**Machine Learning (CS-471)**

**Project Deliverables**

**Project Final Report**

**Class**

BESE 10-A

**Submitted to**

Dr. Seemab Latif

**Group Members**

|  |  |
| --- | --- |
| **Names** | **CMS** |
| Muhammad Hamza Imran | 314378 |
| Ameer Hamza Baig | 283845 |
| Ahsan Akbar | 284918 |

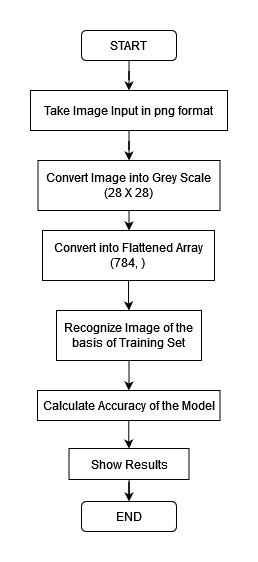
Project Name: Digit Recognizer

**Introduction**

The handwritten digit recognition is the ability of computers to recognize human handwritten digits. It is a hard task for the machine because handwritten digits are not perfect and can be made with many different ways. The handwritten digit recognition is the solution to this problem which uses the image of a digit and recognizes the digit present in the image.

Handwritten digits are a very practical problem. It is largely used by the postal services to recognize the digits and addresses written of the posts. For example, the postal service in the US processes around 500 million pieces of mails per day but using ML recognizer for recognition of handwritten addresses and handwritten digits means that postal works don’t individually have to handle and examine millions and millions of mails and parcels every day.

**Our Model**



**Data-Set Used**

We are going to used MNIST Data-Set by the National Institute of Standards and Technology. Data-Set is available their official website under URL <http://yann.lecun.com/exdb/mnist/> .

Data is available in the form of 4 files train images, train labels, test images and test labels.

**Model Evaluation**

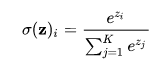
We Trained our Model on MNIST dataset. For Testing purposes, we divided the dataset into two parts. One is for the training and the other is for the testing purpose. Our training dataset is actually of 60000 examples and testing data is of 10000 examples.

We have trained our model using neural networks and the TensorFlow for it.

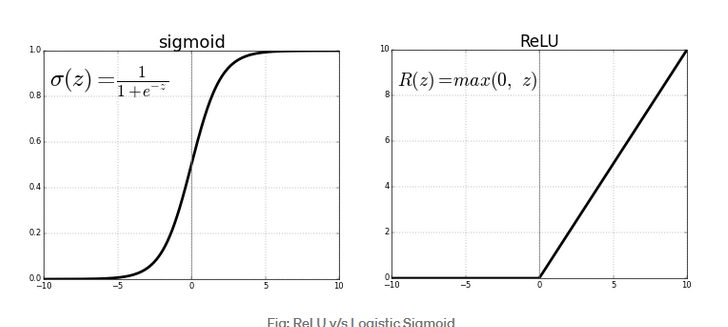
**Activation Functions**

In neural network we have used 2 different activation functions for the accurate results.

**Softmax** is used in the hidden layers of the neural network. It is a mathematical function that converts a vector of numbers into a vector of probabilities, where the probabilities of each value are proportional to the relative scale of each value in the vector.



The rectified linear activation function or **ReLU** in used in the last layer. It is a piecewise linear function that will output the input directly if it is positive, otherwise, it will output zero.



**Optimizer**

We have used Adam Optimizer because it is a replacement optimization algorithm for stochastic gradient descent for training deep learning models. Adam combines the best properties of the AdaGrad and RMSProp algorithms to provide an optimization algorithm that can handle sparse gradients on noisy problems.

**Accuracy Metrix**

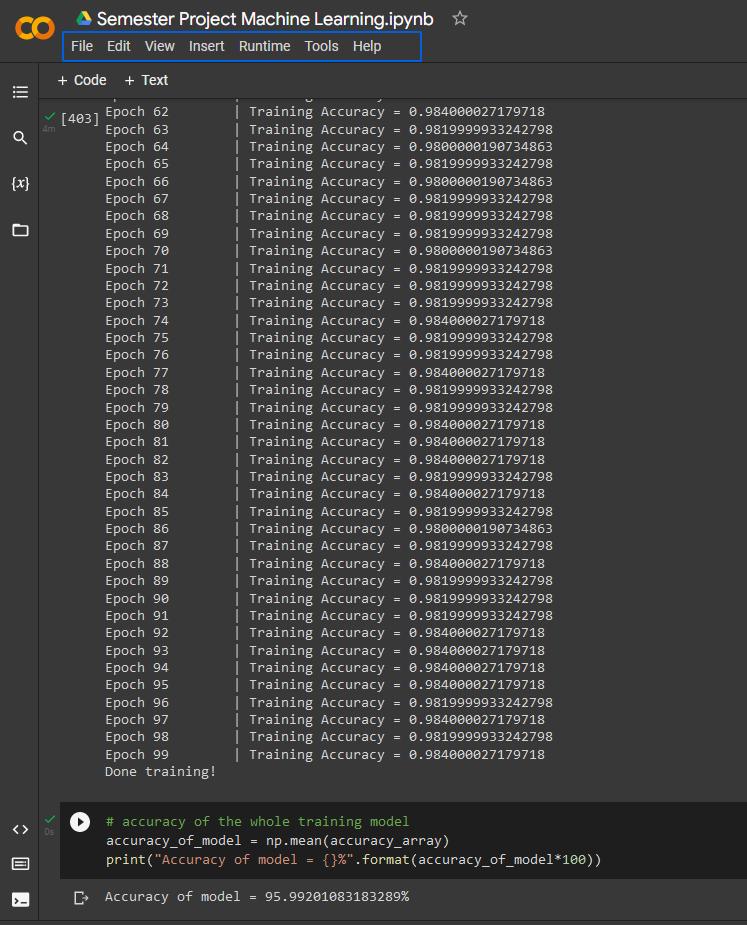
We have used **tf.reduce.mean** a TensorFlow method to calculate the accuracy matrix.

**Algorithm**

1. First of all, it takes data from the dataset files provided and visualize and clean the dataset.
2. Creating the tensors using the TensorFlow library.
3. Providing with the hyperparameters used for the training of the dataset.
4. Layers setup of the 2 hidden layers and other input and output layer.
5. Softmax function between the input and the hidden layers.
6. Relu function is used between the last hidden layer and the output layer.
7. Calculating the accuracy of the model after training.

**Fine Tuning**

We changed the following parameters and get the following accuracy

1. ****First time

No. of epoch = 100

Iterations = 100

Learning rate = 0.0001

Batch size = 500

Accuracy of Model = 95.99%

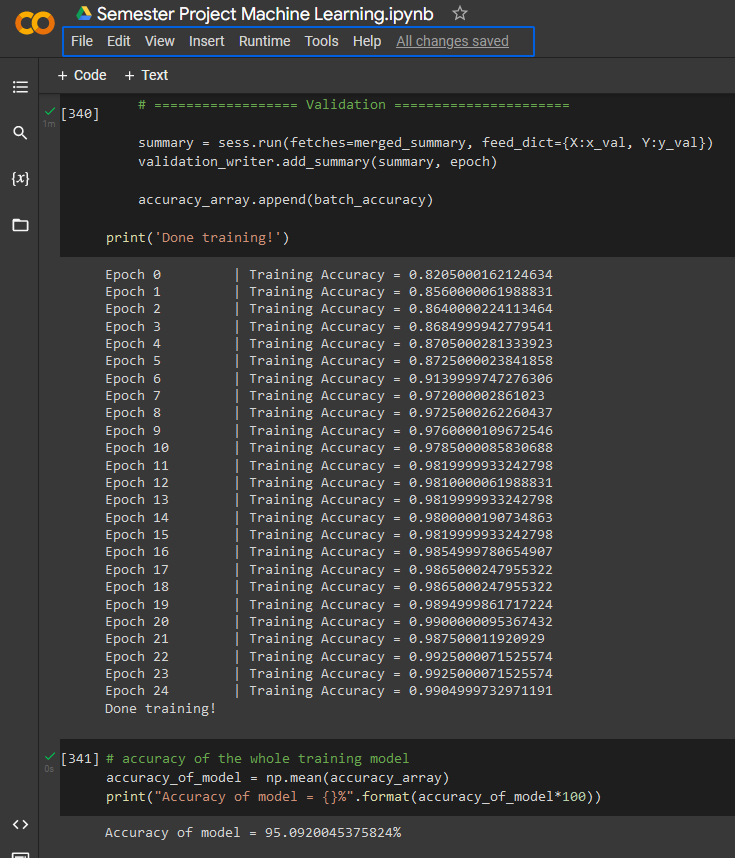
1. Second time

No, of epoch = 25

Iterations = 25

Batch Size = 2000

Accuracy of Model = 95.09%

****

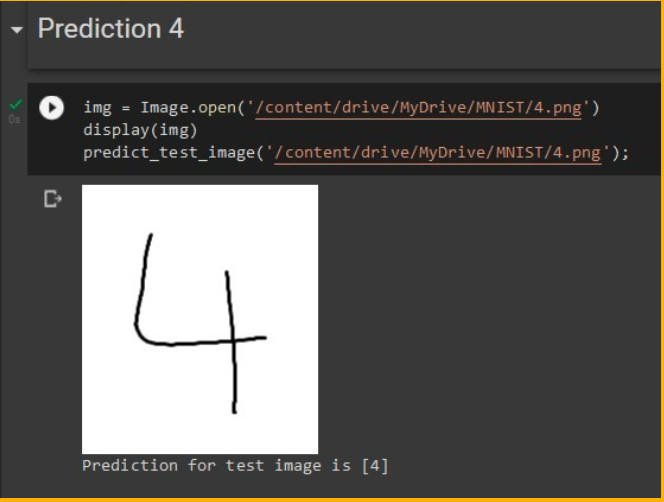
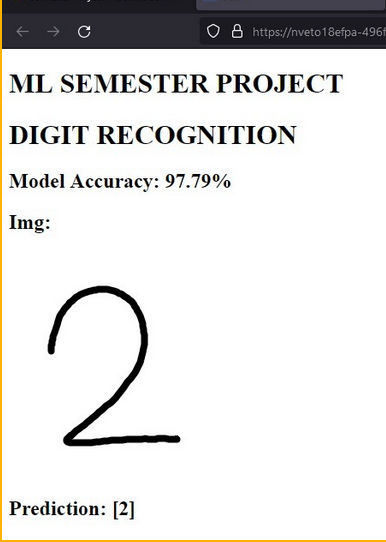
**Final Model**

|  |  |
| --- | --- |
| **Hyperparameter** | **Value** |
| **Number of Epochs** | **50** |
| **Number of iterations** | **50** |
| **Learning rate** | **0.001** |
| **Activation function** | **softmax for all layers and relu for last output layer** |
| **Number of hidden layers in NN** | **2** |
| **Number of neurons in each hidden layer** | **1st = 512, 2nd = 64** |
| **Training, validation, testing split** | **60000, 10000, 50000** |
| **Batch size** | **1000** |
| **Number of features (feature engineering)** | **784** |
| **Accuracy/Loss/Error of the model** | **97.88%** |

**Analysis**

When we changed the epoch and the batch size our accuracy decreases in both the cases. In this case our model is over fitted to the training data.

**Results**

****