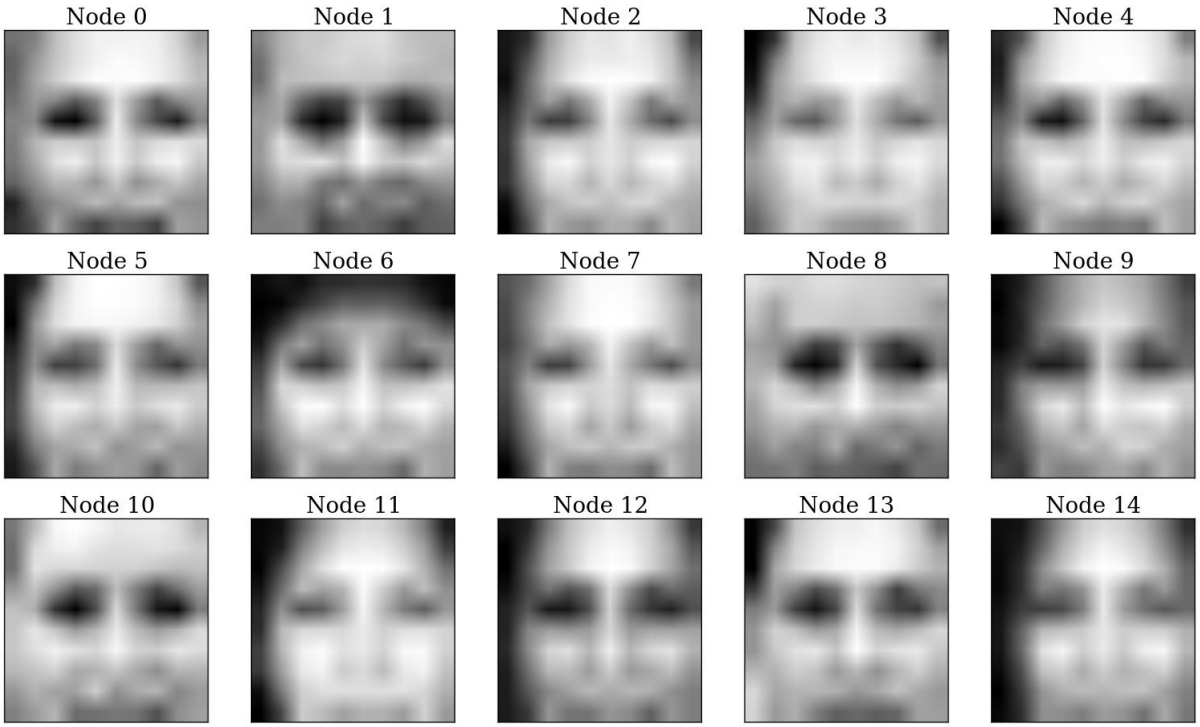


**Figure 13**: Cluster weights for female network with 15 clusters

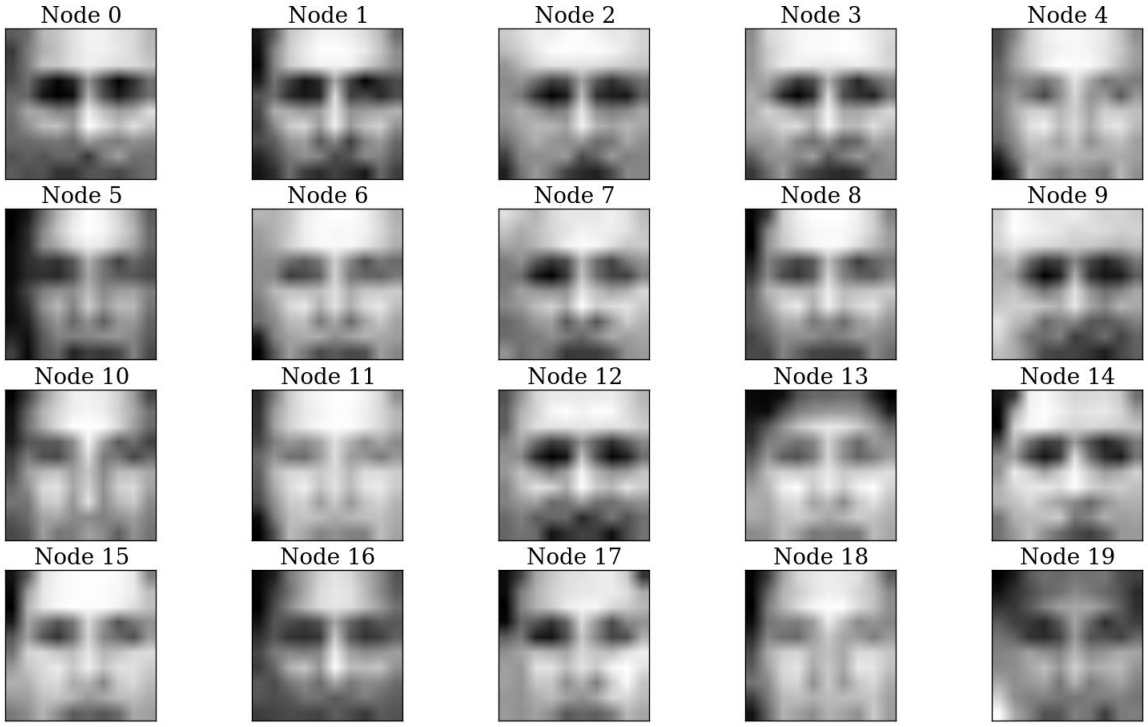


**Figure 14**: Cluster weights for male network with 20 clusters

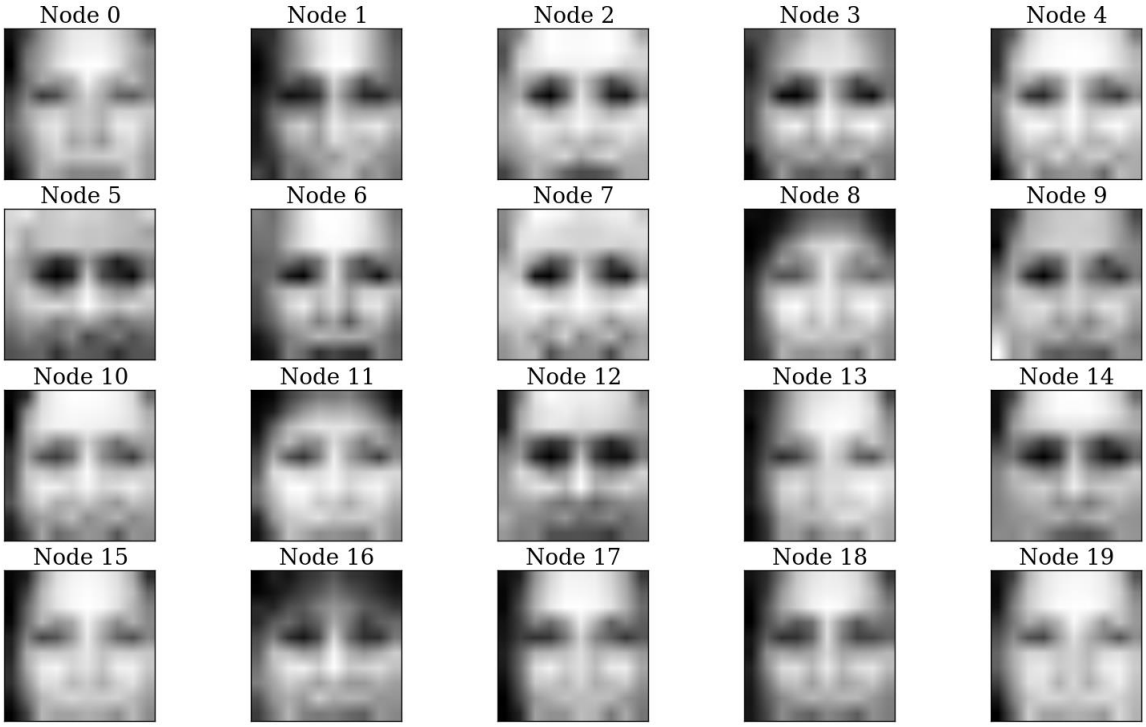


**Figure 15**: Cluster weights for female network with 20 clusters

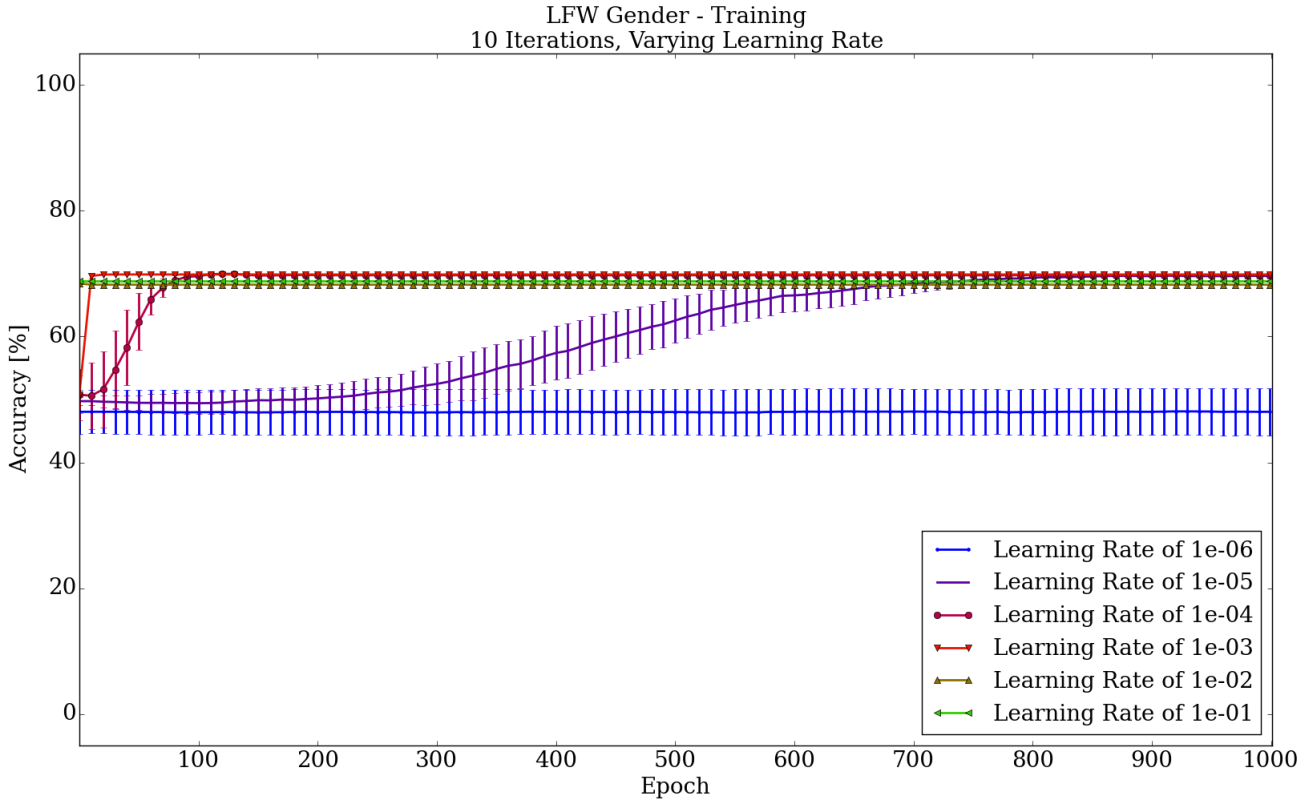


**Figure 16**: Training results for varying the learning rate

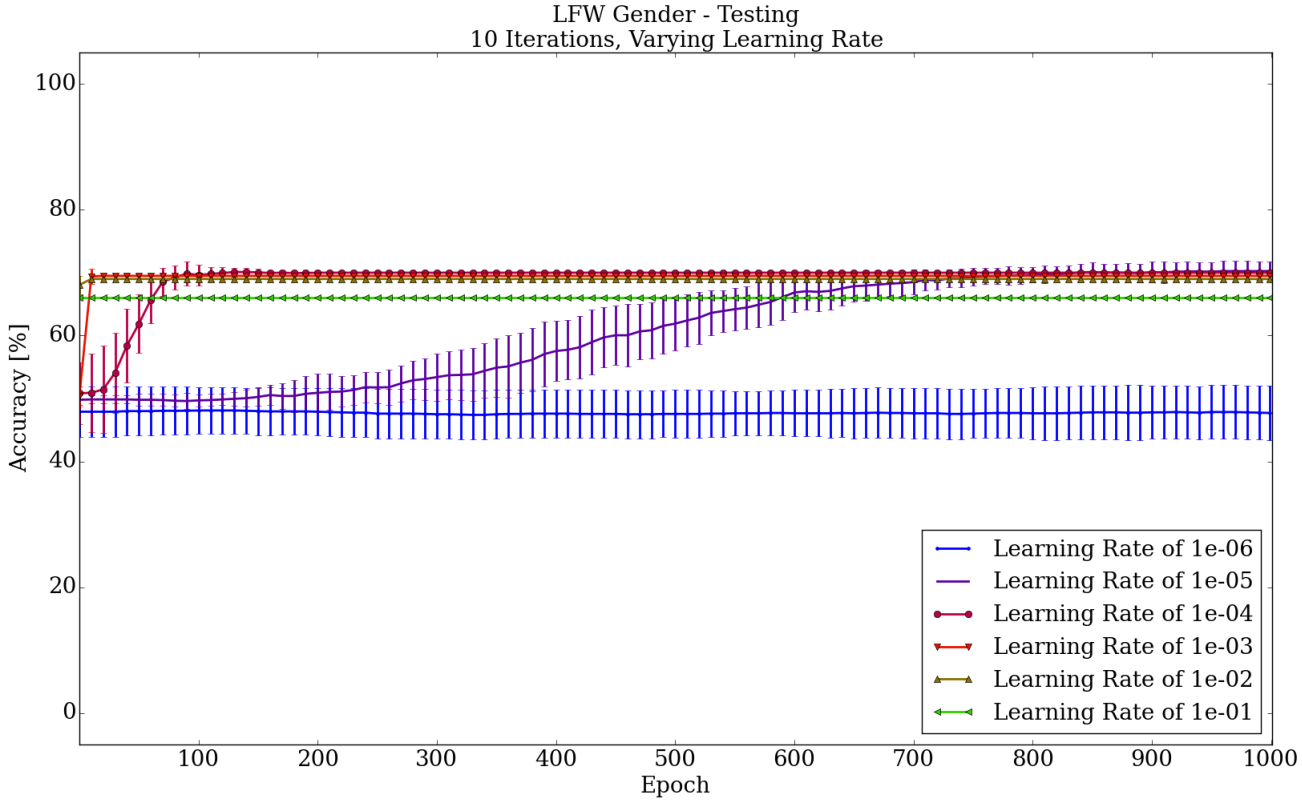


**Figure 17**: Testing results for varying the learning rate

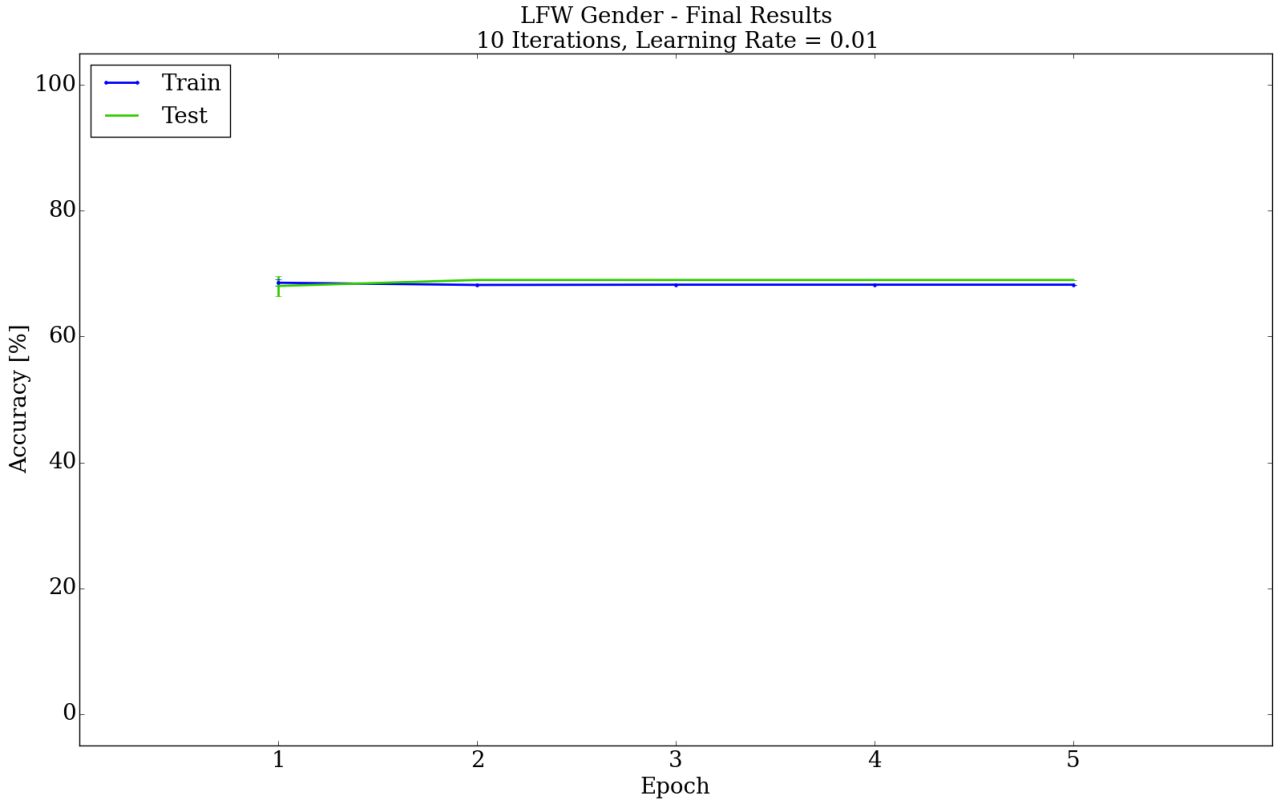


**Figure 18**: Final results after tuning parameters for most simplistic hardware design, while still obtaining good accuracy