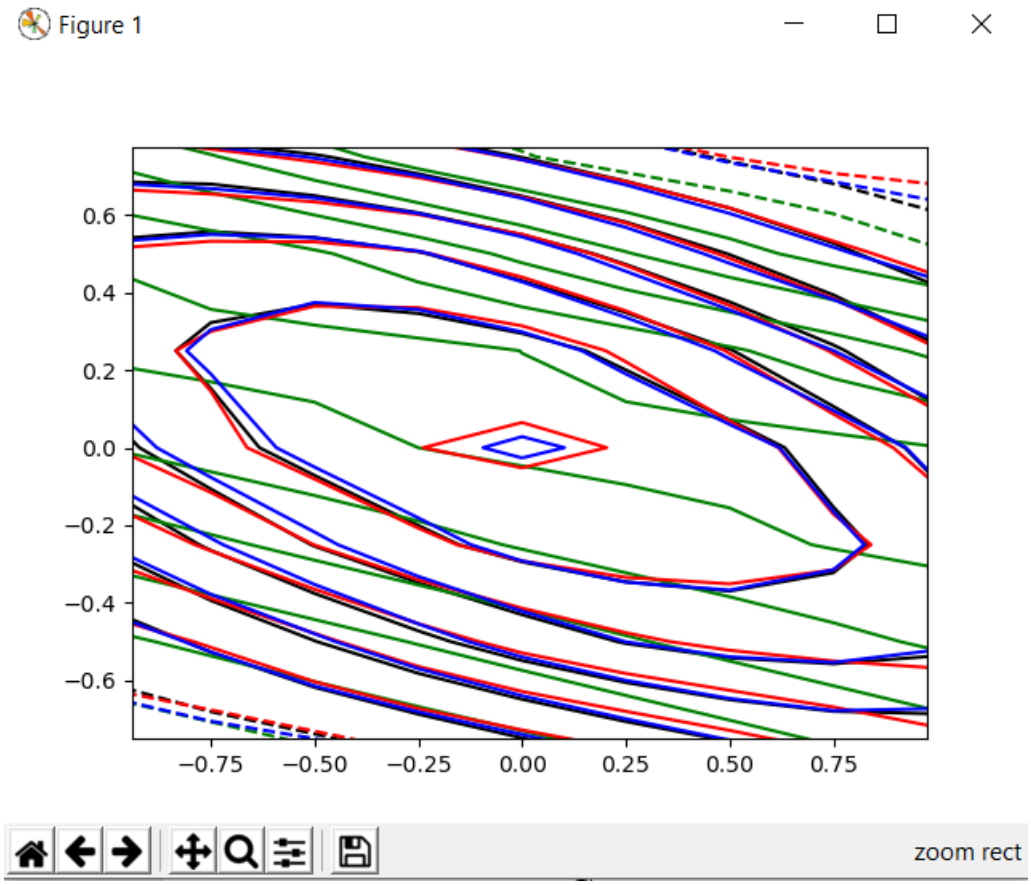
Question 1:

- As specified the assignment, 3 data sets will be used for the question 1: The training set, the testing set and the validation set. The generation of data sets can be found in generate_data (testing and training) and generate_data_randomly (validation) functions.

a.

1. The program uses gradient descent optimizer with the learning rate of 0.05. The MSE we want to approach is MSE = 0.02. The standard deviation of the variable in the model is 0.7 and the batch size is 25.
   Because the result strongly depends on the training set, therefore, sometime the program generates some un-nice graph. But mostly, it will generate the graph looks as the graph below when the 2 neurons approaches to the convergence.
   The standard is drawn in black line. The green is presented for 2 neurons, red is for 8 neurons and the 50 neurons is presented as blue color.
2. 

3. The table is recorded respectively 3 number of epochs to convergence (just to make sure that the convergence is achieved)

   The range for number of epochs is (0,2000) and the message: “The current MSE from the train data” is printed out when the number of epochs % 300 == 0. That means there will be 6 messages will be printed out as pictures below.

The current table for each individual neuron will be printed out as the record of the first 3 number of epochs reaching the convergence.

For 2 neurons:

```
The current MSE from the train data  0.62263894
The current MSE from the train data  0.08602825
The current MSE from the train data  0.035791133
The current MSE from the train data  0.022830538
Converged at epoch 1153 with MSE 0.019998498
Converged at epoch 1154 with MSE 0.019989207
Converged at epoch 1155 with MSE 0.019979935
The current MSE from the train data  0.019581363
The current MSE from the train data  0.017869476
The current MSE from the train data  0.016902968

 The table is defined as
[[0.019950444, 0.019942224, 0.01993402], [1153, 1154, 1155]]
```

For 8 neurons:

```
calculating MSE for  8  neurons
The current MSE from the train data  0.40559566
Converged at epoch 123 with MSE 0.01990174
Converged at epoch 124 with MSE 0.019668886
Converged at epoch 125 with MSE 0.019444128
The current MSE from the train data  0.0073204297
The current MSE from the train data  0.004000942
The current MSE from the train data  0.0022574605
The current MSE from the train data  0.001440037
The current MSE from the train data  0.0011075482
The current MSE from the train data  0.0009454762

 The table is defined as
[[0.015790934, 0.015606368, 0.015428786], [123, 124, 125]]
```

For 50 neurons:

```
calculating MSE for  50  neurons
The current MSE from the train data  0.6505955
Converged at epoch 28 with MSE 0.019217528
Converged at epoch 29 with MSE 0.018495841
Converged at epoch 30 with MSE 0.017851558
The current MSE from the train data  0.0017881187
The current MSE from the train data  0.0009981488
The current MSE from the train data  0.0008095091
The current MSE from the train data  0.0006970862
The current MSE from the train data  0.0006182955
The current MSE from the train data  0.0005593872

 The table is defined as
[[0.014646829, 0.014112836, 0.013636975], [28, 29, 30]]
```

4. As the requirements of the assignment, the activation function is the hyperbolic tangent activation function: “tf.nn.tanh”

b. There will be 3 different forms of training in this part:
Traingd: *GradientDescentOptimizer*
traingdm: *MomentumOptimizer*
traingrms: *RMSPropOptimizer*
The standard learning rate for this part is 0.02, the range of epochs is 100, MSE is still be the same the previous part 0.02, the batch size is 5.

1. Here is one sample of the experiment:
A table for the gradient descent optimizer:

```
The current MSE from the train data  0.38926476
The current MSE from the train data  0.14441246
The current MSE from the train data  0.037465286
The convergence is reached at epoch 51 with MSE 0.018992307
Converged at epoch 51 with MSE 0.018992307
The convergence is reached at epoch 52 with MSE 0.018089337
Converged at epoch 52 with MSE 0.018089337
The convergence is reached at epoch 53 with MSE 0.01726855
Converged at epoch 53 with MSE 0.01726855
The current MSE from the train data  0.013219407
The current MSE from the train data  0.0086773615
[(0.018992307, 51), (0.018089337, 52), (0.01726855, 53)]
```

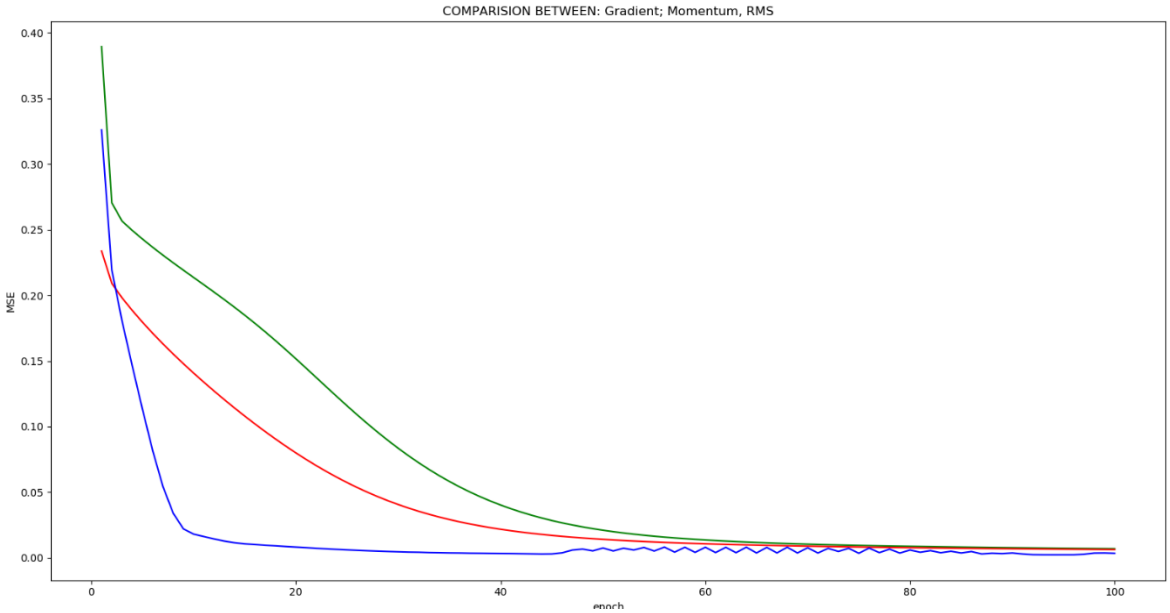
A table for momentum optimizer:

```
calculating MSE for strategy Momentum
The current MSE from the train data  0.23374084
The current MSE from the train data  0.074940726
The current MSE from the train data  0.020695822
The convergence is reached at epoch 45 with MSE 0.016421633
Converged at epoch 45 with MSE 0.016421633
The convergence is reached at epoch 46 with MSE 0.015773881
Converged at epoch 46 with MSE 0.015773881
The convergence is reached at epoch 47 with MSE 0.015180158
Converged at epoch 47 with MSE 0.015180158
The current MSE from the train data  0.010533194
The current MSE from the train data  0.0077024014
[(0.016421633, 45), (0.015773881, 46), (0.015180158, 47)]
```

A table for RMS optimizer:

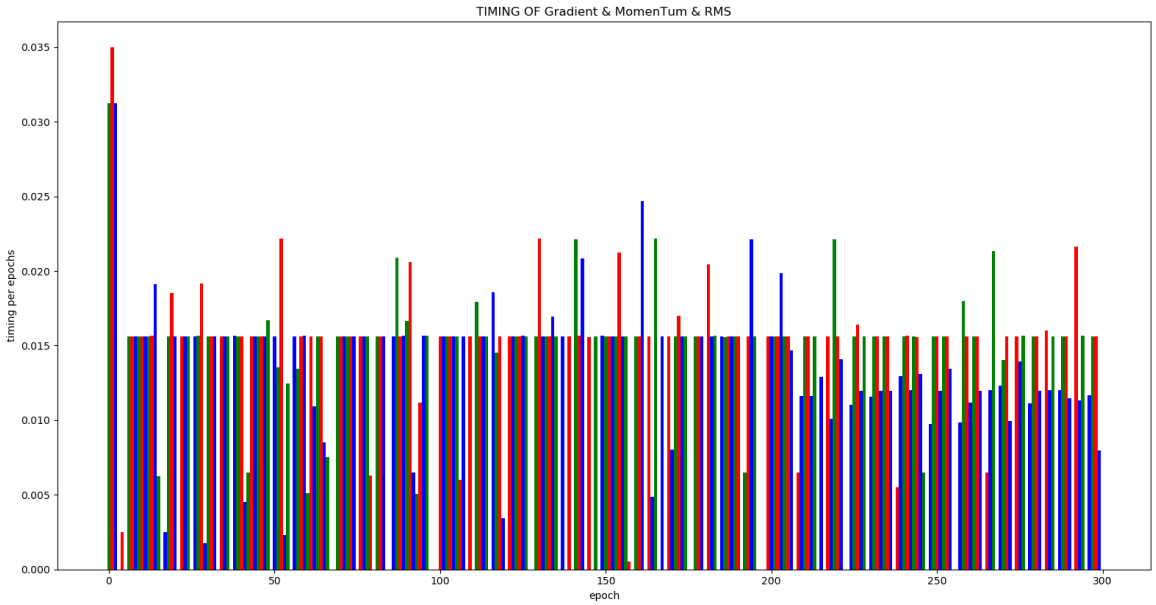
```
calculating MSE for strategy RMS
The current MSE from the train data  0.3259018
The convergence is reached at epoch 9 with MSE 0.018101584
Converged at epoch 9 with MSE 0.018101584
The convergence is reached at epoch 10 with MSE 0.01626031
Converged at epoch 10 with MSE 0.01626031
The convergence is reached at epoch 11 with MSE 0.014424048
Converged at epoch 11 with MSE 0.014424048
The current MSE from the train data  0.007814974
The current MSE from the train data  0.0033228504
The current MSE from the train data  0.0040626894
The current MSE from the train data  0.0043057958
[(0.018101584, 9), (0.01626031, 10), (0.014424048, 11)]
```

2. The MSE against epoch number of each of 3 methods for 1- 100 epochs.



We can claim that the RMS is the fast training method approaching to the convergence. The second is the momentum optimizer and the slowest is gradient descent optimizer.

3. The bar char of the CPU time taken per epoch for 3 methods:



The number of epochs is 300 because for each strategy, we have 100 records for each epoch. As the result, the combination of record from three strategies is 300. The green line is *GradientDescentOptimizer*; the red line is *MomentumOptimizer* and the blue line is *RMSPropOptimizer*.

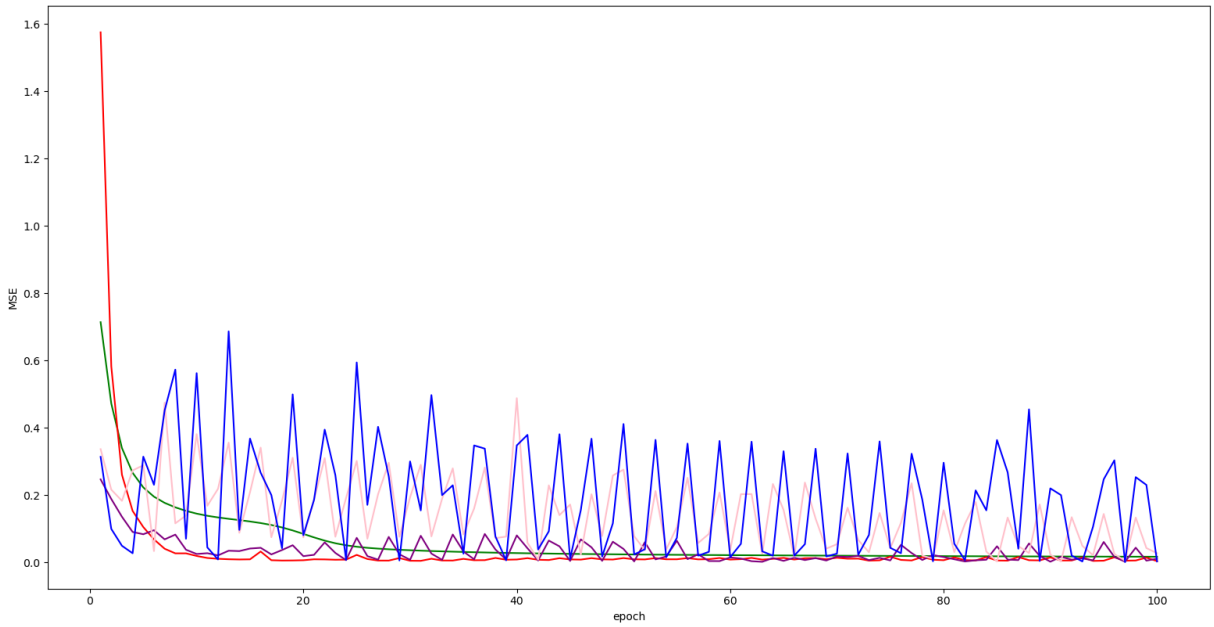
4. For this sample, the smallest MSE at epoch 100 is 0.0035176247 with the strategy is RMS

5. For this sample, the smallest MSE when the training error is reached is 0.016421633 with the strategy is Momentum

```
 The table is defined as
The smallest MSE at epoch 100 is  0.0035176247 with the strategy is RMS
The smallest MSE when the training error is reached is 0.016421633 with the strategy is Momentum
```

c.

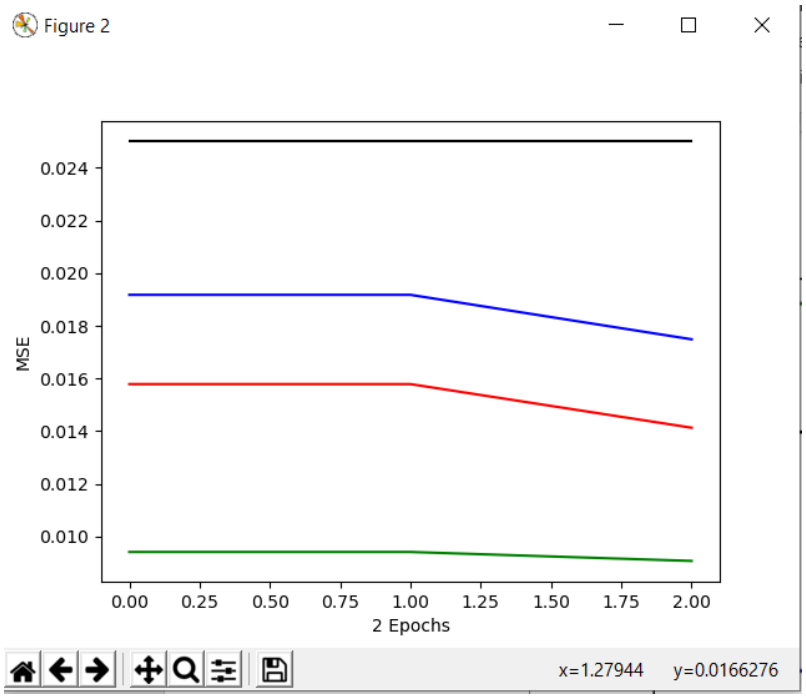
1. The 8 are a good choice for number of neurons in the hidden layer for the assignment, let's exam the experiment below.
2. The program first run an experience with 2,8,20,40,50 neurons: The standard MSE is: 0.02; range of epoch is 100; the training is RMS strategy with the learning rate 0.02; the batch size is 10. Here is the result:



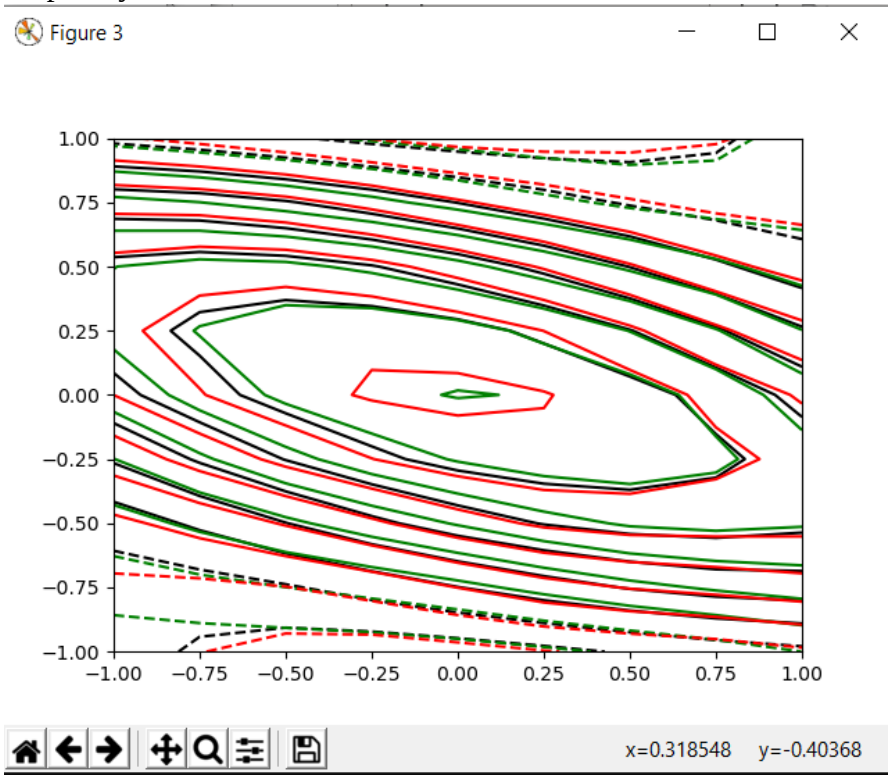
(2 for green; red for 8; purple for 20; 40 for pink and 50 is blue) Note: for some reasons the label in my python does not work (I figured out that I forget to add plot.legend() but please accept this).

The 8 reach the convergence at epoch 10 with MSE 0.013250093
The 20 reach the convergence at epoch 19 with MSE 0.018647287
The 40 reach the convergence at epoch 41 with MSE 0.008372547
The 50 reach the convergence at epoch 11 with MSE 0.008899873
Obviously, 8 neurons are good choice for the problem because it reaches the convergence at epoch 10 and at 100 epoch it still performs extremely good.

3. For the early stopping, the experiment is set up as: batch_size = 10; MSE = 0.02; learning rate is 0.005 the number of invalidation failures is 10 with the amount of iteration 0.025.
   The picture same as Fig 6:



Stop early:



(green is with early stopping and red is without early stopping)

Due to input data, therefore, the network is over training a little bit. However, the graph is still looking good.

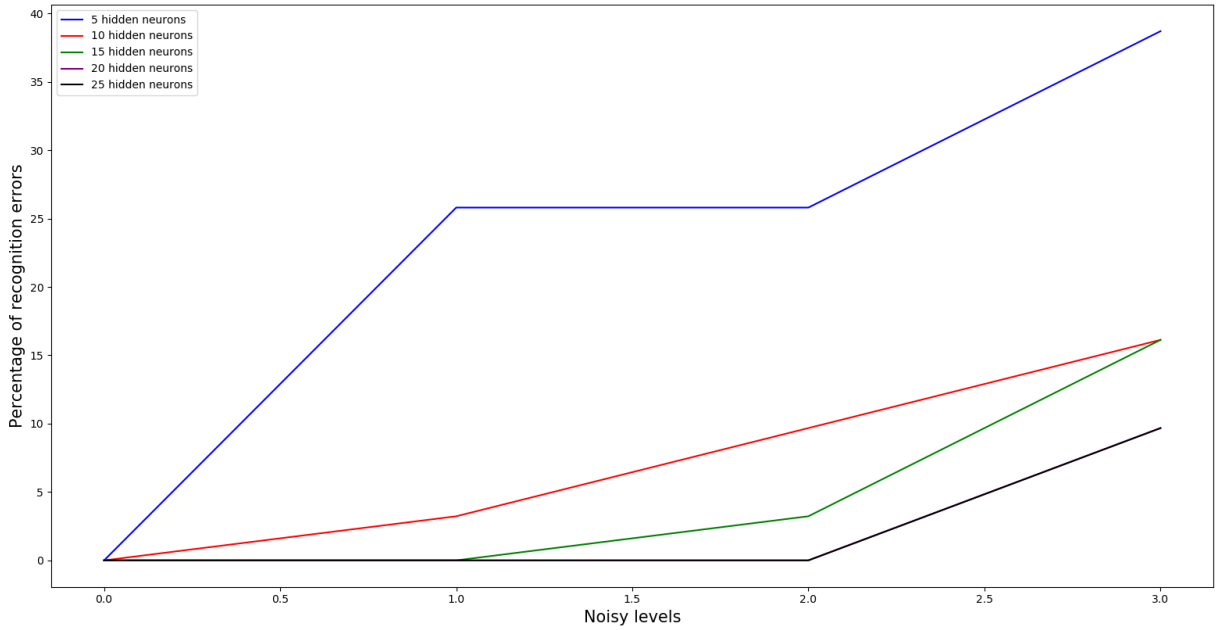
Commit a validation error at 220 the current validation error is: 1

Commit a validation error at 240 the current validation error is: 2

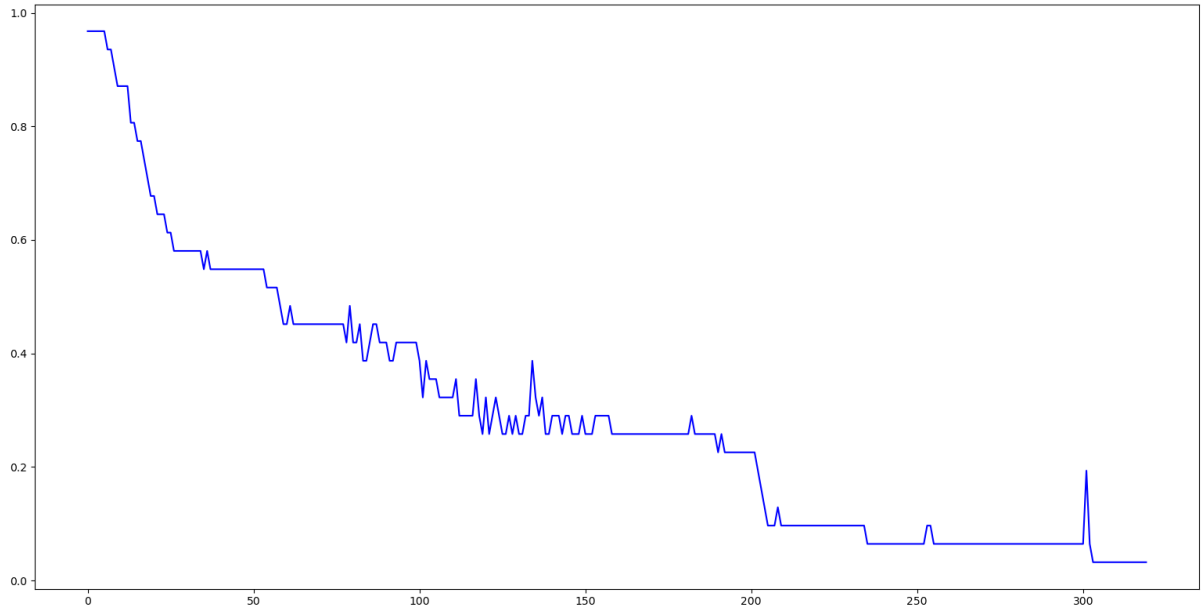
Commit a validation error at 245 the current validation error is: 3
Commit a validation error at 250 the current validation error is: 4
Commit a validation error at 255 the current validation error is: 5
Commit a validation error at 260 the current validation error is: 6
Commit a validation error at 266 the current validation error is: 7
Commit a validation error at 274 the current validation error is: 8
Commit a validation error at 287 the current validation error is: 9
Commit a validation error at 298 the current validation error is: 10
break at epochs 298
0.0020764393
0.0064128614
0.0023892254

Question 2:

a. For this part, the training data is the combination of non-noise data set and 3 bits noise data set. The experiment is set up with standard deviation: 0.01; the *AdamOptimizer* is used as specified in the assignment with the learning rate 0.03; the epochs for the assignment is 399.
Here is the graph of the experiment with hidden neuron numbers in the range 5-25

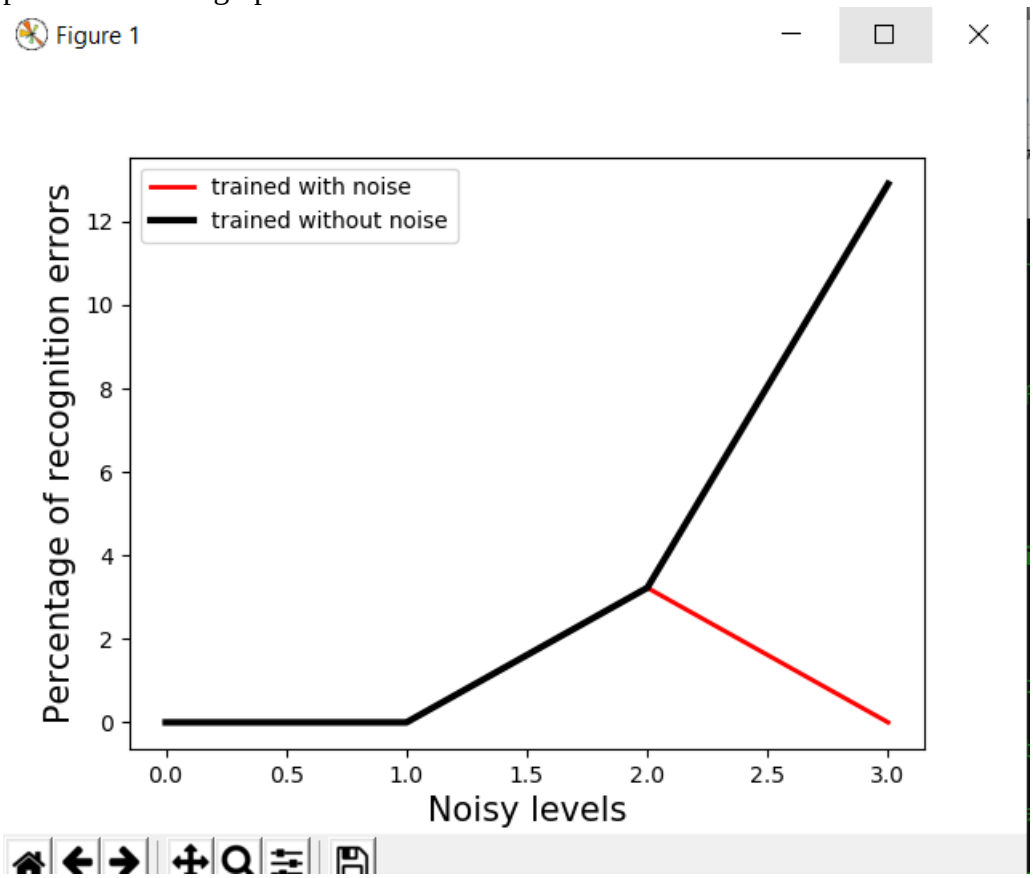


b. To examine the effect of the training strategy described in the assignment that is: First, train the network for zero decision error. Second, train the noisy data then re-train with the again for zero decisions error. I firstly, train the data with non- noise data for 300 epochs, then train the network with noisy data until the performance goal reach 0.01. Finally, re-train the network 20 times with the non-noise data. Here is the graph:



From the graph, we can see that the performance goal at epoch 300 increase a little bit, but while in the 20training process, the training error suddenly decrease.

c. Use same testing data from part 2 and the training strategy from part b, the result can be presented in the graph:



In my sample, the trained with noise is outperform the trained without noise. The graph can confirm that the training strategy mentioned in part b is reasonable to be used and applied for the current problem.