NALAIYA THIRAN - IBM PROJECT REPORT

(19IT410T Professional Readiness for Innovation, Employability and Entrepreneurship)

ON

A NOVEL METHOD FOR HANDWRITTEN DIGIT RECOGNITION SYSTEM

Submitted by

TEAM ID: PNT2022TMID23384

RAKHUL K R (113219031119)

KANKANALA DHANUSH (113219031069)

VINOTH R (113219031163)

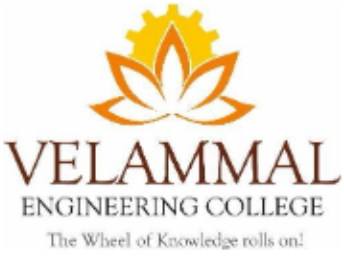
DHANUSHRAM S (113219031036)

in partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING



VELAMMAL ENGINEERING COLLEGE, CHENNAI-66.

(An Autonomous Institution, Affiliated to Anna University, Chennai)

2022-2023

CONTENTS

CHAPTER 1
INTRODUCTION

1. PROJECT OVERVIEW

Machine learning and deep learning play an important role in computer technology and artificial intelligence. With the use of deep learning and machine learning, human effort can be reduced in recognizing, learning, predictions and in many more areas.

Handwritten Digit Recognition is the ability of computer systems to recognise handwritten digits from various sources, such as images, documents, and so on. This project aims to let users take advantage of machine learning to reduce manual tasks in recognizing digits.

2. PURPOSE

Handwritten digits are not perfect and can be made with many different flavours. 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.

Digit recognition systems are capable of recognizing the digits from different sources like emails, bank cheque, papers, images, etc. and in different real-world scenarios for online handwriting recognition on computer tablets or system, recognize number plates of vehicles, processing bank cheque amounts, numeric entries in forms filled up by hand (tax forms) and so on.

CHAPTER 2
LITERATURE SURVEY

1. EXISTING PROBLEM

The issue is that there's a wide range of handwriting – good and bad. This makes it tricky for programmers to provide enough examples of how every character might look. Sometimes, characters look very similar, making it hard for a computer to recognise accurately.

The fundamental problem with handwritten digit recognition is that handwritten digits do not always have the same size, width, orientation, and margins since they vary from person to person. Additionally, there would be issues with identifying the numbers because of similarities between numerals like 1 and 7, 5 and 6, 3 and 8, 2 and 5, 2 and 7, etc. Finally, the individuality and variation of each individual's handwriting influence the structure and appearance of the digits.

2. REFERENCES

An Enhanced Handwritten Digit Recognition Using Convolutional Neural Network(2021)

M. S, C. N. Vanitha, N. Narayan, R. Kumar and G. R

Handwritten digit recognition has a great impact in the applications of deep learning. Convolutional Neural Network in deep learning has become one of the major methods and one of the important factors in the various success in recent times and deep learning is used majorly in the area of object recognition. In the paper work, the speech output feature is integrated along with the text output. Convolutional Neural Network model is applied in the image classification. The dataset used to train and test is the MNIST dataset. There are various applications of handwritten digit recognition in real time. It is applied in detection of vehicle number, reading of bank cheques, the arrangement of letters in the post office.

An Efficient And Improved Scheme For Handwritten Digit Recognition Based On Convolutional Neural Network (2019)

Ali, Saqib and Shaukat, Zeeshan and Azeem, Muhammad and Sakhawat, Zareen and Mahmood, Tariq and others

This study uses rectified linear units (ReLU) activation and a convolutional neural network (CNN) that incorporates the Deeplearning4j (DL4J) architecture to recognize handwritten digits. The proposed CNN framework has all the necessary parameters for a high level of MNIST digit classification accuracy. The system's training takes into account the time factor as well. The system is also tested by altering the number of CNN layers for additional accuracy verification. It is important to note that the CNN architecture consists of two convolutional layers, the first with 32 filters and a 5x5 window size and the second with 64 filters and a 7x7 window size. In comparison to earlier proposed systems, the experimental findings show that the proposed CNN architecture for the MNIST dataset demonstrates great performance in terms of time and accuracy. As a result, handwritten numbers are detected with a recognition rate of 99.89% and

high precision (99.21%) in a short amount of time.

Handwritten Digit Recognition using Convolutional Neural Network in Python with Tensorflow and Observe the Variation of Accuracies for Various Hidden Layers (2019)
Fathma Siddique, Shadman Sakib, Md. Abu Bakr Siddique

In recent times, with the increase of Artificial Neural Network (ANN), deep learning has brought a dramatic twist in the field of machine learning by making it more Artificial Intelligence (AI). Deep learning is remarkably used in vast ranges of fields because of its diverse range of applications such as surveillance, health, medicine, sports, robotics, drones etc. In deep learning, Convolutional Neural Network (CNN) is at the center of spectacular advances that mixes Artificial Neural Network (ANN) and up to date deep learning strategies. It has been used broadly in pattern recognition, sentence classification, speech recognition, face recognition, text categorization, document analysis, scene, and handwritten digit recognition. The goal of this paper is to observe the variation of accuracies of CNN to classify handwritten digits using various numbers of hidden layers and epochs and to make the comparison between the accuracies. For this performance evaluation of CNN, we performed our experiment using the Modified National Institute of Standards and Technology (MNIST) dataset. Further, the network is trained using stochastic gradient descent and the backpropagation algorithm

.

Handwritten Digit Recognition Using Machine And Deep Learning Algorithms (2021)
Pashine, Samay and Dixit, Ritik and Kushwah, Rishika

In this study, they developed three deep and machine learning-based models for handwritten digit recognition using MNIST datasets. To determine which model was the most accurate, they compared them based on their individual properties.
Support vector machines are among the simplest classifiers, making them faster than other algorithms and providing the highest training accuracy rate in this situation. However, due to their simplicity, SVMs cannot categorize complicated and ambiguous images as accurately as MLP and CNN algorithms can. In their research, they discovered that CNN produced the most

precise outcomes for handwritten digit recognition. This led them to the conclusion that CNN is the most effective solution for all types of prediction issues, including those using picture data. Next, by comparing the execution times of the algorithms, they determined that increasing the number of epochs without changing the configuration of the algorithm is pointless due to the limitation of a certain model, and they discovered that beyond a certain number of epochs, the model begins overfitting the dataset and provides biased predictions.

3. PROBLEM STATEMENT DEFINITION

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 digitized to reduce human effort. Hence, there comes a need for handwritten digit recognition in many realtime applications. The MNIST data set is widely used for this recognition process and it has 70000 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 to the UI. MNIST ("Modified National Institute of Standards and Technology") is considered an unofficial computer vision "hello-world" dataset. This is a collection of thousands of handwritten pictures used to train classification models using Machine Learning techniques.

CHAPTER 3

IDEATION AND PROPOSED SOLUTION

1. EMPATHY MAP

2. IDEATION AND BRAINSTORMING

Ideas laid out by each Team Member

1. RAKHUL K R

• Idea 1: Contour based character segmentation
• Idea 2: Cartesian Coordinates based digit grouping
• Idea 3: Text auto-formatting using selected standard
• Idea 4: Operator detection and evaluation

2. KANKANALA DHANUSH

• Idea 1: Digit recognition on antiques
• Idea 2: Vehicle number plate recognition
• Idea 3: Help the blind to recognize currency value.
• Idea 4: Auto recognition of digits of different languages.

3. VINOTH R

• Idea 1: Currency detection using currency symbol
• Idea 2: Depositing money using deposit form in ATM
• Idea 3: Digitizing the digit records which are handwritten.
• Idea 4: Improving hand writing on screen during online classes.

4. DHANUSHRAM

• Idea 1: Phone number detection using the country code
• Idea 2: Detection of matrix and the values in it.
• Idea 3: Cheque processing in ATM and bank
• Idea 4: Identifying the prices on the price tags on products which the prices are punched.

Shortlisted Ideas

- Idea 1: Contour based character segmentation
- Idea 2: Improving hand writing on screen during online classes.
- Idea 3: Text auto-formatting using selected standard.

Rakhul

- Contour based character segmentation
- Cartesian Coordinates based digit grouping
- Text auto-formatting using selected standard
- Operator detection and evaluation

Dhanush

- Digit recognition on antiques
- Vehicle number plate recognition
- Help the blind to recognize the currency Value
- Auto recognition of digits of different languages

Dhanushuram

- phone number detection using the country code
- Detection of matrix and the values in it
- cheque processing in ATM and bank
- identifying the prices on the price tags on products which the prices are punched

Vinoth

- Currency detection using currency symbol
- Depositing money using Deposit form in ATM
- Digitizing the digit records which are hand written
- improving Hand writing on screen during online classes

Handling Financial Information

- Currency detection using currency symbol
- Help the blind to recognize the currency Value
- Depositing money using Deposit form in ATM
- cheque processing in ATM and bank
- identifying the prices on the price tags on products which the prices are punched

Digitizing Physical Documents

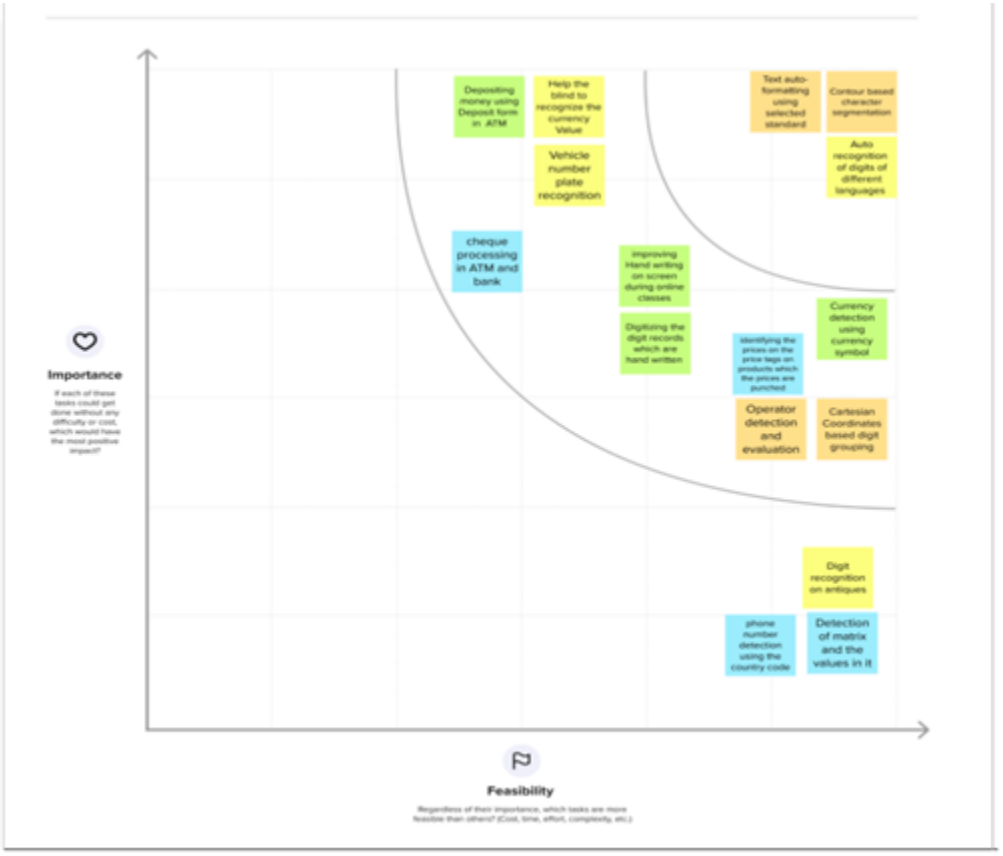
- Digitizing the digit records which are hand written
- Auto recognition of digits of different languages
- Vehicle number plate recognition
- Digit recognition on antiques
- phone number detection using the country code

Online Education Tools

- Operator detection and evaluation
- improving Hand writing on screen during online classes
- Detection of matrix and the values in it
- Cartesian Coordinates based digit grouping

General Functionality

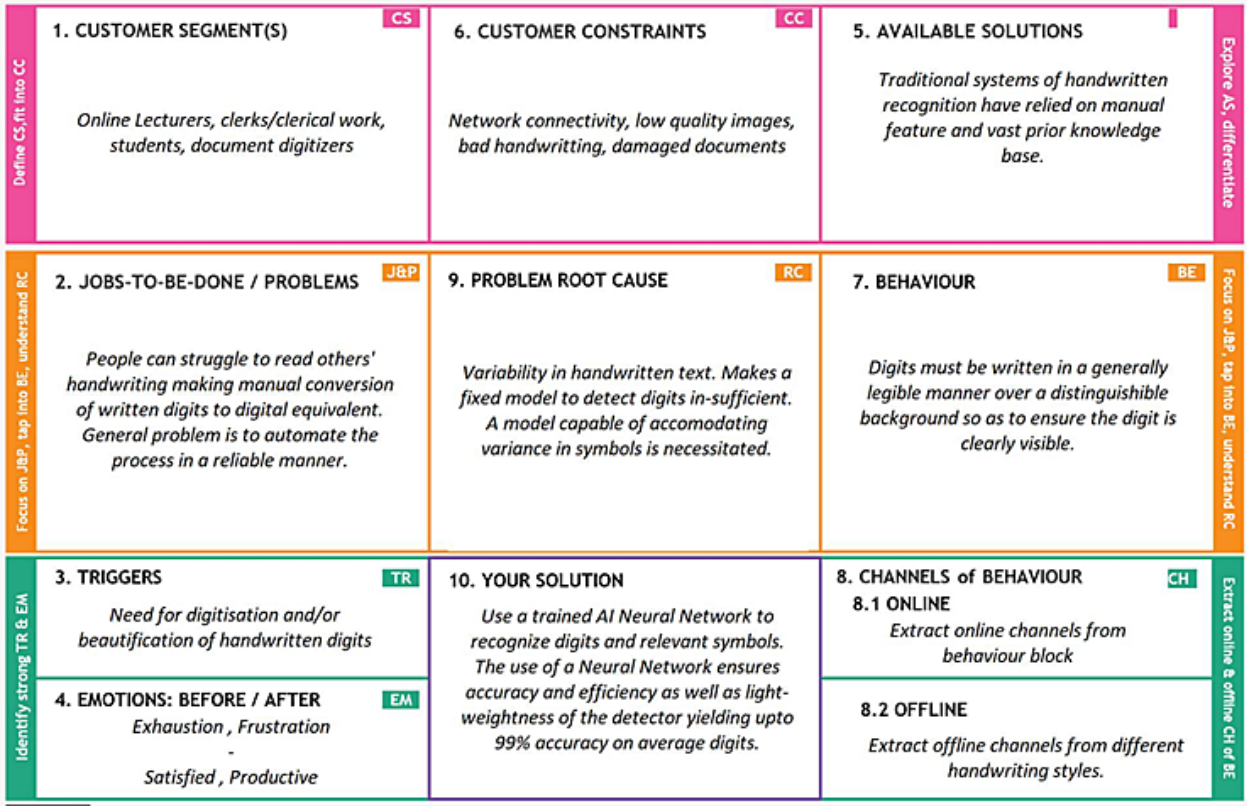
- Text auto-formatting using selected standard
- Contour based character segmentation



3. PROPOSED SOLUTION

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	• To classify handwritten digits. • To take an image of a handwritten digit and determine what that digit and character is.
2.	Idea / Solution description	Implementation of an AI predictive model to predict handwritten digits and to construct a neural network trained to detect digits and relevant symbols.
3.	Novelty / Uniqueness	The system not only produces a classification of the digit but also can detect relevant symbols such as mathematical operators and perform necessary calculations if prompted.
4.	Social Impact / Customer Satisfaction	Handwritten digits can be recognized easily without any strenuous efforts. This reduces time and improves productivity for people.
5.	Business Model (Revenue Model)	It is used in online classes as well as the detection of vehicle numbers, banks for reading cheques, post offices for arranging letters, and many other tasks.
6.	Scalability of the Solution	To attain higher performances in the domain of character recognition and pattern recognition, due to its excellent feature extraction and working as best classifier characteristics. As it is a trained model capable of being deployed on the client side, it is highly scalable in terms of in the number of digits that can be recognized.

4. **PROBLEM SOLUTION FIT**



CHAPTER 4
REQUIREMENT ANALYSIS

1. FUNCTIONAL REQUIREMENTS

Following are the functional requirements of the proposed solution.

FR No.	Sub Requirement (Story / Sub-Task)
FR-1	Source Image: Handwritten digit recognition refers to the ability of a computer to identify human handwritten digits from various sources, such as photographs, documents, touch screens, etc., and classify them into ten established classifications (0-9). In the field of deep learning, this has been the subject of countless studies.
FR-2	Digit Classifier Model: To train a convolutional network to predict the digit from an image, use the MNIST database of handwritten digits. get the training and validation data first.
FR-3	Cloud Services: The cloud offers a range of computing services including virtual storage, networks, servers, databases and applications. Simply put, cloud computing is described as a virtual platform that allows unlimited storage and access to your data over the internet.
FR-4	Modified National Institute of Standards and Technology dataset: The abbreviation MNIST stands for MNIST data set. It's a collection of 60,000 tiny square grayscale photos, each 28 by 28, containing simple handwritten numbers between 0 and 9.

1. NON FUNCTIONAL REQUIREMENTS

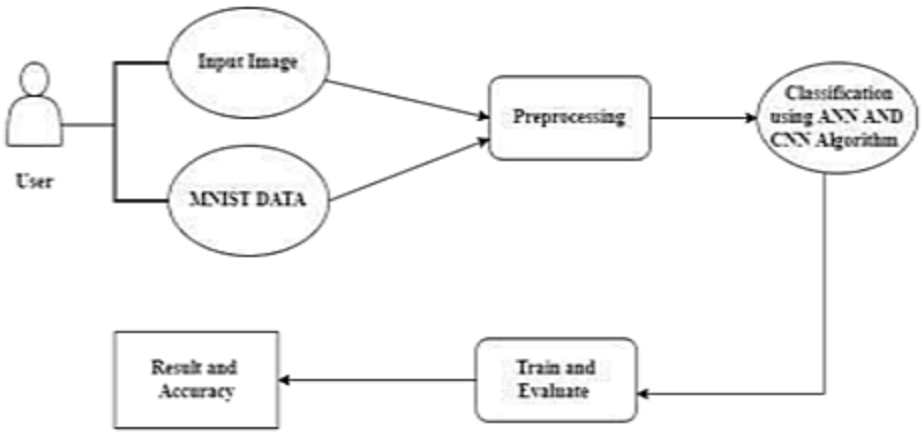
Following are the non-functional requirements of the proposed solution.

NFR-1	**Reliability**	The samples are used by the neural network to automatically derive rules for reading handwritten digits. Moreover, the network can learn more about handwriting and thus improve its accuracy by increasing the number of training instances. Many techniques and algorithms, e.g. Deep Learning/CNN, SVM, Gaussian Naive Bayes, KNN, Decision Trees, Random Forests, etc. can be used to recognize handwritten numbers.

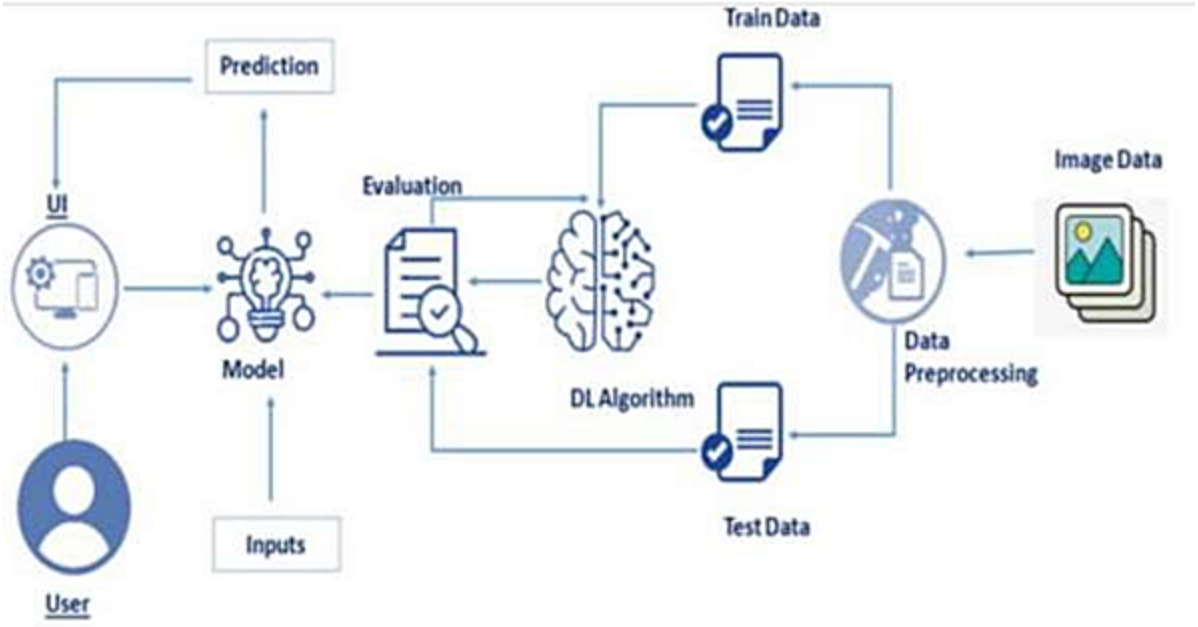
NFR-2	**Accuracy**	For text entered in high-quality photos, Optical Character Recognition (OCR) technology delivers accuracy rates of over 99%. However, spacing discrepancies, handwriting anomalies, and the diversity of human handwriting Styles cause less accurate character recognition.
NFR-3	**Functionality**	One of the very significant problems in pattern recognition applications is the recognition of handwritten characters. Applications for digit recognition include filling out forms, processing bank checks, and sorting mail.
NFR-4	**Robustness**	The system should be able to handle a variety of input values seamlessly.

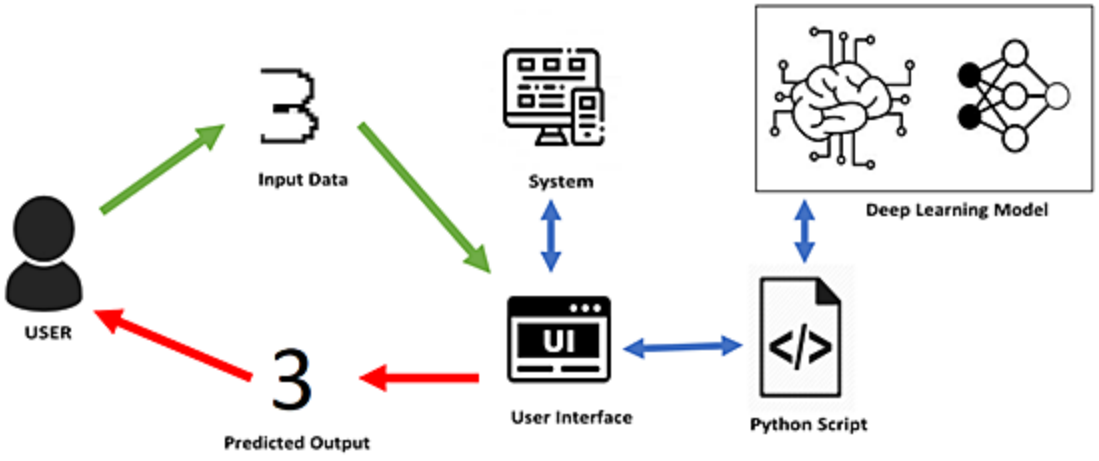
CHAPTER 5
PROJECT DESIGN

1. **DATA FLOW DIAGRAM**



2. SOLUTION & TECHNICAL ARCHITECTURE





3. USER STORIES

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer	Home	USN-1	As a user, I can view the guide and awareness to use this application.	I can view the awareness to use this application and its limitations.	Low	Sprint-1
		USN-2	As a user, I'm allowed to view the guided video 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 use this application.	I can read instructions also to use it in a userfriendly method.	Low	Sprint-2
	Recognize	USN-4	As a user, In this prediction page I get to choose 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 choose the image to be used	I can choose the image from the system storage	Medium	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 data set	I can access the MNIST data set to produce the accurate result.	Medium	Sprint-3

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer	Home	USN-1	As a user, I can view the guide and awareness to use this application.	I can view the awareness to use this application and its limitations.	Low	Sprint-1
		USN-2	As a user, I'm allowed to view the guided video 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 use this application.	I can read instructions also to use it in a user friendly method.	Low	Sprint-2
	Recognize	USN-10	As a user, I can use the application virtually anywhere.	I can use the application portably anywhere.	High	Sprint-1
		USN-12	As it is a portable application, it is installation free	I can use it without the installation of the application or any software.	Medium	Sprint-4
	Predict	USN-13	As a user, I'm Allowed to choose the image to be used	I can choose the image from the system storage	Medium	Sprint-3

CHAPTER 6
PROJECT PLANNING AND SCHEDULING

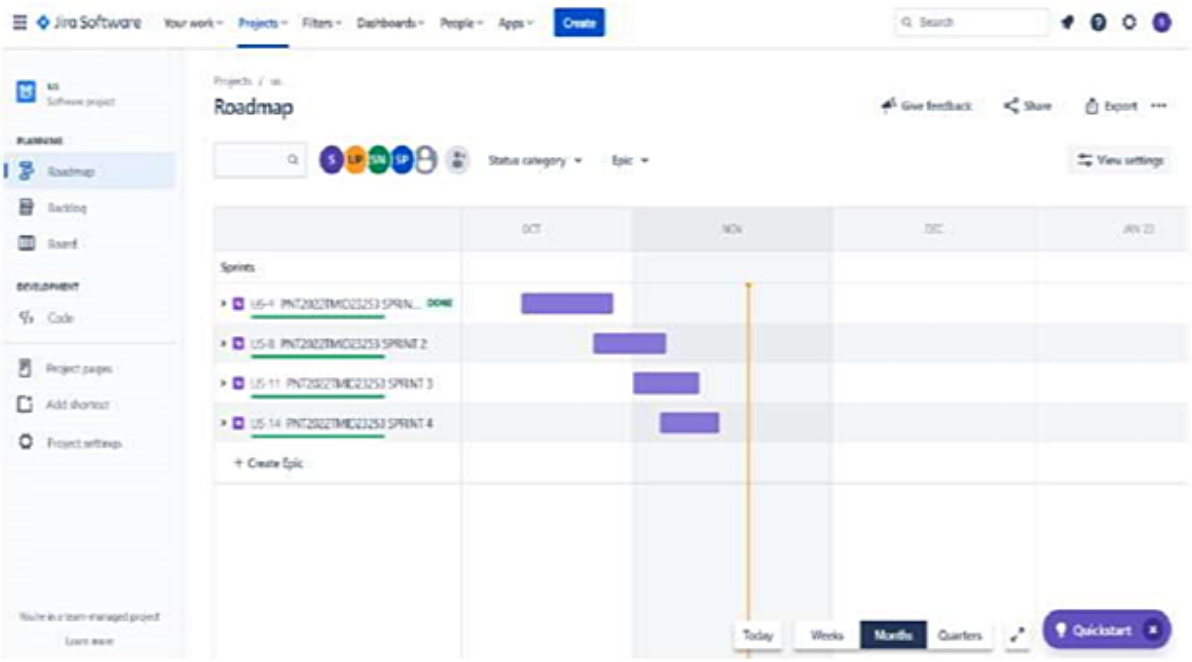
1. SPRINT PLANNING AND ESTIMATION

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Registration	USN-1	Registration through email and password	2	High	Rakhul K R
Sprint-1	Login	USN-2	User login using email and password	1	High	Dhanushram
Sprint-2	Upload Image of digital document	USN-3	Selecting the mode of input, i.e image or realtime	2	Medium	Vinoth R
Sprint-2	Prediction	USN-4	Digit prediction in image input	1	Medium	Rakhul K R
Sprint-3	Prediction	USN-5	Digit prediction in real-time input	2	High	Kankanala Dhanush
Sprint-3	Recognize text	USN-6	Output font formatting options	1	Medium	Dhanushram
Sprint-4	Recognize digit	USN-7	Export the generated content, i.e recognized digits and symbols	1	Medium	Vinoth R
Sprint-4	Recognize digit	USN-8	Perform mathematical operations based on present operators and symbols.	2	High	Rakhul K R

2. SPRINT DELIVERY SCHEDULE

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	2	6 Days	24 Oct 2022	29 Oct 2022	2	29 Oct 2022
Sprint-2	2	6 Days	31 Oct 2022	05 Nov 2022	2	05 Nov 2022
Sprint-3	2	6 Days	07 Nov 2022	12 Nov 2022	2	12 Nov 2022
Sprint-4	2	6 Days	14 Nov 2022	19 Nov 2022	2	19 Nov 2022

3. **REPORTS FROM JIRA:**



CHAPTER 7
CODING & SOLUTIONING

FEATURE 1

Importing the Required Libraries

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow.keras.datasets import mnist
from tensorflow.keras.models import Sequential
from tensorflow.keras import layers
from tensorflow.keras.layers import Dense,Flatten,MaxPooling2D
from tensorflow.keras.layers import Conv2D
from keras.optimizers import Adam
from keras.utils import np_utils
from tensorflow.keras.models import load_model
```

Loading the Data

```
(X_train,y_train),(X_test,y_test) = mnist.load_data()
```

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

```
print(X_train.shape)
print(X_test.shape)
```

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

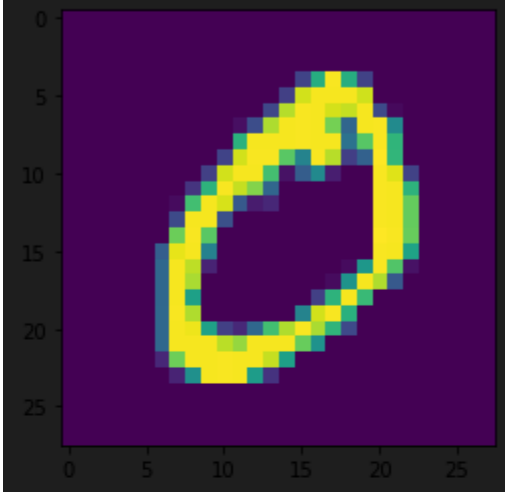
Analyzing the data

```
X_train[0]
```

```
Output exceeds the size limit. Open the full output data in a text editor
array([[  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
```

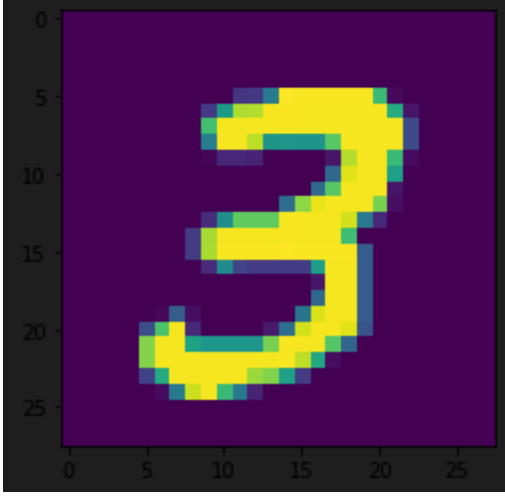
```
plt.imshow(X_train[1])
```

```
<matplotlib.image.AxesImage at 0x7fb7c978f890>
```




```
plt.imshow(X_train[7])
```

```
<matplotlib.image.AxesImage at 0x7fb7c926fe10>
```



Reshaping the data

```
X_train = X_train.reshape(60000,28,28,1).astype('floa
X_test = X_test.reshape(10000,28,28,1).astype('float3
```

```
X_train
```

```
Output exceeds the size limit. Open the full output data
array([[[[0.],
         [0.],
         [0.],
         ...,
         [0.],
         [0.],
         [0.]],

        [[0.],
         [0.],
         [0.],
         ...,
         [0.],
         [0.],
         [0.]],

        [[0.],
         [0.],
         [0.],
         ...,
         [0.],
         [0.],
         [0.]],
```

```
y_train = np_utils.to_categorical(y_train,number_of_classes)
y_test = np_utils.to_categorical(y_test,number_of_classes)
```

```
y_train[0]
```

```
array([0., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0.], dtype=float32)
```

```
y_test
```

```
array([[0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 1., ..., 0., 0., 0.],
       [0., 1., 0., ..., 0., 0., 0.],
       ...,
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.]], dtype=float32)
```

model creation

```
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(MaxPooling2D((2,2)))
model.add(Flatten())
model.add(Dense(number_of_classes,activation="softmax"))
```

Model compilation

```
model.compile(loss="categorical_crossentropy",optimizer="Adam",metrics=["accuracy"])
```

Train the model

```
model.fit(X_train,y_train,epochs=10,validation_data=(X_test,y_test),batch_size=32)
```

FEATURE 2

```
data_url = str(request.get_data())
offset = data_url.index(',')+1
img_bytes = base64.b64decode(data_url[offset:])
image_bytes = BytesIO(img_bytes)
image_bytes.seek(0)
img = Image.open(image_bytes)
```

```
with graph.as_default():
    total = ""

    for Rect in Rectangles:

        subimg = img.crop( Rect )
        #subimg.show()
        subimg = subimg.convert('L')
        subimg = subimg.resize((28, 28))
        print("Type : ", type(subimg))
        subimg = np.array(subimg)
        subimg = subimg.reshape(1, 28, 28, 1)


        out = model.predict( subimg )
        print(np.argmax(out, axis=1))
        total  += np.array_str(np.argmax(out, axis=1))

    total = total.replace( "[" , "" )
    total = total.replace( "]" , " ")
    response = {
        'data': total,
        'probencoded': str("probencoded"),
        'interpretencoded': str("interpretencoded"),
    }
    return jsonify(response)
```

CHAPTER 8
TESTING

1. TEST CASES

Test case ID	Feature Type	Component	Test Scenario	Expected Result	Actual Result	Status
UI_TC_001	UI	WebPage	Verify UI elements in the Home Page	The Home page must be displayed properly	Working as expected	PASS
UI_TC_002	UI	WebPage	Check if the UI elements are displayed properly in different screen sizes	The Home page must be displayed properly in all sizes	The UI is not displayed properly in screen size 1366x768	FAIL
F_TC_003	Functional	WebPage	Check if user can provide input	The input should be recorded successfully	Working as expected	PASS
F_TC_004	Functional	WebPage	Check if user's pattern is read as an image	The application read the input as an appropriate image format	Input is read as an image but of improper type	FAIL
F_TC_005	Functional	WebPage	Check if the page Displays the result once the input is given	The page should Display the result	Working as expected	PASS
F_TC_006	Functional	Model	Check if the model can handle various image sizes	The model should rescale the image and predict the results	Working as expected	PASS
F_TC_007	Functional	Model	Check if the model predicts the digit	The model should predict the number	Working as expected	PASS
F_TC_008	Functional	Model	Check if the model can handle complex input image	The model should predict the number in the complex image	The model fails to identify the digit since the model is not built to handle such data	FAIL

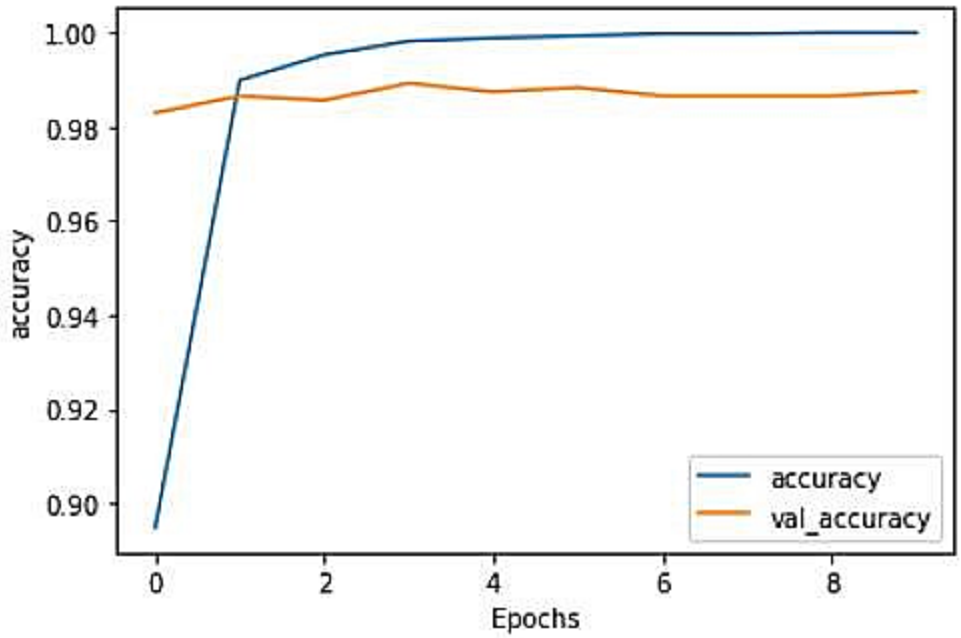
4. **USER ACCEPTANCE TESTING**

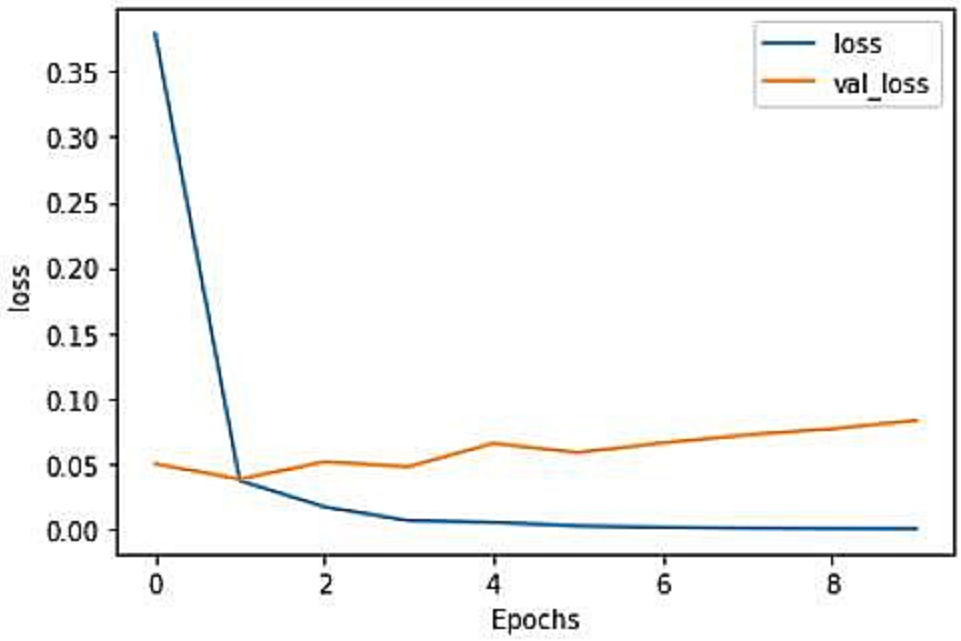
1. DEFECT ANALYSIS:

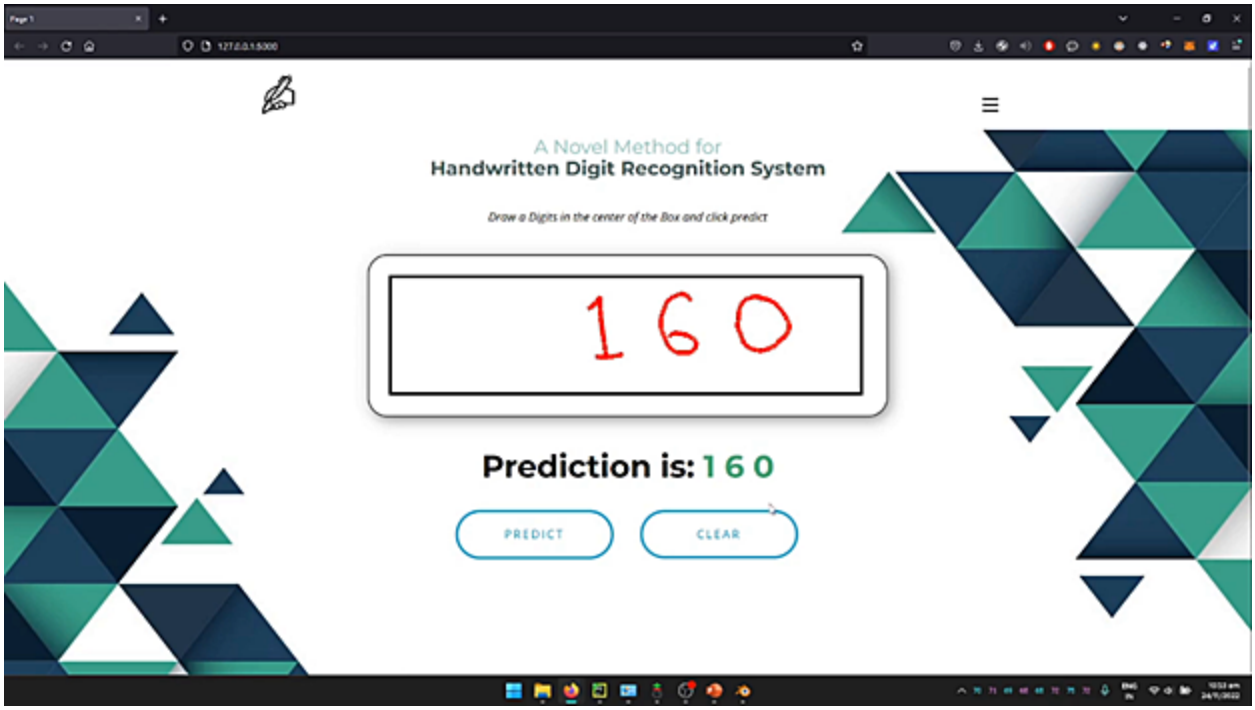
Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Total
By Design	1	0	1	0	2
Duplicate	0	0	0	0	0
External	0	0	2	0	2
Fixed	4	1	0	1	6
Not Reproduced	0	0	0	1	1
Skipped	0	0	0	1	1
Won't Fix	1	0	1	0	2
Total	6	1	4	3	14

CHAPTER 9
RESULTS

1. PERFORMANCE METRICS






A Novel Method for
Handwritten Digit Recognition System
Draw a Digits in the center of the Box and click predict
1 6 0
Prediction is: 1 6 0
PREDICT
CLEAR

CHAPTER 10
ADVANTAGES & DISADVANTAGES

ADVANTAGES

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

DISADVANTAGES

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

CHAPTER 11
CONCLUSION

This project demonstrated a web application that uses machine learning to recognise handwritten numbers. Flask, HTML, CSS, JavaScript, and a few other technologies were used to create this project. The model predicts the handwritten digit using a CNN network. During testing, the model achieved a 99.61% recognition rate. The proposed project is scalable and can easily handle a huge number of users. Since it is a web application, it is compatible with any device that can run a browser. This project is extremely useful in real-world scenarios such as recognizing number plates of vehicles, processing bank cheque amounts, numeric entries in forms filled up by hand (tax forms) and so on. There is so much room for improvement, which can be implemented in subsequent versions.

CHAPTER 12
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:

1. Add support to detect from digits multiple images and save the results
2. Add support to detect multiple digits
3. 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.

APPENDIX

SOURCE CODE

MODEL CREATION

Importing the Required Libraries

```
In [2]:
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow.keras.datasets import mnist
from tensorflow.keras.models import Sequential
from tensorflow.keras import layers
from tensorflow.keras.layers import Dense,Flatten,MaxPooling2D
from tensorflow.keras.layers import Conv2D
from keras.optimizers import Adam
from keras.utils import np_utils
from tensorflow.keras.models import load_model
```

Loading the 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
11490434/11490434 [==============================] - 0s 0us/step
```

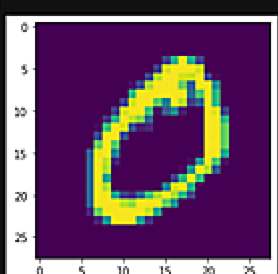
```
In [4]:
print(X_train.shape)
print(X_test.shape)

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

Analyzing the data

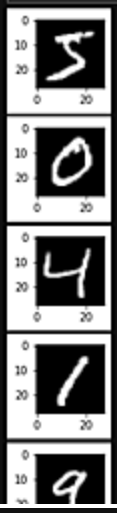
```
In [5]:
X_train[0]

Out[5]: array([[  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
        [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
        [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
        [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
        [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
```

```
In [6]:
y_train[2]

Out[6]: 4
```

```
In [7]:
plt.imshow(X_train[1])

Out[7]:
```



In [10]:
```
for i in range(9):
    plt.subplot(330 + 1 +i)
    plt.imshow(X_train[i], cmap = plt.get_cmap('gray'))
    plt.show()
```



In [11]:
```
X_train = X_train.reshape(60000,28,28,1).astype('float32')
X_test = X_test.reshape(10000,28,28,1).astype('float32')
```

In [12]:
```
X_train
```

Out[12]:
```
array([[[[0.],
```

In [14]:
```
number_of_classes = 20
```

In [15]:
```
y_train = np_utils.to_categorical(y_train,number_of_classes)
y_test = np_utils.to_categorical(y_test,number_of_classes)
```

In [16]:
```
y_train[0]
```

Out[16]:
```
array([0., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0.], dtype=float32)
```

In [17]:
```
y_test
```

Out[17]:
```
array([[0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 1., ..., 0., 0., 0.],
       [0., 1., 0., ..., 0., 0., 0.],
       ...,
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.]], dtype=float32)
```

model creation

In [18]:
```
model = Sequential()
```

In [19]:
```
model.add(Conv2D(64,(3,3),input_shape=(28,28,1),activation="relu"))
model.add(Conv2D(32,(3,3),activation="relu"))
model.add(MaxPooling2D((2,2)))
model.add(Flatten())
model.add(Dense(number_of_classes,activation="softmax"))
```

In [20]:
```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 26, 26, 64)        640
```

In [21]:
```
model.compile(loss="categorical_crossentropy",optimizer="Adam",metrics=["accuracy"])
```

Train the model

In [22]:
```
model.fit(X_train,y_train,epochs=10,validation_data=(X_test,y_test),batch_size=32)
```

```
Epoch 1/10
1875/1875 [==============================] - 205s 109ms/step - loss: 0.2086 - accuracy: 0.9527 - val_loss: 0.1306 - val_accuracy: 0.9633
Epoch 2/10
1875/1875 [==============================] - 198s 106ms/step - loss: 0.0678 - accuracy: 0.9789 - val_loss: 0.0683 - val_accuracy: 0.9790
Epoch 3/10
1875/1875 [==============================] - 200s 107ms/step - loss: 0.0527 - accuracy: 0.9832 - val_loss: 0.0616 - val_accuracy: 0.9821
Epoch 4/10
1875/1875 [==============================] - 194s 104ms/step - loss: 0.0397 - accuracy: 0.9877 - val_loss: 0.0617 - val_accuracy: 0.9809
Epoch 5/10
1875/1875 [==============================] - 194s 103ms/step - loss: 0.0336 - accuracy: 0.9897 - val_loss: 0.0779 - val_accuracy: 0.9789
Epoch 6/10
1875/1875 [==============================] - 195s 104ms/step - loss: 0.0278 - accuracy: 0.9913 - val_loss: 0.0727 - val_accuracy: 0.9822
Epoch 7/10
1875/1875 [==============================] - 196s 105ms/step - loss: 0.0245 - accuracy: 0.9923 - val_loss: 0.0883 - val_accuracy: 0.9813
Epoch 8/10
1875/1875 [==============================] - 193s 103ms/step - loss: 0.0229 - accuracy: 0.9929 - val_loss: 0.0777 - val_accuracy: 0.9835
Epoch 9/10
1875/1875 [==============================] - 192s 103ms/step - loss: 0.0207 - accuracy: 0.9937 - val_loss: 0.1031 - val_accuracy: 0.9826
Epoch 10/10
1875/1875 [==============================] - 193s 103ms/step - loss: 0.0199 - accuracy: 0.9947 - val_loss: 0.1164 - val_accuracy: 0.9809
```

Out[22]:

Observing the metrics

In [23]:
```
metrics=model.evaluate(X_test,y_test,verbose=0)
print("Metrics(Test Loss & Test Accuracy):")
print(metrics)
```

```
Metrics(Test Loss & Test Accuracy):
[0.11644930392503738, 0.98089998960495]
```

Test the model

In [24]:
```
prediction = model.predict(X_test[:4])
print(prediction)
```

```
1/1 [==============================] - 0s 86ms/step
[[5.06196601e-20 1.16798596e-23 2.76244476e-17 1.14661170e-13
  2.75562821e-22 6.83551556e-20 3.83435211e-25 1.00000000e+00
  7.52750449e-20 1.48724864e-13 1.33045053e-17 1.46596453e-18
  1.00764603e-17 3.57548117e-18 9.85595936e-19 2.06104188e-18
  9.82975614e-18 4.63783088e-18 1.92106576e-18 1.19297099e-19]
 [1.33132827e-14 1.61122770e-13 1.00000000e+00 8.10183457e-15
```

```
2.145485 35e-16 7.447676 98e-18 9.361997 09e-17 4.068557 60e-17]
[1.00000000e+00 2.14885948e-23 1.28019535e-12 9.80378986e-21
 8.50560178e-21 8.10558987e-19 1.03540414e-19 1.77055102e-19
 1.97019207e-15 1.47467732e-11 3.30529882e-21 2.22166108e-24
 4.99030947e-23 1.11927780e-20 2.43696197e-20 1.04738628e-25
 1.44798338e-20 7.46245251e-22 8.71647871e-21 1.45710951e-20]
[2.16067980e-17 1.96034222e-14 2.57757764e-22 1.54627602e-20
 1.00000000e+00 2.56732693e-19 3.95305353e-16 2.60721191e-17
 1.40730391e-13 1.13418755e-13 4.88156916e-18 1.48735685e-20
 1.55573003e-21 1.92202591e-18 8.20883597e-19 2.83245872e-23
 4.00272988e-18 1.74067669e-18 1.04656711e-18 1.30661925e-18]]
```

In [28]:
```
print(np.argmax(prediction,axis = 1))
print(y_test[1:5])
```

```
[2 1 0 4]
[[0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]]
```

Save The Model

In [29]:
```
model.save("Detector.h5")
```

FLASK APP

```
 import base64

import os

import re

import sys

from io import BytesIO

import random

import numpy as np

from PIL import Image , ImageFilter

from flask import Flask, render_template, request, jsonify

from imageio.v2 import imread

import cv2

import model

sys.path.append(os.path.abspath("./model"))



app = Flask(__name__)

global model, graph



import tensorflow as tf

from tensorflow import keras

import keras.models

import cv2

from keras import layers
```

```
@app.route('/')
def index():
    return render_template("default.html")



@app.route('/predict/', methods=['GET', 'POST'])
def predict():
    # get data from drawing canvas and save as image
    parseImage(request.get_data())

    # read parsed image back in 8-bit, black and white mode (L)
    x = imread('output.jpg', mode='L')
    x = np.invert(x)
    x = cv2.resize(x, (28, 28))

    # reshape image data for use in neural network
    x = x.reshape(1, 28, 28, 1)
    with graph.as_default():
        out = model.predict(x)
        print(out)
        print(np.argmax(out, axis=1))
        response = np.array_str(np.argmax(out, axis=1))
```

```
        return response



@app.route("/process", methods=["GET", "POST"])
def process():
    tf.compat.v1.disable_eager_execution()
    num_classes = 10
    img_rows, img_cols = 28, 28
    input_shape = (img_rows, img_cols, 1)
    # model = Sequential()


    model = keras.Sequential(
        [
            keras.Input(shape=input_shape),
            layers.Conv2D(32, kernel_size=(3, 3), activation="relu"),
            layers.MaxPooling2D(pool_size=(2, 2)),
            layers.Conv2D(64, kernel_size=(3, 3), activation="relu"),
            layers.MaxPooling2D(pool_size=(2, 2)),
            layers.Flatten(),
            layers.Dropout(0.5),
            layers.Dense(num_classes, activation="softmax"),
        ]
```

```
    )



    model.load_weights("Detector.h5")

    print("Loaded Model from disk")

    print(model.summary())



    # compile and evaluate loaded model


model.compile(loss="categorical_crossentropy",optimizer="Adam",metrics=["accur
acy"]) #(loss=keras.losses.categorical_crossentropy,
optimizer=keras.optimizers.Adadelta(),metrics=['accuracy'])




    graph = tf.compat.v1.get_default_graph()







    data_url = str(request.get_data())

    offset = data_url.index(',')+1

    img_bytes = base64.b64decode(data_url[offset:])

    image_bytes = BytesIO(img_bytes)

    image_bytes.seek(0)

    img = Image.open(image_bytes)
```

```
    pil_image = img.convert('RGB')

    open_cv_image = np.array(pil_image)

    # Convert RGB to BGR

    open_cv_image = open_cv_image[:, :, ::-1].copy()

    result = open_cv_image.copy()



    kernel = np.ones((5, 5), np.uint8)

    open_cv_image = cv2.dilate(open_cv_image, kernel, iterations=3)

    open_cv_image = cv2.GaussianBlur(open_cv_image, (7, 7), 0)



    imgray = cv2.cvtColor(open_cv_image, cv2.COLOR_BGR2GRAY)

    ret, thresh = cv2.threshold(imgray, 5, 255, cv2.THRESH_BINARY)

    contours, hier = cv2.findContours(thresh, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)

    colour = (255, 0, 0)

    thickness = 1

    i = 0



    Rectangles = []



    for cntr in contours:

        x1, y1, w, h =  cv2.boundingRect(cntr)
```

```
        if w>h:
            h = w
        else:
            w = h

        Rectangles += [ ( x1-int(w/2) , y1 , x1+w , y1+h ) ]
        x1, y1, w, h = Rectangles[-1]
        x2 = w
        y2 = h
        cv2.rectangle(result, (x1, y1), (x2, y2), colour, thickness)
        print("Object:", i + 1, "x1:", x1, "x2:", x2, "y1:", y1, "y2:", y2)
        i += 1

    Rectangles = sorted(Rectangles , key=lambda x : x[0])
    print(Rectangles)



    img = img.filter(ImageFilter.GaussianBlur(5))

    with graph.as_default():
        total = ""

        for Rect in Rectangles:
```

```
        subimg = img.crop( Rect )
        #subimg.show()
        subimg = subimg.convert('L')
        subimg = subimg.resize((28, 28))
        print("Type : ", type(subimg))
        subimg = np.array(subimg)
        subimg = subimg.reshape(1, 28, 28, 1)



        out = model.predict( subimg )
        print(np.argmax(out, axis=1))
        total  += np.array_str(np.argmax(out, axis=1))


    total = total.replace( "[" , "" )
    total = total.replace( "]" , " ")
    response = {
        'data': total,
        'probencoded': str("probencoded"),
        'interpretencoded': str("interpretencoded"),
    }
    return jsonify(response)
```

```
def parseImage(imgData):

    # parse canvas bytes and save as output.png

    imgstr = re.search(b'base64,(.*)', imgData).group(1)

    with open('output.jpg', 'wb') as output:

        output.write(base64.decodebytes(imgstr))




if __name__ == '__main__':

    app.debug = True

    port = int(os.environ.get("PORT", 5000))

    app.run(host='0.0.0.0', port=port)
```

HOME PAGE (HTML)

```
 <!DOCTYPE html>

<html style="font-size: 16px;" lang="en"><head>

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <meta charset="utf-8">

    <meta name="keywords" content="Happiness &amp;amp; Mindfulness Courses,
Welcome Message, Benefits of working with us, 01, 02, 03, 04, 05, 06, How
Coaching Works, How and where to learn mindfulness, Meet The Team

Our Professionals, Start using Our App for free">

    <meta name="description" content="">
```

```
    <title>Page 1</title>

    <link rel="stylesheet" href="{{url_for('static', filename='nicepage.css')
}}" media="screen">

<link rel="stylesheet" href="{{url_for('static', filename='nicepage-site.css')
}}" media="screen">

<link rel="stylesheet" href="{{url_for('static', filename='Page-1.css') }}"
media="screen">

    <script class="u-script" type="text/javascript"
src="//capp.nicepage.com/assets/jquery-3.5.1.min.js" defer=""></script>

    <script class="u-script" type="text/javascript"
src="//capp.nicepage.com/f348af15ed2cdeede576d36e56b797a3c242e24f/nicepage.js"
defer=""></script>

    <meta name="generator" content="Nicepage 5.0.19, nicepage.com">

    <link id="u-theme-google-font" rel="stylesheet"
href="https://fonts.googleapis.com/css?family=Montserrat:100,100i,200,200i,300
,300i,400,400i,500,500i,600,600i,700,700i,800,800i,900,900i|Open+Sans:300,300i
,400,400i,500,500i,600,600i,700,700i,800,800i">

    <link id="u-page-google-font" rel="stylesheet"
href="https://fonts.googleapis.com/css?family=Montserrat:100,100i,200,200i,300
,300i,400,400i,500,500i,600,600i,700,700i,800,800i,900,900i|Montserrat:100,100
i,200,200i,300,300i,400,400i,500,500i,600,600i,700,700i,800,800i,900,900i">

    <script type="text/javascript" src="{{url_for('static',
filename='jquery.min.js') }}"></script>

<link rel="stylesheet" type="text/css" href="{{url_for('static',
filename='style.css') }}">

    <script type="text/javascript">

    var canvas, ctx, flag = false,

        prevX = 0,

        currX = 0,

        prevY = 0,
```

```
    currY = 0,
    dot_flag = false;


var x = "red",
    y = 8;


function init() {
    canvas = document.getElementById('can');


    ctx = canvas.getContext("2d");
    w = canvas.width;
    h = canvas.height;


    canvas.addEventListener("mousemove", function (e) {
        findxy('move', e)
    }, false);
    canvas.addEventListener("mousedown", function (e) {
        findxy('down', e)
    }, false);
    canvas.addEventListener("mouseup", function (e) {
        findxy('up', e)
        console.log("clicked")
```

```
        }, false);
        canvas.addEventListener("mouseout", function (e) {
            findxy('out', e)
        }, false);
        console.log("Initialised")
    }


    function draw() {
        ctx.beginPath();
        ctx.moveTo(prevX, prevY);
        ctx.lineTo(currX, currY);
        ctx.strokeStyle = x;
        ctx.lineWidth = y;
        ctx.stroke();
        ctx.closePath();
    }


    function erase() {
        ctx.clearRect(0, 0, w, h);
        document.getElementById("canvasimg").style.display = "none";
        document.getElementById("prediction").style.display = "none";
```

```
    }

    function save() {
        document.getElementById("prediction").style.display = "block";

        var final_image = canvas.toDataURL( "image/jpeg" );
        console.log( final_image )
        var a = document.createElement('a');
        a.href = final_image;
        a.download = 'process.jpg';
        document.body.appendChild(a);
        // a.click();
        $.ajax({
            url: "{{ url_for('process') }}",
            type: 'POST',
            data: final_image,
            success: function (response) {
                endresult = JSON.parse(JSON.stringify(response))
                console.log(endresult)
                $('#prediction').html('Prediction is: <span id="text">' +
endresult.data + '</span>')
                $('#probs').prop('src', '' +
```

```
endresult.probencoded)

                $('#interpret').prop('src', '' +
endresult.interpretencoded)

            }

        });

    }



    function findxy(res, e) {

        if (res == 'down') {

            prevX = currX;

            prevY = currY;

                currX = e.clientX - canvas.offsetLeft - 564;

                currY = e.clientY - canvas.offsetTop - 310;



            flag = true;

            dot_flag = true;

            if (dot_flag) {

                ctx.beginPath();

                ctx.fillStyle = x;

                ctx.fillRect(currX, currY, 2, 2);

                ctx.closePath();

                dot_flag = false;

            }
```

```
        }
        if (res == 'up' || res == "out") {
            flag = false;
        }
        if (res == 'move') {
            if (flag) {
                prevX = currX;
                prevY = currY;
                currX = e.clientX - canvas.offsetLeft - 564;
                currY = e.clientY - canvas.offsetTop - 310;
                console.log(currX + " - " + currY)
                draw();
                console.log("drawn")
            }
        }
        console.log("called")
    }
</script>

    <script type="application/ld+json">{
      "@context": "http://schema.org",
      "@type": "Organization",
      "name": "",
```

```
      "url": "/",

      "logo":
"//images01.nicepagecdn.com/c7babaed154234a3f5292c18/79e065495802520188662215/
2600896-200.png"

}</script>

    <meta name="theme-color" content="#95ccc3">

    <meta property="og:title" content="Page 1">

    <meta property="og:type" content="website">

    <link rel="canonical" href="/">

  </head>

  <body data-home-page="https://website2993530.nicepage.io/Page-
1.html?version=049dd8d8-9717-40e5-8ffc-461e18cf63bb" data-home-page-title="Page
1" class="u-body u-xl-mode" data-lang="en"><header class="u-clearfix u-header
u-header" id="sec-cdb9"><div class="u-clearfix u-sheet u-valign-middle u-sheet-
1">




      <a href="https://nicepage.com" class="u-image u-logo u-image-1" data-
image-width="200" data-image-height="200">

          <img
src="//images01.nicepagecdn.com/c7babaed154234a3f5292c18/79e065495802520188662
215/2600896-200.png" class="u-logo-image u-logo-image-1">

        </a>

        <nav class="u-menu u-menu-hamburger u-offcanvas u-menu-1" data-
responsive-from="XL">

          <div class="menu-collapse" style="font-size: 1rem; letter-spacing:
0px; font-weight: 700;">

            <a class="u-button-style u-custom-left-right-menu-spacing u-
```

```
custom-padding-bottom u-custom-text-active-color u-custom-top-bottom-menu-
spacing u-nav-link u-text-active-palette-1-base u-text-hover-palette-2-base"
href="#">

              <svg class="u-svg-link" viewBox="0 0 24 24"><use
xlink:href="#menu-hamburger"></use></svg>

              <svg class="u-svg-content" version="1.1" id="menu-hamburger"
viewBox="0 0 16 16" x="0px" y="0px" xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns="http://www.w3.org/2000/svg"><g><rect y="1" width="16"
height="2"></rect><rect y="7" width="16" height="2"></rect><rect y="13"
width="16" height="2"></rect>

</g></svg>

            </a>

          </div>

          <div class="u-custom-menu u-nav-container">

            <ul class="u-nav u-unstyled u-nav-1"><li class="u-nav-item"><a
class="u-button-style u-nav-link u-text-active-palette-2-base u-text-hover-
palette-2-base" href="#" style="padding: 10px 20px;">Home</a>

</li></ul>

          </div>

          <div class="u-custom-menu u-nav-container-collapse">

            <div class="u-black u-container-style u-inner-container-layout u-
opacity u-opacity-95 u-sidenav">

              <div class="u-inner-container-layout u-sidenav-overflow">

                <div class="u-menu-close"></div>

                <ul class="u-align-center u-nav u-popupmenu-items u-unstyled
u-nav-2"><li class="u-nav-item"><a class="u-button-style u-nav-link"
href="#">Home</a>

</li></ul>

              </div>
```

```
            </div>

            <div class="u-black u-menu-overlay u-opacity u-opacity-70"></div>

          </div>

        </nav>

      </div></header>

    <section class="u-align-center u-clearfix u-image u-section-1" id="sec-edee" data-image-width="1980" data-image-height="1134">

      <div class="u-clearfix u-sheet u-sheet-1">

        <h1 class="u-text u-text-palette-1-base u-text-1" data-animation-name="customAnimationIn" data-animation-duration="1500" data-animation-delay="0"> A Novel Method for <span style="font-weight: 700;" class="u-text-palette-1-dark-2">

            <br>Handwritten Digit Recognition System

          </span>

        </h1>

        <p class="u-text u-text-2" data-animation-name="customAnimationIn" data-animation-duration="1500" data-animation-delay="500"> Draw a Digits in the center of the Box and click predict<br>

        </p>

        <div class="u-border-2 u-border-grey-75 u-container-style u-group u-radius-30 u-shape-round u-white u-group-1">

          <div class="u-container-layout u-container-layout-1">
```

```
          <canvas id="can" width="720px" height="175px"></canvas>

          <img id="canvasimg">




          </div>

        </div>



        <h2 class="u-text u-text-3" id="prediction">Sample Headline</h2>

        <a onclick="save()" class="u-active-palette-4-base u-border-4 u-
border-active-palette-4-base u-border-hover-palette-4-base u-border-palette-4-
base u-btn u-btn-round u-button-style u-hover-palette-4-base u-radius-50 u-
text-active-white u-text-hover-white u-btn-1" data-animation-
name="customAnimationIn" data-animation-duration="1750" data-animation-
delay="500">PREDICT</a>

        <a onclick="erase()" class="u-active-palette-4-base u-border-4 u-
border-active-palette-4-base u-border-hover-palette-4-base u-border-palette-4-
base u-btn u-btn-round u-button-style u-hover-palette-4-base u-radius-50 u-
text-active-white u-text-hover-white u-btn-2" data-animation-
name="customAnimationIn" data-animation-duration="1750" data-animation-
delay="500">CLEAR</a>



      </div>

    </section>







    <footer class="u-align-center u-clearfix u-footer u-grey-80 u-footer"
id="sec-44c2"><div class="u-clearfix u-sheet u-sheet-1">
```

```
        <p class="u-small-text u-text u-text-variant u-text-1">Never gonna
give you up<br>

        </p>

      </div></footer>

    <section class="u-backlink u-clearfix u-grey-80">

      <a class="u-link" href="https://nicepage.com/website-mockup"
target="_blank">

        <span>Website Mockup</span>

      </a>

      <p class="u-text">

        <span>created with</span>

      </p>

      <a class="u-link" href="https://nicepage.best" target="_blank">

        <span>Best Free Website Builder</span>

      </a>.

    </section>

    <script type="text/javascript">

    init()

    </script>

</body></html>
```

TRAIN THE MODEL

```
import numpy as np
```

```
from tensorflow import keras

from keras import layers



"""

## Prepare the data

"""



# Model / data parameters

num_classes = 10

input_shape = (28, 28, 1)



# Load the data and split it between train and test sets

(x_train, y_train), (x_test, y_test) = keras.datasets.mnist.load_data()



# Scale images to the [0, 1] range

x_train = x_train.astype("float32") / 255

x_test = x_test.astype("float32") / 255

# Make sure images have shape (28, 28, 1)

x_train = np.expand_dims(x_train, -1)

x_test = np.expand_dims(x_test, -1)

print("x_train shape:", x_train.shape)

print(x_train.shape[0], "train samples")

print(x_test.shape[0], "test samples")
```

```
# convert class vectors to binary class matrices
y_train = keras.utils.to_categorical(y_train, num_classes)
y_test = keras.utils.to_categorical(y_test, num_classes)

"""
## Build the model
"""

model = keras.Sequential(
    [
        keras.Input(shape=input_shape),
        layers.Conv2D(32, kernel_size=(3, 3), activation="relu"),
        layers.MaxPooling2D(pool_size=(2, 2)),
        layers.Conv2D(64, kernel_size=(3, 3), activation="relu"),
        layers.MaxPooling2D(pool_size=(2, 2)),
        layers.Flatten(),
        layers.Dropout(0.5),
        layers.Dense(num_classes, activation="softmax"),
    ]
)
```

```
model.summary()


"""
## Train the model
"""


batch_size = 128
epochs = 15


model.compile(loss="categorical_crossentropy", optimizer="adam",
metrics=["accuracy"])


model.fit(x_train, y_train, batch_size=batch_size, epochs=epochs,
validation_split=0.1)


"""
## Evaluate the trained model
"""


score = model.evaluate(x_test, y_test, verbose=0)
print("Test loss:", score[0])
print("Test accuracy:", score[1])
model.save("Detector.h5")
```

GITHUB

https://github.com/IBM-EPBL/IBM-Project-5308-1658757138/tree/master

PROJECT DEMO

https://youtu.be/6Lej7RdrRCk