PROJECT REPORT

A NOVEL METHOD FOR HANDWRITTEN DIGIT RECOGNITION

submitted by

PNT2022TMID04401

Jagadeeshwaran VV -737819CSR068

Logesh R -737819CSR095

Maheshvar A -737819CSR100

Manoj S -737819CSR102

TABLE OF CONTENTS

CHAPTER 1
INTRODUCTION

a. PROJECT OVERVIEW

Machine learning and deep learning are critical components of computer technology and artificial intelligence. Human effort can be reduced by using deep learning and machine learning in recognising, learning, predicting, and many other areas.

Handwritten Digit Recognition refers to computer systems' ability to recognise handwritten digits from various sources such as images, documents, and so on. This project aims to enable users to use machine learning to reduce manual tasks in digit recognition.

b. PURPOSE

Digit recognition systems can recognise digits from various sources such as emails, bank cheques, papers, images, and so on, as well as in various real-world scenarios such as online handwriting recognition on computer tablets or systems, recognising vehicle licence plates, processing bank cheque amounts, numeric entries in forms filled out by hand (tax forms), and so on.

CHAPTER 2

LITERATURE SURVEY

a. EXISTING PROBLEM

The main issue with handwritten digit recognition is that because handwriting varies from person to person, handwritten digits do not always have the same size, width, orientation, and margins. In addition, it would be difficult to distinguish the numbers due to similarities between the numerals, such as 1 and 7, 5 and 6, 3 and 8, 2 and 5, and 2 and 7. The distinctiveness and variety of each person's handwriting also has an impact on the digits' shape and appearance.

b. REFERENCES

Effective Handwritten Digit Recognition using deep Convolution Neural Network (2020)-Bharadwaj Yellapragada, Bhanu Prakash Kolla

The model consists of binary classification and feature extraction with convolution. To extract the features from the image, convolution and max-pooling are used. At the conclusion, the flatten layer will convert the images into a series of values that will be mapped to a dense layer of neurons connected to the layer of neurons responsible for categorical output. The predicted label for the image is the e neuron with the highest value.

Handwritten Digit Recognition using CNN (2019)-Vijayalaxmi R Rudraswamimath, BhavanishankarK

The user can upload either an image of the digit to be detected or data from the MNIST dataset. Pre-processing is performed on the input images. The accuracy of recognised digits is compared using different classifiers, and the result is obtained. The obtained results, as well as the accuracy, are displayed.

Handwritten Digit Recognition (2022)-E. Lavanya

To optimise recognition accuracy and interval, a model of the convolutional neural network is developed and analysed for appropriate differentiation. We propose to investigate the needs of CNN designs with three layers (CNN 3L) and CNN designs with four layers (CNN 4L). For CNN with three layer design, a total of six cases (e one to case 6) are considered.

An improved zone-based hybrid feature extraction model for handwritten alphabets recognition using Euler number (2012)-O. P. Sharma, M. K. Ghose, and K. B. Shah

An Improved Zone-based Hybrid Feature Extraction Model Based on the Euler Number, which not only improves the feature extraction process implemented in Diagonal Based Feature Extraction, but also aids in the efficient classification of the handwritten alphabet.

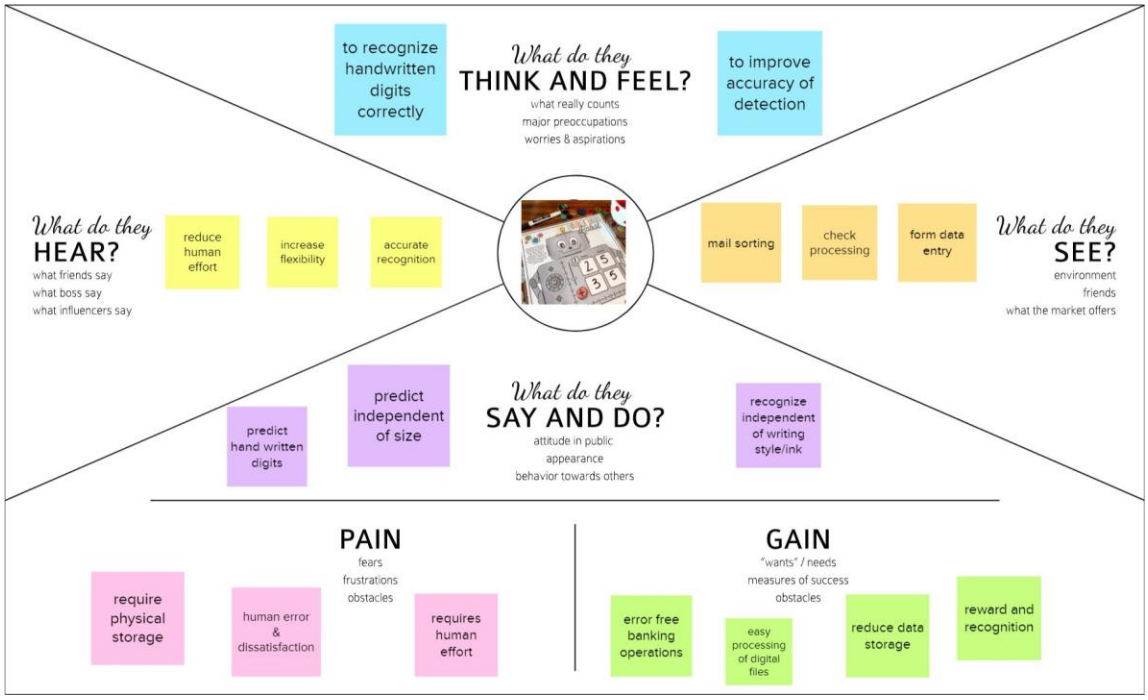
c. PROBLEM STATEMENT DEFINITION

The traffic department has been fighting traffic law violators for years. These criminals endanger not only their own lives, but also the lives of others. Punishing these offenders is critical to preventing others from becoming like them. Identification of these offenders is nearly impossible because the average person cannot write down a reckless driver's licence plate. As a result, the goal of this project is to assist the traffic department in identifying these offenders and, as a result, reduce traffic violations.

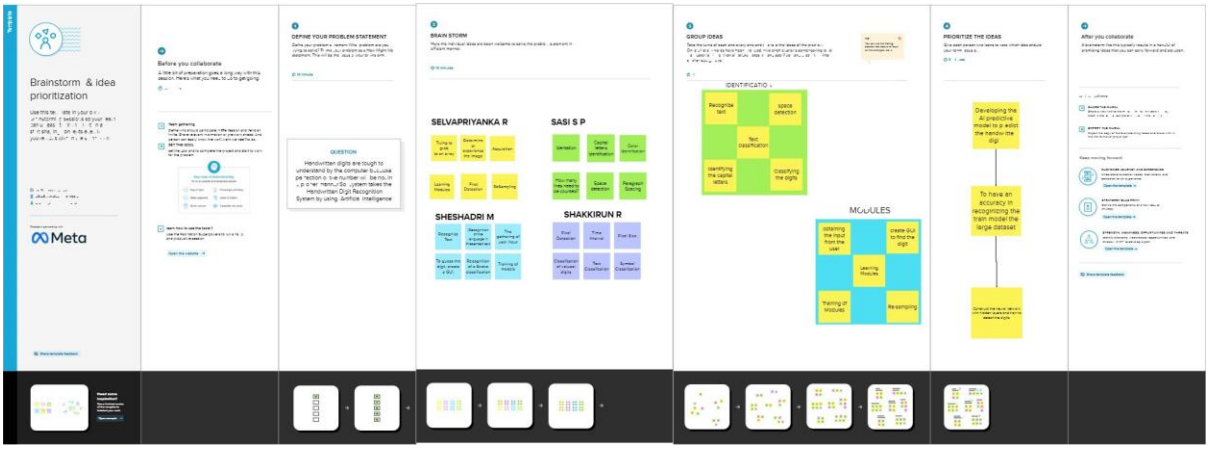
CHAPTER 3

IDEATION AND PROPOSED SOLUTION

a. EMPATHY MAP CANVAS



b.IDEATION & BRAINSTORMING



c. PROPOSED SOLUTION

S. NO	PARAMETER	DESCRIPTION
1	Problem Statement	To develop a programme that recognises handwritten digits.
2	Idea / Solution Description	The application uses an image as input and detects the digits in it accurately.
3	Novelty / Uniqueness	Instead of recognising every word, the app recognises only the digits.
4	Social Impact / Customer Satisfaction	This application reduces the number of manual tasks that must be completed. This increases workplace productivity.
5	Business Model	To recognise vehicle licence plates, the application can be integrated with traffic surveillance cameras. To effectively recognise pin codes, the application can be integrated with postal systems.
6	Scalability of the Solution	To increase efficiency, the application can easily be scaled to accept multiple inputs and process them in parallel.

d. PROBLEM-SOLUTIONTION FIT

Problem Statement: Handwritten Digit Recognition:

MNIST ("Modified National Institute of Standards and Technology") is an unofficial "hello-world" computer vision dataset. This is a collection of thousands of handwritten images used to train Machine Learning classification models. As part of this problem statement, we will use TensorFlow to train a multi-layer Perceptron to recognise handwritten digits.

MNIST Dataset Description:

Because everyone in the world has their own writing style, handwriting recognition is one of the most intriguing research projects currently underway. It automatically quantifies and understands handwritten digits or characters. Because of advances in science and technology, everything is being digitalized in order to reduce human effort.

As a result, handwritten digit recognition is required in many real-time applications. The MNIST data set, which contains 70000 handwritten digits, is widely used in this recognition process. To train these images and create a deep learning model, we use artificial neural networks. We deep-learning is developing a website where users can upload an image of a handwritten digit. This image is analysed by the model, and the detected result is returned to the user interface.

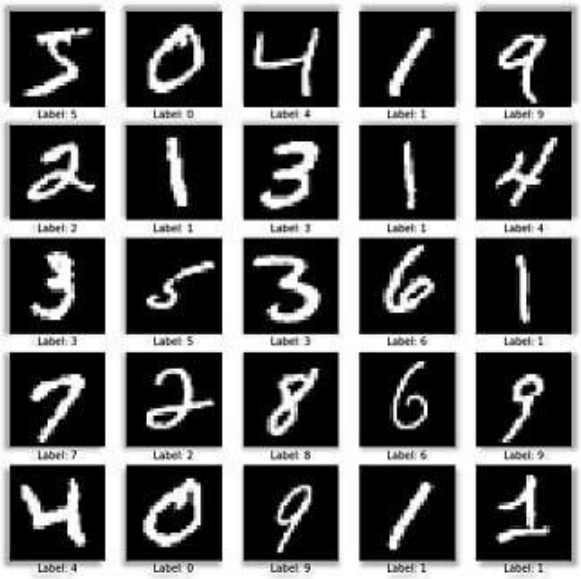
The MNIST Handwritten Digit Recognition Dataset contains 60,000 labelled handwritten digit pictures for training and 10,000 for testing. Each image has a height of 28 pixels and a width of 28 pixels, for a total of 784 (2828) pixels. Each pixel is associated with a single pixel value. It indicates whether a pixel is bright or dark (larger numbers indicate darker pixel). This pixel value is an integer between 0 and 255.

PROCEDURE:

- Install the latest TensorFlow library.
- Prepare the dataset for the model.
- Develop Single Layer Perceptron model for classifying the handwritten digits.
- Plot the change in accuracy per epochs.
- Evaluate the model on the testing data.
- Analyze the model summary.
- Add a hidden layer to the model to make it Multi-Layer Perceptron.
- Add Dropout to prevent over fitting and check its effect on accuracy.
- Increasing the number of Hidden Layer neurons and check its effect on accuracy.

- Use different optimizer and check its effect on accuracy.
- Increase the hidden layers and check its effect on accuracy

Manipulate the batch size and epochs and check its effect on accuracy.



MNIST is a widely used dataset for handwritten digit recognition. There are 60,000 training images and 10,000 test images in the dataset. Artificial neural networks can all most closely mimic the human brain and are an important component in the image processing field.

Handwritten digit recognition using the MNIST dataset is a large project completed with the assistance of a Neural Network. It recognises scanned images of handwritten digits.

Our handwritten digit recognition system goes a step further in that it can now recognise handwritten numbers written directly on the screen with the aid of an integrated GUI in addition to detecting them in scanned images.

Approach:

We will approach this project by using a three-layered Neural Network.

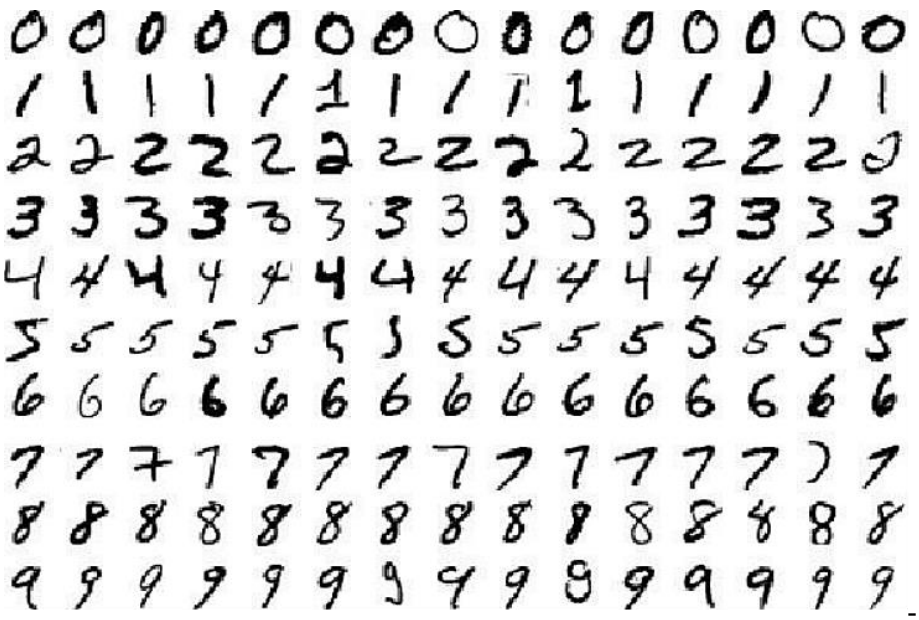
- **The input layer:** It distributes the features from our example layers to the following layer so that the subsequent layer's activations can be calculated.
- **The hidden layer:** They are made of hidden units called activations providing nonlinear ties for the network. A number of hidden layers can vary according to our requirements.
- **The output layer:** The nodes here are called output units. It provides us with the final prediction of the Neural Network on the basis of which final

predictions can be made.

A neural network is a model of the brain's operations. It is made up of numerous layers with a variety of activations; these activations mimic the neurons in our brain. An attempt is made by a neural network to learn a set of parameters from a set of data that might aid in understanding the underlying relationships. Since neural networks are capable of adapting to changing input, they can produce the best possible results without having to change the output criteria.

METHODOLOGY:

A neural network with one hidden layer and 100 activation units has been put into practise (excluding bias units). A.mat file containing the data was loaded, and features (X) and labels (Y) were then extracted. The characteristics are then rescaled to a range of [0,1] by dividing them by 255 in order to prevent calculation overflow. 10,000 testing cases and 60,000 training examples make up the data. With the training set, feed forward is used to calculate the hypothesis, and back propagation is then used to lower the error between the layers. To combat overfitting, the regularisation parameter lambda is set to 0.1. To find the model that fits the situation the optimizer runs for 70 times.



ALGORITHM:

Forward Propagation Architecture:

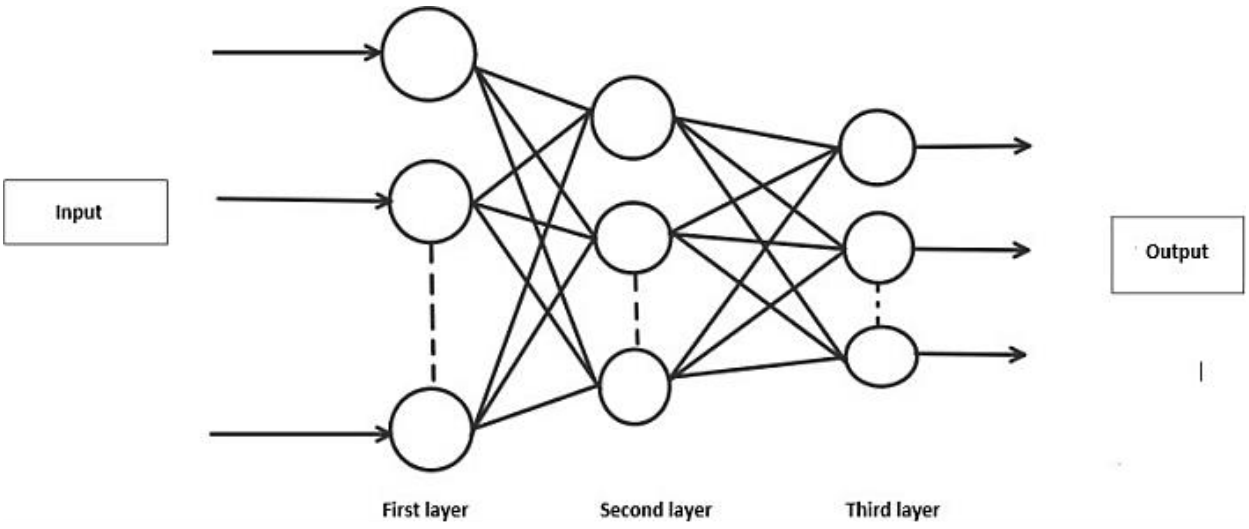
A simple process for the CNN module's feature extraction and picture classification is shown below. The network's input layer, hidden layers, and output layer are all displayed in the architecture. The feature extraction phase of the network involves many layers and uses convolution and sub sampling.

EXPLANATION OF THE PROPOSED SYSTEM

1. The User layer is the top layer in the architecture. The users who interact with the app and get the desired results make up the user layer.
2. The next three layers is the frontend architecture of the application.

The application will be developed using which is the open-source platform for HTML, CSS and JavaScript.

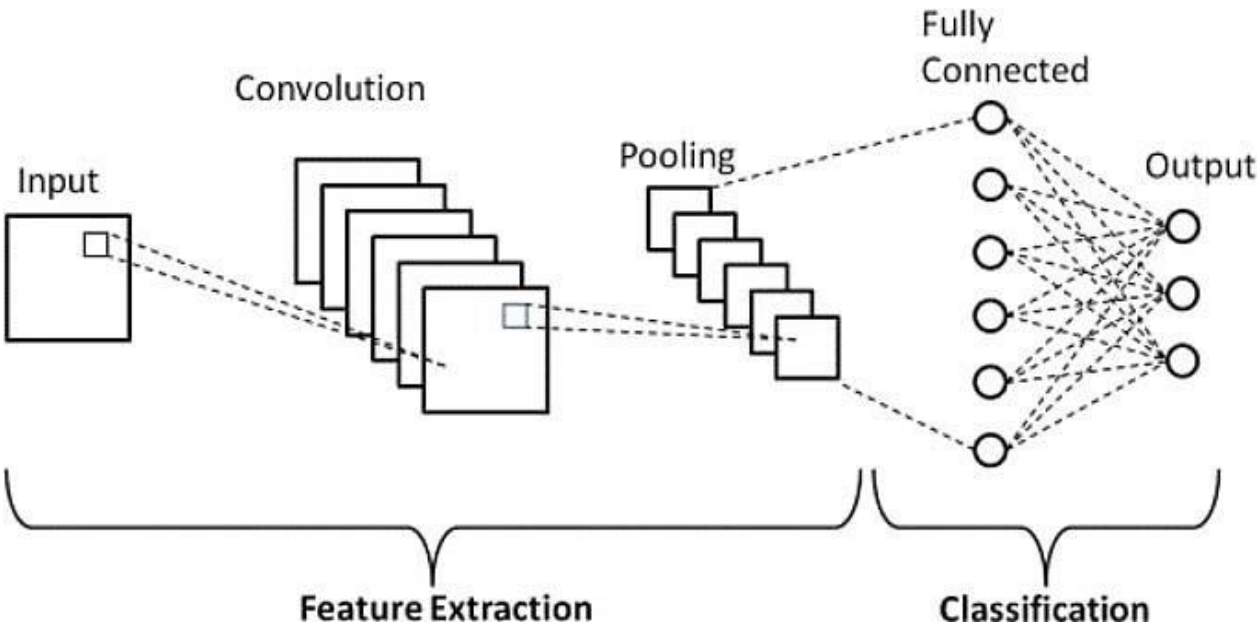
The localhost, which is displayed in the browser, is where the programme is deployed. The user will be able to upload images of the handwritten numbers to the app to have them digitalized. • The business layer, which consists of logical calculations based on the client's request, sits between the database and view layers. The service interface is also included. • Training Data and Test Data make up the backend layer's two datasets. The training set, which consists of 60,000 cases, and the test set, which consists of 10,000 examples, have already been separated out in the MNIST database. Convolution neural network training is the employed training algorithm. By doing this, the trained model will be ready to be used to categorise the digits found in the test data. As a result, we may categorise the digits visible in the photos as: Class 0,1,2,3,4,5,6,7,8,9.



WORKING

1. Neural Networks receive an input and transform it through a series of hidden layers.
2. Each hidden layer is made up of a set of neurons, where each neuron is fully connected to all neurons in the previous layer.

One layer of neurons have complete independence from one another. The "output layer" is the final layer that is completely connected.



Convolution Layer: The foundational component of a CNN is the convolutional layer. The parameters of the layer are made up of a number of learnable filters (or kernels), each of which has a limited receptive field but covers the entire depth of the input volume. Each filter is convoluted across the width and height of the input volume during the forward pass, computing the dot product between each filter entry and the input to create a two-dimensional activation map of the filter.

As a result, the network picks up filters that turn on when it detects a particular kind of feature at a particular spatial location in the input.

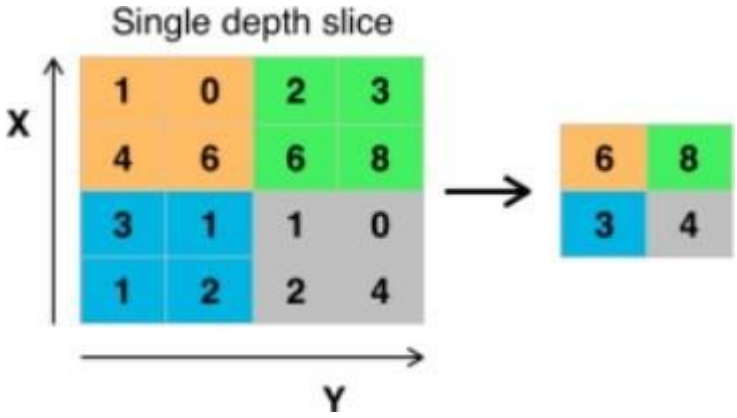
Feature Extraction:

The weights of each neuron in a feature are the same. In this manner, the same feature is recognised by all neurons at various locations in the input image. Limit the number of unrestricted parameters.

Sub sampling Layer: Reducing the overall size of a signal is referred to as sub sampling, also known as down sampling. The spatial resolution of each feature map is decreased by the sub sampling layers. Reduce the impact of noises and shift or variance distortion is accomplished.

Pooling layer: In a Convent design, it is typical to sporadically introduce a Pooling layer between succeeding Convolution layers. Its purpose is to gradually shrink the representation's spatial size in order to simplify the network's computations and parameters while also exerting some control against overfitting. Every depth slice of the input is independently processed by the Pooling Layer, which then applies the MAX operation to resize each slice spatially.

TensorFlow: An open-source machine learning library for both research and production is called TensorFlow. TensorFlow provides developers of all skill levels with APIs for desktop, mobile, web, and cloud applications. To get started, refer to the sections below. We can get text output and sound output by scanning the numerical digit and converting it to png format using the python3 command in terminal.



Pooling layer

RESULT:

We do not consider our results to be flawless, as with any study or project undertaken in the field of machine learning and image processing.

There is always opportunity for improvement in your methods because machine learning is a topic that is continually developing; there will always be a fresh new idea that solves a given problem more effectively. Three models were used to test the application: Multi-Layer Perceptron (MLP), Convolution Neural Network, and (CNN). We obtain a different classifier accuracy with each model, indicating which is superior.

CHAPTER 4

REQUIREMENT ANALYSIS

a. FUNCTIONAL REQUIREMENTS

FR.NO	FUNCTIONAL REQUIREMENTS	SUB REQUIREMENTS
FR-1	Model Creation	Get access to the MNIST dataset
		Analyze the dataset
		Define a CNN model
		Train and Test the Model
FR-2	Application Development	Create a website to let the userrecognize handwritten digits.
		Create a home page to uploadimages
		Create a result page to displaythe results

		Host the website to let the usersuse it from anywhere
FR-3	Input ImageUpload	Let users upload images of various formats.
		Let users upload images of various sizes
		Prevent users from uploadingunsupported image formats
		Pre-Process the image to use it on the model

		Create a database to store allthe input images
FR-4	Display Results	Display the result from the model
		Display input image

		Display the accuracy ac y the result
		Display other possible predictions with their respectiveaccuracy

b. NON FUNCTIONAL REQUIREMENTS

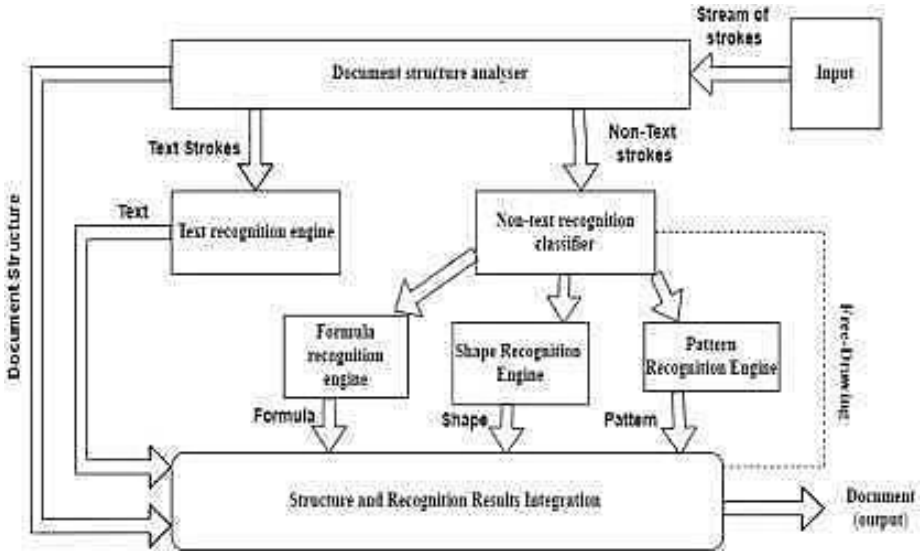
NFR	NON-FUNCTIONAL REQUIREMENTS	DESCRIPTION
NFR-1	Usability	The application must be usableon all devices
NFR-2	Security	The application must protectuser uploaded image
NFR-3	Reliability	The application must give as accurate results as much as possible
NFR-4	Performance	The application must be fast and quick to load up
NFR-5	Availability	The application must be available to use all the time
NFR-6	Scalability	The application must scale along with the user base

CHAPTER 5

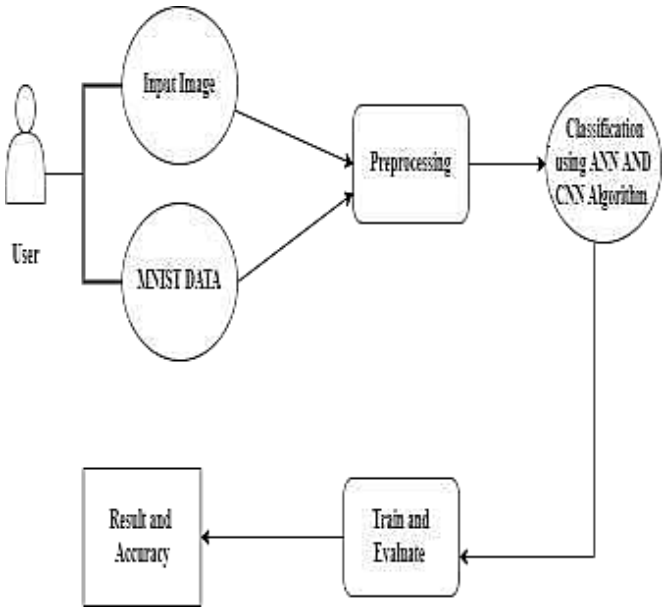
PROJECT DESIGN

a. DATA FLOW DIAGRAM

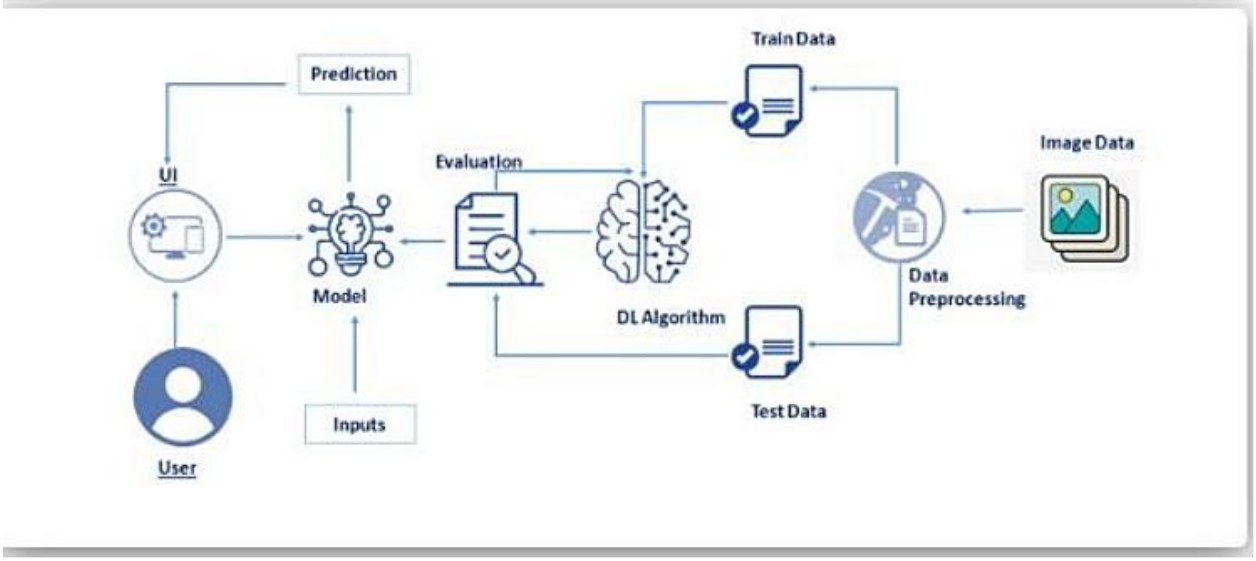
Example: DFD Level 0 (Industry Standard)



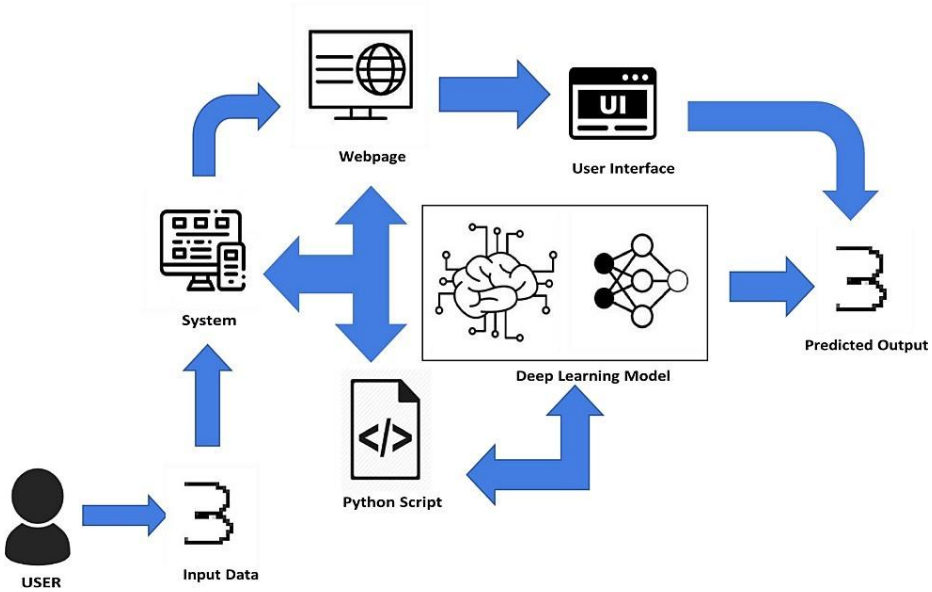
Simplified diagram:



b. SOLUTION & TECHNICAL ARCHITECTURE



Technical Architecture
Team ID : PNT2022TMID04870



c. USER STORIES

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobileuser)	Home	USN-1	As a user,I can view the guide and awareness to use this application.	I can view the awarenessto use this application and its limitations.	Low	Sprint-1
		USN-2	As a user, I'm allowed to view the guidedvideo to use the interface of this application.	I can gain knowledge to use this application by a practical method.	Low	Sprint-1
		USN-3	As a user,I can read the instructions to usethis application.	I can read instructions also to use it in a user friendly method.	Low	Sprint-2
	Recognize	USN-4	As a user,In this prediction page I get tochoose the image.	I can choose the image from our local system and predict the output.	High	Sprint-2
	Predict	USN-6	As a user, I'm Allowed to upload andchoose the image to be uploaded	I can upload and choose the image from the system storage and also in any virtual storage.	Medi um	Sprint-3
		USN-7	As a user, I will train and test the input to get the maximum accuracy of output.	I can able to train and test the application until it gets maximum accuracy of the result.	High	Sprint-4

		USN-8	As a user, I can access the MNIST dataset	I can access the MNISTdata set to produce the accurate result.	Medi um	Sprint-3

CHAPTER 6
PROJECT PLANNING AND SCHEDULING

SPRINT	USER STORY / TASK	STORY POINTS	PRIORITY	TEAM MEMBERS
Sprint - I	Get the dataset	3	High	Manoj S
	Explore the data	2	Medium	Logesh R Maheshvar A
	Data Pre-Processing	3	High	Maheshvar A Jagadeeswaran V V
	Prepare training and testing data	3	High	Logesh R

Sprint - II	Create the model	3	High	Jagadeeswaran V V
	Train the model	3	High	Manoj S
	Test the model	3	High	Maheshvar A
Sprint - III	Improve the model	2	Medium	Logesh R
	Save the model	3	High	Maheshvar A
	Build the Home Page	2	Medium	Logesh R
	Setup a connection to store input images	3	High	Jagadeeswaran V V

Sprint - IV	Build the results page	3	High	Manoj S Maheshvar A

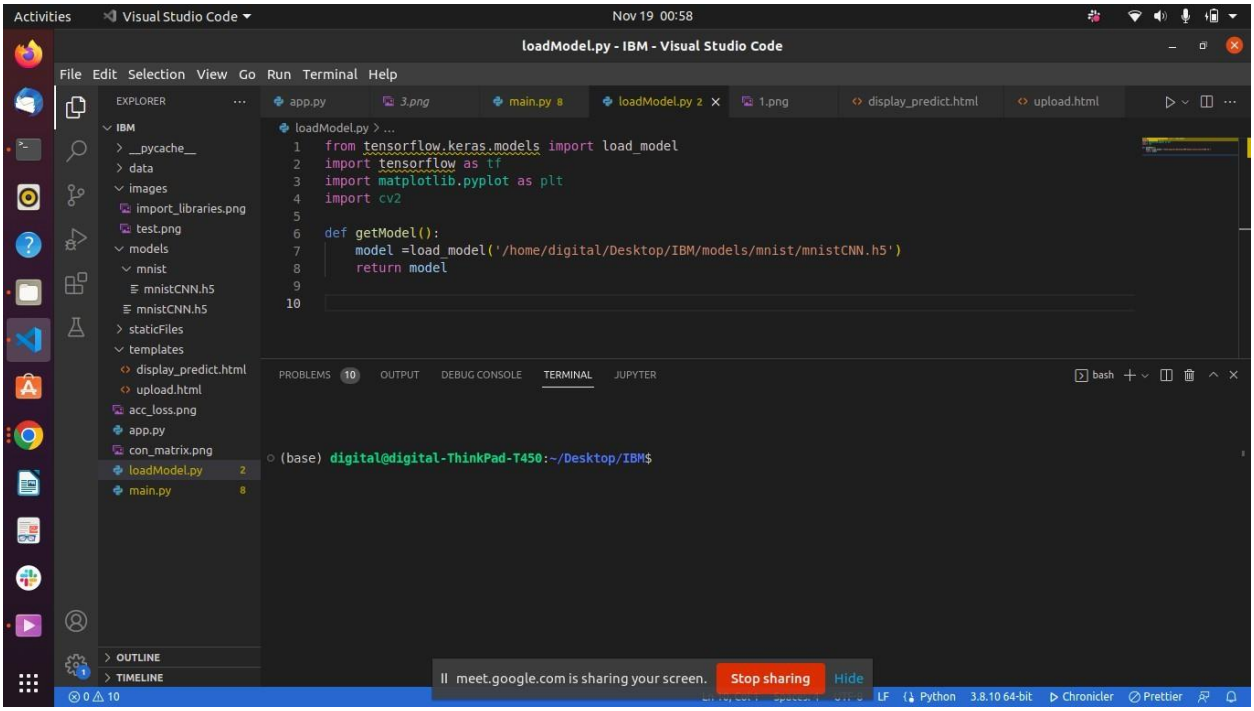
	Integrate the modelwith the application	3	High	Maheshvar A Logesh R
	Test the application and deploy Model	3	High	Jagadeeswaran VV Logesh R Maheshvar A Manoj S

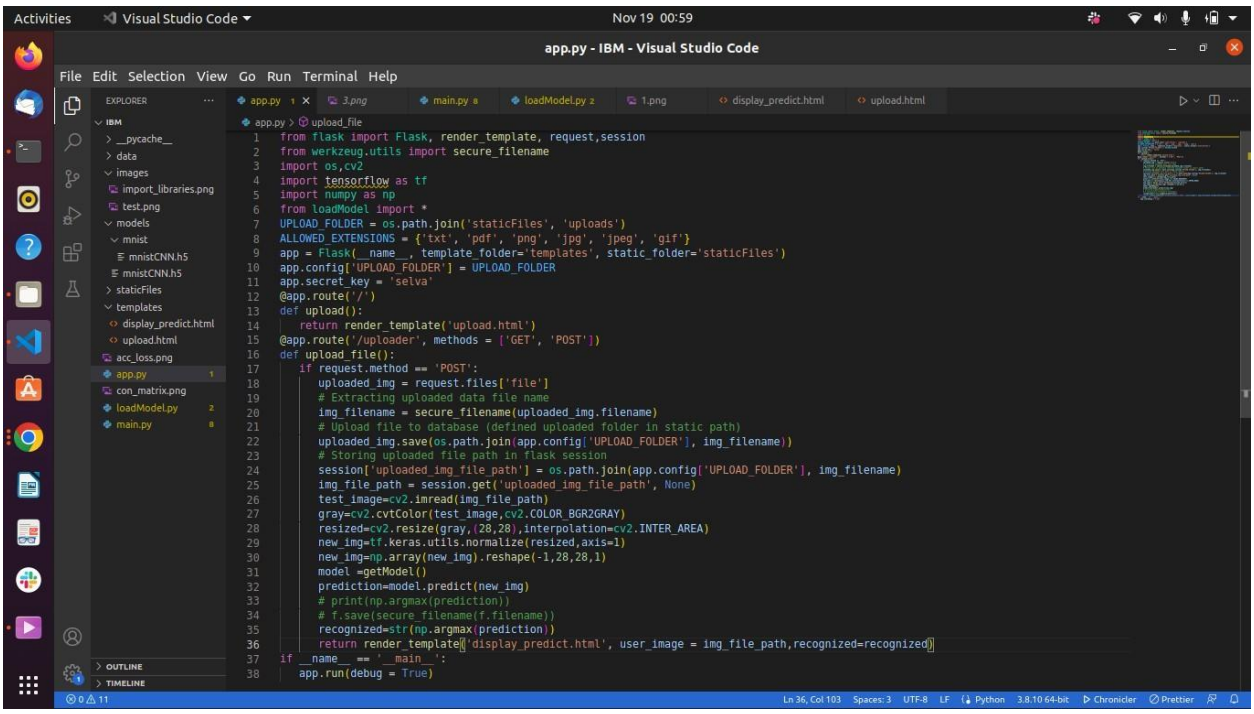
SPRINT DELIVERY SCHEDULE

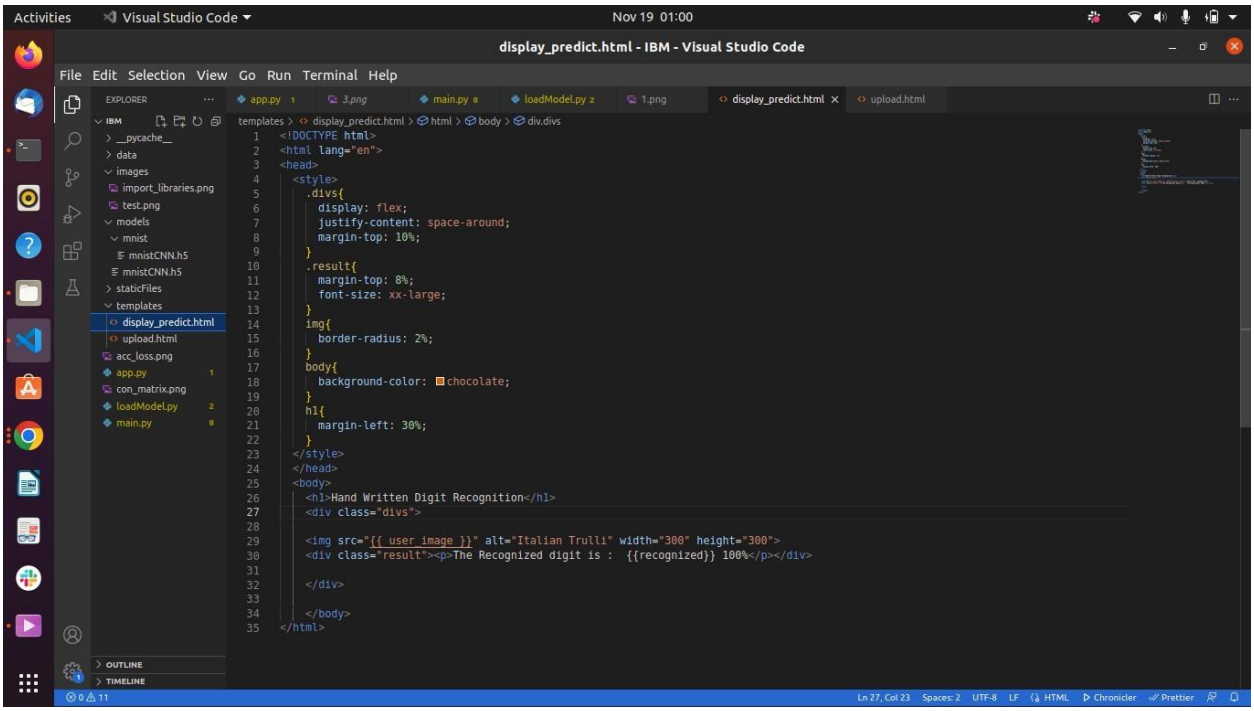
SPRINT	TOTAL STORY POINTS	DURATION	SPRINT START DATE	SPRINT END DATE (PLANNED)	STORY POINTS COMPLETED (AS ON PLANNED DATE)	SPRINT RELEASE DATE (ACTUAL)
Sprint - I	11	6 Days	24 Oct 2022	29 Oct 2022	11	29 Oct 2022
Sprint - II	9	6 Days	31 Oct 2022	05 Nov 2022	9	05 Nov 2022
Sprint - III	10	6 Days	07 Oct 2022	12 Nov 2022	10	12 Nov 2022
Sprint - IV	9	6 Days	14 Nov 2022	19 Nov 2022	9	19 Nov 2022

CHAPTER 7

CODING & SOLUTIONING

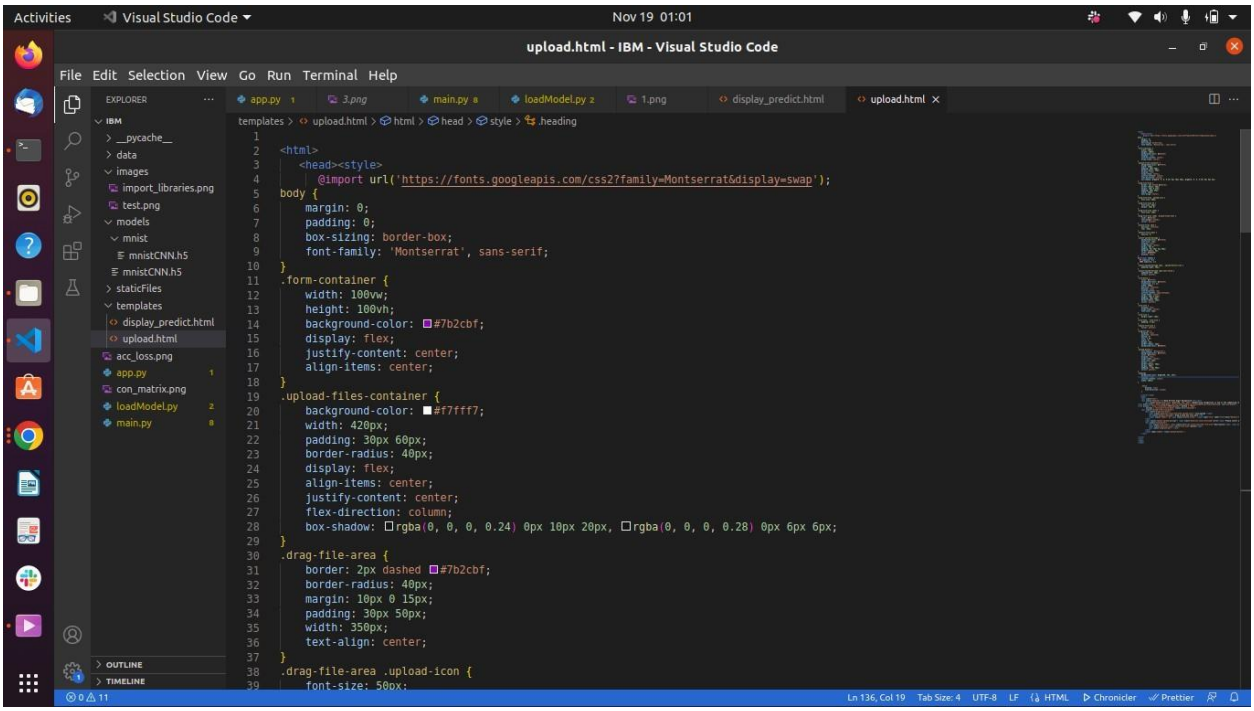


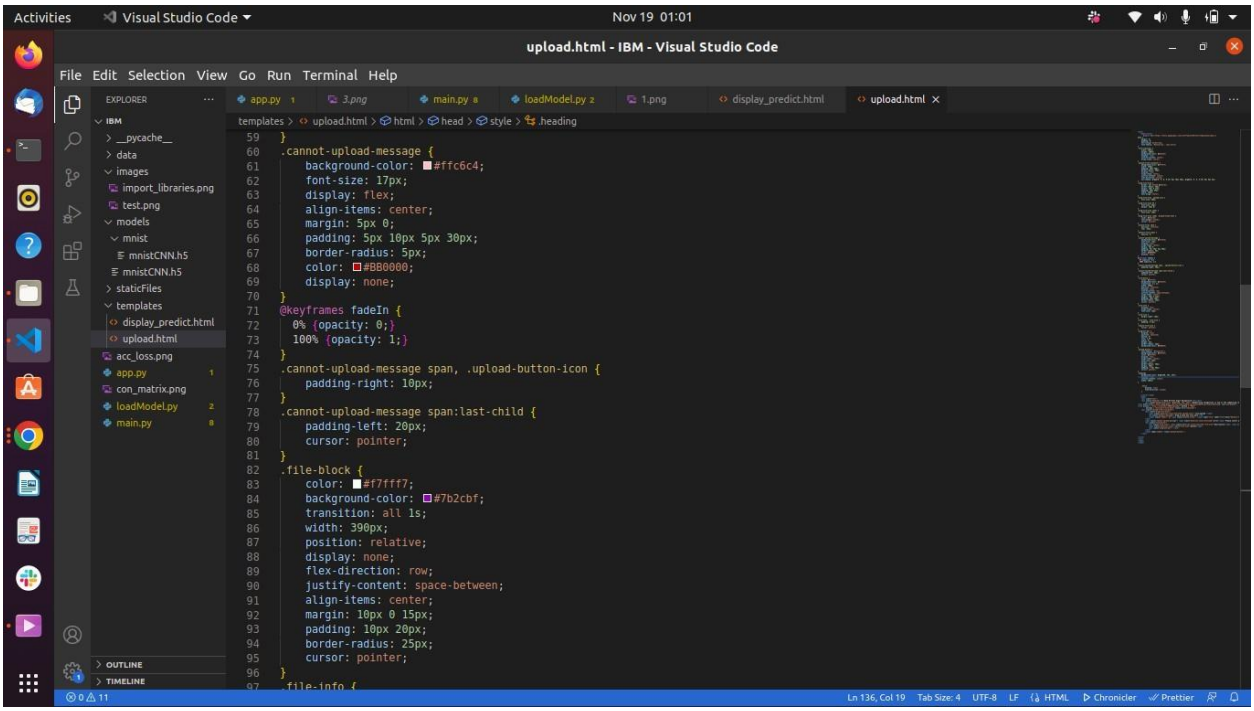


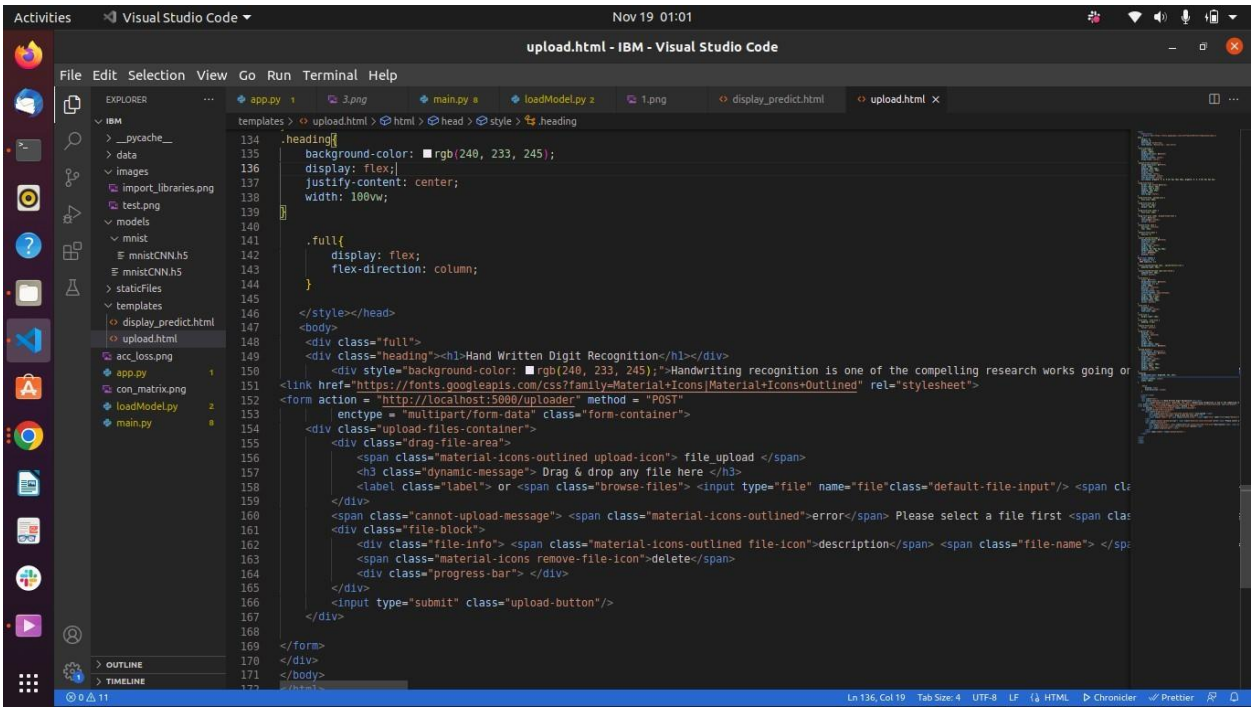

Activities
Visual Studio Code
Nov 19 01:00
display_predict.html - IBM - Visual Studio Code
File Edit Selection View Go Run Terminal Help
EXPLORER
IBM
__pycache__
data
images
import_libraries.png
test.png
models
mnist
mnistCNN.h5
mnistCNN.h5
staticFiles
templates
display_predict.html
upload.html
acc_loss.png
app.py
con_matrix.png
loadModel.py
main.py
OUTLINE
TIMELINE
app.py 1
3.png
main.py 8
loadModel.py 2
1.png
display_predict.html
upload.html
templates > display_predict.html > html > body > div.divs
<!DOCTYPE html>
<html lang="en">
<head>
<style>
.divs{
display: flex;
justify-content: space-around;
margin-top: 10%;
}
.result{
margin-top: 8%;
font-size: xx-large;
}
img{
border-radius: 2%;
}
body{
background-color: chocolate;
}
h1{
margin-left: 30%;
}
</style>
</head>
<body>
<h1>Hand Written Digit Recognition</h1>

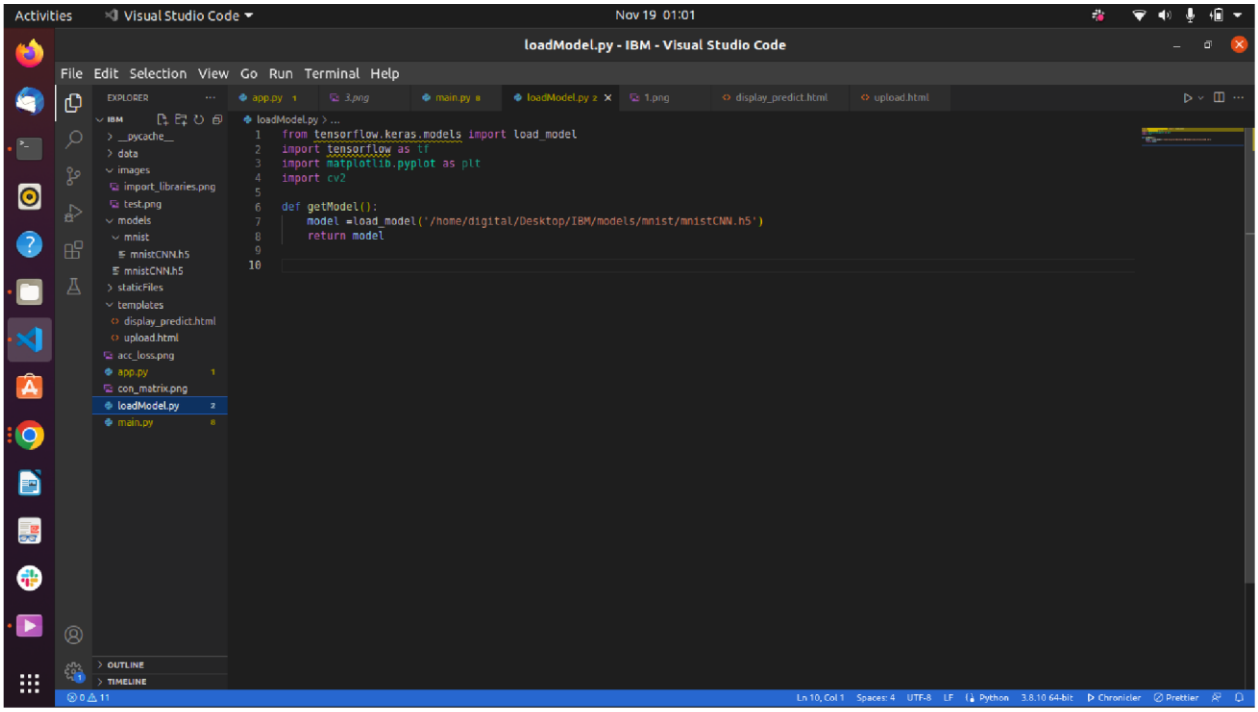
<img src="{{ user_image }}" alt="Italian Trulli" width="300" height="300">
The Recognized digit is : {{recognized}} 100%

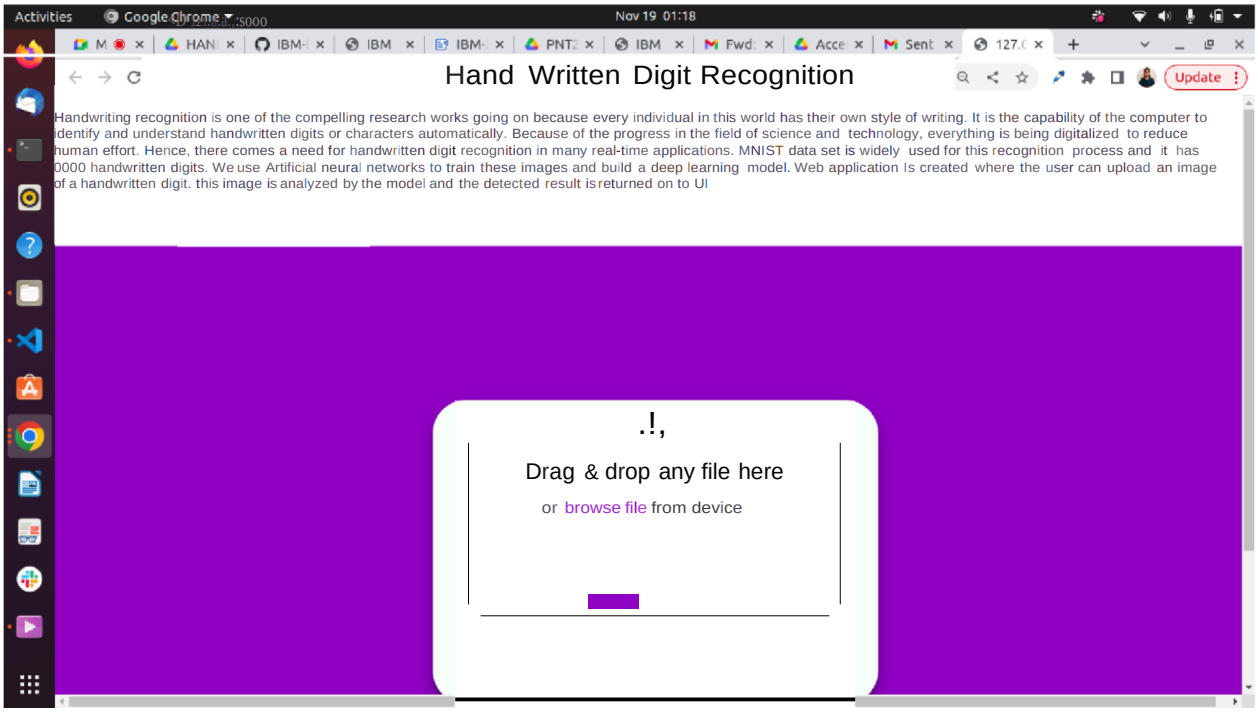
</body>
</html>
Ln 27, Col 23
Spaces: 2
UTF-8
LF
HTML
Chronicler
Prettier

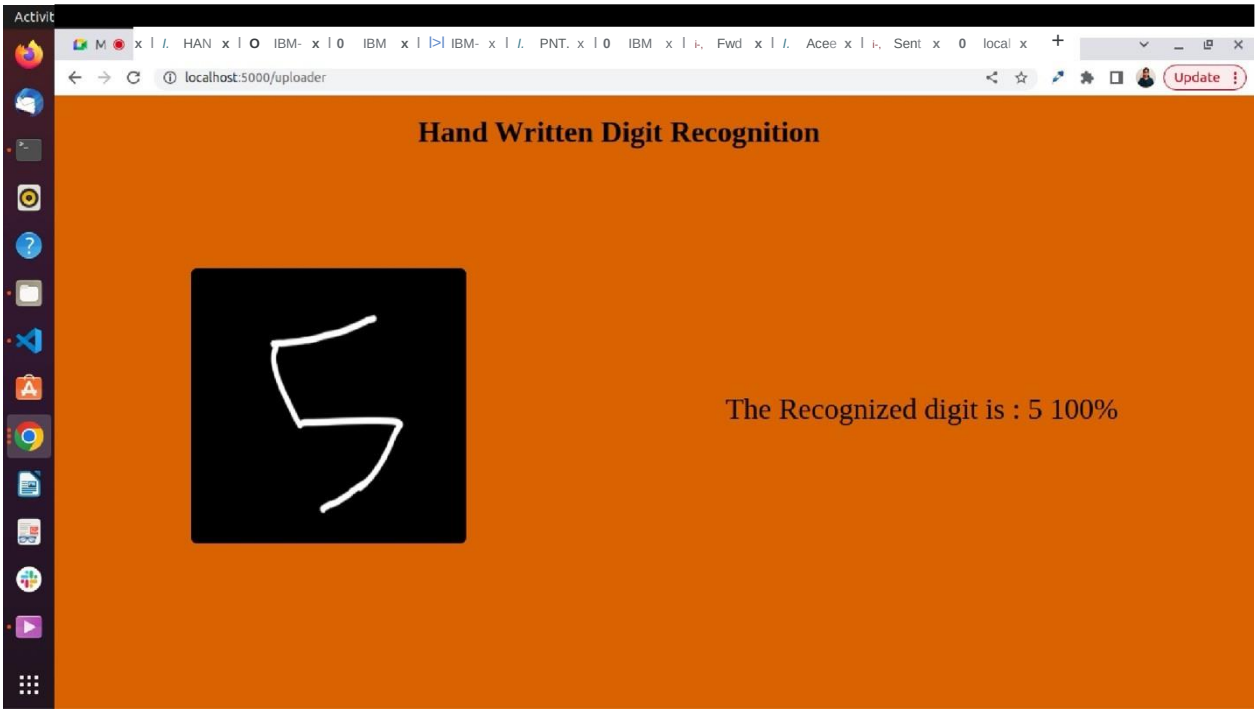

Activities
Visual Studio Code
Nov 19 01:01
upload.html - IBM - Visual Studio Code
File Edit Selection View Go Run Terminal Help
EXPLORER
IBM
__pycache__
data
images
import_libraries.png
test.png
models
mnist
mnistCNN.h5
mnistCNN.h5
staticFiles
templates
display_predict.html
upload.html
acc_loss.png
app.py
con_matrix.png
loadModel.py
main.py
OUTLINE
TIMELINE
app.py 1
3.png
main.py 8
loadModel.py 2
1.png
display_predict.html
upload.html
templates > upload.html > html > head > style > .heading
<html>
<head><style>
@import url('https://fonts.googleapis.com/css2?family=Montserrat&display=swap');
body {
margin: 0;
padding: 0;
box-sizing: border-box;
font-family: 'Montserrat', sans-serif;
}
.form-container {
width: 100vw;
height: 100vh;
background-color: #7b2cbf;
display: flex;
justify-content: center;
align-items: center;
}
.upload-files-container {
background-color: #f7fff7;
width: 420px;
padding: 30px 60px;
border-radius: 40px;
display: flex;
align-items: center;
justify-content: center;
flex-direction: column;
box-shadow: rgba(0, 0, 0, 0.24) 0px 10px 20px, rgba(0, 0, 0, 0.28) 0px 6px 6px;
}
.drag-file-area {
border: 2px dashed #7b2cbf;
border-radius: 40px;
margin: 10px 0 15px;
padding: 30px 50px;
width: 350px;
text-align: center;
}
.drag-file-area .upload-icon {
font-size: 50px;
Ln 136, Col 19
Tab Size: 4
UTF-8
LF
HTML
Chronicler
Prettier


Activities
Visual Studio Code
Nov 19 01:01
upload.html - IBM - Visual Studio Code
File Edit Selection View Go Run Terminal Help
EXPLORER
IBM
__pycache__
data
images
import_libraries.png
test.png
models
mnist
mnistCNN.h5
mnistCNN.h5
staticFiles
templates
display_predict.html
upload.html
acc_loss.png
app.py
con_matrix.png
loadModel.py
main.py
OUTLINE
TIMELINE
app.py 1
3.png
main.py 8
loadModel.py 2
1.png
display_predict.html
upload.html
templates > upload.html > html > head > style > .heading
59 }
60 .cannot-upload-message {
61 background-color: #ffc6c4;
62 font-size: 17px;
63 display: flex;
64 align-items: center;
65 margin: 5px 0;
66 padding: 5px 10px 5px 30px;
67 border-radius: 5px;
68 color: #BB0000;
69 display: none;
70 }
71 @keyframes fadeIn {
72 0% {opacity: 0;}
73 100% {opacity: 1;}
74 }
75 .cannot-upload-message span, .upload-button-icon {
76 padding-right: 10px;
77 }
78 .cannot-upload-message span:last-child {
79 padding-left: 20px;
80 cursor: pointer;
81 }
82 .file-block {
83 color: #f7fff7;
84 background-color: #7b2cbf;
85 transition: all 1s;
86 width: 390px;
87 position: relative;
88 display: none;
89 flex-direction: row;
90 justify-content: space-between;
91 align-items: center;
92 margin: 10px 0 15px;
93 padding: 10px 20px;
94 border-radius: 25px;
95 cursor: pointer;
96 }
97 .file-info {
Ln 136, Col 19 Tab Size: 4 UTF-8 LF HTML Chronicler Prettier


Activities
Visual Studio Code
Nov 19 01:01
upload.html - IBM - Visual Studio Code
File Edit Selection View Go Run Terminal Help
EXPLORER
IBM
__pycache__
data
images
import_libraries.png
test.png
models
mnist
mnistCNN.h5
mnistCNN.h5
staticFiles
templates
display_predict.html
upload.html
acc_loss.png
app.py
con_matrix.png
loadModel.py
main.py
OUTLINE
TIMELINE
app.py 1
3.png
main.py 8
loadModel.py 2
1.png
display_predict.html
upload.html
templates > upload.html > html > head > style > .heading
134 .heading{
135 background-color: rgb(240, 233, 245);
136 display: flex;
137 justify-content: center;
138 width: 100vw;
139 }
140
141 .full{
142 display: flex;
143 flex-direction: column;
144 }
145
146 </style></head>
147 <body>
148
149 <h1>Hand Written Digit Recognition</h1>
150 Handwriting recognition is one of the compelling research works going or
151 <link href="https://fonts.googleapis.com/css?family=Material+Icons|Material+Icons+Outlined" rel="stylesheet">
152 <form action = "http://localhost:5000/uploader" method = "POST"
153 enctype = "multipart/form-data" class="form-container">
154
155
156 file_upload
157 <h3 class="dynamic-message"> Drag & drop any file here </h3>
158 <label class="label"> or <input type="file" name="file"class="default-file-input"/> <span cla
159
160 error Please select a file first <span clas
161
162 description </spa
163 delete
164
165
166 <input type="submit" class="upload-button"/>
167
168
169 </form>
170
171 </body>
Ln 136, Col 19 Tab Size: 4 UTF-8 LF HTML Chronicler Prettier


Activities
Visual Studio Code
Nov 19 01:01
loadModel.py - IBM - Visual Studio Code
File Edit Selection View Go Run Terminal Help
EXPLORER
app.py
3.png
main.py
loadModel.py
1.png
display_predict.html
upload.html
loadModel.py > ...
1 from tensorflow.keras.models import load_model
2 import tensorflow as tf
3 import matplotlib.pyplot as plt
4 import cv2
5
6 def getModel():
7 model =load_model('/home/digital/Desktop/IBM/models/mnist/mnistCNN.h5')
8 return model
9
10
__pycache__
data
images
import_libraries.png
test.png
models
mnist
mnistCNN.h5
mnistCNN.h5
staticFiles
templates
display_predict.html
upload.html
acc_loss.png
app.py
con_matrix.png
loadModel.py
main.py
OUTLINE
TIMELINE


Activities
Google Chrome
Nov 19 01:18
Hand Written Digit Recognition
Update
Handwriting recognition is one of the compelling research works going on because every individual in this world has their own style of writing. It is the capability of the computer to identify and understand handwritten digits or characters automatically. Because of the progress in the field of science and technology, everything is being digitalized to reduce human effort. Hence, there comes a need for handwritten digit recognition in many real-time applications. MNIST data set is widely used for this recognition process and it has 0000 handwritten digits. We use Artificial neural networks to train these images and build a deep learning model. Web application is created where the user can upload an image of a handwritten digit. this image is analyzed by the model and the detected result is returned on to UI
Drag & drop any file here
or browse file from device


Activit
localhost:5000/uploader
Update
Hand Written Digit Recognition
The Recognized digit is : 5 100%

CHAPTER 8

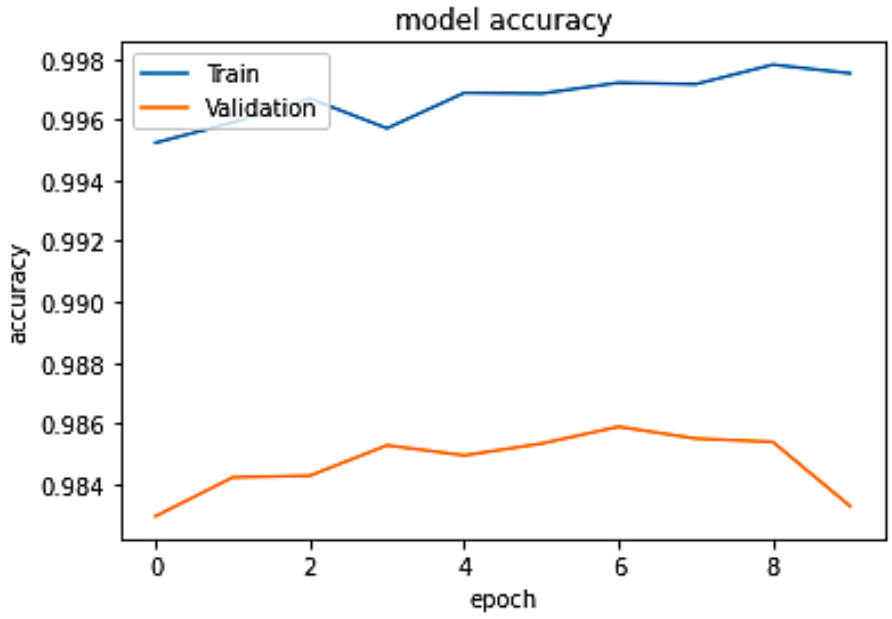
RESULTS

PERFORMANCE METRICS

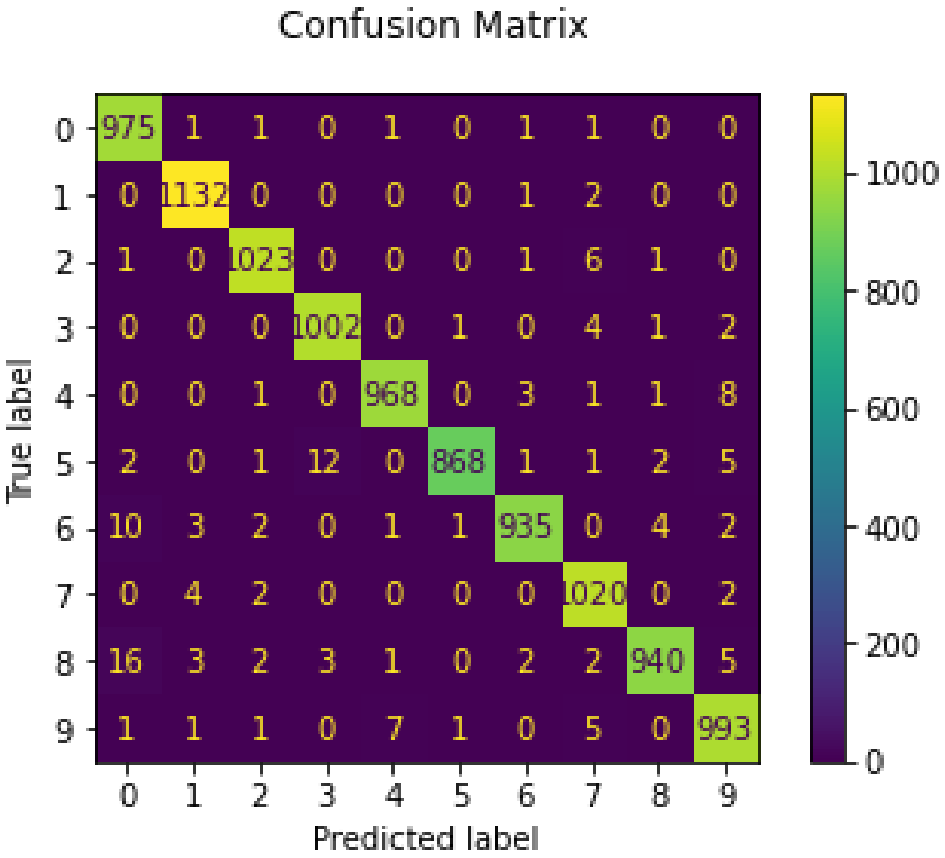
a. MODEL SUMMARY

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_2 (Conv2D)            (None, 20, 20, 64)        5248

conv2d_3 (Conv2D)            (None, 12, 12, 32)        165920

flatten_1 (Flatten)          (None, 4608)              0

dense (Dense)                (None, 10)                46090

=================================================================
Total params: 217,258
Trainable params: 217,258
Non-trainable params: 0
```

b. ACCURACY



c. CONFUSION MATRIX



d. CLASSIFICATION REPORT

```
              precision    recall  f1-score   support

           0       0.99      0.99      0.99       980
           1       0.99      0.99      0.99      1135
           2       0.97      0.99      0.98      1032
           3       0.99      0.99      0.99      1010
           4       0.99      0.98      0.99       982
           5       0.99      0.99      0.99       892
           6       0.98      0.99      0.99       958
           7       0.99      0.98      0.98      1028
           8       0.98      0.99      0.99       974
           9       0.99      0.98      0.98      1009

    accuracy                           0.99     10000
   macro avg       0.99      0.99      0.99     10000
weighted avg       0.99      0.99      0.99     10000

[[ 972    1    1    0    0    1    3    0    2    0]
 [   0 1121    4    0    0    0    5    4    1    0]
 [   1    1 1026    0    0    0    0    1    3    0]
 [   0    0    4  997    0    4    0    3    1    1]
 [   0    0    2    0  967    0    4    0    4    5]
 [   1    0    0    5    0  882    2    1    0    1]
 [   3    3    1    0    1    2  947    0    1    0]
 [   2    0   17    1    1    0    0 1003    1    3]
 [   1    0    3    3    1    2    1    0  961    2]
 [   1    1    1    2    8    4    0    5    3  984]]
```

CHAPTER 9
ADVANTAGES & DISADVANTAGES

ADVANTAGES

1. Reduces manual work
2. More accurate than average human
3. Capable of handling a lot of data
4. Can be used anywhere from any device

DISADVANTAGES

5. Cannot handle complex data
6. All the data must be in digital format
7. Requires a high performance server for faster predictions
8. Prone to occasional errors

CHAPTER 10
CONCLUSION

In this project, a web application that recognises handwritten digits using machine learning was demonstrated. This project was made using a variety of technologies, including JavaScript, HTML, CSS, and Flask. The model uses a CNN network to predict the handwritten digit. The model scored a 99.61% recognition rate during testing. The suggested idea is easily scalable and capable of supporting a large number of users. It is compatible with any device that can run a browser because it is a web application. This project is very helpful in real-world scenarios like reading licence plates of moving vehicles, processing the amounts on bank checks, entering numbers manually filled out forms (like tax forms), and so on. There is so much room for development that can be added in later iterations.

CHAPTER 11
FUTURE SCOPE

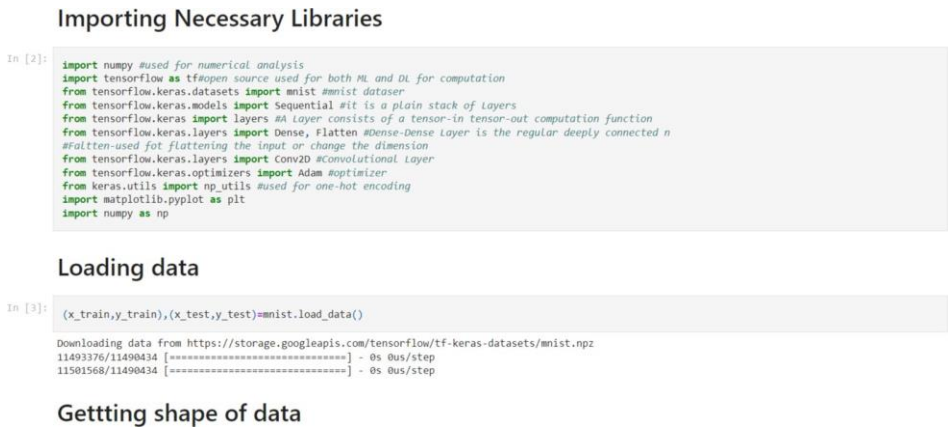
This project is far from complete and there is a lot of room for improvement.
Some of the improvements that can be made to this project are as follows:

- Add support to detect from digits multiple images and save the results
- Add support to detect multiple digits
- Improve model to detect digits from complex images
- Add support to different languages to help users from all over the world

This project has endless potential and can always be enhanced to become better.Implementing this concept in the real world will benefit several industries and reduce the workload on many workers, enhancing overall work efficiency.

CHAPTER 12
APPENDIX

MODEL CREATION



Importing Necessary Libraries

In [2]:
```
import numpy #used for numerical analysis
import tensorflow as tf#open source used for both ML and DL for computation
from tensorflow.keras.datasets import mnist #mnist dataset
from tensorflow.keras.models import Sequential #it is a plain stack of Layers
from tensorflow.keras import layers #A Layer consists of a tensor-in tensor-out computation function
from tensorflow.keras.layers import Dense, Flatten #Dense-Dense Layer is the regular deeply connected n
#Falltten-used fot flattening the input or change the dimension
from tensorflow.keras.layers import Conv2D #Convolutional Layer
from tensorflow.keras.optimizers import Adam #optimizer
from keras.utils import np_utils #used for one-hot encoding
import matplotlib.pyplot as plt
import numpy as np
```

Loading data

In [3]:
```
(x_train,y_train),(x_test,y_test)=mnist.load_data()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11493376/11490434 [==============================] - 0s 0us/step
11501568/11490434 [==============================] - 0s 0us/step
```

Gettting shape of data

Gettting shape of data

```
print("Shape of training and testing dataset ")
print(x_train.shape)
print(x_test.shape)
```

```
Shape of training and testing dataset
(60000, 28, 28)
(10000, 28, 28)
```

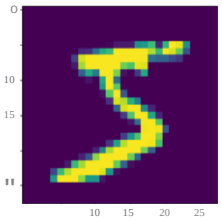
Understanding of data

```
printCAnalyzing the data")
print("Printing the first data and its label")
print(x_train[0])
print(y_train[0])
printCvisualising the data ")
plt .ir11show(x_train[0])
plt.show()
```

```
Analyzing the data
Printing the first data and its label
([ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0]
 0 0 0 0 0 0 0 0 0 0
 0 ]
 0 0 0 0 0 0 0 0 0 0
 0 ]
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 ]
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 18 18 18126136
 175 26 166 255 247 127 0 0 0 0]
 f 0 0 0 0 0 0 0 0 l0 l6 94 1S4 170 J53 7:53 153 J53 ]53
 J.J.'> 1/J. .!!.;1 J.4J. l!:1'> b4 u u itj
 [0 0 0 0 0 0 0 49 238 253 253 253 253 253 253 253 253 251
 93 82 82 56 39 0 0 0 0 }
 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0)
 0 0 0 0 0 0 0 0 80 156 107 253 253 205 11 0 43 154
 0 0 0 0 0 0 0 0 0 0)
 [ 0 0 0 0 0 0 0 0 0 14 1 154 253 90 0 0 0 0
 0)
 0 0 139 253 190 2 0 0 0
 0 0 0 0 0 0 0 0 0 0)
 0 0 0 0 0 0 0 0 0 0 11 190 253 70 0 0 0
 0 0 0 0 0 0 0 0)
 0 0 0 0 0 0 0 0 0 0 0 0 35 241 225 160 108 1
 0 0 0 0 0 0 0 0 0 0]
 0 0 0 0 0 81 240 253 253 119
 I e 0 0 0 0 0 0 0 45 186 253 253
 I 150 e 27 0 0 0 0 0 0 0 0 16 93 252
 253187 0 0 0 0
 I 253 e 249 0 64 0 0 0 0 0 0 0 0 0 249
 I e 0 0 0 0 46 130 183 253
 I e e 0 0 0 0 39 148 229 253 253 253
 253 207 2 0 0
 I 250 e 182 e 0 0 0 0 0 24 114 221 253 253 253 253 201
 78 0 0 0 0 0
 23 66 213 253 253 253 253 198 81 2
 0 0]
 I 0 0 0 0 0 18 171 21 9 253 253 253 253 195 80 9 0
 0 0 0 0 0
 0 0 0 0 55 172 226 253 253 253 253 244 133 11 0 0 0 0
 0 0 0 0 0 0]
 0 0 0 0 136 253 253 253 212 135 132 16 0 0 0 0 0 0
 0]
 0 0 0 0 0 0 0 0] 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0
```

```
[[  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   3  18  18  18 126 136
  175  26 166 255 247 127   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0  30  36  94 154 170 253 253 253 253 253
  225 172 253 242 195  64   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0  49 238 253 253 253 253 253 253 253 253 251
   93  82  82  56  39   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0  18 219 253 253 253 253 253 198 182 247 241
    0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0  80 156 107 253 253 205  11   0  43 154
    0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0  14   1 154 253  90   0   0   0   0
    0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0 139 253 190   2   0   0   0
    0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0  11 190 253  70   0   0   0
    0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0  35 241 225 160 108   1
    0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0  81 240 253 253 119
   25   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0  45 186 253 253
  150  27   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0  16  93 252
  253 187   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 249
  253 249  64   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0  46 130 183 253
  253 207   2   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0  39 148 229 253 253 253
  250 182   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0  24 114 221 253 253 253 253 201
   78   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0  23  66 213 253 253 253 253 198  81   2
    0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0  18 171 219 253 253 253 253 195  80   9   0   0
    0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0  55 172 226 253 253 253 253 244 133  11   0   0   0   0
    0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0 136 253 253 253 212 135 132  16   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0   0   0]]
```

Visualising the data



Reshaping the data

In [6]:
```
x_train=x_train. reshape(60000, 28, 28, 1). astype( 'float32')
x_test=x_test. reshape(10000,28, 28,1) .astype( 'float32·)
print(x_train.shape)
print(x_test.shape)
```

```
(60000, 28, 28, 1)
(10000, 28, 28, 1)
```

Applying One Hot Encoding && Normalization"

In P]:
```
no_of_classes=10
y_train=np_utils.to_categorical (y_train, no_of_classes)
y_test=np_utils. to_categorical (y_test, no_of_ classes)
x_train=tf. kera s. utils. normalize (x_train, axis=l)
x_test=tf. keras. utils. normalize (x_test, axis=l)
print(y_train[0])
```

```
[0. 0. 0. 0. 0. 1. 0. 0. 0. 0.]
```

Creating the model

In [8]:
```
model =Sequential()
model. add (Conv2D(64, (3•3), input_shape =( 28, 28, 1) ,activation=' relu' ))
model. add (Conv2D(32, (3•3) ,activation=' relu' ))
model.add(Flatten())
model. add (Dense (no_of _classes,activation=' softmax'))
```

Compiling the model

In r]:
```
model. compile(loss="categorical_crossentropy",optimizer=' adam' ,metrics=['accuracy'])
model.fit (x_train, y_train, epochs=S, validation_split=0. 3, batch_size=S)
```

Evaluation

In [12]:
```
test_loss, test_acc=model .evaluate(x_test ,y_test)
```

Evaluation

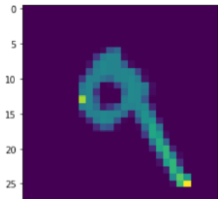
```
n [l.2  test_loss, test_acc=model .evaluate( x_test ,y_test)
        print("Test accuracy: ",test_acc)
        print("Test validation loss: ",test_loss)
```

```
313/313 [------------------------------] - 6s18ms/step - loss: 0.0827 - accuracy: 0.9847
Test accuracy: 0.9847000241279602
Test validation loss: 0.08267717063426971
```

Testing

```
n [14  predictions=model. predict([x_test])
       print(predictions(7])
       print (np.argmax( predictions [7]))
       plt.imshow(x_test[7])
       plt.show()
```

```
[1.3543255e-21 5.4878523e-16 7.2801753e-19 5.0472089e-15 2.6604361e-09
 9.2844643e-14 9.2869199e-23 7.0519296e-21 1.1336766e-10 1.0000000e+00]
9
```



Saving the model

```
1. I  model.save( 'mnistetm.hS')
      !tar -zcvf hand-written-digit-recognition. tgzmnistetm. h5
```

```
mnistCNN.hS
```

```
In *  ls -1
```

```
hand-written-digit-recognition.tgz
mnistCNN.hS
```

Model Deployment

```
In "  !pip install watson-machine-learning-client --upgrade
```

```
Collecting watson-machi ne- learning-client
  Dow loadinf watson machine learning1; \!· ;1 !-: e; ; l (538 kB)
Requirement already satisfied; certi fi in /opt/conda/envs/Python-3.9/lib/python3.9/si te-packages (from watson-machine-learning-client) {2022.9.24)
Requirement already satisfied: boto3 in /opt/conda/envs/Python-3.9/lib/python3. 9/site -packages (from watson-machine-learning-client) (1.18. 21)
Requirement already satisfied: pandas in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from watson-machine-learning-client) (1.3.4)
Requirement already satisfied: lomond in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from watson-machine-learning-client) (0.3. 3)
Requirement already satisfied: ibm-cos-sdk in /opt/conda/envs/Python- 3.9/lib/python3.9/site-packages (from watson-machine-learning-client) (2.11.0)
Requirement already satisfied: requests in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from watson-machine-learning-client) (2. 26 .0)
Requirement already satisfied: urllib3 in /opt/conda/envs/Python-3.9/lib/python3.9/si te-packages (from watson-machine-learning-client) (1. 26.7)
Requirement already satisfied: tqdm in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from watson-machine- learning-client) (4.62. 3)
Requirement already satisfied: tabulate in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from watson-machine-learning-client) (0.8.9)
Requirement already satisfied: s3transfer<0.6.0, >:0.5.0 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from boto3->watson-machine-learning
-client) (0.5.0)
Requirement already satisfied: botocore<l. 22 .0, >:1.21. 21 in /opt/conda/envs/Python-3.9/lib/python3.9/si te-packages (from boto3- >watson-machine-learnin g-client) {1.21.41)
Requirement already satisfied: jmespath<l. 0.0, >"-0.7.1 in /opt/conda/envs/Python -3.9/lib/python3.9/site-packages (from boto3->watson-machine-learning-c lient) (0.10.0)
```

```
tensorflow_2.4-py3.8             fe185c44-9a99-5425-986b-59bd1d2eda46  base
----------------------------------  ------------------------------------  ----

In [27]:
software_space_uid=client.software_specifications.get_uid_by_name("tensorflow_rt22.2-py3.10")
print(software_space_uid)

f65bd165-f057-55de-b5cb-f97cf2c0f393

In [28]:
ls

hand-written-digit-recognition.tgz  mnistCNN.h5

In [29]:
model_details=client.repository.store_model(model='hand-written-digit-recognition.tgz',meta_props={
    client.repository.ModelMetaNames.NAME:"Digit Model",
    client.repository.ModelMetaNames.TYPE:'tensorflow_2.9',
    client.repository.ModelMetaNames.SOFTWARE_SPEC_UID:software_space_uid
})

In [30]:
model_id=client.repository.get_model_id(model_details)

In [31]:
model_id

Out[31]: '34ed2f6a-1d50-493e-82bb-00c19a768ced'

In [32]:
client.repository.download(model_id,'mnist.tar.gb')

Successfully saved model content to file: 'mnist.tar.gb'
Out[32]: '/home/wsuser/work/mnist.tar.gb'

In [ ]:
```

GITHUB

https://github.com/IBM-EPBL/IBM-Project-12041-1659367827.git

PROJECT DEMO

https://drive.google.com/file/d/1qpzW9C4UkjE_k4kNzhSmJ67qOj2jW2kh/view?usp=sharing