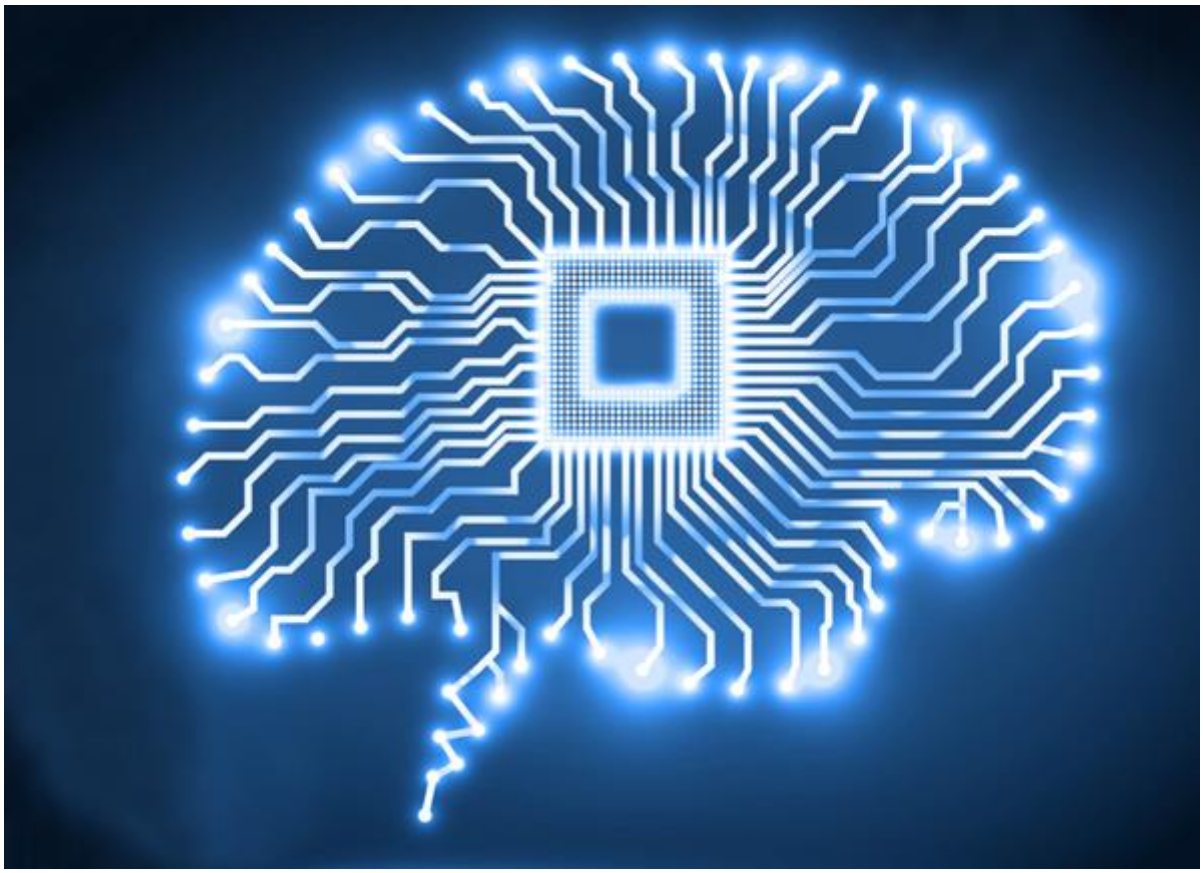


INFO 6205 Neural Network Project Report

--- Digit Recognition

--- Object Recognition

Team Members

DinakaraSaiSantosh Burugupalli. (001491075)

Vivek Shakya (001495865)

Project GitHub Link

https://github.com/dburugupalli/INFO6205-FinalProject

Introduction

What is Neural Network?

A neural network is a “connectionist” computational system. The system is basically a set of algorithms, modelled loosely after the human brain, that are designed to recognize patterns. The idea is to train your model the same way a human brain learns. Learning system is called train the neural network. Here comes the concept of dataset, where we teach the system how it should perform. For example, to recognize digits, we train the system using various digits from 0-9. One of the key elements of a neural network is its ability to *learn*.

A neural network is not just a complex system, but a complex *adaptive* system, meaning it can change its internal structure based on the information flowing through it. This process is achieved by updating the weights.

In Fig 1.0 below every link with forward arrow is a connection with some random weights assigned to it. If the output generated is correct then there is no need to change the weights, otherwise we go back adjust the weights and again generate the output, repeat this until we reach closer to the actual output. This is also called as backpropagation.

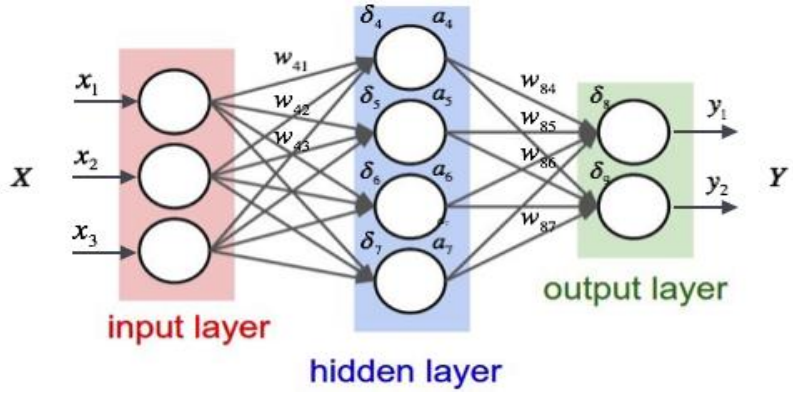
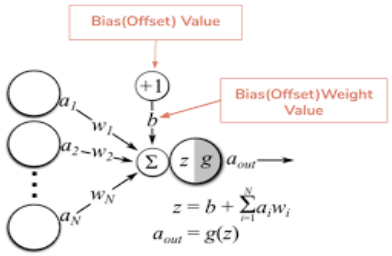


Fig_1: A simple Neural Network



Calculating value of each Neuron at Hidden Layer.

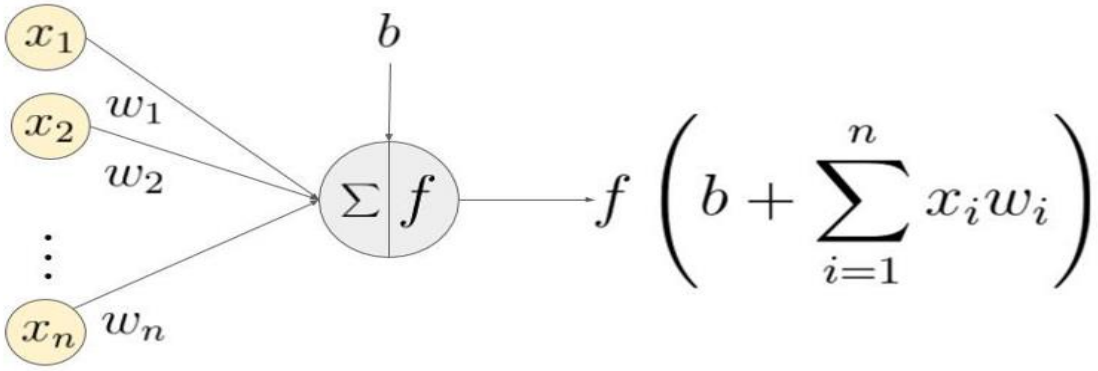


Fig_2: Output of a single Neuron.

- Above is an example of the neuron showing the input (x1- xn), their corresponding weights (w1-w0), a bias (b) and the activation function f applied to the weighted sum of the inputs.

Activation Function

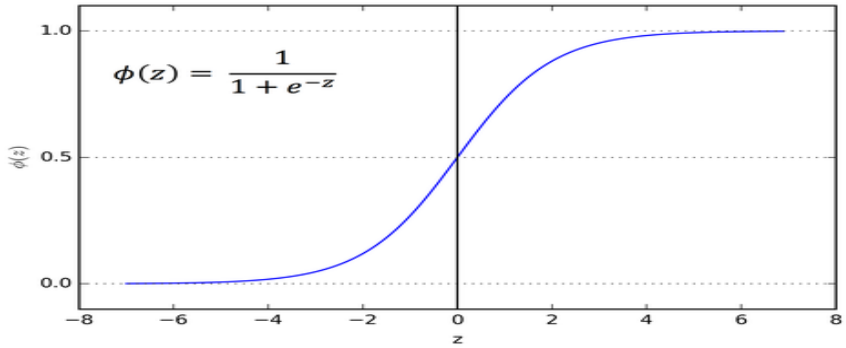


Fig_3: Output of a single Neuron

- Figure above displays the sigmoid function. It has a huge advantage over linear and step activation.

$$f'(x) = f(x)(1 - f(x))$$

- It is continuously differentiable. The above function is the derivative of above activation function.

Aim of the Project

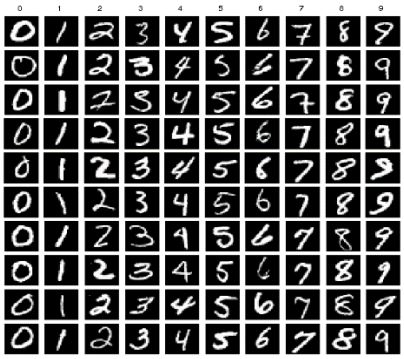
Designed the Artificial Neural Network

- Flexible, Dynamic Neural Network Model.
- Customizing the hidden layers, and Number of Hidden layers as per the user.
- A model which can be mingled with many datasets with less processing.
- Generate Confusion Matrix and various other parameters on the fly.
- MNIST dataset, Fashion Dataset, IRIS dataset from Kaggle.

Datasets used in the Project

In the project, datasets used are MNIST, object-based dataset and IRIS from Kaggle.

- MNIST dataset



- Fashion Dataset:

- IRIS Dataset:

Iris Versicolor | Iris Setosa | Iris Virginica

Project Description

Created a Neural Network Model using Object Oriented Programming concepts and Designs.

Like Considering Layer as an array of neurons and weights, inputs and outputs being the attributes.

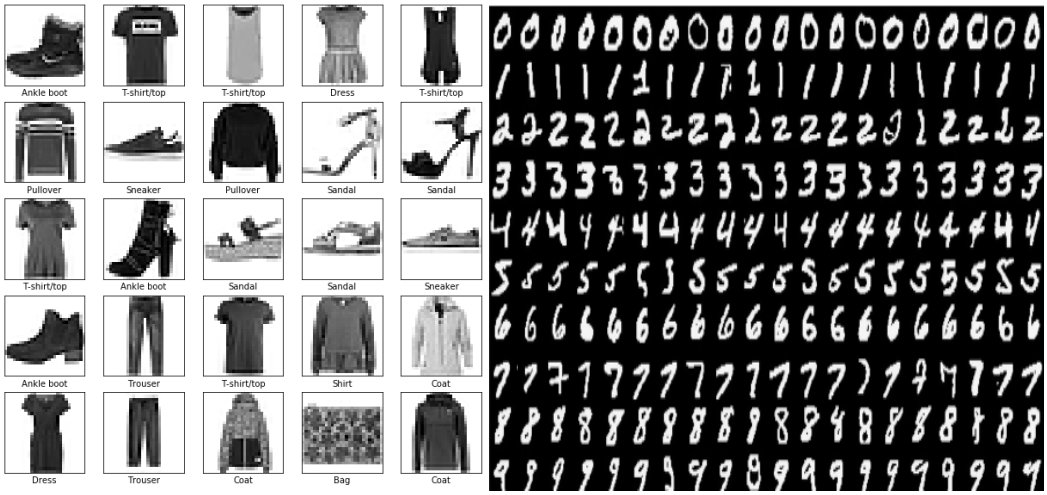
We used matrix manipulation concepts to achieve the output of each neuron. Subsequently each layer.

Using the above designed model, and CSV manipulation and Some image processing we achieved

1. Digit Recognition
   - The basic idea to solve this problem is to feed our neural network a set of 28x28 pixels image of handwritten digits labelled from 0-9.
   - We convert each pixel into binary i.e. 0 and 1, if the image background is black then it's 0 else 1.
   - We train the model using 32000 examples which is part of the MNIST dataset and then test its accuracy using the 10000 testing examples.
   - The more training examples we feed our system the better it will predict.

2. Fashion image recognition
   - It is also similar to the digit recognition problem; we have used the same model but the examples we feed to train the system are the greyscale images of the 10 different clothing/fashion items.
   - We train the system with various pictures of the cloths, footwear. We make sure, the system is trained properly.

- Once we are done with the training, we feed forward the network and predict the desired object.
- Test Images passed

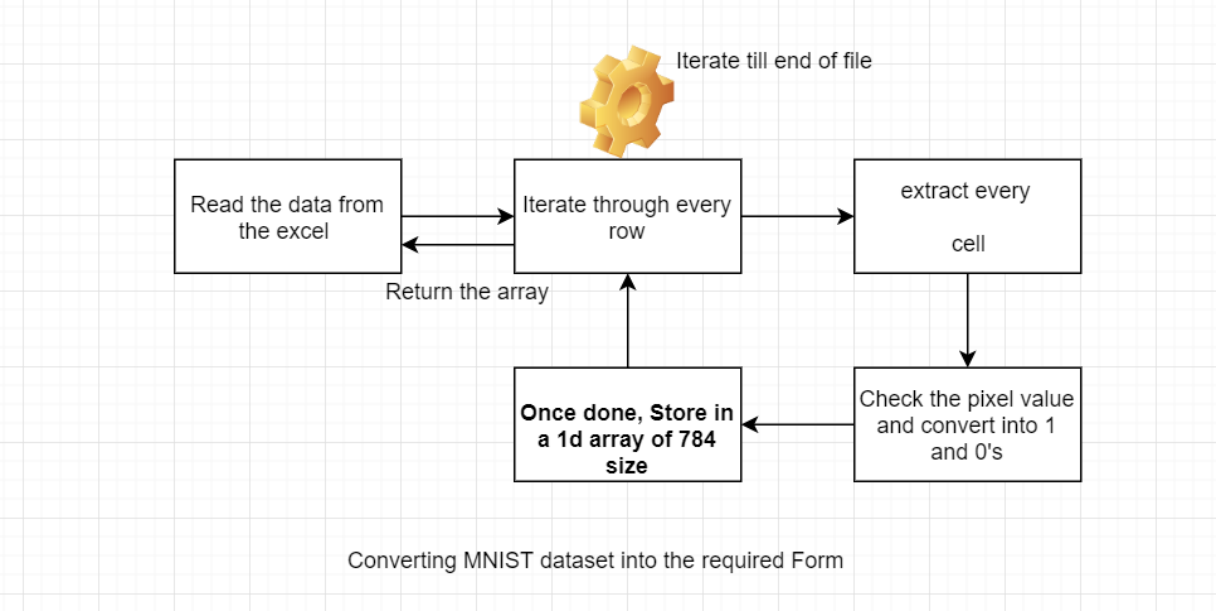


Implementation

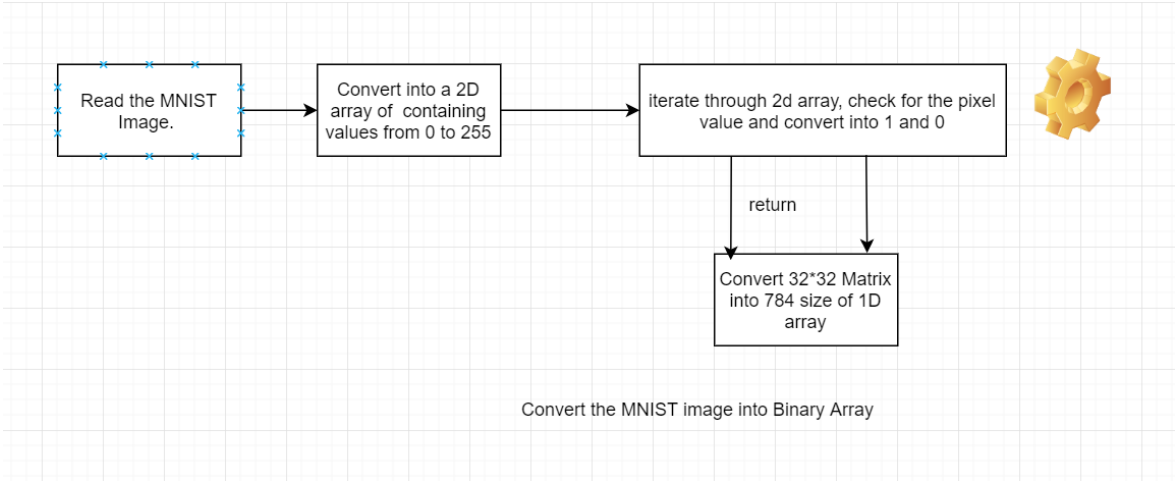
```
Initialise all the network inputs and outputs.
Use Random Number Generator to Initialise weights randomly
repeat
    For Every Pattern present in the Training present

        // Feed Forward the inputs

        for : Layer in the Array of Layer objects

          for: Every neuron in the Layer

            1. weight sum of inputs to the neuron.
            2. Pass it through activation Function
            3. Calculate the activation for the node.

            end.
        end.

        // Feed Forward is completed Successfully. Propagate Backwards.

        for : every output in the output Layer
            calculate the error signal (Current-Target)
        end.

        for all hidden Layers
            for every neuron in every Layers
                1. calculate the nodes signal error
                2. Update each nodes weights int the network
            end
        end
    end

while(iterations > 0)
```

Overall Pseudo Code for Neural Net Train.

Converting the dataset as the Neural Network



Converting MNIST dataset into the required Form



Convert the MNIST image into Binary Array

Implementation & Backpropagation

One initializes how many Hidden layers user wants in this class in the constructor if the user needs n hidden layers, he can configure the same.

For this Analysis we used 2 hidden Layers with

The table Below shows **neurons** in each layer

Dataset	Input	Hidden Layer1	Hidden Layer2	Output
IRIS	4	6	6	3
MNIST dataset	784	315	20	10
Object dataset	784	30	15	10

```
public BackpropNeuralNetwork(int inputSize, int hiddenSize1, int outputSize) {
    layers = new Layer[2];
    layers[0] = new Layer(inputSize, hiddenSize1);
    layers[1] = new Layer(hiddenSize1, outputSize);
}
```

Constructor with 1 hidden layer

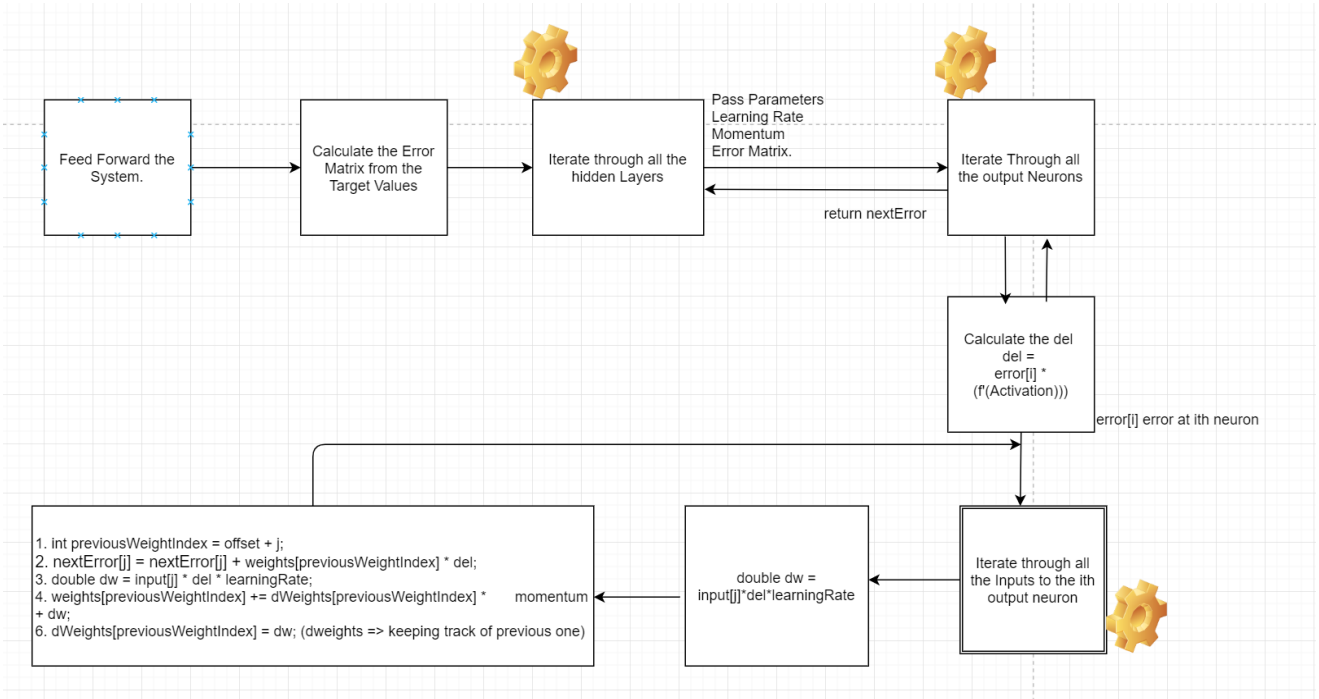
```
// constructor with two hidden layers
public BackpropNeuralNetwork(int inputSize, int hiddenSize1,int hiddenSize2, int outputSize) {
    layers = new Layer[3];
    layers[0] = new Layer(inputSize, hiddenSize1);
    layers[1] = new Layer(hiddenSize1, hiddenSize2);
    layers[2] = new Layer(hiddenSize2, outputSize);
}
```

Constructor with 2 hidden layer

- Run method in this Helps to Feed Forward the system.
- Train method continuously uses back propagation to train the system.

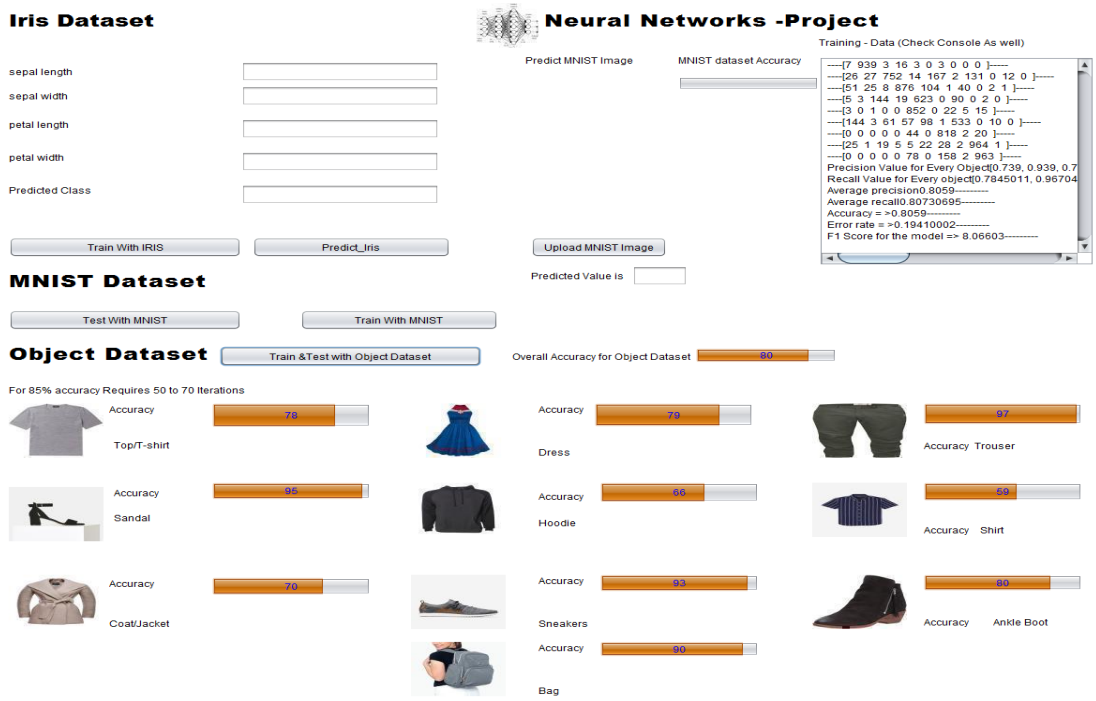
It is the process where the neural network backpropagates information with the error, in reverse through the network, so that it can alter the parameters. This is a step by step process.

1.) It makes wild guess, what could be the data?
2.) The is measured in terms of loss function.
3.) Wrongheaded parameters (weights are updated) by propagating the error.



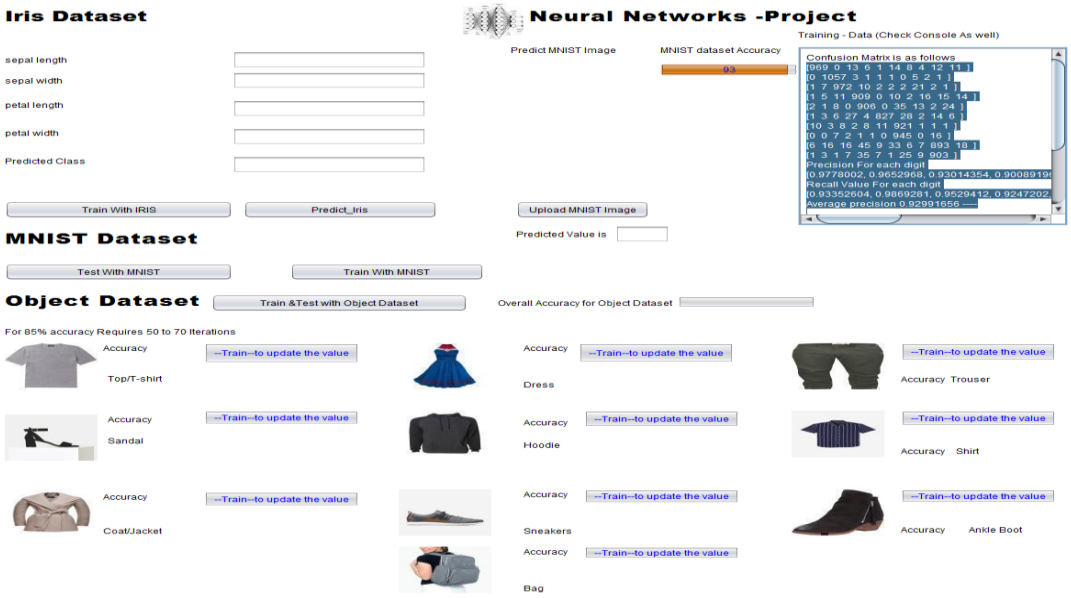
Output

Fashion-Dataset



Object Based Dataset: Overall Accuracy is 80 percent for 15 iterations.

MNIST Dataset



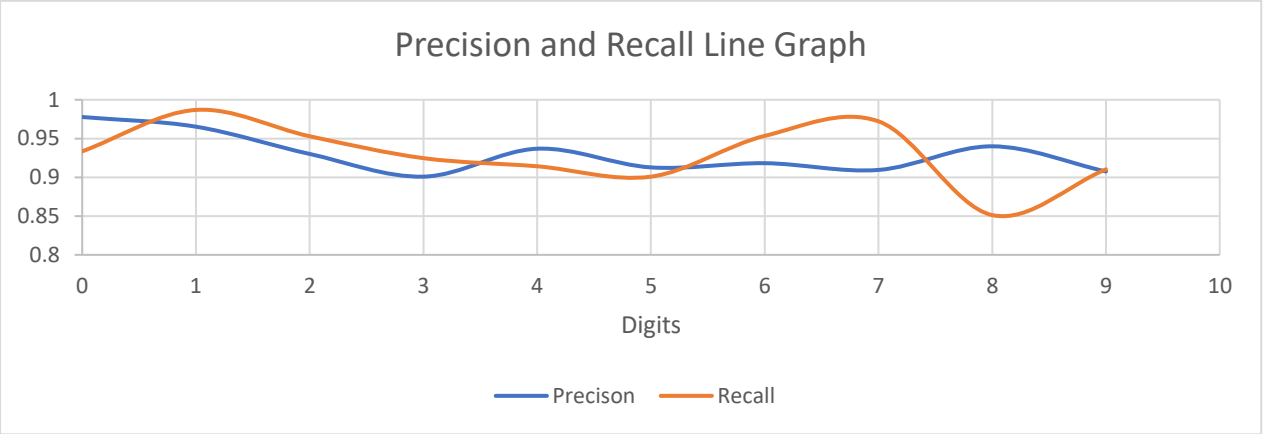
Overall Accuracy of 93 percent for MNIST dataset.

Mathematical Analysis

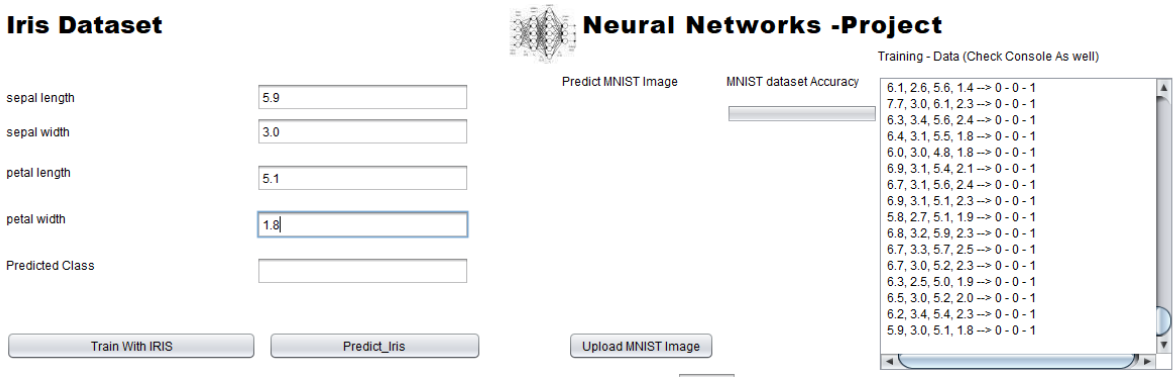
Confusion Matrix for the MNIST Test dataset

	Predicted											
Actual	0	1	2	3	4	5	6	7	8	9		Recall
0	969	0	13	6	1	14	8	4	12	11		0.9335
1	0	1057	3	1	1	1	0	5	2	1		0.9869
2	1	7	972	10	2	2	2	21	2	1		0.9529
3	1	5	11	909	0	10	2	16	15	14		0.924
4	2	1	8	0	906	0	35	13	2	24		0.914
5	1	3	6	27	4	827	28	2	14	6		0.9008
6	10	3	8	2	8	11	921	1	1	1		0.953
7	0	0	7	2	1	1	0	945	0	16		0.972
8	6	16	16	45	9	33	6	7	893	18		0.851
9	1	3	1	7	35	7	1	25	9	903		0.910
Precision	0.9778	0.965296 8	0.9301435 4	0.9008919 6	0.9369183	0.9128035 3	0.9182452 6	0.9095284	0.94	0.907538	Accuracy	0.9302
											F1 Score	0.92

Precision vs Recall Graphs for 100 iterations



IRIS Dataset Training and Prediction

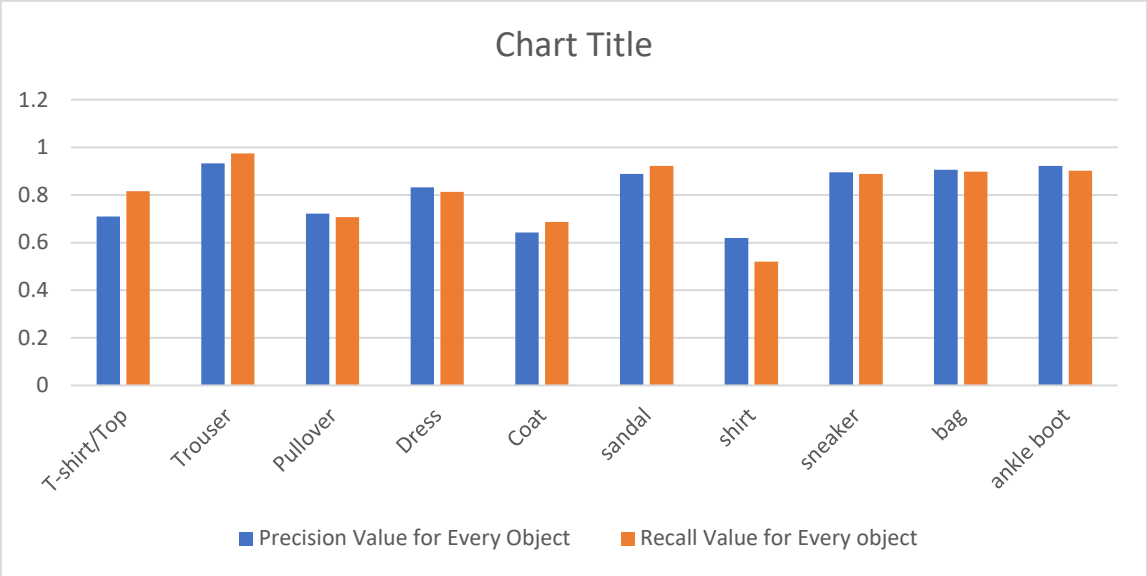


Confusion Matrix for the Object Test dataset

Actual	Predicted										recall
	T-shirt/Top	Trouser	Pullover	Dress	Coat	sandal	shirt	sneaker	bag	ankle boot	
T-shirt/Top	709	0	11	14	0	1	133	0	1	0	0.81588
Trouser	2	933	2	14	1	0	5	0	0	0	0.974922
Pullover	28	24	722	15	107	3	99	0	24	0	0.706458
Dress	44	26	6	832	77	2	34	0	2	0	0.813294
Coat	6	9	146	39	643	0	78	0	13	2	0.686966
sandal	0	0	1	0	1	889	2	47	4	20	0.922199
shirt	180	6	101	75	164	0	619	0	45	0	0.520168
sneaker	0	0	0	0	0	52	0	895	5	55	0.888779
bag	30	1	11	11	7	11	30	1	906	1	0.897919
ankle boot	1	1	0	0	0	42	0	57	0	922	0.901271

	0.709	0.933	0.722	0.832	0.643	0.889	0.619	0.895	0.906	0.922	
	Accuracy	0.807									
	F1-Score	0.809									

Precision vs Recall Graphs for 15 iterations on Object dataset



Conclusion and Future Implementations

- On the basis of the experiments we have learned that neural accuracy of the neural network increases when we train the system for larger iterations.
- For example the accuracy we observed for the fashion dataset was close to 67% when trained over 10 iterations but it increased to 80 when number of iterations were increased to 15.
- We can make our neural network more optimised by different activation functions.
- Advanced Image Processing Techniques, for better object detection.

References

1. Understanding activation functions in java
   https://medium.com/the-theory-of-everything/understanding-activation-functions-in-neural-networks-9491262884e0
2. Deep Learning
   https://www.investopedia.com/terms/d/deep-learning.asp
3. Beginners guide to Neural Networks and Deep Learning
   https://skymind.com/wiki/neural-network
4. Nature of code Daniel Shiffman
   https://natureofcode.com/book/chapter-10-neural-networks/