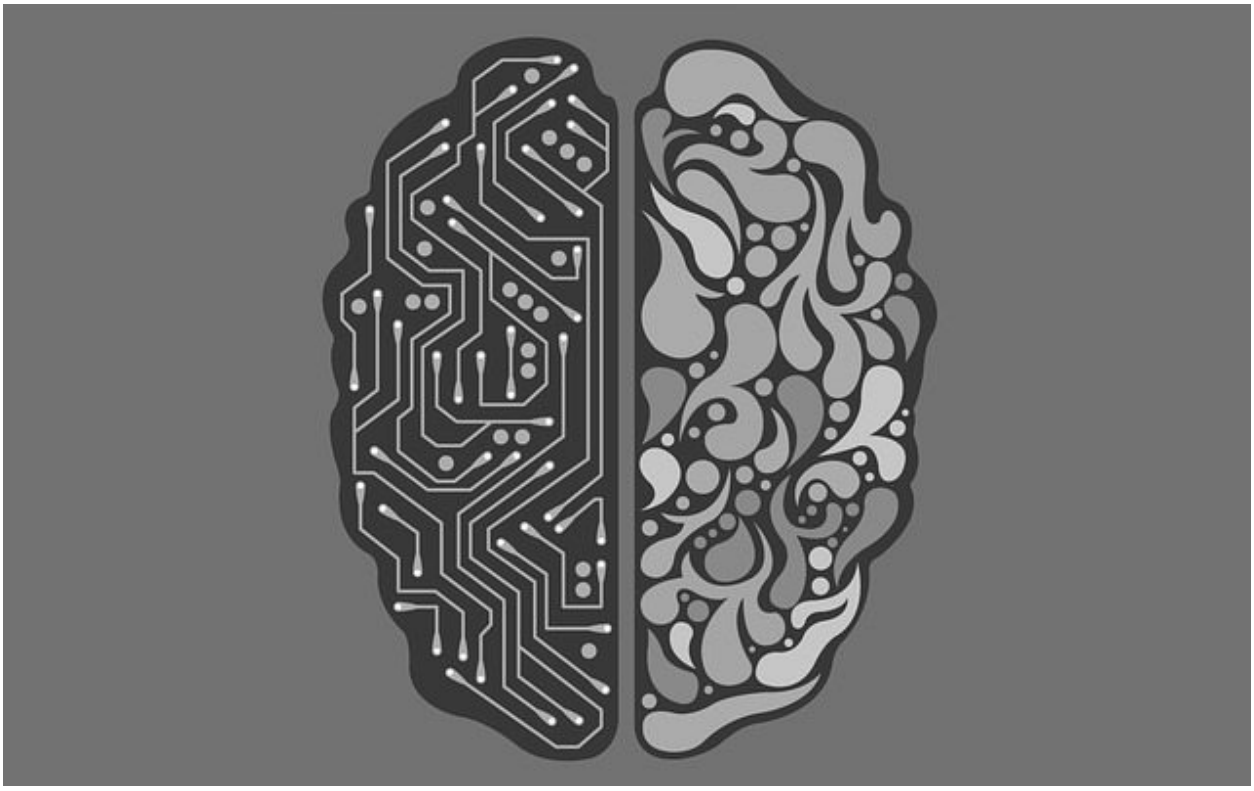


Neural Networks

- Digit Recognition (Excel data)
- Face Recognition (Images)

-By
Ajay Goel
(001897443)

Akshay N. Mahajanshetti
(001893697)

Table of Contents

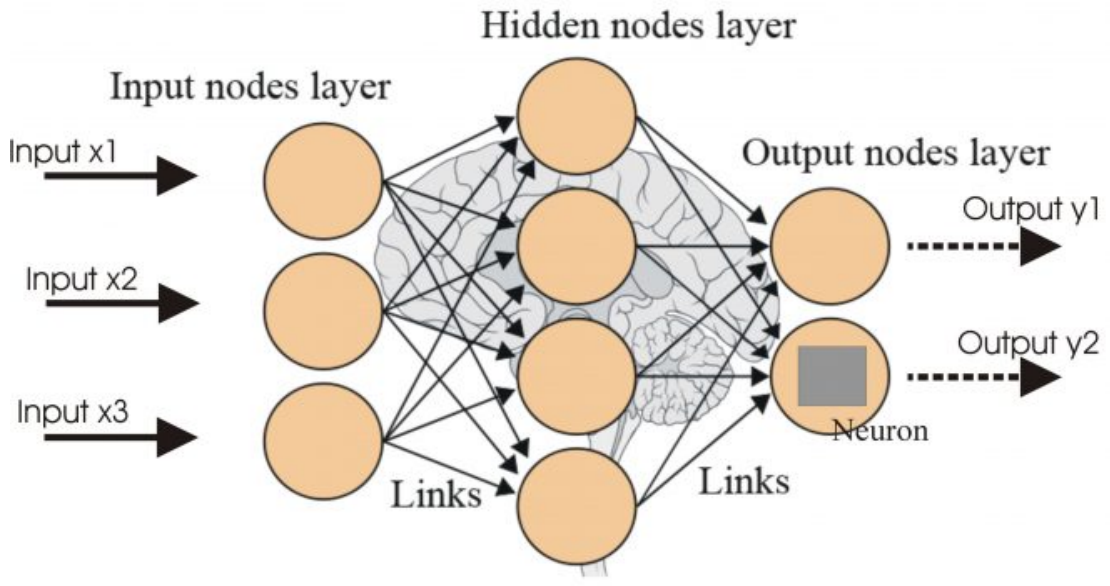
Introduction

Artificial neural networks (ANNs) or connectionist systems are computing systems which perform tasks by considering examples, generally without being programmed with any task-specific rules.
For example, In image recognition, system might learn to identify images that contain cats by analyzing training images that have been manually labelled as "cat" or "no cat" and using the results to identify cats in other images. It does this without any prior knowledge about cats. The system learns things like cats have fur, tails, whiskers and other cat attributes from the training data.

An ANN is based on a collection of connected units or nodes which can transmit a signal from one artificial neuron to another. An artificial neuron that receives a signal can process it and then signal additional artificial neurons connected to it. Artificial neurons and edges typically have a weight that adjusts as learning proceeds. The weight increases or decreases the strength of the signal at a connection. Different layers may perform different kinds of transformations on their inputs. Signals travel from the first layer (the input layer), to the last layer (the output layer), possibly after traversing the layers multiple times.

Goal:
To construct a system that solves problems in the same way that a human brain would.

Neural Network Skeleton



Problem Statement

The neural network in our application is used to solve two major problems:

Digit Recognition

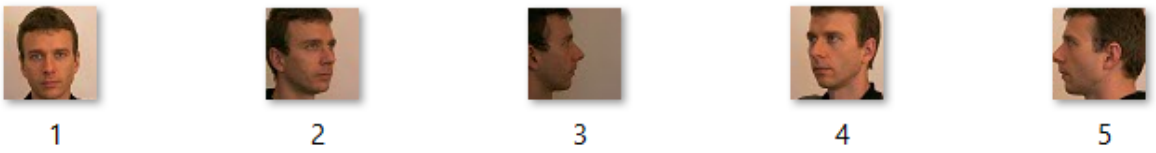
In this problem, The idea is to take a large number of handwritten digits **(in excel format)**. In our system, we have taken data set of 42000 digits to train the system then develop a system which can learn from those training examples, we test it by passing 11000 testing digits to the system and at least 85% digits should be predicted correctly.
Furthermore, by increasing the number of training examples, the network can learn more about handwriting, and so improve its accuracy.

Face Recognition

In this problem, our aim is to pass images (.png format) of 10 different people in 5 different angles and train the system. After this, to see if the system has learnt to identify different people, we test it by passing close to 35 images to the system and at least 80% images should be recognised correctly.

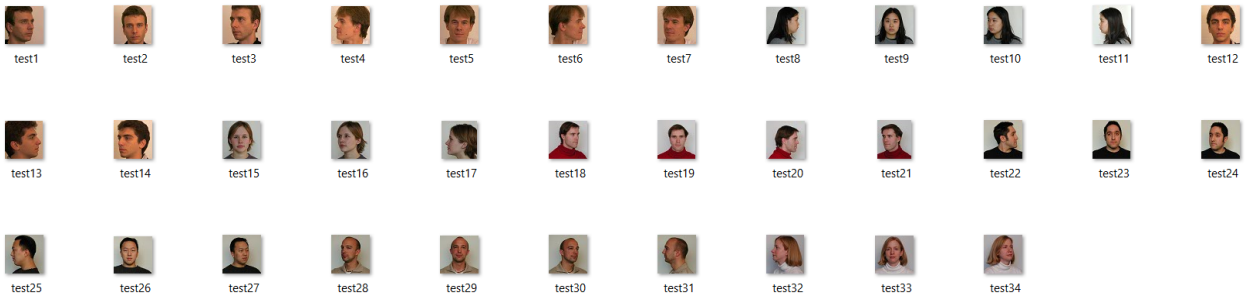
Image samples that we have passed.

First image set



Similarly we have passed 5 images in different angles of 10 different people.

The Test images we am passing are as follows



Implementation Details

We have made classes like Matrix, Sketch, Neural Net and Image Conversion class to carry out the different techniques to form the system.

Driver (Main class) or Sketch

In this class, We are passing two sets of datasets:

- 42000 digits datasets.
- 50 images of different individuals from different angles.

The system is trained individually for digits first and then testing datasets is passed to checkout the output. Similarly, the separate system is created for face recognition and then tested using 34 different testing images to check whether the predicted output is correct or not.

Neural Net

In this class, neural net is created 3 layers ie.,

- Input Layer
- Hidden Layer
- Output Layer

Problems	Neurons in each layer		
	Input Layer	Hidden Layer	Output Layer
Face Recognition	2500	1250	10
Digit Recognition	784	392	10

Note: Output layer neurons are the maximum number of outcomes possible.
Example: In digit Recognition: only possible outcomes are 10 as 0,1,2,3,4,5,6,7,8,9.

Matrix

It is the library class that we have designed to carry out the functioning in the important functions like feedback and backpropagation.
We will convert the input data array into matrix format which will denote our different layers and neurons in it.

- Rows: Number of neurons in each layer
- Columns: Number of layers in the neuron network.

It has functions to convert input array to matrix, dot multiplication, cross multiplication, addition, sigmoid function, random, subtract.

Image Conversion

- This class is used only in face recognition only.
- Image load function is used to convert the image of any format to array. Suppose, if the image is 50*50 dimensions then the array or input layer is formed of 2500 neurons.
- It returns an array which is formed after conversion.

IMPORTANT FUNCTIONS

Essential Techniques used

$Y = \sigma(\sum w*a + b)$

For further concepts, it is important to understand above mentioned formula:

Here, Y = output of any layer

σ = Sigmoidal Function

w = weights on the edge connected to the neuron

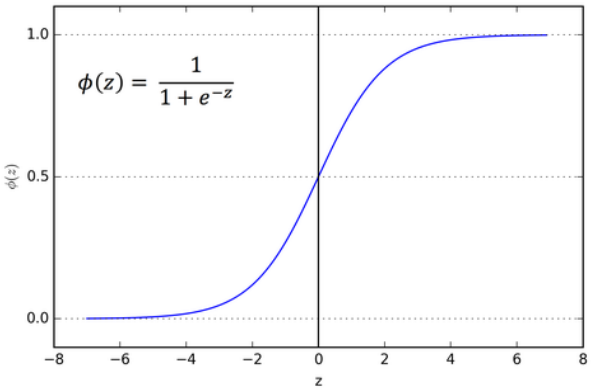
a = neuron value

b = bias

Sigmoidal Function

$y = \frac{1}{1+e^{-x}}$, e is the

Sigmoid functions have finite limits at negative infinity and infinity, most often going either from 0 to 1 or from −1 to 1, depending on convention. **e** is the base of the Natural Logarithms (Euler's formula)



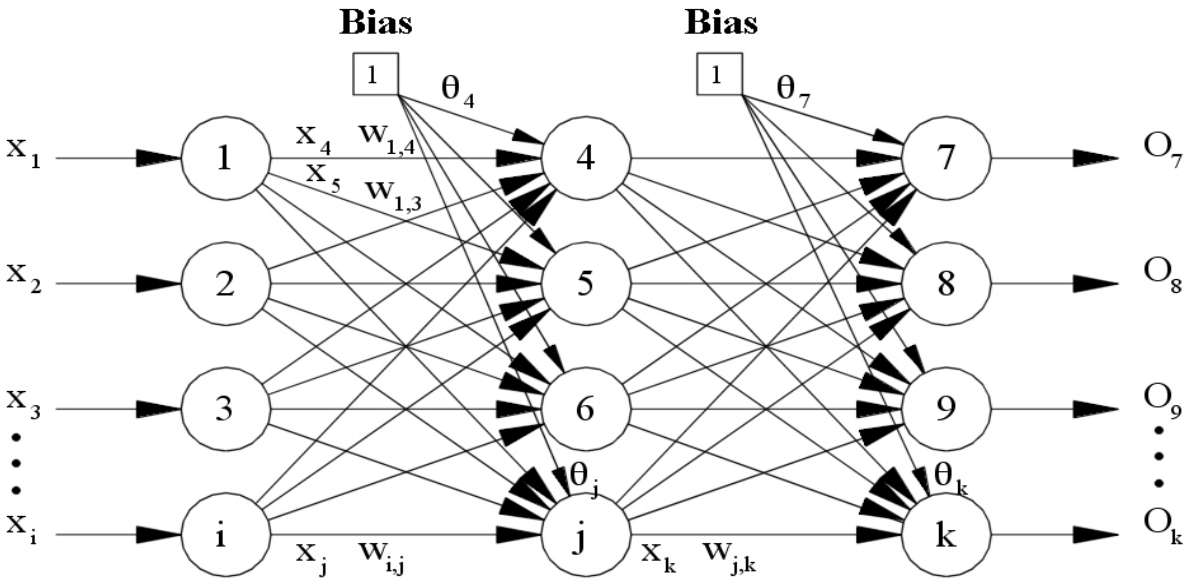
Sigmoidal Function

Bias

We need bias in the network as if the inputs are 0, then the total weight with the value will be 0. So, we need bias in a given direction. It will help in tuning.

Neuron Value: It is the random value in the initial stage. Then, Delta value is added which is calculated by multiplying gradient and inputs of the layer is added to a neuron as the value.

Output of any layer: It is the maximum number of possible outcomes possible while training. We consider the output as the neuron which has maximum value.



BackPropagation

The Backpropagation algorithm is a supervised learning method for multilayer feed-forward networks. A given function by modifying internal weightings of input signals to produce an expected output signal.
The system is trained using a supervised learning method, where the error between the system's output and a known expected output is presented to the system and used to modify its internal state.

Each neuron computes its own activation. It computes a weighted sum of the outputs of the previous layer and applies an activation function before forwarding it on to the next layer.

Pseudo Code for Back Propagation

Inputs, targets - Array which are converted to matrix
//Generating the output by hidden layer
Hidden Matrix = **Inputs * Weights input Hidden**
Hidden +=Bias Hidden

//Activation Function
Sigmoidal Function to Hidden Matrix

//Generating the output
Output Matrix = **Weights Hidden to Output * Hidden Matrix**
Output Matrix = **Bias Hidden + Output Matrix**
Sigmoidal Function(Output Matrix)

Target Matrix (Converted from Target Array)

//Error = Target - Output

Output Error = **Target Matrix - Output Matrix**

//Calculating Gradient
Output Gradient = **Derivative of sigmoid(Output Matrix)**
Output Gradient = **Output Gradient * Output Error**
Output Gradient = **Learning Rate * Output Gradient**

//Calculating Deltas
Weights hidden output Deltas = **Output Gradient * Transpose(Hidden Matrix)**

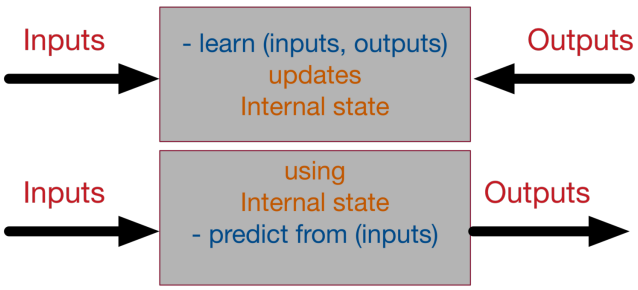
//Adjusting the weights by deltas
Weights hidden output += Weights hidden output Deltas
Bias Output += **Output Gradient**

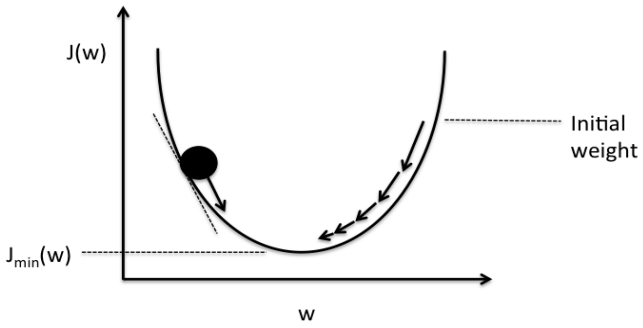
//Calculating the hidden layer errors
Hidden Errors += **Transpose(Weights Hidden Output) * Output Errors**

//Calculating the hidden gradient
Hidden Gradient = Derivative of Sigmoid(Hidden Matrix)
Hidden Gradient *= Hidden Errors
Hidden Gradient *= **Learning Rate**

//Calculating Input to hidden deltas
Weights input Hidden Layer Deltas = **Hidden gradient * transpose(Input)**
Weights input hidden += **Weights input Hidden Layer**
Bias Hidden += Hidden Gradient

In each iteration, our System decides the contribution of the weights from each neuron of the previous layer. While, back propagation helps to tune the weights according to the contribution of each neuron.





Schematic of gradient descent.

Feed Forward

Our network is built on the feed forward model, meaning that an input arrives at the first neuron and the output of that neuron flows across the connections to the right until it exits as output from the network itself.

Input is given in the form of an array which is converted into matrix for the processing in the input, hidden & output layer. Output is stored and analyzed.

Pseudo Code for Feed - Forward

```
Input Matrix = Matrix converted from input array

//Generating the output of hidden layer
Hidden Matrix = Weights Input Hidden * Input Matrix
Hidden Matrix += Bias Hidden

//Activation Function
Hidden Matrix  = Sigmoidal Function(Hidden Matrix)

//Generating the output
Output= Weights Hidden Output  * Hidden Matrix
Output +=Bias Output

Output = Sigmoidal Function(Output)

return Output Array converted from Output Matrix
```

Activation Function

It calculates a "weighted sum" of its input, adds a bias and then decides whether it should be "fired" or not.

$$Y = \sum (weight * input) + bias$$

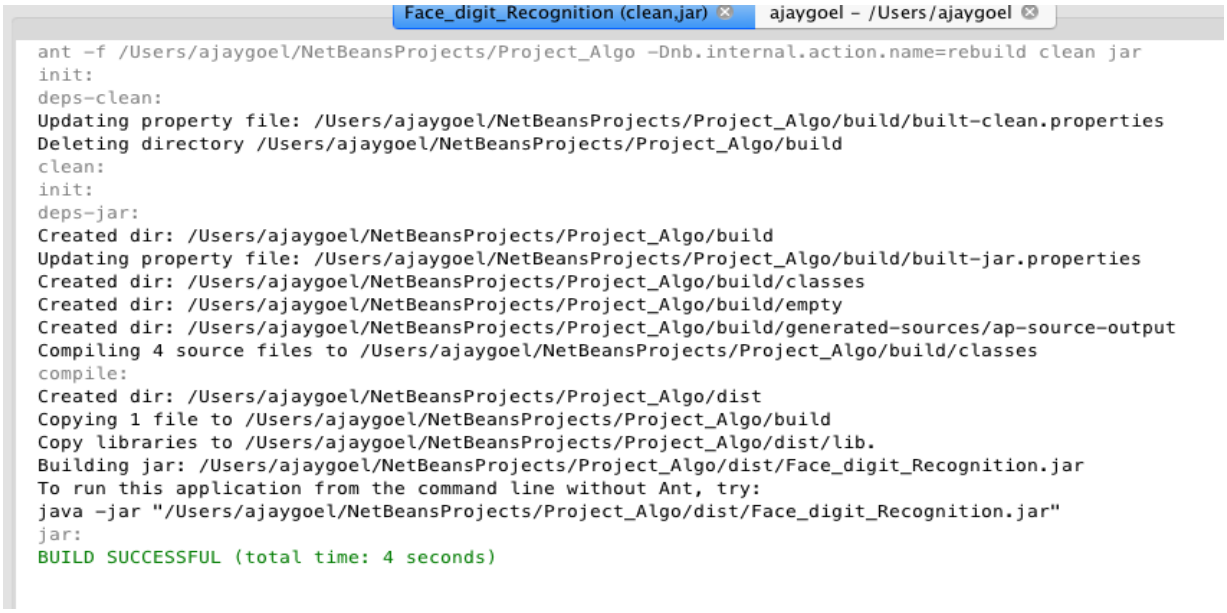
There are different kinds of activation functions. In our project, we have used Sigmoidal Function which is already explained above.

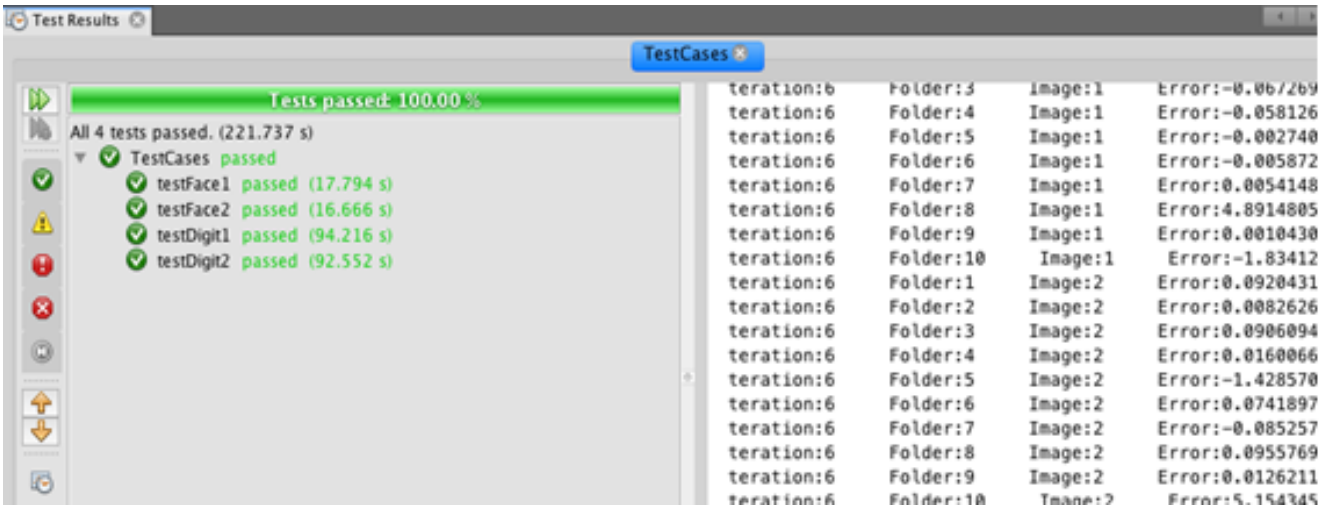
OUTPUT

To note the consistency, we have tested our project on two datasets.

Digit Recognition (Excel format)	Face Recognition(.png file)
- 30,000 rows of data was trained. - 11,000 rows of training data was passed for testing. -Output prediction was correct upto 89%.	-50 images were trained - 34 training images were passed for testing. - Output prediction was correct upto 82%.

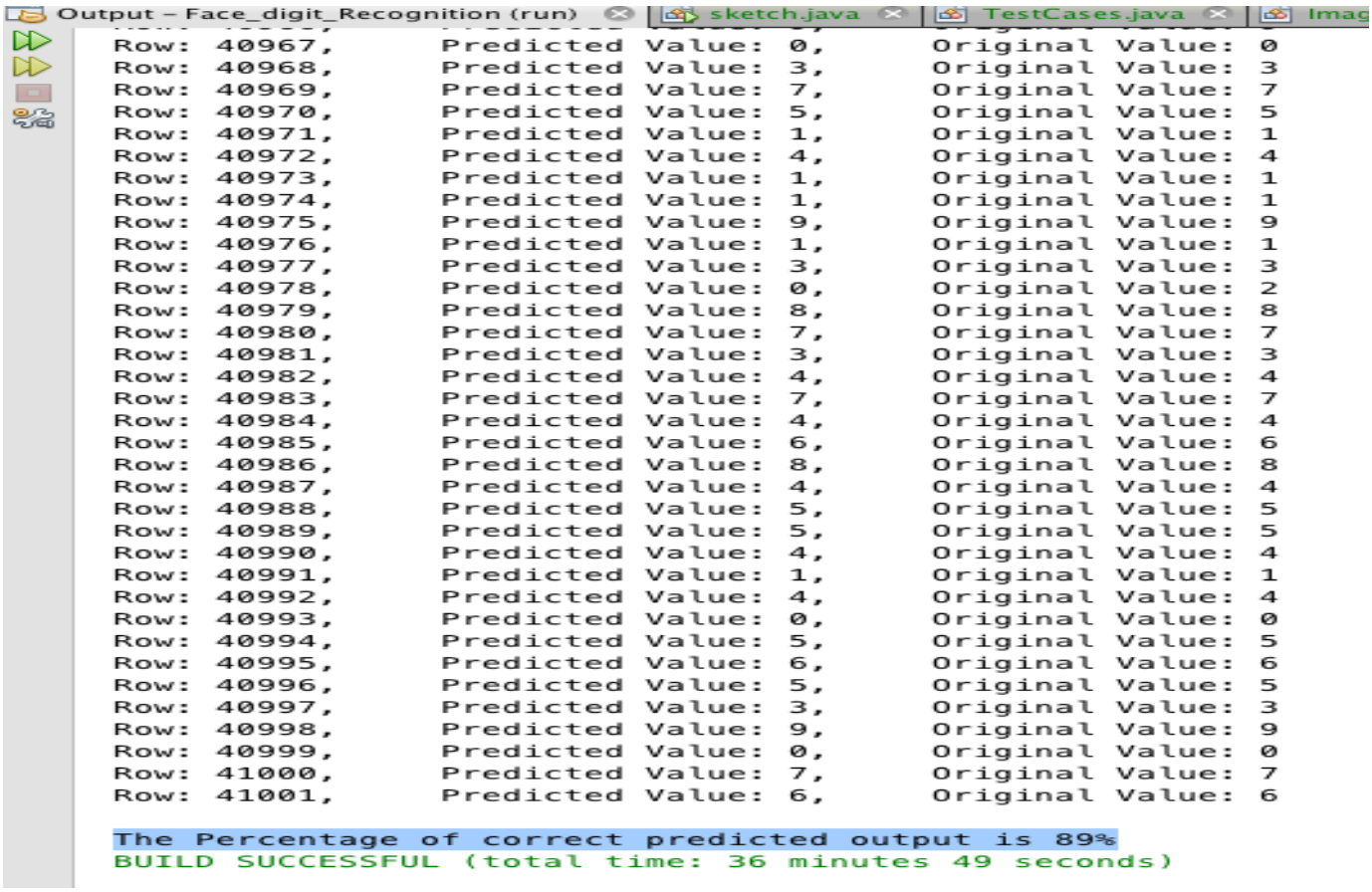
Maven build and test cases screenshot :



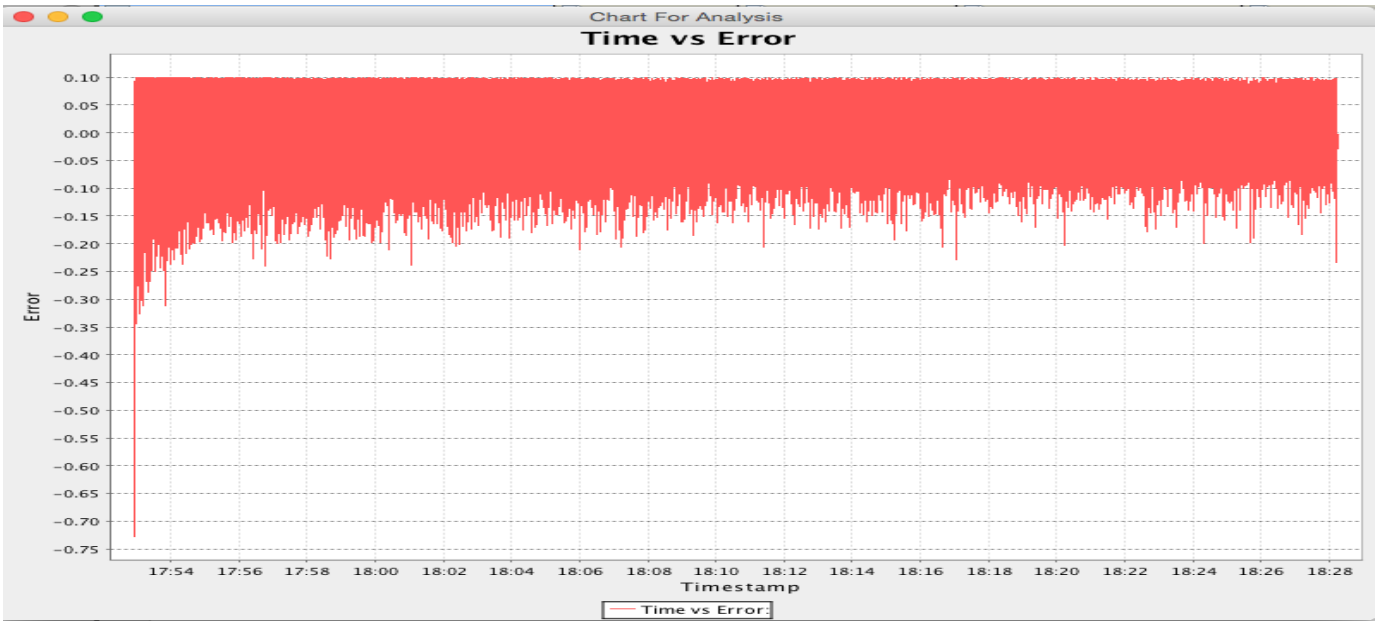


Confusion Matrix : Data Set 1 (Digit Recognition)

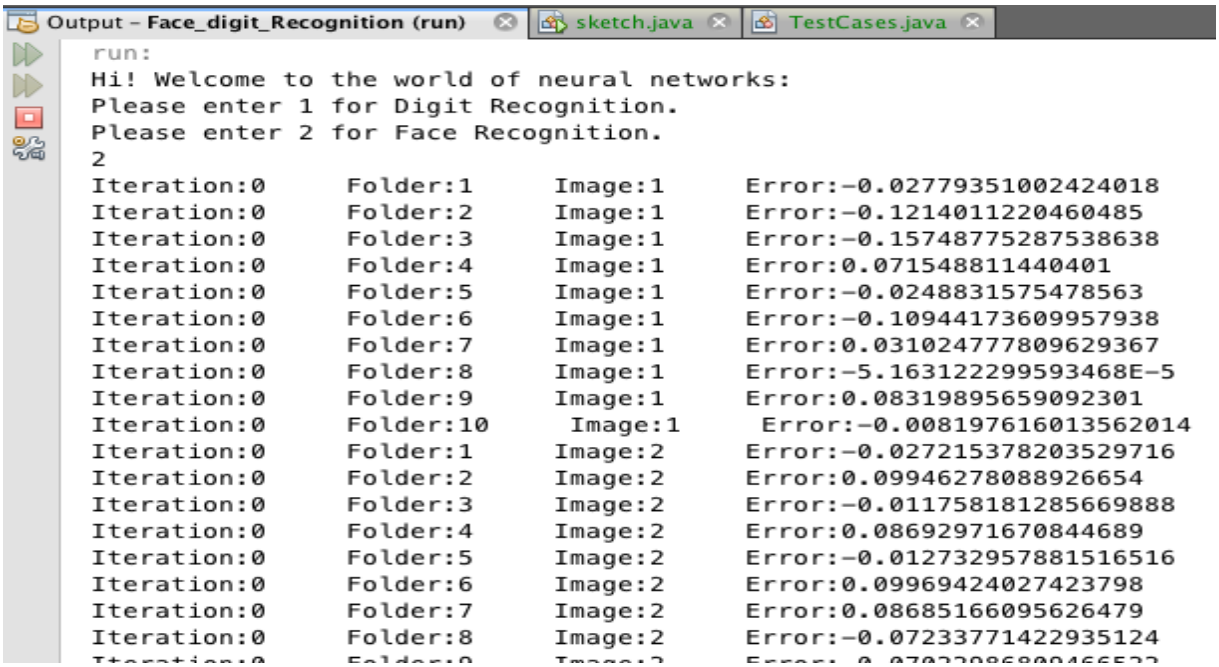
Output : Percentage of predicted output correct = 89%



Graph analysis of Data Set 1 (Error in backpropagation vs Time)



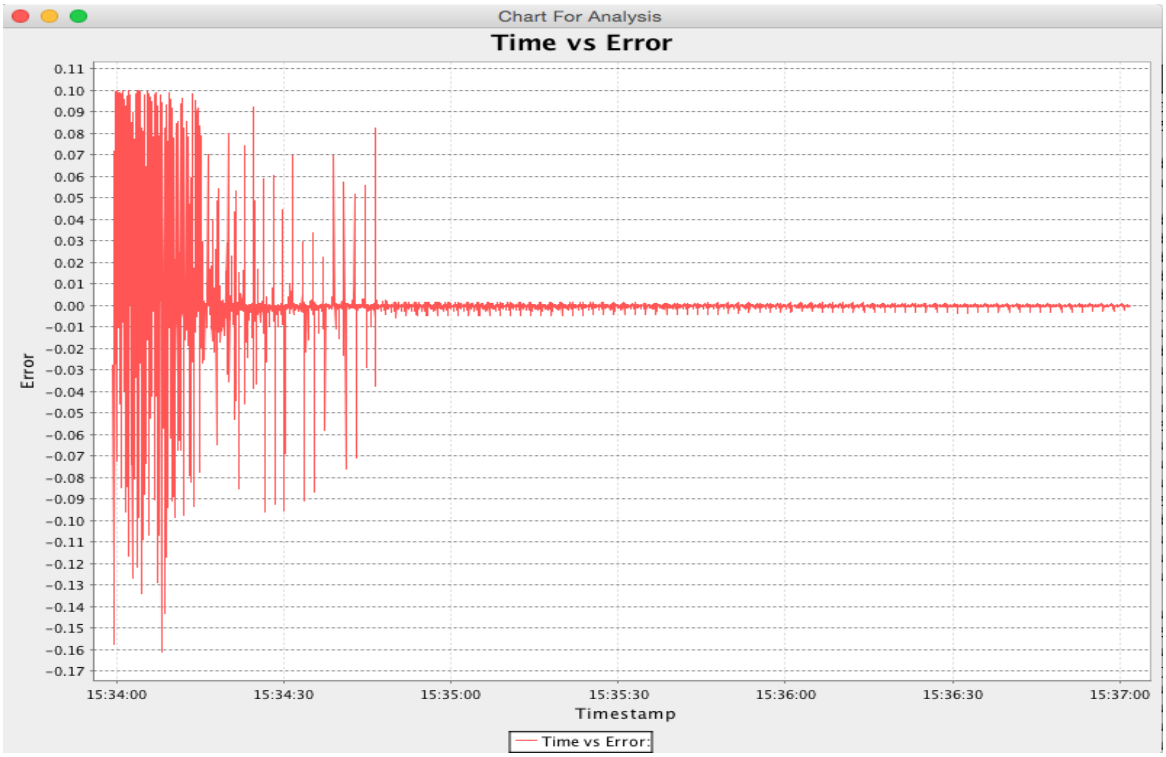
Confusion Matrix : Data Set 2 (Face Recognition)



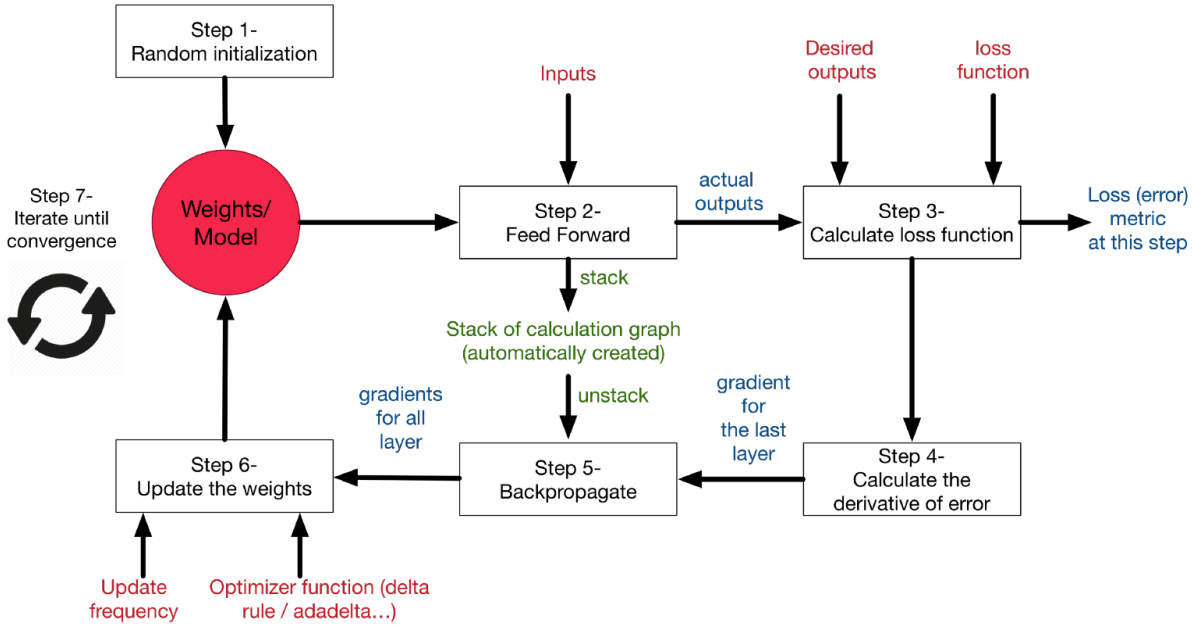
Output : Percentage of predicted output correct = 82%

```
Iteration:99     Folder:9     Image:5     Error:-7.061100752443338E-4
Iteration:99     Folder:10     Image:5     Error:-2.2278244343015719E-4
Training has been done!
1 image,  The maximum value: 0.9853701375773115,    Predicted Subject number 1,    Original Subject Number: 1
2 image,  The maximum value: 0.07914390293439631,    Predicted Subject number 8,    Original Subject Number: 1
3 image,  The maximum value: 0.0028606273708136607,    Predicted Subject number 1,    Original Subject Number: 1
4 image,  The maximum value: 0.9998362299640284,    Predicted Subject number 2,    Original Subject Number: 2
5 image,  The maximum value: 0.7663367953152196,    Predicted Subject number 2,    Original Subject Number: 2
6 image,  The maximum value: 0.5787601188915803,    Predicted Subject number 2,    Original Subject Number: 2
7 image,  The maximum value: 0.9223273464916677,    Predicted Subject number 2,    Original Subject Number: 2
8 image,  The maximum value: 0.9902561401240568,    Predicted Subject number 3,    Original Subject Number: 3
9 image,  The maximum value: 0.546467695449406,    Predicted Subject number 3,    Original Subject Number: 3
10 image,  The maximum value: 0.5719879682246147,    Predicted Subject number 3,    Original Subject Number: 3
11 image,  The maximum value: 0.891863160559619,    Predicted Subject number 8,    Original Subject Number: 3
12 image,  The maximum value: 0.6635024326389081,    Predicted Subject number 4,    Original Subject Number: 4
13 image,  The maximum value: 0.7434724050942615,    Predicted Subject number 4,    Original Subject Number: 4
14 image,  The maximum value: 0.9994794480882019,    Predicted Subject number 4,    Original Subject Number: 4
15 image,  The maximum value: 0.06837722563234032,    Predicted Subject number 5,    Original Subject Number: 5
16 image,  The maximum value: 0.9999662399624324,    Predicted Subject number 5,    Original Subject Number: 5
17 image,  The maximum value: 0.7304404909799813,    Predicted Subject number 5,    Original Subject Number: 5
18 image,  The maximum value: 0.9961095008330396,    Predicted Subject number 6,    Original Subject Number: 6
19 image,  The maximum value: 0.49651530033130853,    Predicted Subject number 6,    Original Subject Number: 6
20 image,  The maximum value: 0.750185123817891,    Predicted Subject number 7,    Original Subject Number: 6
21 image,  The maximum value: 0.8557857067337523,    Predicted Subject number 8,    Original Subject Number: 6
22 image,  The maximum value: 0.9961611308711182,    Predicted Subject number 7,    Original Subject Number: 7
23 image,  The maximum value: 0.9904924676455812,    Predicted Subject number 7,    Original Subject Number: 7
24 image,  The maximum value: 0.004595802072927042,    Predicted Subject number 7,    Original Subject Number: 7
25 image,  The maximum value: 0.998628460275053 9,    Predicted Subject number 8,    Original Subject Number: 8
26 image,  The maximum value: 0.01681335231425537,    Predicted Subject number 1,    Original Subject Number: 8
27 image,  The maximum value: 0.7070780347701531,    Predicted Subject number 9,    Original Subject Number: 8
28 image,  The maximum value: 0.99236905746383,    Predicted Subject number 9,    Original Subject Number: 9
29 image,  The maximum value: 0.8462903804853865,    Predicted Subject number 9,    Original Subject Number: 9
30 image,  The maximum value: 0.6858369564060532,    Predicted Subject number 9,    Original Subject Number: 9
31 image,  The maximum value: 0.8305497418060761,    Predicted Subject number 9,    Original Subject Number: 9
32 image,  The maximum value: 0.9988174581963732,    Predicted Subject number 10,    Original Subject Number: 10
33 image,  The maximum value: 0.9999251259136981,    Predicted Subject number 10,    Original Subject Number: 10
34 image,  The maximum value: 0.9769853287178399,    Predicted Subject number 10,    Original Subject Number: 10
The percentage of testing data correct =82%
```

Graph analysis of Data Set 2 (Error in backpropagation vs Time)



Complete Flow diagram of our Neural Network



CONCLUSION AND OPTIMIZATION

Conclusion

- To increase the efficiency of our Neural Network system, we will need to train the data for more iterations.
- The Learning rate range best suited for our system based on our manual tests is **0.001 - 0.2** with 0.001 being the best and 0.2 being the worst.

Optimization

- *OverFitting of the data* : **Overfitting** is " the production of an analysis that corresponds too closely or exactly to a particular set of data, and may therefore fail to fit additional data or predict future observations reliably.
    a. By testing both the dataset we have checked the OverFitting of data. System was well balanced that it was able to very accurate output.
- To further improve the performance of our system, we can use multi-threading.
- We can also improve the performance of our neural network by increasing the number of hidden layers.

References

- Neural and Deep Learning - **Michael Nielsen**
  http://neuralnetworksanddeeplearning.com/chap1.html
- Excellent book Nature of code - **Daniel Shiffman**
  https://natureofcode.com/book/chapter-10-neural-networks/
- How to make Neural Network - **Daniel Shiffman**
  https://www.youtube.com/watch?v=ntKn5TPHHAk
- Optimising Learning Rate - **David Mack**
  https://medium.com/octavian-ai/which-optimizer-and-learning-rate-should-i-use-for-deep-learning-5acb418f9b2
- Make your own Neural Network. A gentle journey through the Mathematics of Neural Networks. Create space independent publishing platform, 2016 – Tariq Rashid
- Neural Networks: A Systematic Introduction,1996 - **Raúl Rojas**