Neural Network

Final Project Report

The project for this course will be to build an artificial neural network from the ground up.

Presented by -
Ajinkya Rode -NUID 001221702
Ketul Shukla- NUID 001225292

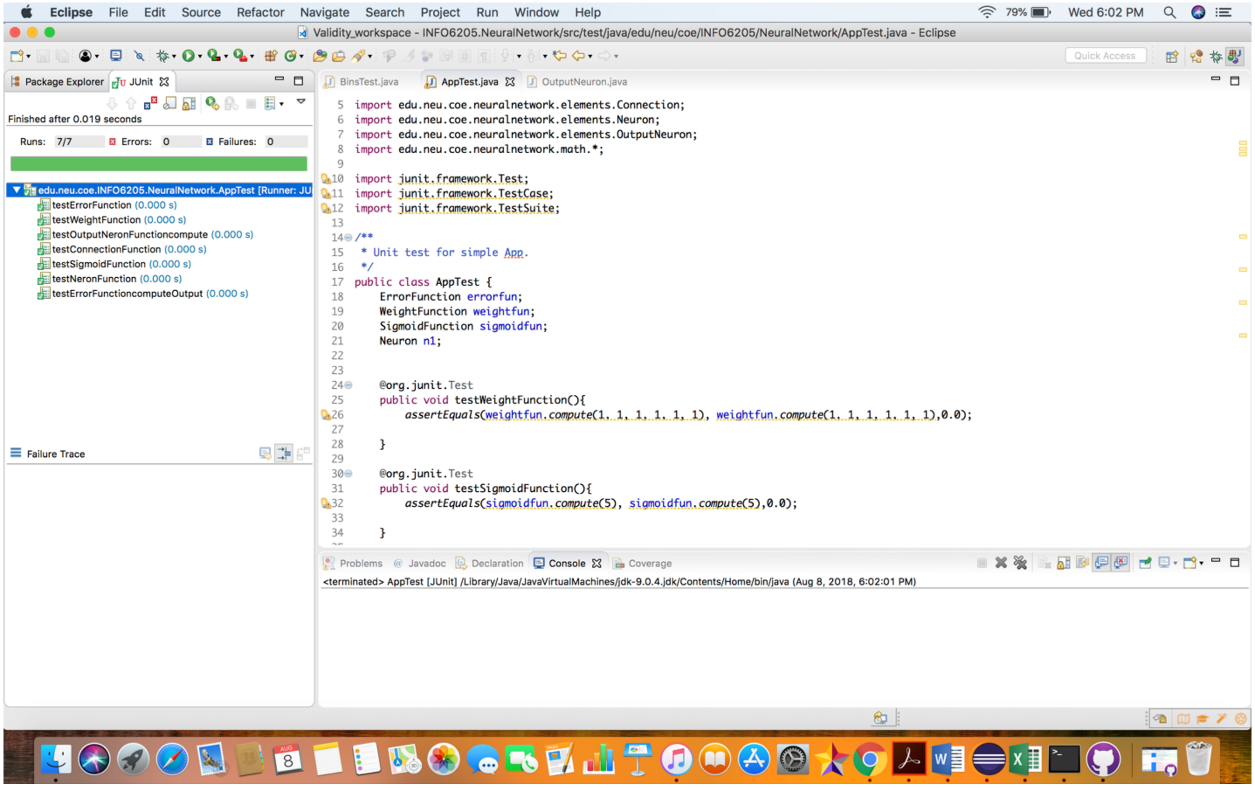
Introduction-

In this project we have developed a classic neural network example of recognizing handwritten digits using the MNIST dataset. The network can be trained and tested using this dataset provided in supporting files folder.

Unit Test for the experiment:

Below is the screenshot of successfully completed unit tests. It contains four tests which are.

1. Test Weight Function class's compute method.
2. Test Sigmoid Function class's compute method.
3. Test Error Function class's compute method and compute Output method.
4. Test Neuron creation and its parameters such as output and error.
5. Test Connection creation and its parameters such as weight, to Neuron and from Neuron that connection is.
6. Test Output Neuron creation and its parameters such as target setting and getting it.


Eclipse File Edit Source Refactor Navigate Search Project Run Window Help
79%
Wed 6:02 PM
Validity_workspace - INFO6205.NeuralNetwork/src/test/java/edu/neu/coe/INFO6205/NeuralNetwork/AppTest.java - Eclipse
Quick Access
Package Explorer
JUnit
BinsTest.java
AppTest.java
OutputNeuron.java
Failure Trace
Problems
Javadoc
Declaration
Console
Coverage

Finished after 0.019 seconds

Runs: 7/7 Errors: 0 Failures: 0

```
edu.neu.coe.INFO6205.NeuralNetwork.AppTest [Runner: JU
    testErrorFunction (0.000 s)
    testWeightFunction (0.000 s)
    testOutputNeronFunctioncompute (0.000 s)
    testConnectionFunction (0.000 s)
    testSigmoidFunction (0.000 s)
    testNeronFunction (0.000 s)
    testErrorFunctioncomputeOutput (0.000 s)
```

```java
import edu.neu.coe.neuralnetwork.elements.Connection;
import edu.neu.coe.neuralnetwork.elements.Neuron;
import edu.neu.coe.neuralnetwork.elements.OutputNeuron;
import edu.neu.coe.neuralnetwork.math.*;

import junit.framework.Test;
import junit.framework.TestCase;
import junit.framework.TestSuite;

/**
 * Unit test for simple App.
 */
public class AppTest {
    ErrorFunction errorfun;
    WeightFunction weightfun;
    SigmoidFunction sigmoidfun;
    Neuron n1;


    @org.junit.Test
    public void testWeightFunction(){
        assertEquals(weightfun.compute(1, 1, 1, 1, 1, 1), weightfun.compute(1, 1, 1, 1, 1, 1),0.0);

    }

    @org.junit.Test
    public void testSigmoidFunction(){
        assertEquals(sigmoidfun.compute(5), sigmoidfun.compute(5),0.0);

    }
```

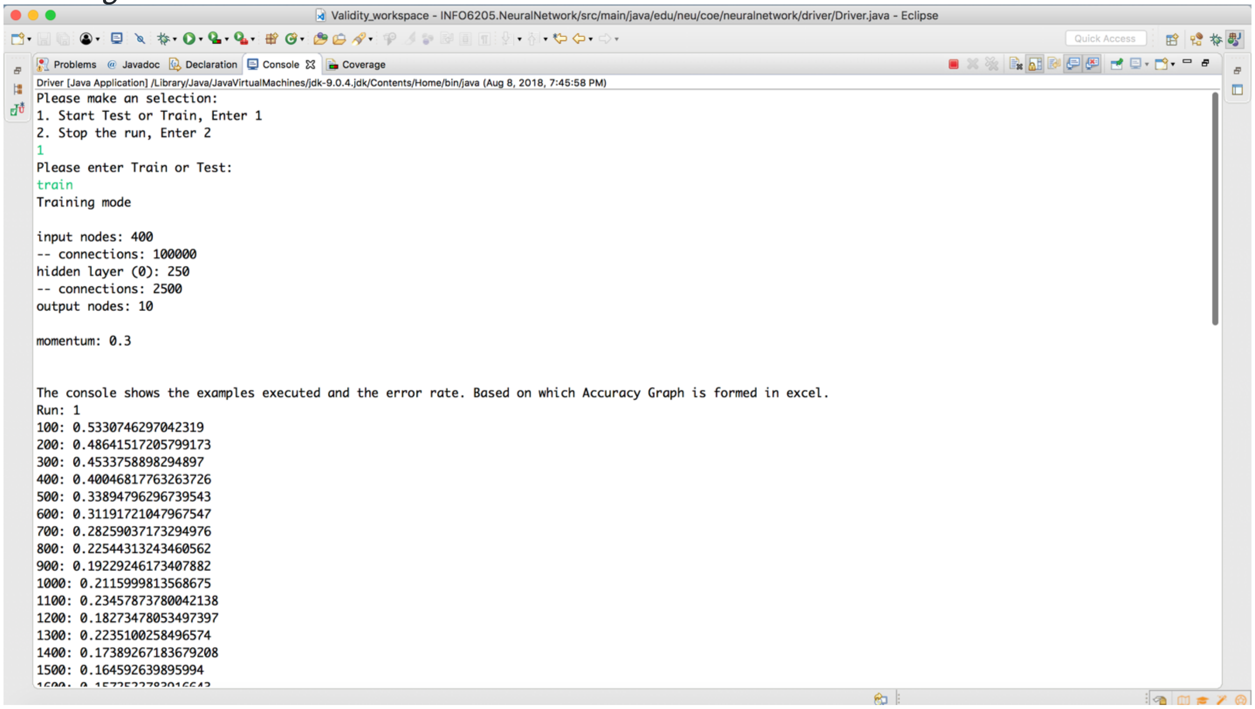
```
<terminated> AppTest [JUnit] /Library/Java/JavaVirtualMachines/jdk-9.0.4.jdk/Contents/Home/bin/java (Aug 8, 2018, 6:02:01 PM)
```

Observations:

The error rate (percentage wrong) and number of examples we have run, which is displayed in console also, csv file gets created with these details for training and test results as well. Also, the csv data is described using excel graphs provide in report below. The csv files are provided in supporting documents folder. Console shows the percentage accuracy at the end of testing the neural network.

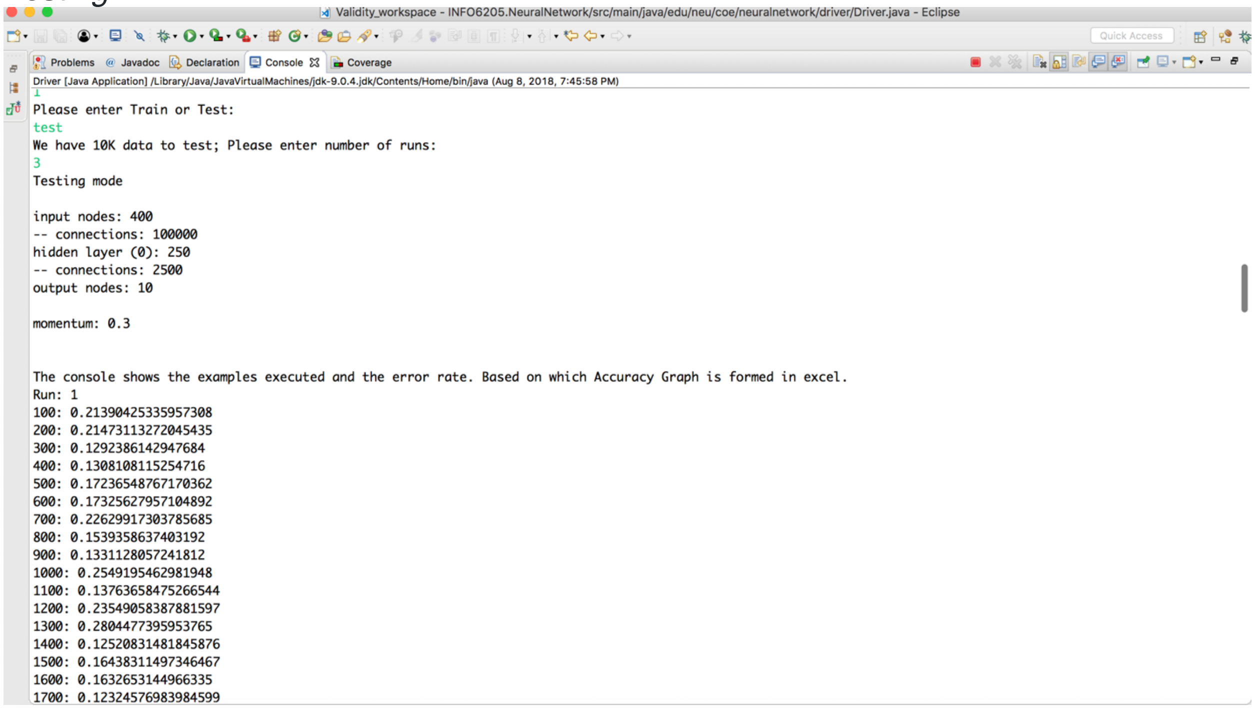
Below is the console screenshots while training and testing

Training

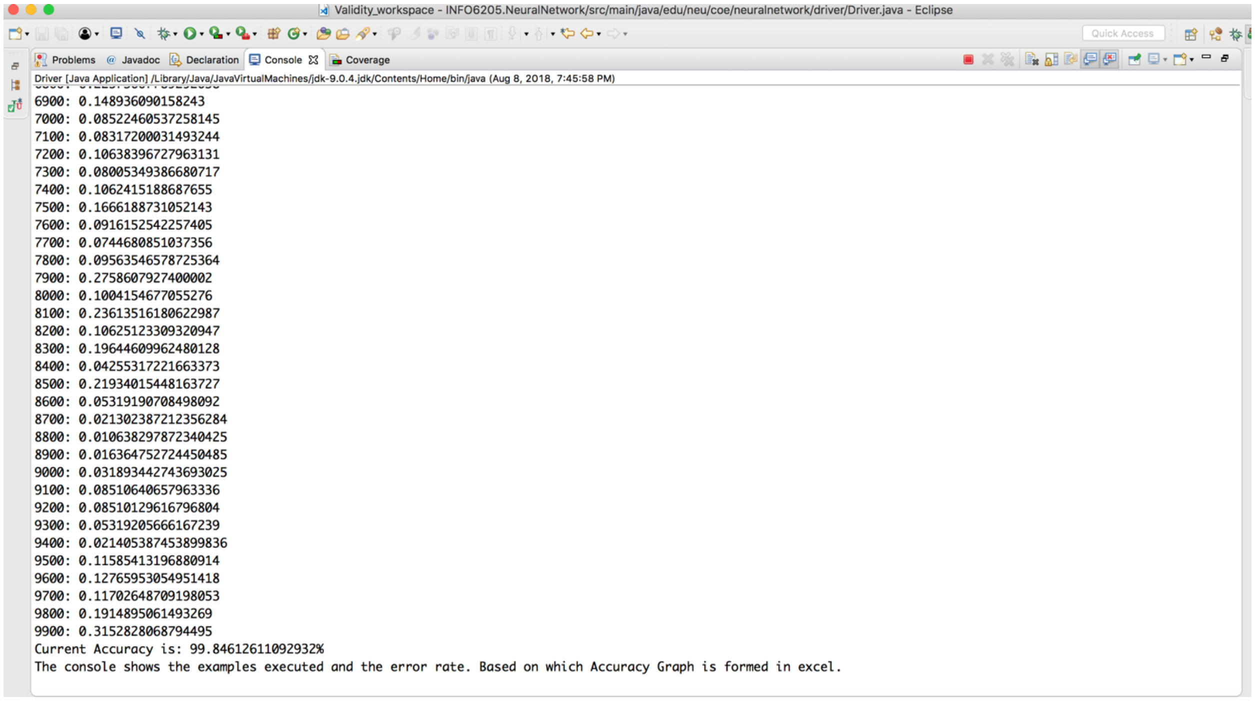

Validity_workspace - INFO6205.NeuralNetwork/src/main/java/edu/neu/coe/neuralnetwork/driver/Driver.java - Eclipse
Quick Access
Problems
Javadoc
Declaration
Console
Coverage

```
Driver [Java Application] /Library/Java/JavaVirtualMachines/jdk-9.0.4.jdk/Contents/Home/bin/java (Aug 8, 2018, 7:45:58 PM)
Please make an selection:
1. Start Test or Train, Enter 1
2. Stop the run, Enter 2
1
Please enter Train or Test:
train
Training mode

input nodes: 400
-- connections: 100000
hidden layer (0): 250
-- connections: 2500
output nodes: 10

momentum: 0.3


The console shows the examples executed and the error rate. Based on which Accuracy Graph is formed in excel.
Run: 1
100: 0.5330746297042319
200: 0.48641517205799173
300: 0.4533758898294897
400: 0.40046817763263726
500: 0.33894796296739543
600: 0.31191721047967547
700: 0.28259037173294976
800: 0.22544313243460562
900: 0.19229246173407882
1000: 0.2115999813568675
1100: 0.23457873780042138
1200: 0.18273478053497397
1300: 0.2235100258496574
1400: 0.17389267183679208
1500: 0.164592639895994
```

Testing

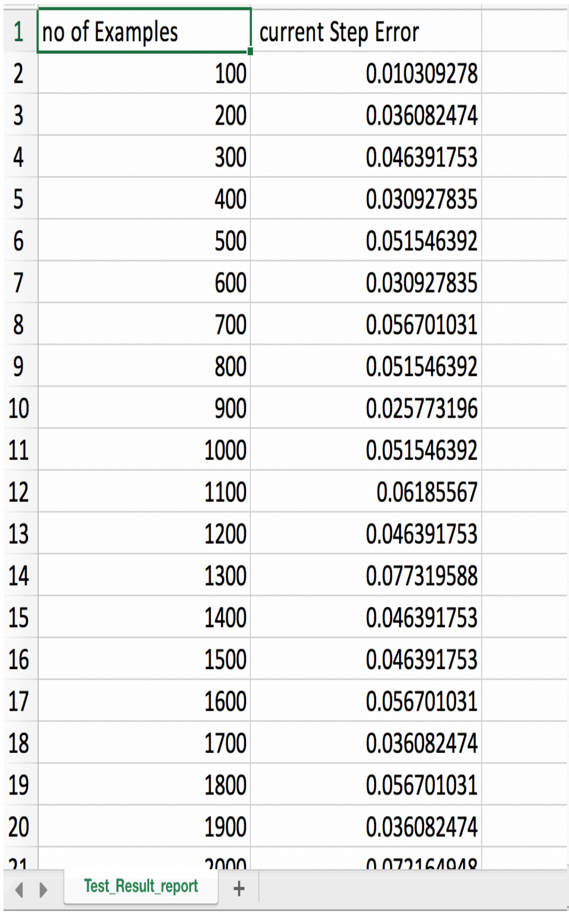

Validity_workspace - INFO6205.NeuralNetwork/src/main/java/edu/neu/coe/neuralnetwork/driver/Driver.java - Eclipse
Quick Access
Problems
Javadoc
Declaration
Console
Coverage

```
Driver [Java Application] /Library/Java/JavaVirtualMachines/jdk-9.0.4.jdk/Contents/Home/bin/java (Aug 8, 2018, 7:45:58 PM)
1
Please enter Train or Test:
test
We have 10K data to test; Please enter number of runs:
3
Testing mode

input nodes: 400
-- connections: 100000
hidden layer (0): 250
-- connections: 2500
output nodes: 10

momentum: 0.3


The console shows the examples executed and the error rate. Based on which Accuracy Graph is formed in excel.
Run: 1
100: 0.21390425335957308
200: 0.21473113272045435
300: 0.1292386142947684
400: 0.1308108115254716
500: 0.17236548767170362
600: 0.17325627957104892
700: 0.22629917303785685
800: 0.1539358637403192
900: 0.1331128057241812
1000: 0.2549195462981948
1100: 0.13763658475266544
1200: 0.23549058387881597
1300: 0.2804477395953765
1400: 0.12520831481845876
1500: 0.16438311497346467
1600: 0.1632653144966335
1700: 0.12324576983984599
```

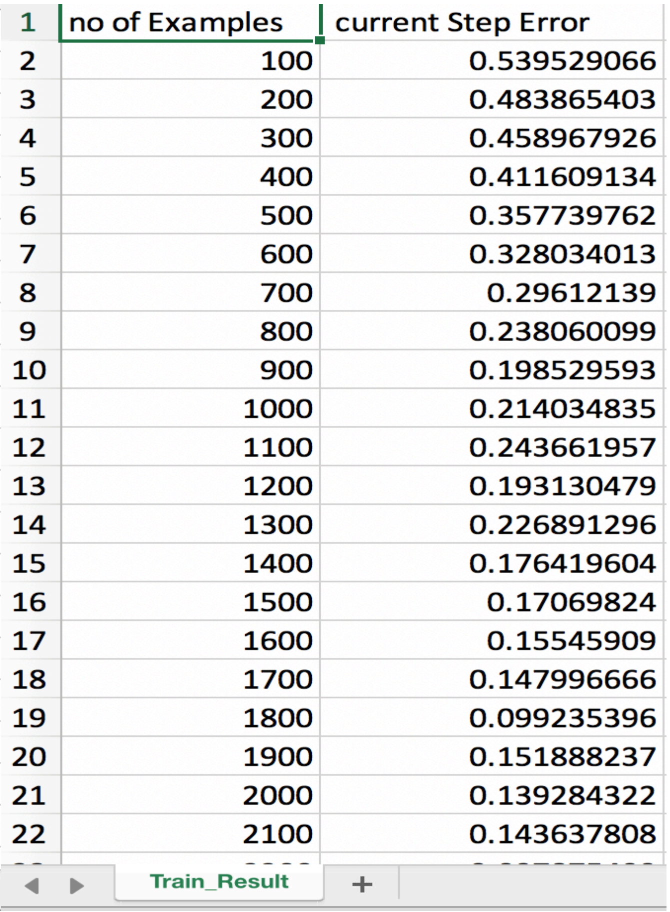

Validity_workspace - INFO6205.NeuralNetwork/src/main/java/edu/neu/coe/neuralnetwork/driver/Driver.java - Eclipse
Quick Access
Problems
Javadoc
Declaration
Console
Coverage

```
Driver [Java Application] /Library/Java/JavaVirtualMachines/jdk-9.0.4.jdk/Contents/Home/bin/java (Aug 8, 2018, 7:45:58 PM)
6900: 0.148936090158243
7000: 0.08522460537258145
7100: 0.08317200031493244
7200: 0.10638396727963131
7300: 0.08005349386680717
7400: 0.1062415188687655
7500: 0.1666188731052143
7600: 0.0916152542257405
7700: 0.0744680851037356
7800: 0.09563546578725364
7900: 0.2758607927400002
8000: 0.1004154677055276
8100: 0.23613516180622987
8200: 0.10625123309320947
8300: 0.19644609962480128
8400: 0.04255317221663373
8500: 0.21934015448163727
8600: 0.05319190708498092
8700: 0.021302387212356284
8800: 0.010638297872340425
8900: 0.016364752724450485
9000: 0.031893442743693025
9100: 0.08510640657963336
9200: 0.08510129616796804
9300: 0.05319205666167239
9400: 0.021405387453899836
9500: 0.11585413196880914
9600: 0.12765953054951418
9700: 0.11702648709198053
9800: 0.1914895061493269
9900: 0.3152828068794495
Current Accuracy is: 99.84612611092932%
The console shows the examples executed and the error rate. Based on which Accuracy Graph is formed in excel.
```

Below is the data snippet of training result and test result.



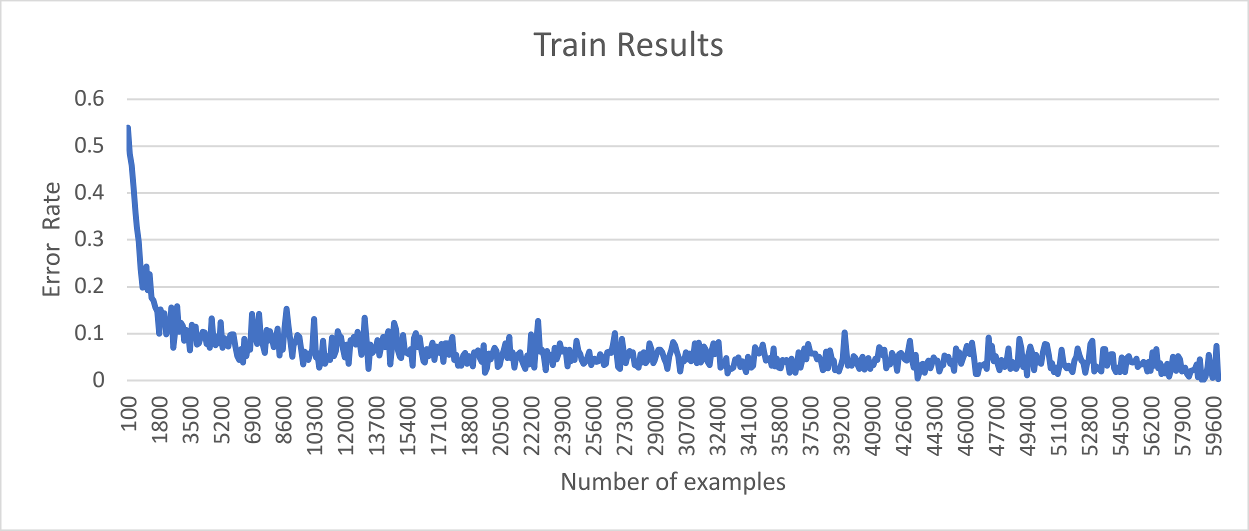
1	no of Examples	current Step Error
2	100	0.010309278
3	200	0.036082474
4	300	0.046391753
5	400	0.030927835
6	500	0.051546392
7	600	0.030927835
8	700	0.056701031
9	800	0.051546392
10	900	0.025773196
11	1000	0.051546392
12	1100	0.06185567
13	1200	0.046391753
14	1300	0.077319588
15	1400	0.046391753
16	1500	0.046391753
17	1600	0.056701031
18	1700	0.036082474
19	1800	0.056701031
20	1900	0.036082474
21	2000	0.072164948



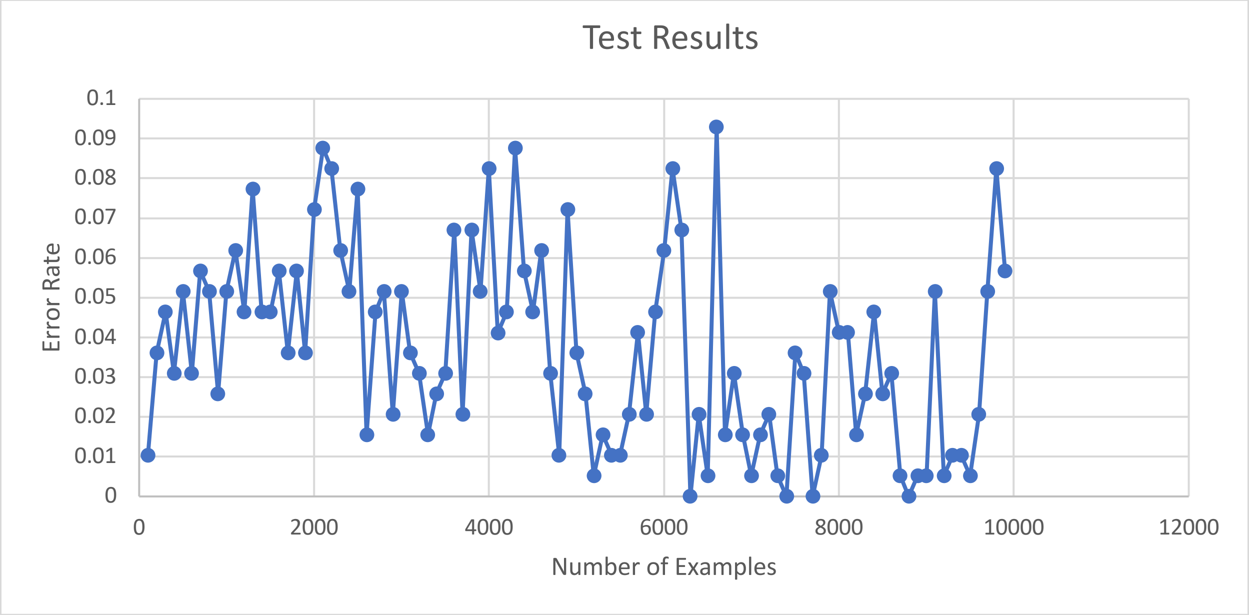
1	no of Examples	current Step Error
2	100	0.539529066
3	200	0.483865403
4	300	0.458967926
5	400	0.411609134
6	500	0.357739762
7	600	0.328034013
8	700	0.29612139
9	800	0.238060099
10	900	0.198529593
11	1000	0.214034835
12	1100	0.243661957
13	1200	0.193130479
14	1300	0.226891296
15	1400	0.176419604
16	1500	0.17069824
17	1600	0.15545909
18	1700	0.147996666
19	1800	0.099235396
20	1900	0.151888237
21	2000	0.139284322
22	2100	0.143637808

Below is the complete data represented on a graph. The X axis in both the graph is the Number of examples and Y axis is the error rate.

Graph while training the Neural network



Graph while testing the Neural Network



Conclusion:

The train result graph shows the error rate goes from 0.5 to below 0.1 as we train the network with number of examples. As the number of examples are increasing then the error rate goes on reducing. While the Test result shows the error rate is below 0.1. It ranges from 0 to 0.09. As we have trained our developed network it can now guess accurately handwritten digits using the MNIST dataset. The percentage accuracy we calculated it shows is 99.84%.