COMP 4107 Assignment #2

Due: 23rd February 2020 at 11:55 pm

Name: Kirk Zhen
Student ID: 101087006

Question 1 [50 marks]:

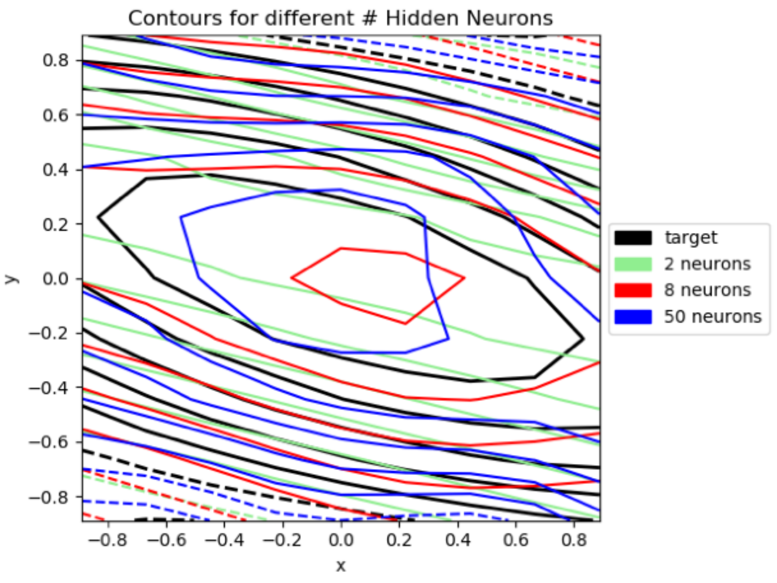
Part (a):

1. Using the *GradientDescentOptimizer*, investigate the converged performance of a 2 layer neural network with 2, 8 and 50 hidden layer neurons.

Number of Neurons	MSE
2	0.014267435
8	0.012023145
50	0.016543256

Note: The data in the table are average value across 10 experiments

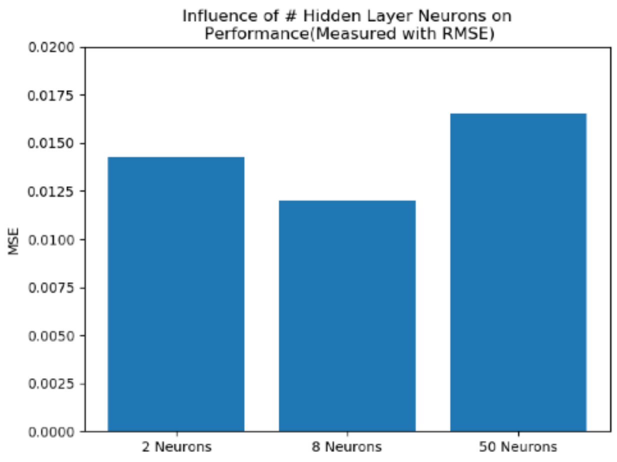
2. You must produce a contour diagram similar to Fig.3.

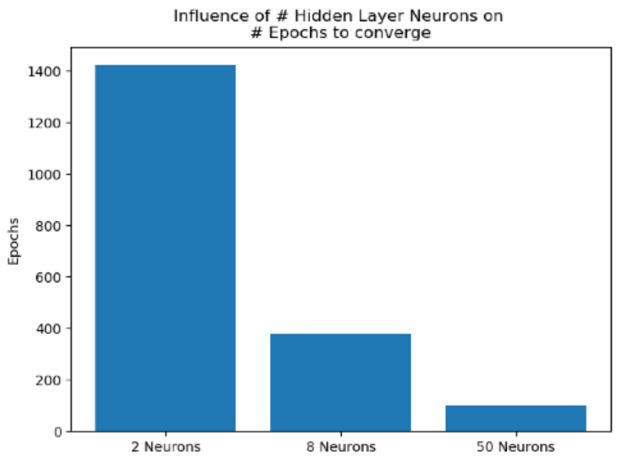


3. Include a table of MSE for the 3 different network sizes and number of epochs to convergence.

Number of Neurons	**MSE**	**Epochs to converge**
2	0.014267435	1423.5
8	0.012023145	378.3
50	0.016543256	102.3

Note: The data in the table are average value across 10 experiments





4. Indicate whether sigmoid of hyperbolic tangent activation functions were used in your experiments.

I used hyperbolic tangent function as activation functions in my experiments.

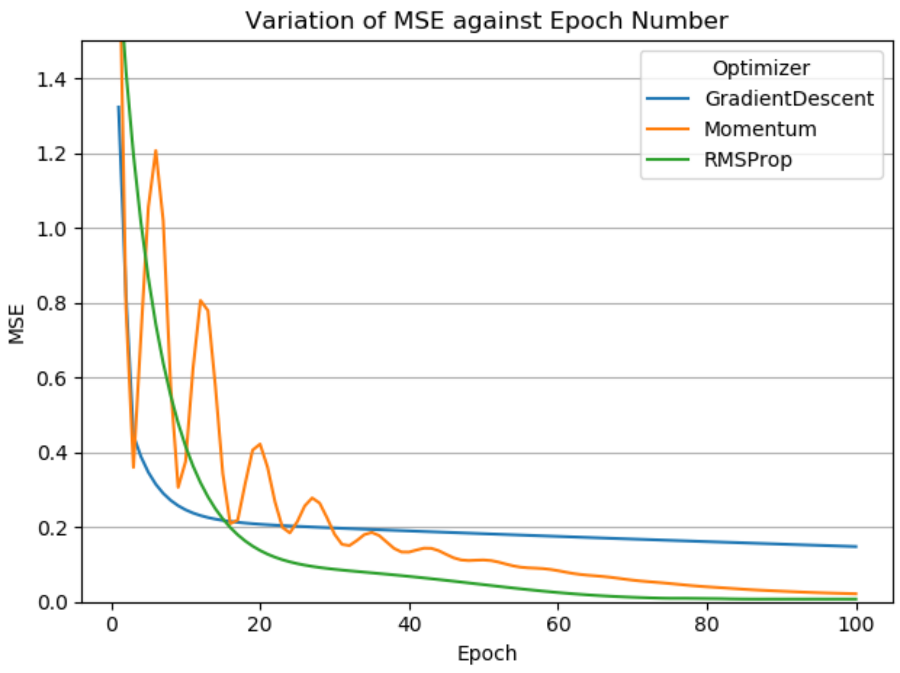
Part (b).

1. include a table containing the appropriate number of epochs for convergence for each training method

Training Method	**Epochs to converge**
GradientDescentOptimizer	605.0
MomentumOptimizer	116.7
RMSPropOptimizer	93.4

Note: The data in the table are average value across 10 experiments

2. Plot the variation of MSE against epoch number for each of the 3 methods for 1-100 epochs.



3. Plot a bar chart of the CPU time taken per epoch for each of the 3 methods.

This Table below and Fig.1 are obtain by setting tolerance to 0.02, note that the Y axis start from 180000 nano-second/epoch.

The Fig.2 is time cost for each first 100 epoch tested on each of the 3 optimizers.

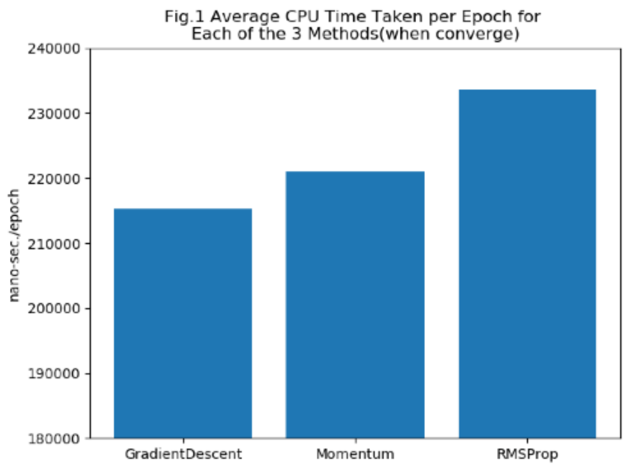
As indicated in Fig.2, if set the maximum epoch to 100, the time cost per for *MomentumOptimizer* before epoch 70 would be relatively higher. This is because the MSE of *MomentumOptimizer* fluctuate in the earlier epochs, and this is a little bit expensive.

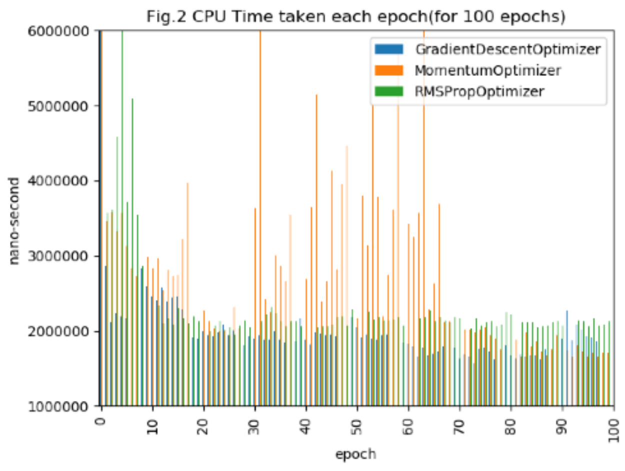
When epoch grows above 70, the time cost per epoch for *MomentumOptimizer* got decreased, and the time cost for *RMSPropOptimizer* is slightly more expensive than the other two optimizer. And when converge, the average CPU Time taken for *RMSPropOptimizer* is the most expensive(shown in Fig.1). But the variance between average time cost of the 3 method is relatively small.

And according to my other experiment(not shown in the report), if we set a smaller tolerance(e.g. 0.01, 0.005, 0.0025), the variance between average time cost of the three optimizer would be bigger.

Training Method	**Time Cost per Epoch(nano-second)**
GradientDescentOptimizer	215255
MomentumOptimizer	220936
RMSPropOptimizer	233615

Note: The data in the table are average value across 10 experiments



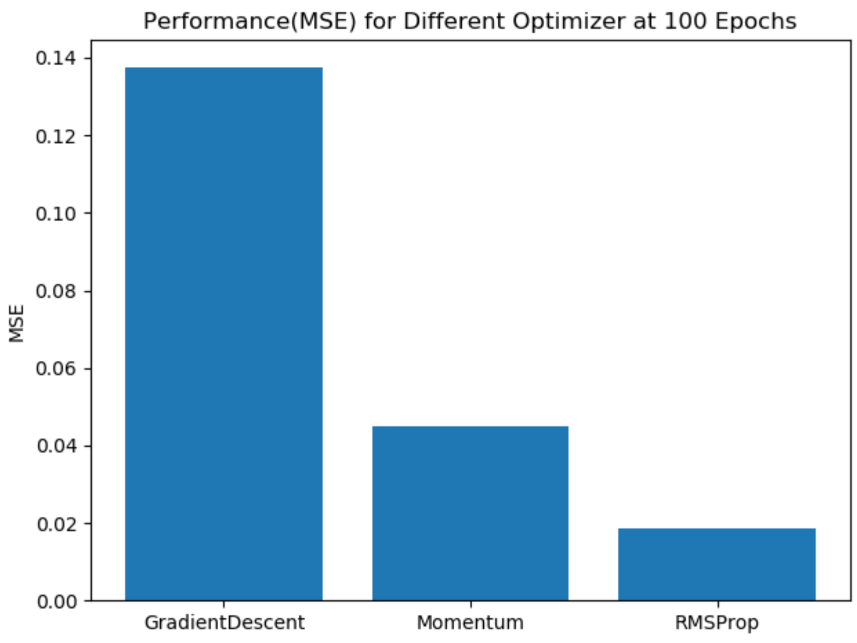


4. Which method provides the best accuracy at the end of 100 epochs of training?

We can see from the table and bar chart, *RMSPropOptimizer* achieves the lowest MSE at the end of 100 epochs among the 3 optimizers. We conclude that, the *RMSPropOptimizer* provides the best accuracy.

Training Method	**Performance(MSE)**
GradientDescentOptimizer	0.1376
MomentumOptimizer	0.0451
RMSPropOptimizer	0.0184

Note: The data in the table are average value across 10 experiments

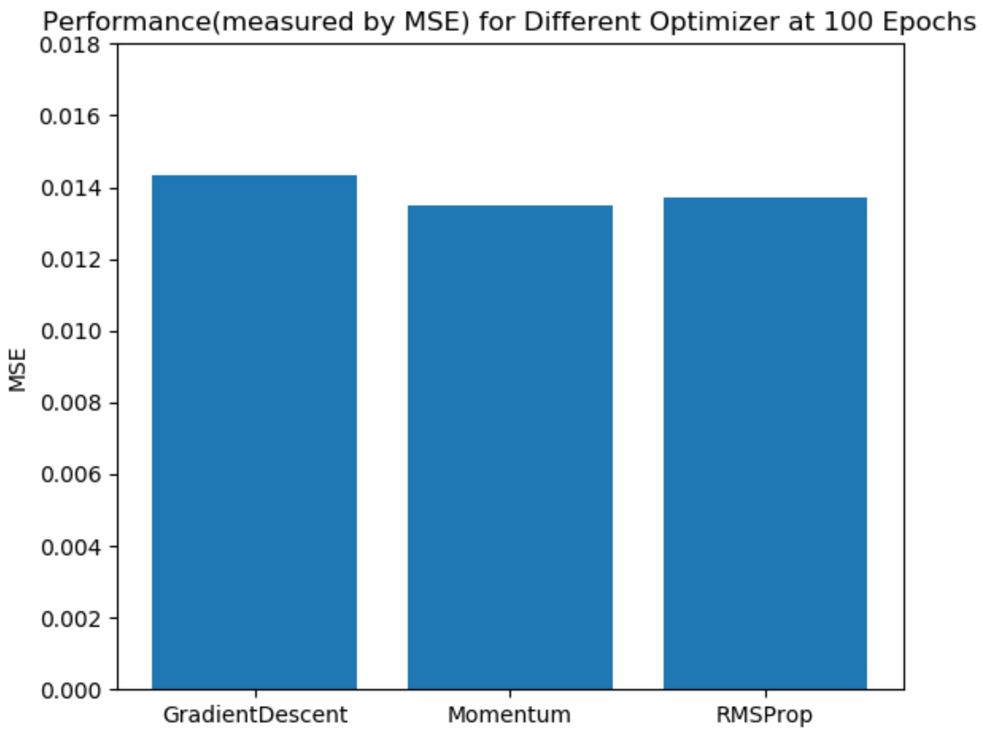


5. Which method is the most accurate when the training error is reached?

According to the table and Bar Chart, *MomentumOptimizer* achieves the lowest MSE when training error is reached. We conclude that *MomentumOptimizer* provides the best accuracy. But the variance between the 3 method is relatively small.

Training Method	**Performance(MSE)**
GradientDescentOptimizer	0.01433
MomentumOptimizer	0.01349
RMSPropOptimizer	0.01372

Note: The data in the table are average value across 10 experiments

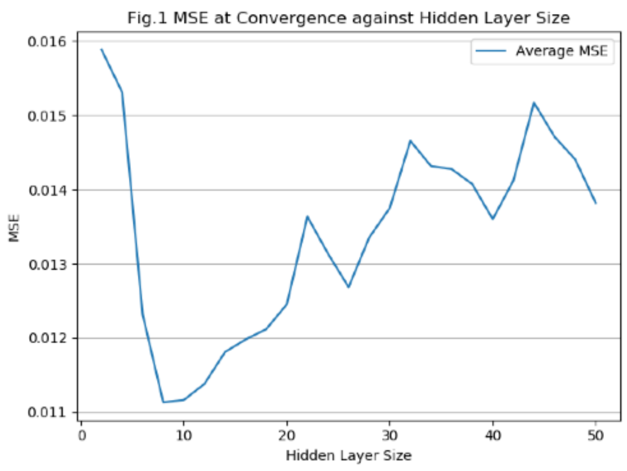


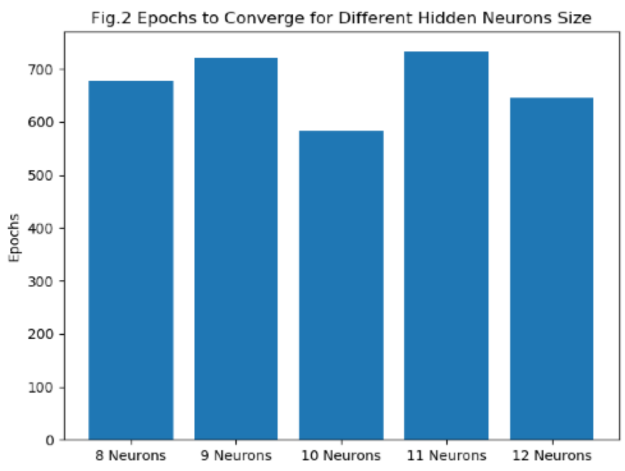
Part (c).

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.

These two question would be answered together, because question c.2 can give evidence to prove question c.1.
According to the results from the previous problems, there's reason to believe that the increase of hidden neurons size could decrease the epochs to converge. And among 2, 8, 50 hidden neurons size, we found that 8 neurons provides a best performance(high accuracy, low MSE). In order to find out whether 8 neurons are a "good choice " for this problem, more hidden layer sizes would be experimented.
In Part(c) q2, I run experiments across a range (in range(2,51,2)) of hidden layer sizes, to see how the MSE change along with hidden layer size increase. We can see from Fig.1 for Part(c) q2, the MSE for neurons size lower than 8 are relatively high, this is because of underfitting. As the size of hidden neurons increases, MSE decrease to a low level when hidden size between 8 and 12, after that, the MSE increase rapidly and become unstable along with the hidden neurons size because of overfitting.
Given that hidden neuron size between 8-12 provides a good performance, we test the epochs needed to convergence for 8-12 hidden layer size. The result is show as in Fig.2. We can see from figure 2, the average epochs to converge for size 8-12 varies but are almost at the same level. So, I judge that hidden neuron size 8-12 are all good to handle this problem.
Combing the result in Part(c) q3, the early stopping is not triggered during the training, this indicate that size 8 is good for the problem.

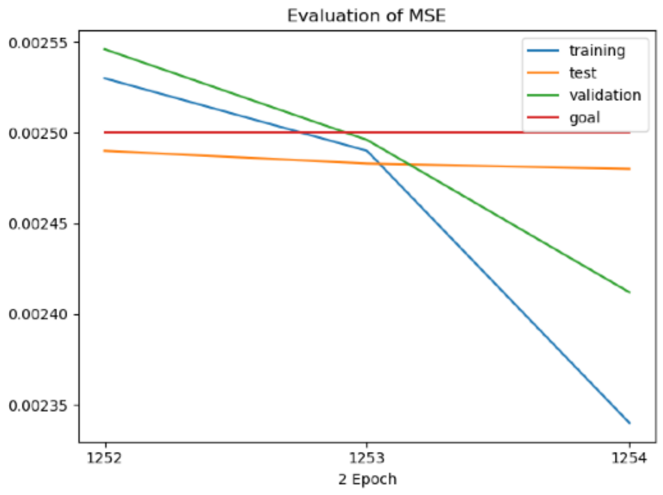
(Note: I just take 10 experiments and take average value across them. But this has already taken me over 40 minute to get the data demanded to plot Fig.1, so I was not able to increase the experiment size to get a more accurate data. But I believe what I have now is enough to show that 8 neurons are a good choice for the current problem)

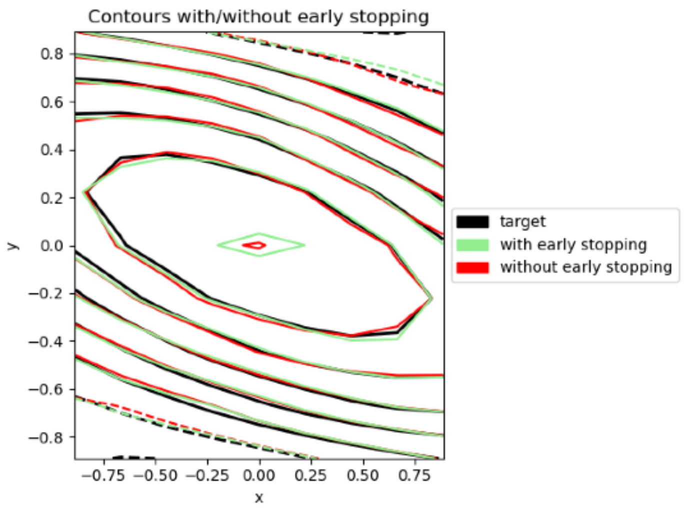




3. Reproduce (approximately) Fig. 6 and Fig. 7. Note: the epochs axis in Fig. 6 represent the 2 epochs around convergence, not epochs 0-2 of a run.

From the reproduction of Fig. 7 (right figure), we can see that the network with and without early stopping performs almost equally well. As indicated by the reproduction of Fig. 6 (left one), the early stopping is not triggered during the training. This is because the validation error keeps decreasing during the whole training process.



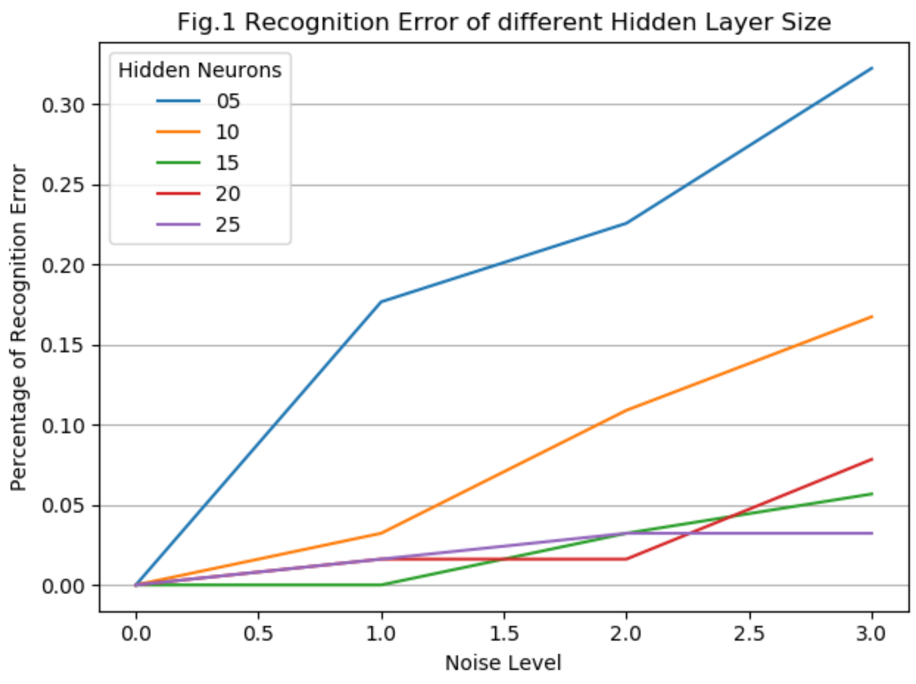


Question 2 [50 marks]

Part (a).

1. Run experiments with hidden neuron numbers in the range 5-25.
2. Plot a chart of recognition error against the number of hidden neurons.

Fig.1 shows the percentage of recognition error for different size of hidden layer. We can see from the graph, neuron size 15, 20, 25 performs relatively better than neuron size 5 and 10.



Part (b).

1. Confirm that Fig.13 is a reasonable representation of performance for the optimal number of hidden layer neurons chosen.
2. Plot a chart in support of (1)

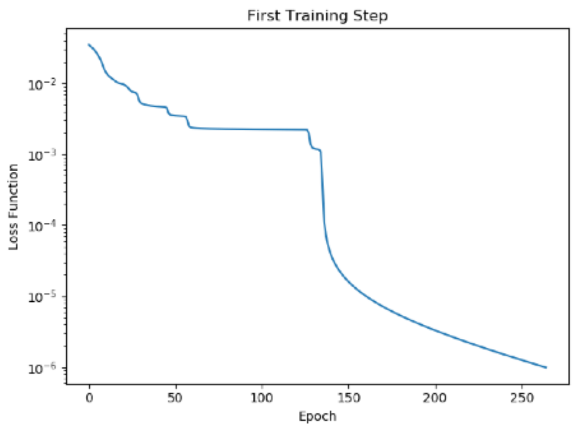
The following two figure are produced using 15 hidden layer size.

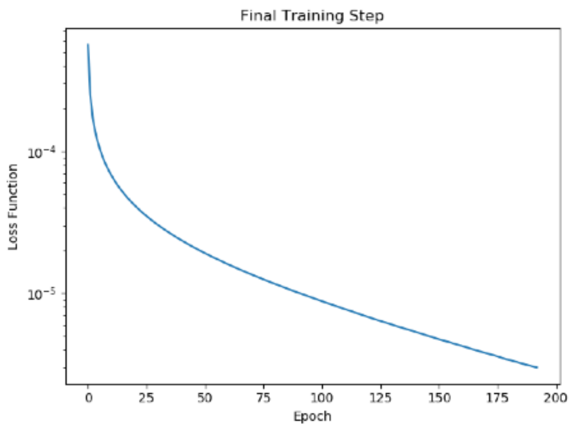
According to the specification of Question 2, the first step and final step in the training process train the network on the ideal data for zero decision errors.

After the first step terminate, noisy data would be fed in the network. This makes the starting cost in step 3 relatively higher than the ending cost in step 1. So, as shown in the following two graph, in the final step, the cost function start at a higher value than in the ending cost inn first step.

Because the hyperparameter in the two step are the same, starting at a lower cost than in first step, the final step takes less epochs to converge. Because the condition of termination are also the same(terminate when zero decision errors), the two cost curve converge at almost same value.

In both step, the cost decrease as epoch grows.

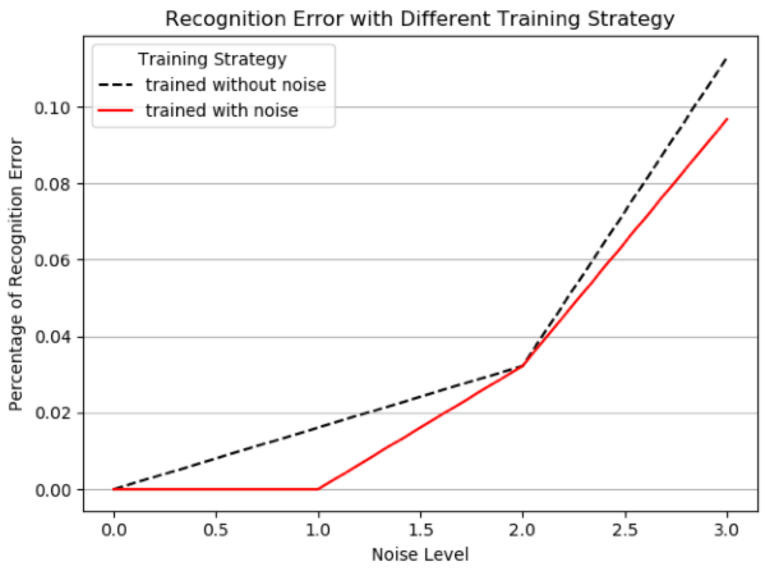




Part (c).

1. Create testing data that has between 0 and 3 bits noise.
2. Confirm that you can produce the recognition accuracy shown in Fig. 14.

Here is my reproduction of Fig.14. The figure indicate that, the network trained with noise provide a higher recognition accuracy in compared with that trained without noise, when the testing data was distorted.


Recognition Error with Different Training Strategy
Training Strategy
trained without noise
trained with noise
Percentage of Recognition Error
Noise Level