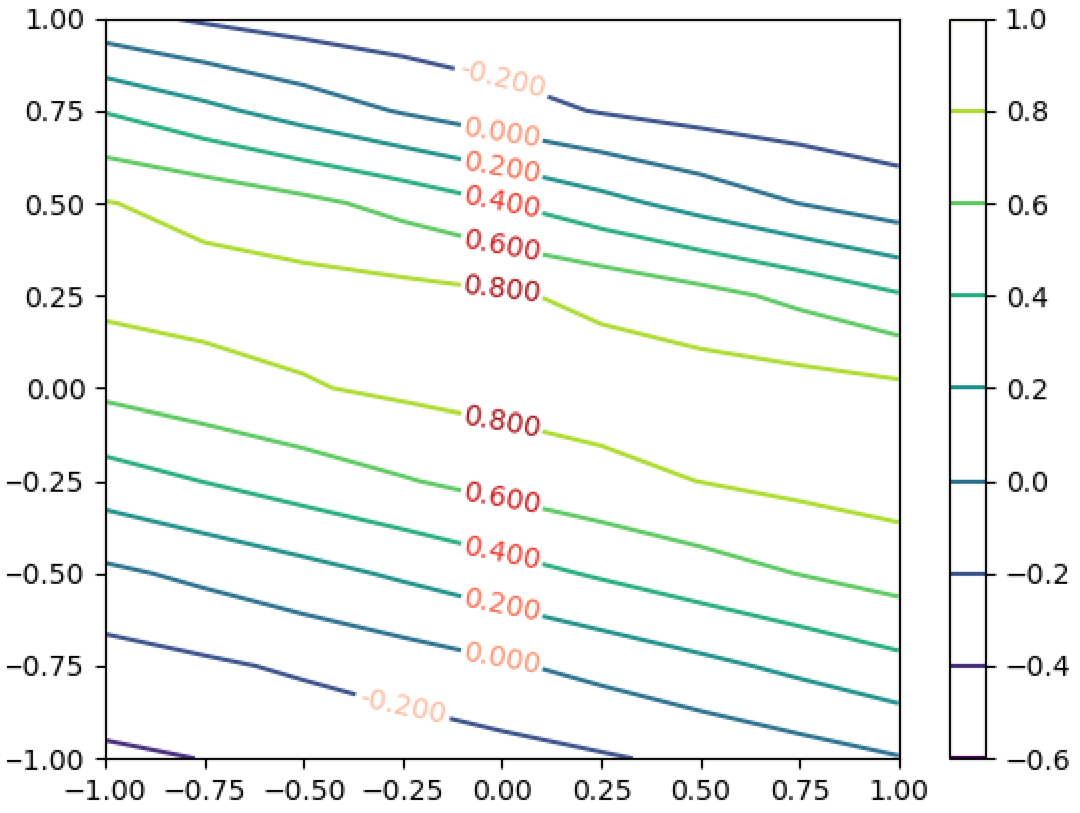
COMP4107 – Neural Networks

Assignment 2 Answers

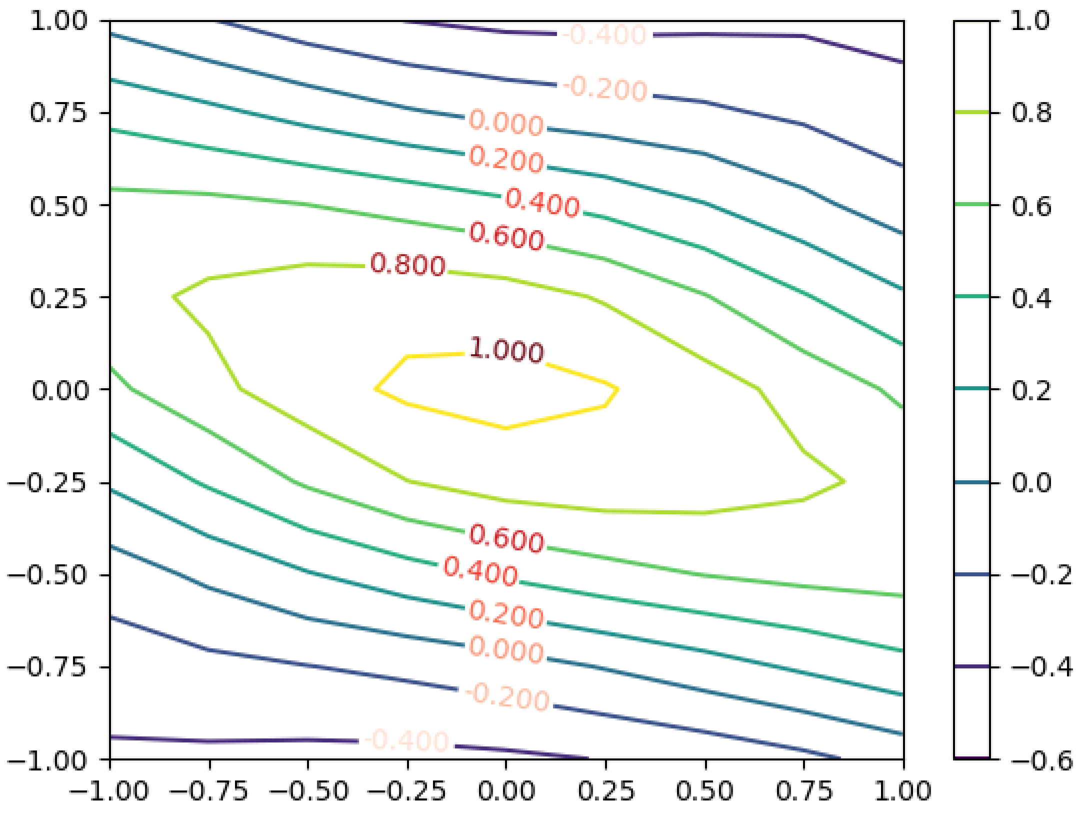
Members: Krystian Wojcicki, Michael Kuang

1. a)

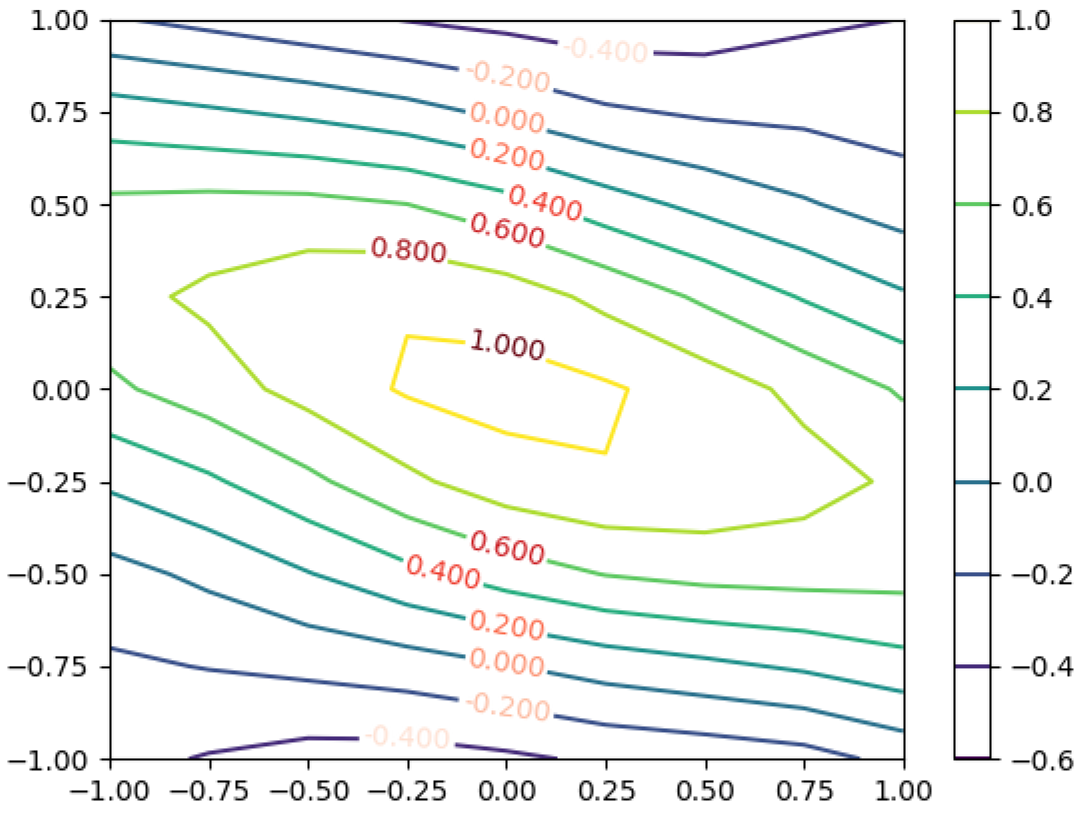
Running `python q1.py a` we get the following graphs. The networks ran for 2000 epochs with convergence at MSE of training = 0.02. The average of 10 runs was taken.



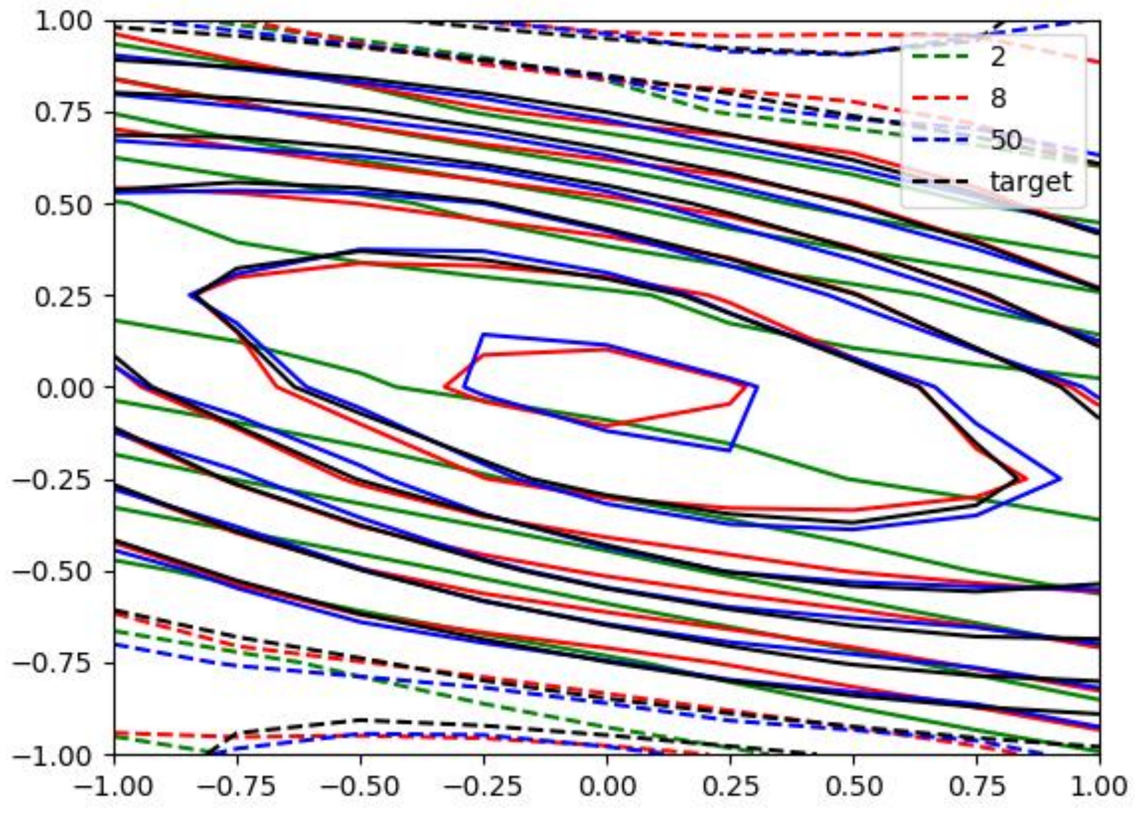
Contour generated by network with 2 hidden layer neurons, with x on the x axis and y on the y axis



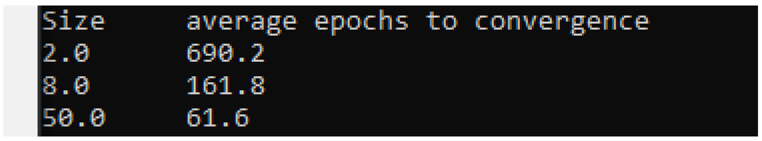
Contour generated by network with 8 hidden layer neurons, with x on the x axis and y on the y axis



Contour generated by network with 50 hidden layer neurons, with x on the x axis and y on the y axis



Reproduction of fig 3 from the assignment, with x on the x axis and y on the y axis



```
Size      average epochs to convergence
2.0       690.2
8.0       161.8
50.0      61.6
```

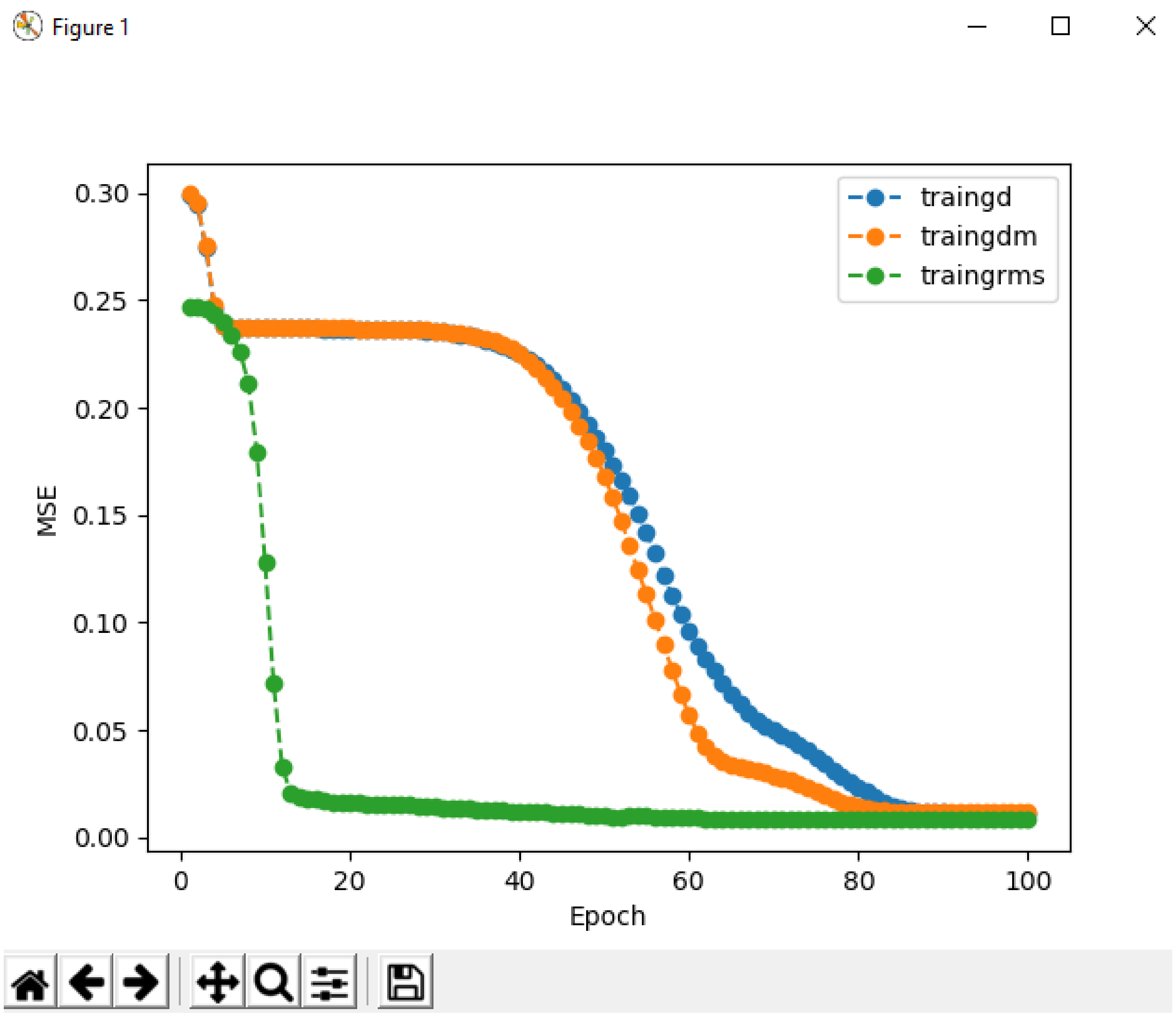
Table showing hidden layer size and average convergence

From the average epochs to converge its easy to see that the higher the # of neurons the quicker the convergence but with a higher number of neurons over fitting seems to be a problem as seen in the reproduction of fig 3. Indicating that 8 neurons is indeed the optimal number of neurons.

The hyperbolic tangent activation function was used.

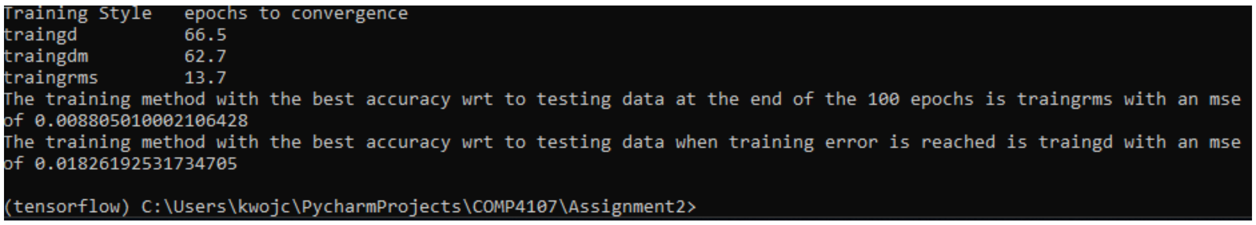
b)

Running `python q1.py b` . The networks with different training styles were trained for 100 epochs with 10 runs per training style of network.



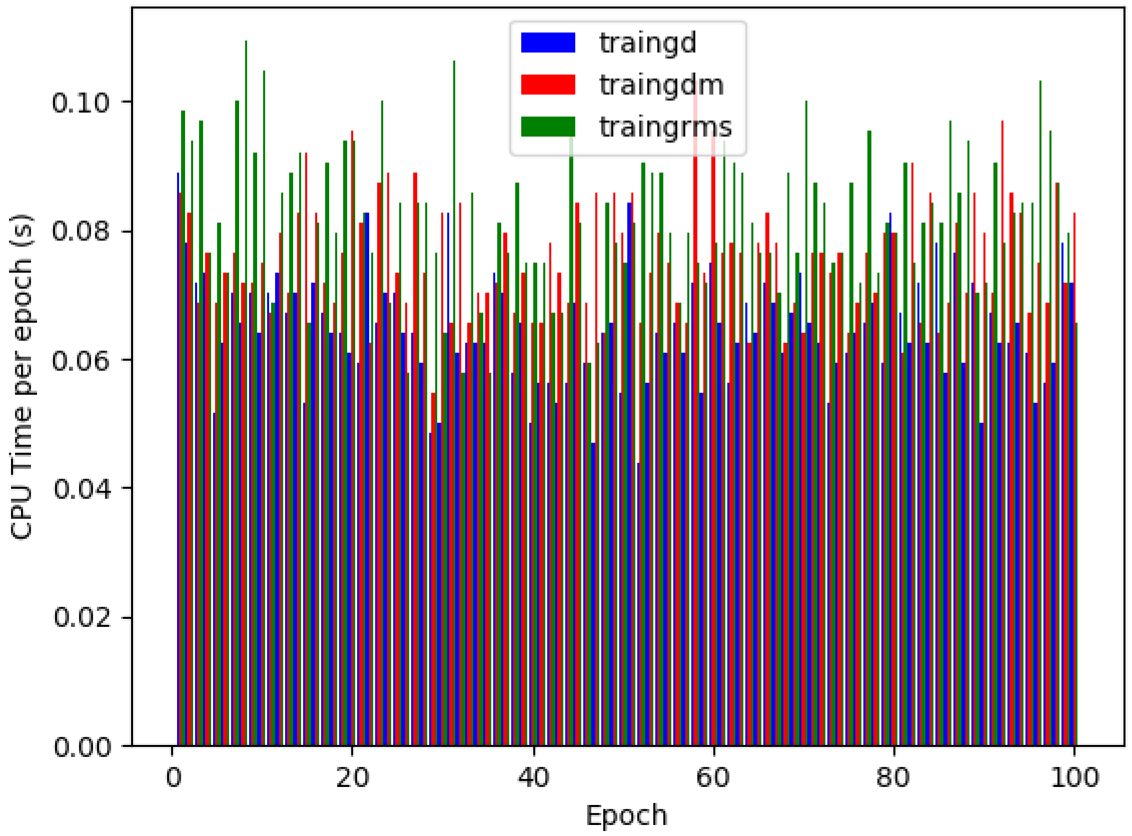
Graph showing average MSE and epoch

From the above graph we can see that the RMSPropOptimzer did converge the fastest, with MomentumOptimizer slightly quicker than GradientDescentOptimizer. As confirmed by the following table



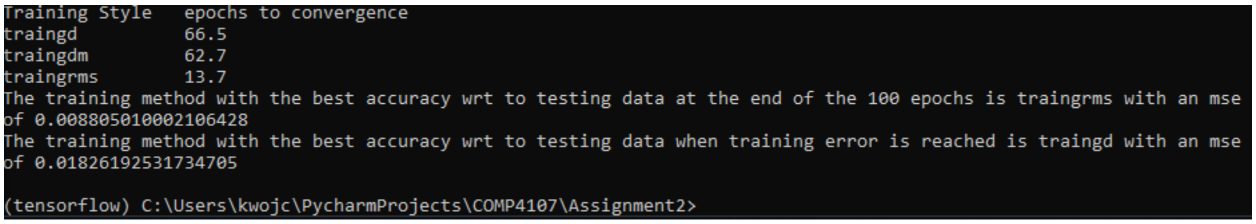
```
Training Style   epochs to convergence
traingd          66.5
traingdm         62.7
traingrms        13.7
The training method with the best accuracy wrt to testing data at the end of the 100 epochs is traingrms with an mse
of 0.008805010002106428
The training method with the best accuracy wrt to testing data when training error is reached is traingd with an mse
of 0.01826192531734705

(tensorflow) C:\Users\kwojc\PycharmProjects\COMP4107\Assignment2>
```

Table showing convergence with rms the fastest followed by dm, d

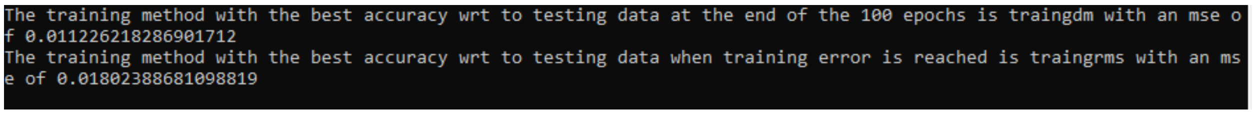


Graph showing average CPU time per epoch

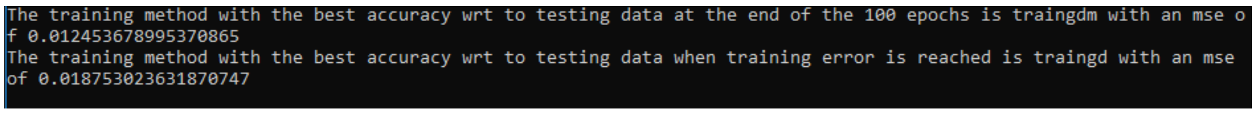
The CPU time per method fluctuated greatly but overall the more complex the training method the longer it took per epoch.



```
Training Style   epochs to convergence
traingd          66.5
traingdm         62.7
traingrms        13.7
The training method with the best accuracy wrt to testing data at the end of the 100 epochs is traingrms with an mse
of 0.008805010002106428
The training method with the best accuracy wrt to testing data when training error is reached is traingd with an mse
of 0.01826192531734705

(tensorflow) C:\Users\kwojc\PycharmProjects\COMP4107\Assignment2>
```



```
The training method with the best accuracy wrt to testing data at the end of the 100 epochs is traingdm with an mse o
f 0.011226218286901712
The training method with the best accuracy wrt to testing data when training error is reached is traingrms with an ms
e of 0.01802388681098819
```



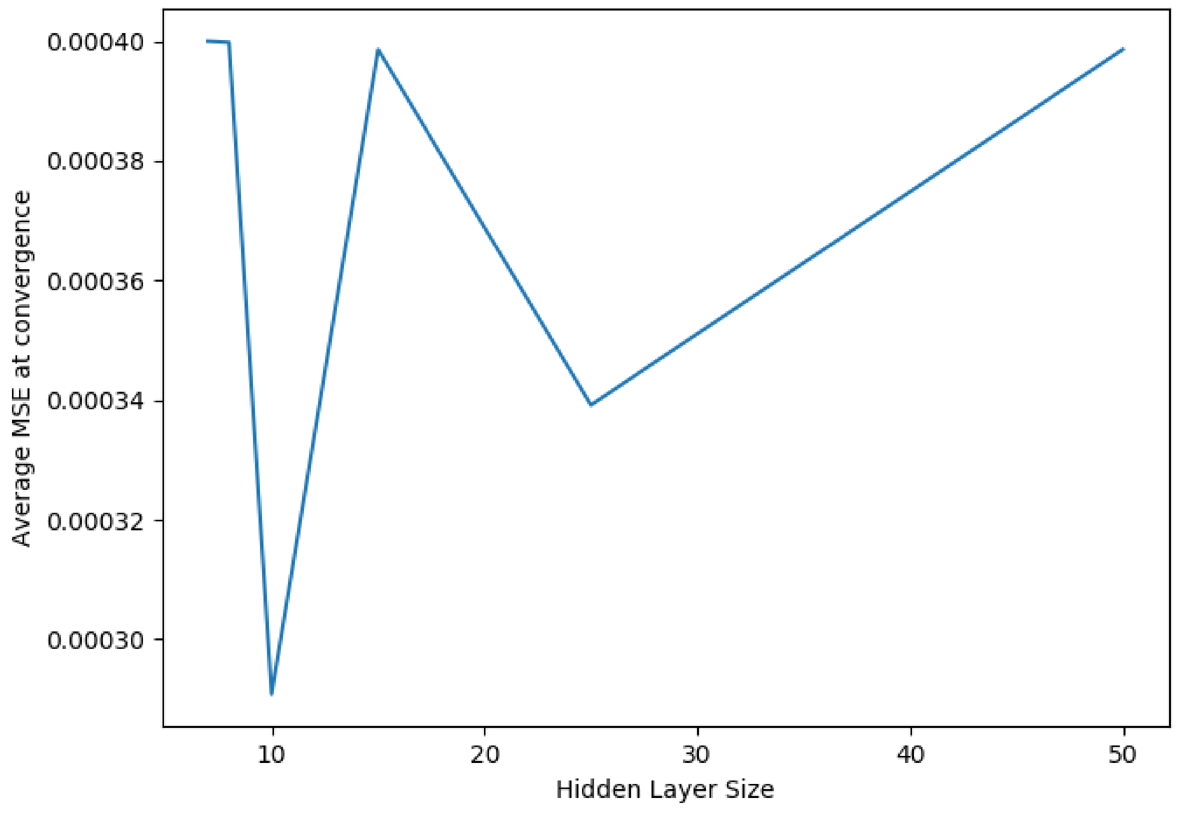
```
The training method with the best accuracy wrt to testing data at the end of the 100 epochs is traingdm with an mse o
f 0.012453678995370865
The training method with the best accuracy wrt to testing data when training error is reached is traingd with an mse
of 0.018753023631870747
```

Showing which method has the best accuracy

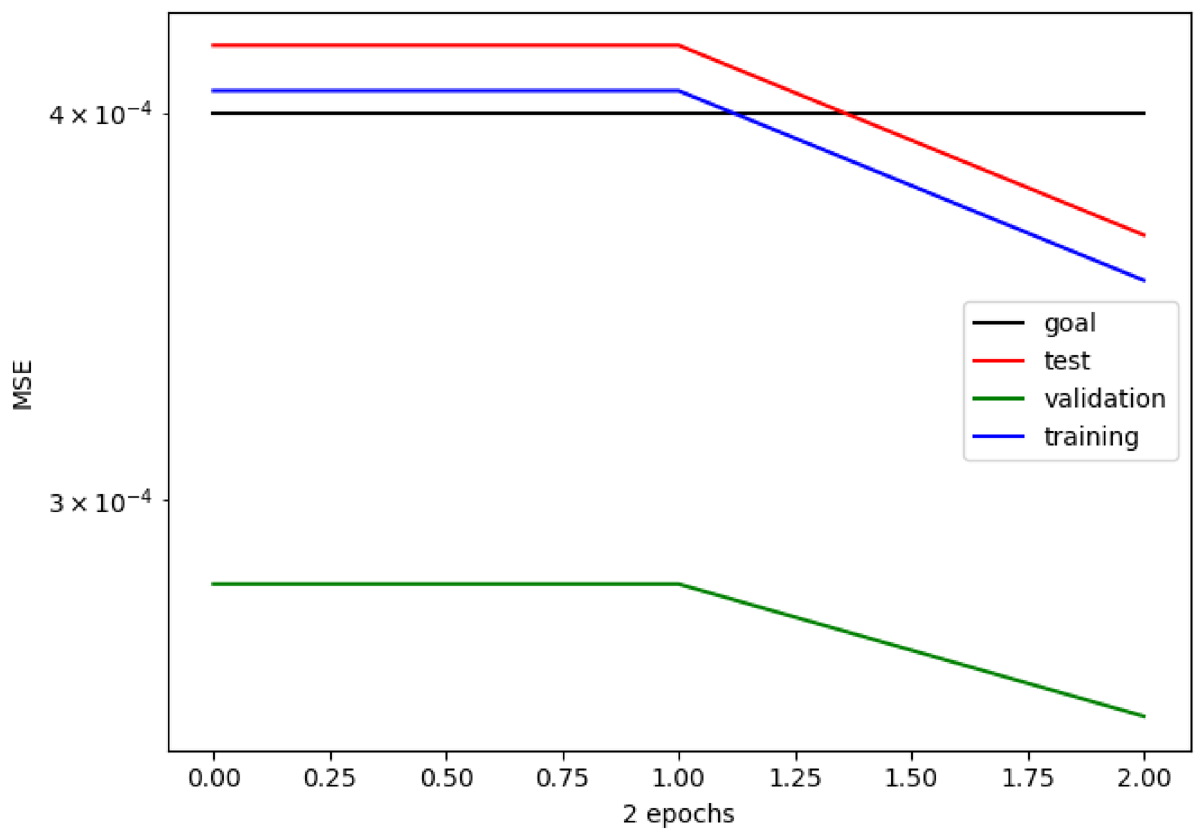
After 100 epochs typically traingrms or traingdm had the best accuracy but sometimes traingd did have the better accuracy. At convergence traingd usually had the best accuracy. This seems to make sense as the more complex training mechanisms converge quicker but do large leaps causing it to potentially over shoot the more optimal solution.

c)

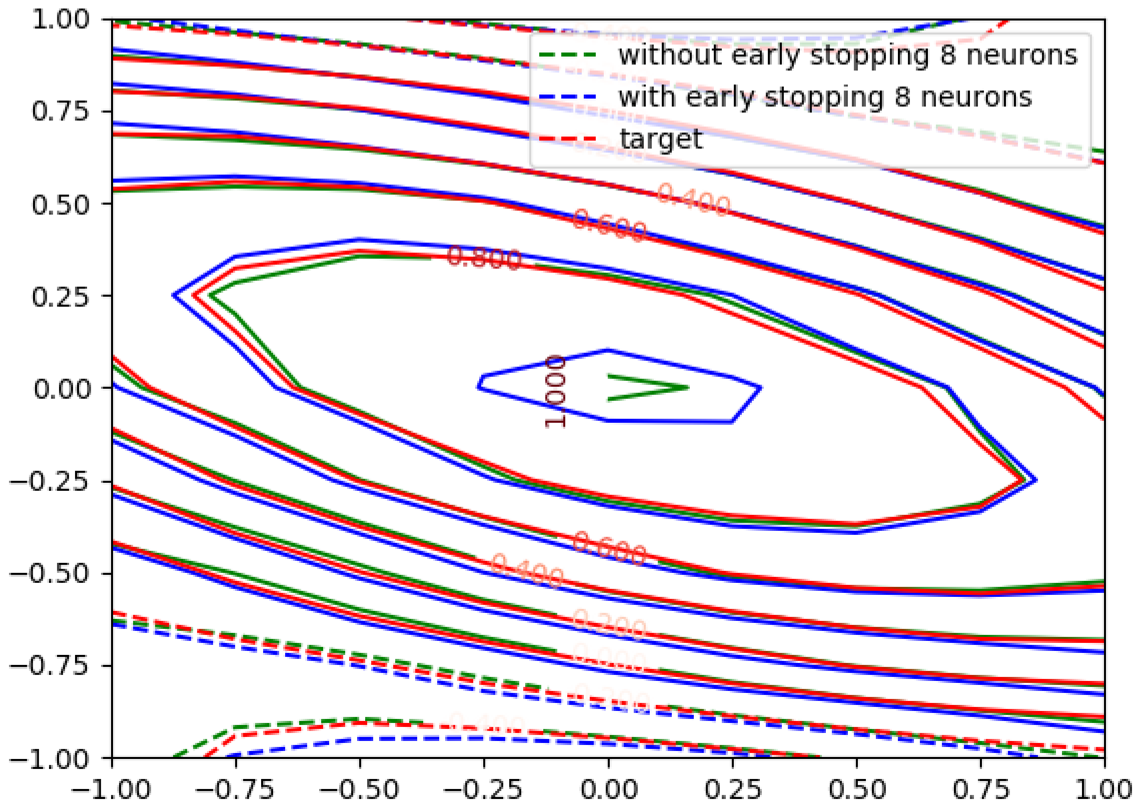
Running `python q1.py c`. 10 experiments were run for getting the average MSE at convergence for different hidden layer sizes. Anything below 7 neurons took extremely long to converge.



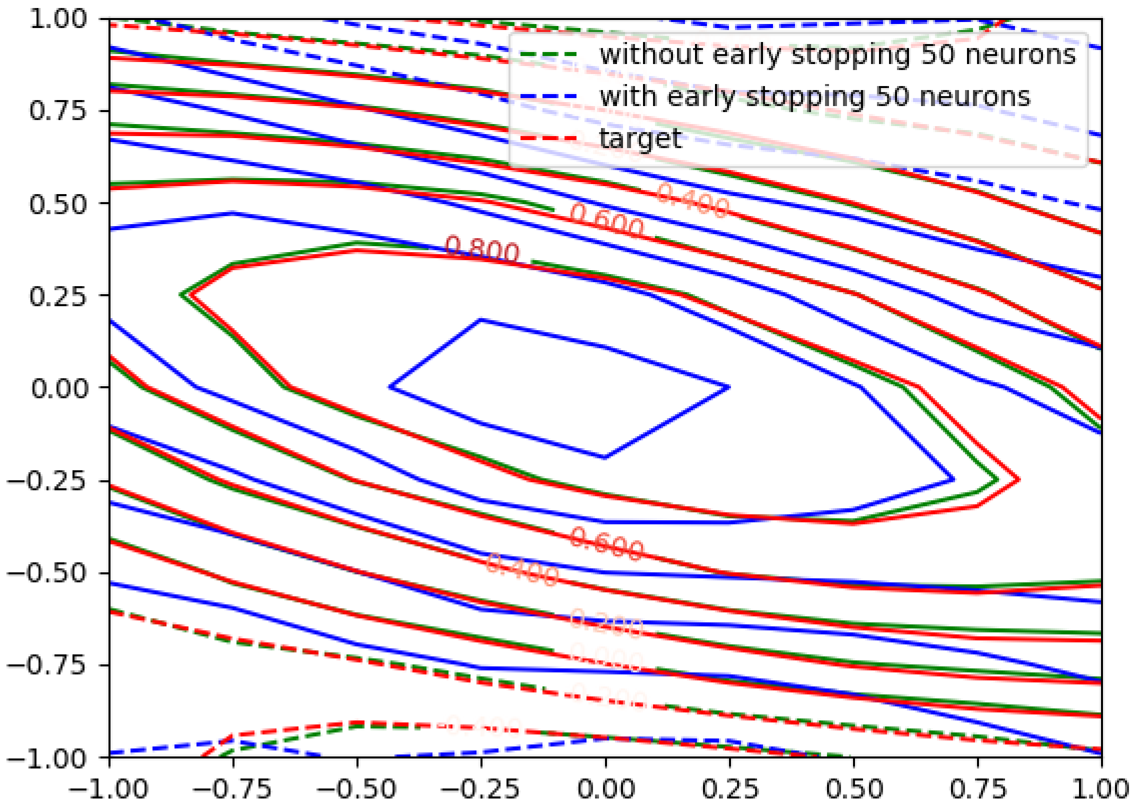
Average MSE vs hidden layer size at convergence



Replication of fig 6 using 8 neurons



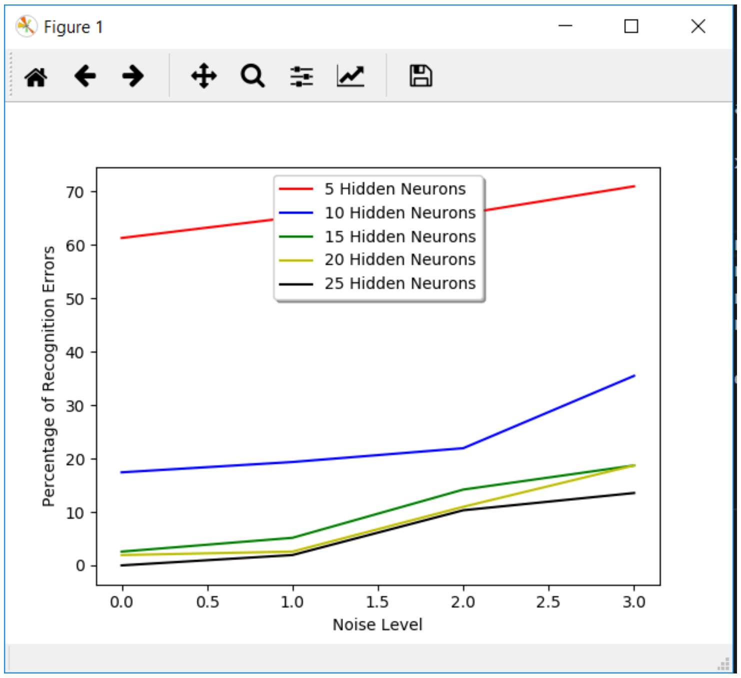
Replication of fig 7 using 8 neurons



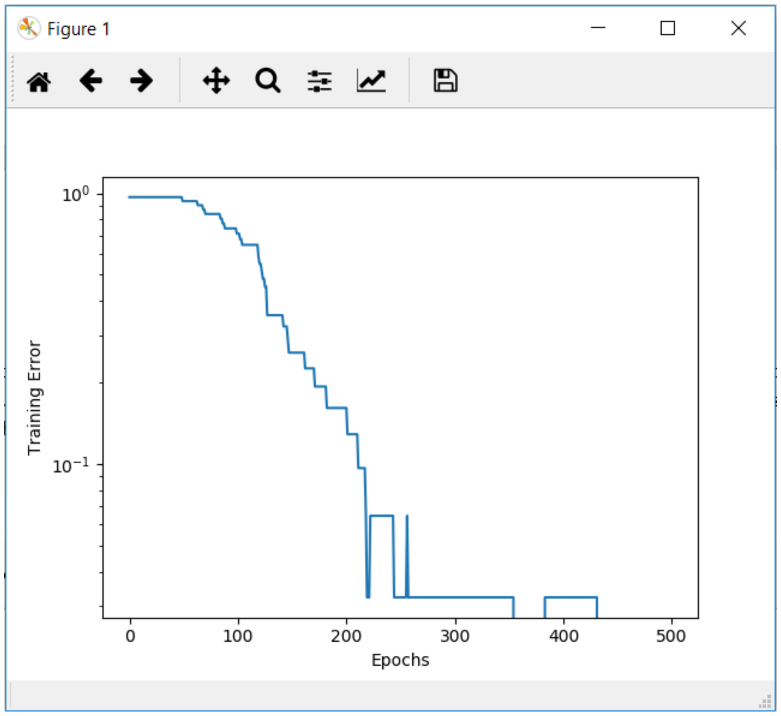
Replication of fig 7 using 50 neurons

As seen by the replication of fig 7 using 8 and 50 neurons. 8 neurons are the optimal choice, this can be seen as the without early stopping and with early stopping networks of 50 neurons do not work equally well. But with 8 neurons both the early stopped and without early stopped networks work equally well.

2. a) The first part of q2.py, we ran experiments with hidden neuron numbers in range of 5-25 with increments of 5 for 300 epochs each. The graph below shows the percentage of recognition error for different noise levels and we can see that generally, with each increment of 5 neurons, the percentage of recognition errors decreases. From this we can see that 25 hidden neurons will likely to produce the best accuracy and is the most optimal choice.

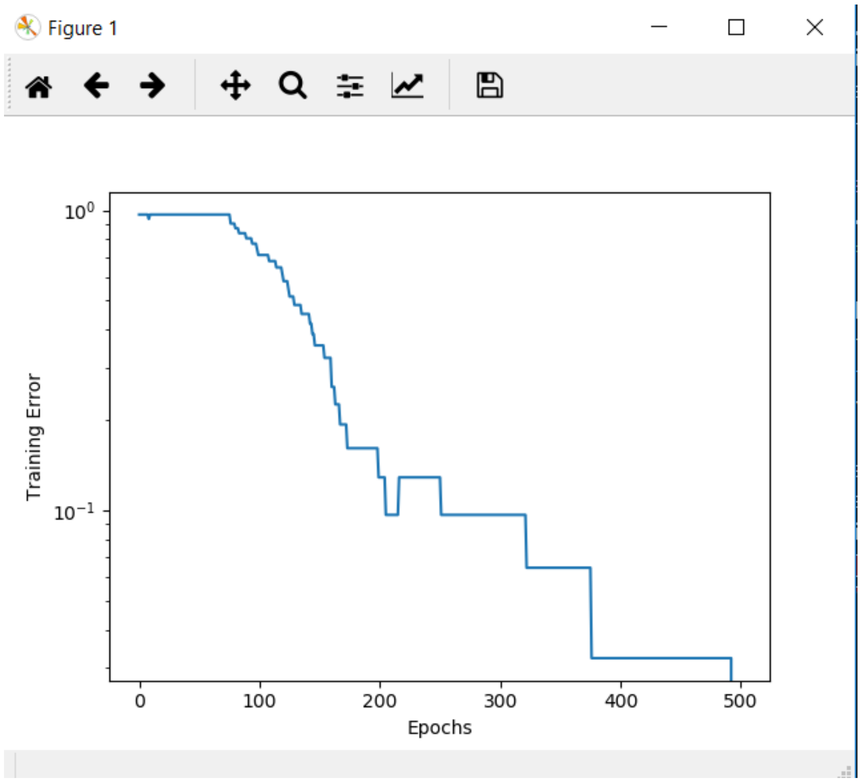


b) In the following graph, we can confirm that Fig.13 from the assignment is a reasonable representation of performance for the optimal number of hidden layer neurons being 25. We can see that at the first hundred epochs, the model does not learn much, then there are a couple sudden drops from approximately 100 to 200 epochs and it flattens out at the end. This makes sense as the network is trained on the ideal data with zero noise. The network should start off at a very high error rate as it should not recognize anything. As more epochs pass, the model should train to the point where it should have very low error rate as shown below.



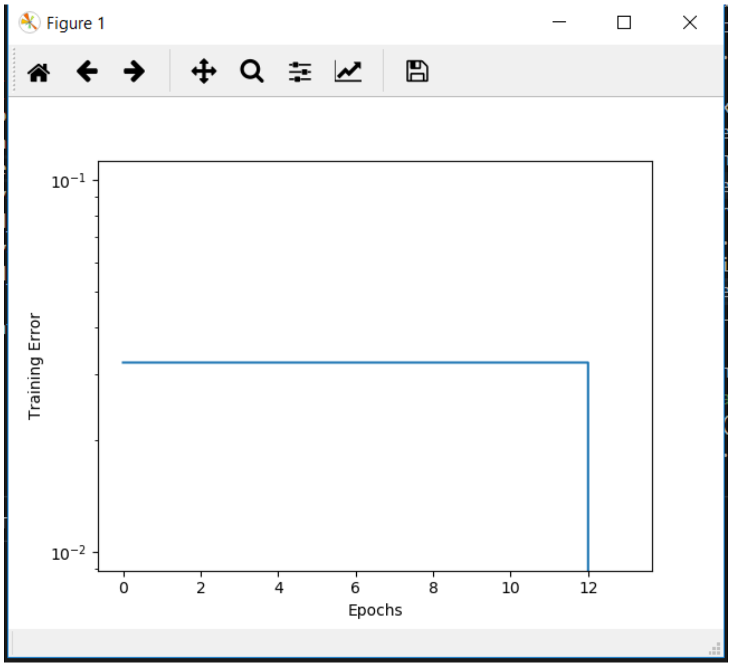
Graph trained with a hidden layer of 25 hidden neurons

If we compare this with a network trained on 15 hidden neurons, we can see that with 25, it is able to learn much faster. With approximately 200 epochs, it reaches around 3% error while using a hidden layer of 15 hidden neurons, it takes about 375 epochs.

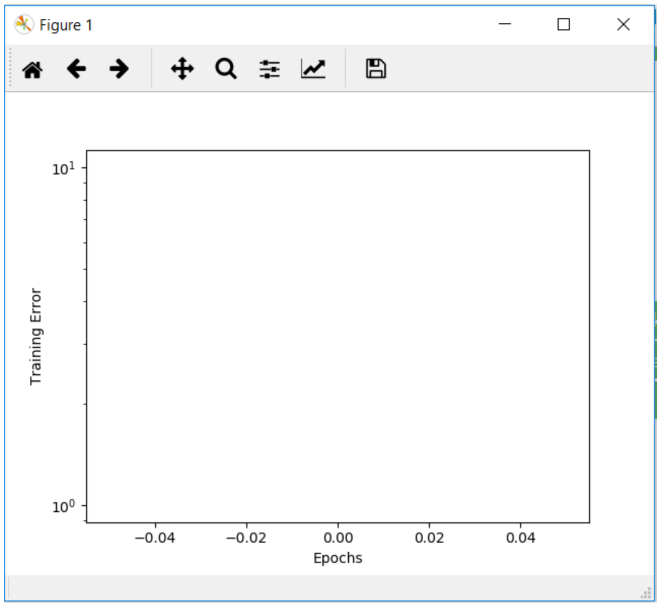


Graph trained with a hidden layer of 15 hidden neurons

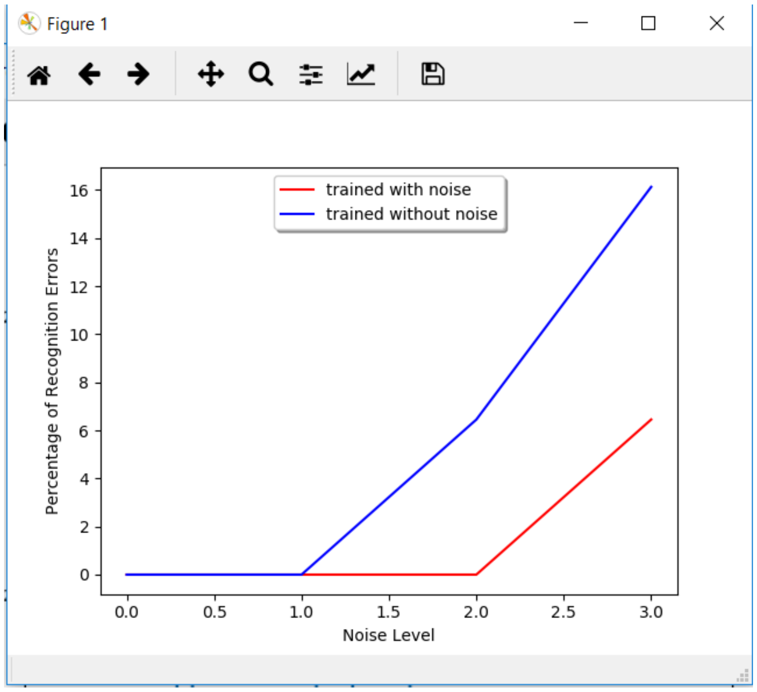
The figure below is a graph of training error at each epoch after we have trained on the ideal set and noisy data. The graph confirms that after training on the noisy data, it forgot a bit of the noiseless data and as such the error at the beginning is a bit higher, then it drops after training on the noiseless data again.



Please note that there are cases when this graph will show nothing (example below), this is because sometimes the network is able to retain its understanding of the ideal targets with zero noise even after training on the noisy data.



2 c) As seen in the graph below, we can produce the recognition accuracy shown in Fig. 14. In general, the model trained with noise outperforms the model that was trained without noise.



Notes:

To run the various parts of q1 do `python q1.py [part letter] `

Q2-patterns.txt as provided on the form.

Tony mentioned in an email for part 1c 2) he may change it to plot accuracy as of handing in the assignment the assignment still asked for plotting MSE.

Part (c). You must:

1. Confirm that approximately 8 neurons are a good choice for the current problem.
2. Run experiments across a range of hidden layer sizes and plot MSE for testing set at convergence against hidden layer size.
3. Reproduce (approximately) Fig. 6 and Fig. 7. Note: the epochs axis in Fig. 6 represent the 2 epochs around convergence, not epochs 0-2 of a run.