

A Novel Method for Handwritten Digit Recognition System

IBM Project

Submitted by

BALAJI G **142219205010**

BALASUBRAMANIAN **142219205012**
KALYAN

ELA BARATH **142219205019**

GOKUL ARAVIND V **142219205022**

in a partial fulfilment for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

INFORMATION TECHNOLOGY



SRM VALLIAMMAI ENGINEERING COLLEGE

(AN AUTONOMOUS INSTITUTION)

SRM NAGAR, KATTANKULATHUR,

ANNA UNIVERSITY: CHENNAI 600 025

DEC 2022

ABSTRACT

Because everyone in the world has a unique writing style, handwriting identification is one of the fascinating research projects now being conducted. It is the ability of a computer to automatically recognise and comprehend handwritten numbers or letters. Every aspect of life is being digitalized to lessen the need for human labour as a result of advancements in science and technology. Thus, handwritten digit recognition is required in many real-time applications. The MNIST data collection, which contains 70000 handwritten digits, is frequently utilised for this recognition method. In order to train these photos and create a deep learning model, we use artificial neural networks. A web application is developed that allows users to upload pictures of handwritten numbers. The model analyses this image, and the detected result is sent back to the user interface.

TABLE OF CONTENTS

CHAPTER	TITLE	PAGE NO
	ABSTRACT	ii
	LIST OF FIGURES	v
	LIST OF TABLES	vi
1	INTRODUCTION	1
	1.1 PROJECT OVERVIEW	2
	1.2 PURPOSE	3
2	LITERATURE SURVEY	4
	2.1 EXISTING PROBLEM	7
	2.2 REFERENCES	8
	2.3 PROBLEM STATEMENT DEFINITION	8
3	IDEATION & PROPOSED SOLUTION	9
	3.1 EMPATHY MAP CANVAS	10
	3.2 IDEATION & BRAINSTORMING	11
	3.3 PROPOSED SOLUTION	14
	3.4 PROBLEM SOLUTION FIT	15
4	REQUIREMENT ANALYSIS	16
	4.1 FUNCTIONAL REQUIREMENTS	17
	4.2 NON-FUNCTIONAL REQUIREMENTS	18
5	PROJECT DESIGN	21
	5.1 DATA FLOW DIAGRAM	22

	5.2 SOLUTION & TECHNICAL ARCHITECTURE	23
	5.3 USER STORIES	25
6	PROJECT PLANNING & SCHEDULING	26
	6.1 SPRINT PLANNING & ESTIMATION	27
	6.2 SPRINT DELIVERY SCHEDULE	27
	6.3 REPORT FROM JIRA	28
7	CODING & SOLUTIONING	29
	7.1 CNN MODEL	30
	7.2 THE USER INTERFACE	33
8	RESULT	37
	8.1 PERFORMANCE METRICS	38
9	ADVANTAGES AND DISADVANTAGES	39
10	CONCLUSION	41
11	FUTURE SCOPE	43
12	APPENDIX	45
	SOURCE CODE	46
	GITHUB LINK	55

LIST OF FIGURES

S.NO	FIG NO	TITLE	PAGE NO
1	3.1	EMPATHY MAP	10
2	3.2	BRAINSTORM	11
3	3.3	GROUP IDEAS	12
4	3.4	PRIORITIZE	13
5	3.5	PROBLEM SOLUTION FIT	15
6	5.1	DATA FLOW DIAGRAM	22
7	5.2	SOLUTION ARCHITECTURE	23
8	5.3	TECHNICAL ARCHITECTURE	24
9	6.1	BURNDOWN CHART	28
10	7.1	HOME PAGE(MAIN.HTML)	34
11	7.2	INDEX6.HTML	35
12	7.3	OUTPUT 1	36
13	7.4	OUTPUT 2	36

LIST OF TABLES

S.NO	TABLE NO	TITLE	PAGE NO
1	3.1	PROPOSED SOLUTION	14
2	5.1	USER STORIES	25
3	6.1	SPRINT PLANNING & ESTIMATION	27
4	6.2	SPRINT DELIVERY SCHEDULE	27

CHAPTER-1

INTRODUCTION

1.1 Project Overview

The ability of a computer to detect human handwritten digits from various sources, such as photographs, papers, touch displays, etc. and classify them into 10 specified classes is known as handwritten digit recognition (0-9). In the realm of deep learning, this has been the subject of countless studies. Numerous uses for digit recognition include processing bank checks, identifying licence plates, and sorting mail at the post office. Because handwritten digit recognition is not optical character recognition, there are numerous difficulties due to the wide variety of writing styles used by different cultures. This study offers a thorough evaluation of various deep learning and machine learning methods for handwritten digit recognition. We have a convolutional neural network for this. Convolutional layer added into CNN decreases high dimensionality of images without sacrificing information. CNN's fundamental advantage over its forerunners is that it uses machine learning to identify key elements without human intervention. Any model's correctness is crucial since more accurate models produce better results. Low precision models are unsuitable for use in practical situations. High precision is essential for automated bank check processing systems that can read the amount and date on the check, for instance. It is not ideal for the system to wrongly identify a digit because this could result in serious harm. In these real-world applications, a high accuracy algorithm is necessary. We train these photos using an artificial convolutional neural network and create a deep learning model. A web application is developed that allows users to upload pictures of handwritten numbers. The model analyses this image, and the identified result is sent back to the user interface.

1.2 Purpose

Due to the fact that everyone in the world has a distinctive writing style, handwriting identification is one of the most interesting areas of research now being conducted. It is the ability of a computer to recognise and comprehend manually entered numbers or letters automatically. Everything is becoming digitalized to minimise human effort as a result of advancements in science and technology. As a result, many real-time applications require the ability to recognise handwritten digits. For this recognition process, the MNIST data collection, which contains 70000 handwritten digits, is frequently employed. The formatting, accurate character segmentation, and word-finding processes are all handled by a handwriting recognition system. Applications for digit recognition include form data entry, processing bank checks, and sorting postal mail.

CHAPTER-2

LITERATURE SURVEY

Literature Survey:

1. Digit Recognition Using Various Machine Learning Algorithms and Models

Handwritten digit recognition is one of the important problems in computer vision these days. There is a great interest in this field because of many potential applications, most importantly where large number of documents must be dealt such as post mail sorting, bank cheque analysis, handwritten form processing etc. So, a system should be designed in such a way that it is capable of reading handwritten digits and provide appropriate results. This paper presents a survey on various neural network approaches to recognize handwritten digits.

2. Diagonal based feature extraction for handwritten character recognition system using neural network

An off-line handwritten alphabetical character recognition system using multilayer feed forward neural network is described in the paper. A new method, called, diagonal based feature extraction is introduced for extracting the features of the handwritten alphabets. Fifty data sets, each containing 26 alphabets written by various people, are used for training the neural network and twenty different handwritten alphabets characters are used for testing. The proposed recognition system performs quite well yielding higher levels of recognition accuracy compared to the systems employing the conventional horizontal and vertical methods of feature extraction. This system will be suitable for converting handwritten documents into structural text form and recognizing handwritten names.

3. Improving Offline Handwritten Text Recognition with Hybrid HMM/ANN Models

The structural part of the optical models has been modeled with Markov chains, and a Multilayer Perceptron is used to estimate the emission probabilities. This paper also presents new techniques to remove slope and slant from handwritten text and to normalize the size of text images with supervised learning methods. Slope correction and size normalization are achieved by classifying local extrema of text contours with Multilayer Perceptrons. Slant is also removed in a nonuniform way by using Artificial Neural Networks. Experiments have been conducted on offline handwritten text lines from the IAM database, and the recognition rates achieved, in comparison to the ones reported in the literature, are among the best for the same task.

4. Recognition of handwritten similar Chinese characters by self-growing probabilistic decision-based neural networks

The self-growing probabilistic decision-based neural network (SPDNN) is a probabilistic type neural networks, which adopts a hierarchical network structure with nonlinear basis functions and a competitive credit-assignment scheme. Based on the SPDNN model, we constructed a three stage recognition system. The prototype system demonstrates a successful utilisation of SPDNN to similar handwritten Chinese recognition on the public database CCL/HCCRI (5401 characters /spl times/200 samples). Regarding the performance, the experiments on the CCL/HCCRI database demonstrated a 90.12% of recognition accuracy with no rejection and 94.11% of accuracy with 6.7% rejection rates, respectively.

2.1 Existing Problem

Numerous processes are possible using number recognition, including processing bank checks, sorting postal mail, and number plate recognition. We face a lot of difficulties in handwritten number recognition. because different people write in various ways and because optical character recognition is not used. This investigation offers a thorough comparison of various deep literacy and machine literacy algorithms for handwritten number recognition.

2.2 References

[1] Sakshic;, Dr.Kusum gupta ,*"Handwritten digit recognition using various neural network approaches"*, *International Journal of Advanced Research in Computer and Communication Engineering*.

[2] J. Pradeep; E. Srinivasan; S. Himavathi, *"Diagonal based feature extraction for handwritten character recognition system using neural network"*

[3] S. España-Boquera; M.J. Castro-Bleda; J. Gorbe-Moya; F. Zamora-Martinez, *"Improving Offline Handwritten Text Recognition with Hybrid HMM/ANN Models"*

[4] Hsin-Chia Fu; Y.Y. Xu, *"Recognition of handwritten similar Chinese characters by self growing probabilistic decision-based neural networks"*

2.3 Problem Statement Definition

The capacity of computer programmes to detect human handwritten digits is known as handwritten digit recognition. Because handwritten figures are not always accurate and can take many various forms and sizes, it is a difficult work for the machine. A solution to this issue is the handwritten digit recognition system, which uses an image of a digit to identify the digit that is contained in the image. To recognise handwritten numbers, a convolutional neural network model was developed using the PyTorch library and the MNIST dataset.

CHAPTER-3

IDEATION & PROPOSED SOLUTION

3.1 Empathy Map Canvas:

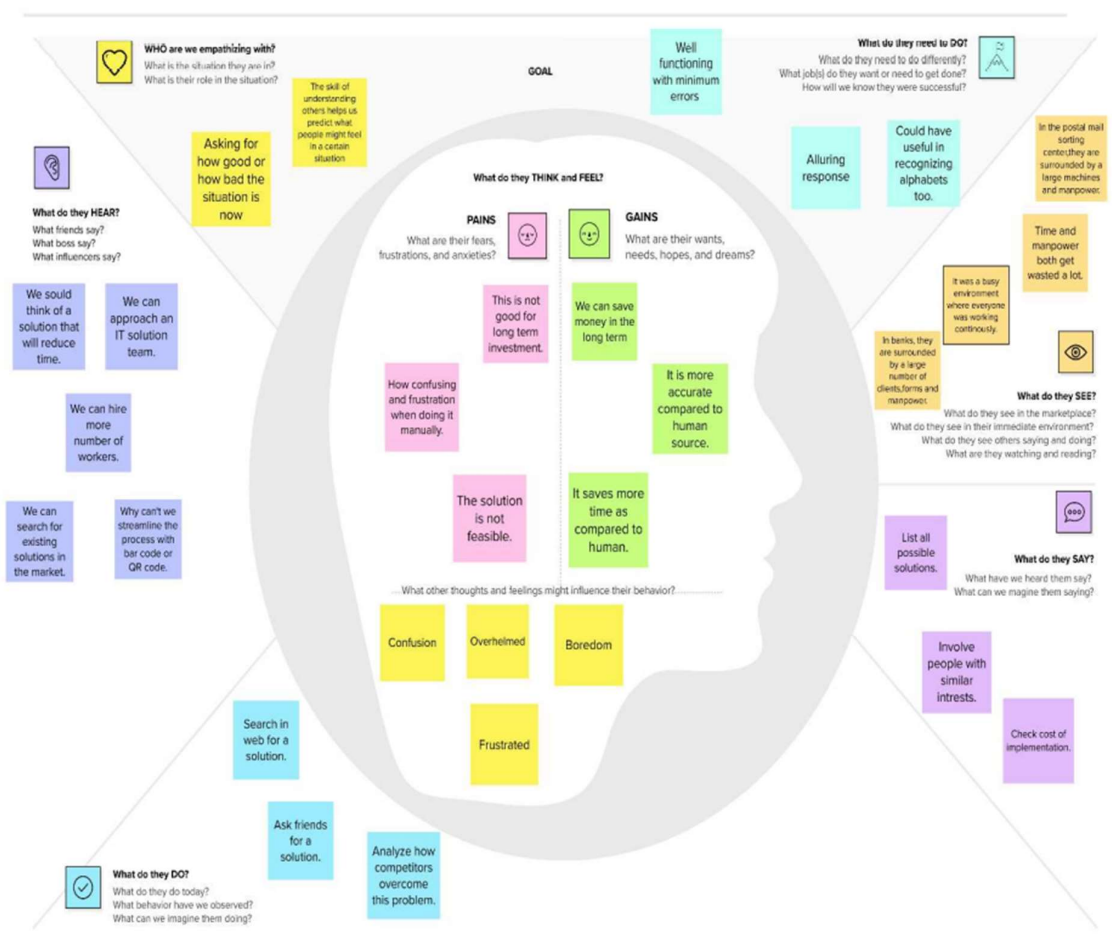
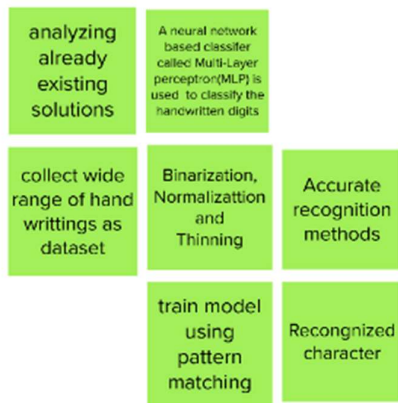


Fig 3.1 Empathy Map

3.2 Ideation & Brainstorming:

Balasubramanian Kalyan



Balaji G



Gokul Aravind V



Ela Barath



Fig 3.2 Brain storm



Fig: 3.3 Group ideas



Fig: 3.4 Prioritize

3.3 Proposed Solution

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	The size, width, orientation, and margin justification of the handwritten numerals are not always the same as those in printed text. Because every person has a unique handwriting.
2.	Idea / Solution description	The MNIST dataset can be used to recognise handwritten digits. The MNIST dataset includes 60000 practise photos of handwritten numbers beginning with 0 to 9 and 10,000 photos for testing.
3.	Novelty / Uniqueness	This system offers authentication so that users can retain their privacy and store data.
4.	Social Impact / Customer Satisfaction	The postal office and courier firms can quickly locate the written digits. There are several uses for handwriting recognition, including: reading postal codes, check amounts from the bank, and forms.
5.	Business Model (Revenue Model)	Utilised in the banking and postal sectors. Numerous handwritten numbers are used in the financial industry. Our method lessens the human errors.
6.	Scalability of the Solution	The handwritten digit is accurately and efficiently recognised.

Table 3.1 Proposed Solution

3.4 Problem Solution Fit

The Problem-Solution Fit simply means that you have found a problem with your customer and that the solution you have realized for it actually solves the customer's problem.

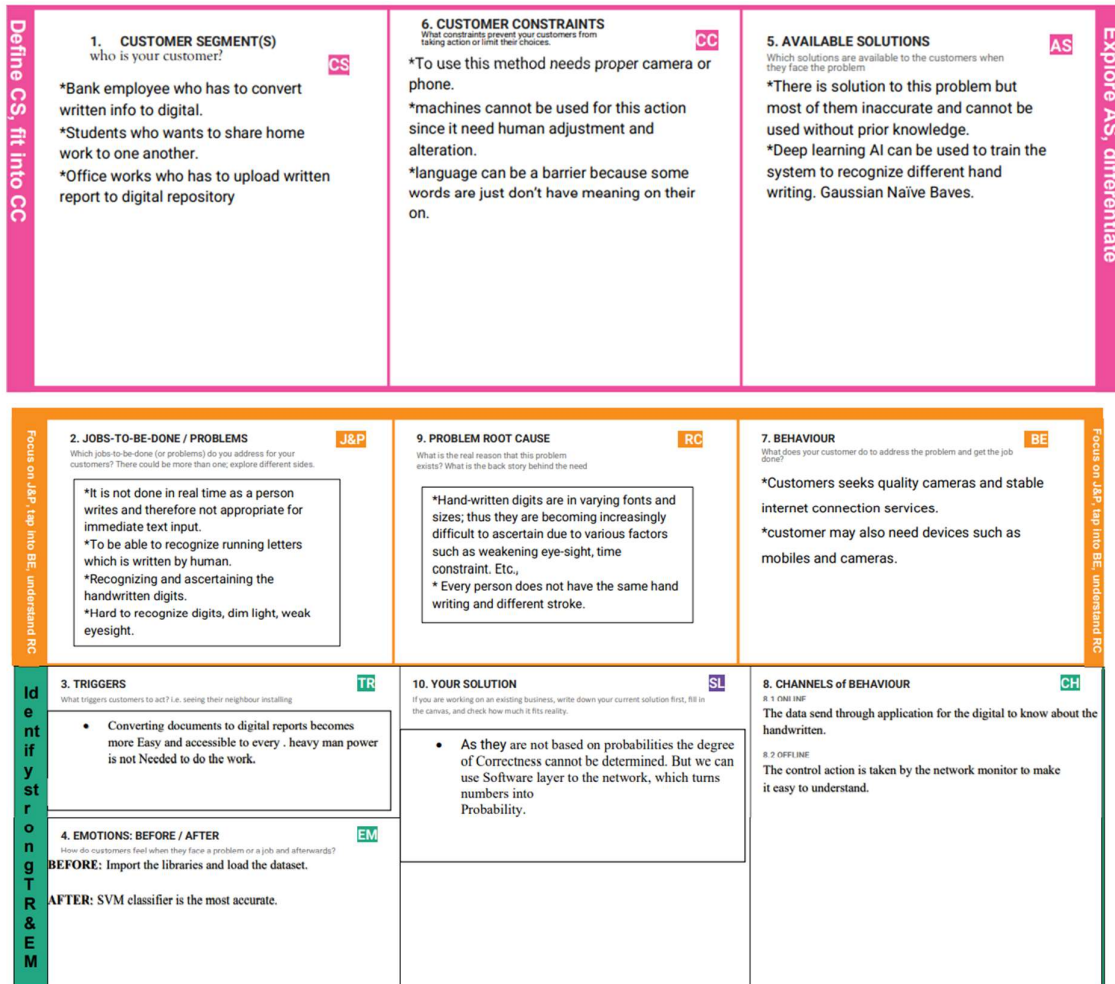


Fig 3.5 Problem Solution Fit

CHAPTER -4

REQUIREMENT ANALYSIS

4.1 Functional Requirements:

The specifics and guidelines that specify how software functions and behaves are known as functional requirements. Depending on the user's industry, functional needs can differ in terms of behaviours, features, and protocols. Software developers can identify the features needed to support a system's seamless operation with the use of functional requirements. The usage of functional criteria allows users to guarantee that software is simple for consumers to use. You can identify a system's features with the aid of functional requirements to determine how you can improve functioning. Functional requirements improve the software's usability. A certain feature that makes a software system easier for people to utilise may be included.

User Input:

One of the most crucial elements of programming principles is user input. Every software should include users in some way, whether it's asking for the name of a character in a game or a password to access a database. For processing functions like training and testing, the model uses images as input. The user's convenience can inform the input. Uploads are used in the model to gather data from the user. Input from users can be recorded and used as user attributes to build conditional flows.

Model:

Models represent an estimation of reality. These are not only a component of AI. Every day, our brains use models. Similar to how we do it in daily life, AI is the process of creating models like this one utilising data. AI models are pieces of software that, in response to inputted data, predict (or estimate) the outcome of an example. To generate a trained model, the MNIST dataset for the novel handwritten digit recognition method should be trained using CNN.

It becomes obvious that the analytics industry is overly focused on predictive modelling tools and methodologies when you consider everything that goes into

an AI model. There are many data science frameworks, but decision automation is seeing relatively little innovation. The process of training a predictive analytic, known as machine learning, as a result becomes separate from the rest of the decision-making process.

Prediction:

AI prediction is used to foresee a variety of events. Prediction models in AI Builder examine trends in the historical data you give. Prediction models pick up the ability to link those patterns to results. Then, we apply AI's power to find previously learnt patterns in fresh data and utilise them to forecast future results. AI prediction makes use of AI to extrapolate future events and values from data. Attention is being drawn to AI prediction in order to both avoid completely personal predictions and pursue predictions that are more accurate. It is necessary to test the trained model using test data provided by MNIST, and the model's accuracy must be greater than 90%.

Evaluation

The development of the discipline depends on the evaluation of artificial intelligence systems and components. Verify the model's output to make sure it is accurate.

4.2 Non-Functional Requirements:

A set of specifications known as non-functional requirements, or NFRs, explain the system's operational capabilities and limitations and make an effort to increase its functionality. These are essentially the specifications that define how well it will function, taking into account factors like speed, security, dependability, and data integrity. The quality attributes of the system are specified by non-functional needs, hence its second term, quality attributes.

Usability:

Because it is too challenging to operate, a system may have adequate capability but poor usability. The system's usability criteria outlines how simple it must be to use. Usability is a non-functional requirement since, at its core, it doesn't identify specific aspects of system functioning; rather, it specifies how the functionality is to be perceived by the user, such as how simple and effective it must be to do user tasks.

The usability requirements must be concrete so that we can check and follow them as we build. They must also be complete in order for us to be certain that we will achieve the usability we desire if we fulfil them. Numbers can be accurately predicted by usability. The model can be utilised for data entry, processing bank checks, etc.

Security:

Security is a non-functional criterion that guarantees that every piece of data inside the system will be safe from malware assaults and unauthorised access. Therefore, the non-functional needs section will introduce particular dangers that will be covered in more detail by the functional requirements. If your security depends on particular standards and encryption techniques, these standards don't actually describe how a system behaves; rather, they provide engineers with implementation guidelines. How well the system and its data are safeguarded from intrusions Because the uploaded image is not kept in a database, security is ensured.

Availability:

Availability refers to the likelihood that a user will be able to use the system at a specific time. You can define it as a percentage of the time the system is available for operation within a given time period, while it can also be represented as an expected percentage of requests that are successful. These three indicators are

linked together, as you can see. And more significantly, if you choose to include them as non-functional requirements for your system, you should address them jointly. Accessible through mobile and web browsers Scalability NFR-6 provides many people with assistance while requiring little time and great accuracy.

Reliability

The reliability of a system or component indicates the likelihood that it will function well for a predetermined amount of time under specific circumstances. The possibility that a product, system, or service will function as intended for a predetermined amount of time or will run faultlessly in a predetermined environment is known as reliability. Example: During a month, the system must operate without error in 95% of use cases. can handle sensitive data without it leaking because it is never kept in a database. improvement in quick prediction performance. For precise prediction, we employ the CNN algorithm.

Maintainability:

Maintainability is the length of time needed to fix a problem, make changes to a solution or its component to improve performance or other characteristics, or adapt to a changing environment. It can be stated as a probability of repair over time, just as reliability. A component has a 75% chance of being fixed within 24 hours, for instance, if its maintainability is 75% during that period. The mean time to restore the system (MTTRS), a metric, is frequently used to assess maintainability.

Compactability:

Compatibility is a further component of portability that describes how two systems can survive in the same environment. Software installed on an operating system, for instance, needs to be compatible with the firewall and antivirus software. In terms of operating systems, hardware, browsers, software, and their versions, portability and compatibility are established.

CHAPTER-5

PROJECT DESIGN

5.1 Data Flow Diagram

A data flow diagram is a graphic or visual representation that describes how data is moved through an organization's processes using a standardised set of symbols and notations. In formal methodologies like the Structured Systems Analysis and Design Method, they are frequently components. Although DFDs may superficially resemble flow charts or the Unified Modelling Language, they are not intended to capture the specifics of programme logic.

The MNIST dataset is pre-processed and separated into two for training and testing in this data flow diagram. The CNN method is then used to evaluate such data. As shown in fig. 5.1 below, the trained model was used to pre-process the evaluated data to produce the results that the user will see.

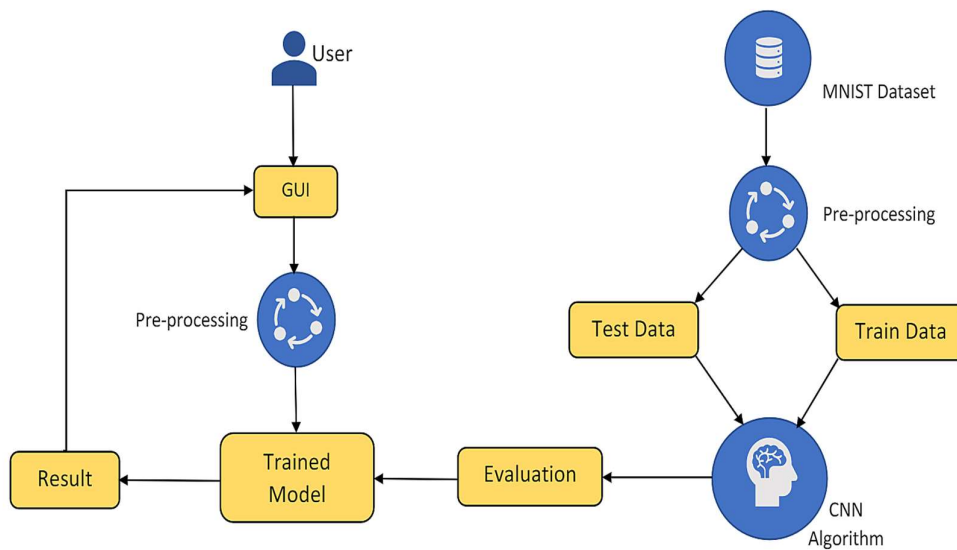


Fig 5.1 Data Flow Diagram

5.2 Solution & Technical Architecture

Solution architecture is the process of creating solutions using established procedures, rules, and best practises with the goal of ensuring that the created solution is compatible with the enterprise architecture in terms of information architecture, system portfolios, integration needs, and other factors.

The image data is pre-processed, trained, and sent to the DL algorithm, which then tests the data. Following that, evaluation requests are sent for both training and tested data. The user then inputs data into the model, which is then forecasted by the model through a user interface (UI), as seen in fig. 5.2 below.

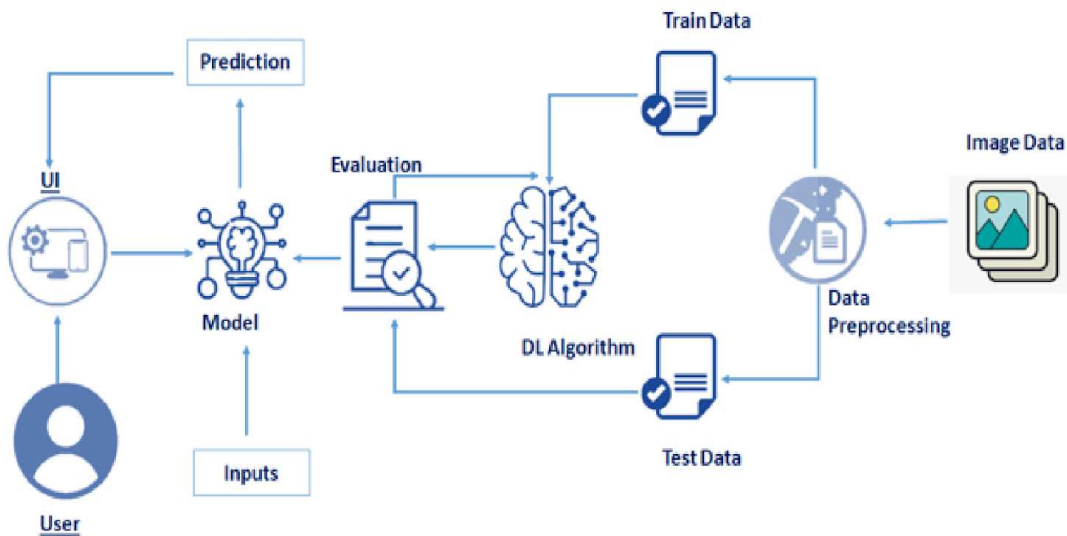


Fig 5.2 Solution Architecture

The infrastructure needed to support applications, operations, and reporting requirements is described in the technical architecture. a framework for creating

an organisation that includes hardware, operating systems, database management systems, and standards for application development.

In this technical architecture, the user supplies both the input and the MNIST data, which are then classified after pre-processing. The model is then trained and evaluated to produce the precise results shown in fig. 5.3 below.

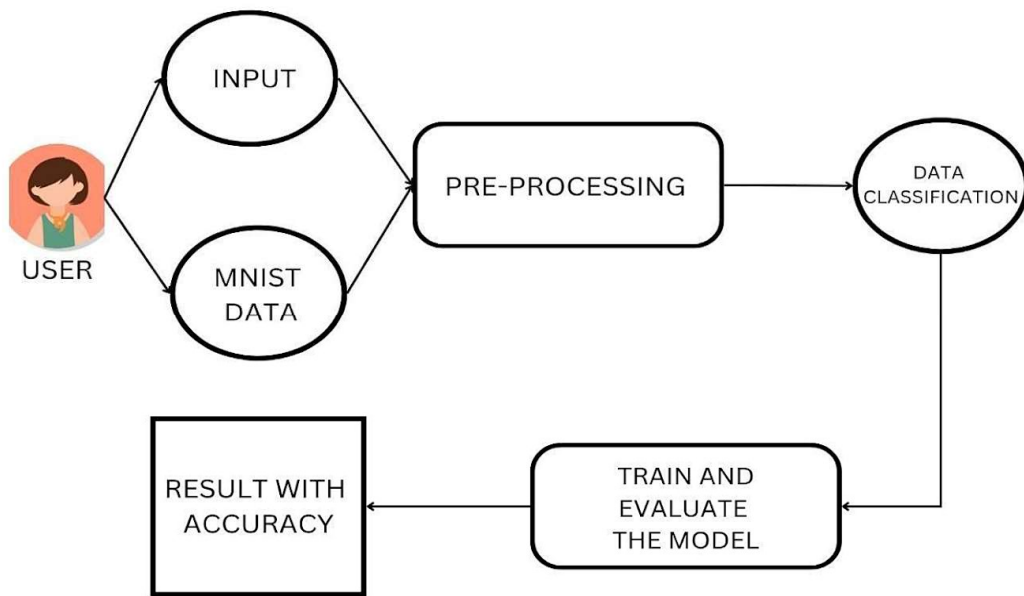


Fig 5.3 Technical Architecture

5.3 User Stories

An informal, general explanation of a software feature written from the viewpoint of the end user is known as a user story. Its objective is to explain how a software feature will benefit the user.

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Web user)	Home	USN-1	In the Home Page, I can view the guidelines of how to use the website	I can view the guidelines	low	Sprint-1
	Dashboard	USN-2	As a user, I can see Home Page & Prediction Page	I can access the dashboard	Low	Sprint-2
	Choose Input	USN-3	In Prediction Page, I can upload an image of handwritten digit for prediction	I can upload my input by browsing the device storage	Medium	Sprint-3
		USN-4	As a user, I can get an accuracy rate with the prediction	I can get different forms of output	High	Sprint-4
	Recognize	USN-5	As a user, I can see that the GUI processing the input using trained model	I can perform handwritten digit prediction	High	Sprint-1
	Prediction	USN-6	As a user, I can get accuracy rate by pressing the predict button	I can get the accuracy of the output	Medium	Sprint-1
Customer (Mobile user)	Home	USN-7	As a user, I can access application in mobile phone	I can access the dashboard with mobile	Medium	Sprint-1
	Recognize	USN-8	I can upload input and retrieve output with accuracy by using the mobile	I can upload input image and get output with a mobile device	High	Sprint-2

Table 5.1 User Stories

CHAPTER-6

PROJECT PLANNING & SCHEDULING

6.1 Sprint Planning & Estimation

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Data Collection & pre processing	USN-1	I, as a user, am able to upload any photograph that has undergone pre-processing.	12	High	Balaji G
Sprint-1		USN-2	I can post the image in any resolution as a user.	8	Medium	
Sprint-2	Building the model	USN-3	I can utilise the machine learning model, which has a high level of recognition accuracy for handwritten digits.	8	High	Ela Barath
Sprint-2		USN-4	I, as a user, can give the model an image of a handwritten digit so it can identify it.	6	Medium	
Sprint-2		USN-5	I can distinguish the digit as a user with the highest level of accuracy.	7	Medium	
Sprint-3	Building User Interface Application	USN-6	I, as a user, will upload the image of the handwritten digits to the programme using the upload feature offered by the UI.	10	High	Balasubramanian Kalyan
Sprint-3		USN-7	I may view the anticipated and recognised digits in the programme as a user.	10	High	
Sprint-4	Train and deployment of model in IBM Cloud	USN-8	I can utilise the online application as a user from anywhere and have access to it.	20	High	Gokul Aravind

Table 6.1 Sprint Planning & Estimation

6.2 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	20	6 Days	22 Oct 2022	28 Oct 2022	20	28 Oct 2022
Sprint-2	20	6 Days	29 Nov 2022	04 Nov 2022	20	03 Nov 2022
Sprint-3	20	6 Days	05 Nov 2022	10 Nov 2022	20	09 Nov 2022
Sprint-4	20	6 Days	11 Nov 2022	16 Nov 2022	20	16 Nov 2022

Table 6.2 Sprint Delivery Schedule

6.3 Reports from JIRA

Velocity:

Consider a sprint that lasts six days and the team's velocity is 20. (points per sprint). Let's determine the group's average velocity (AV) for each iteration (story points per day)

$$\begin{aligned} AV &= \text{sprint duration} / \text{Velocity} \\ &= 20 / 6 \\ &= 3.33 \end{aligned}$$

Burndown Chart:

A burn down chart plots the amount of work remaining to perform against the amount of time. In agile software development approaches like Scrum, it is frequently employed. Burn down charts, however, can be used for any project that makes observable progress over time.

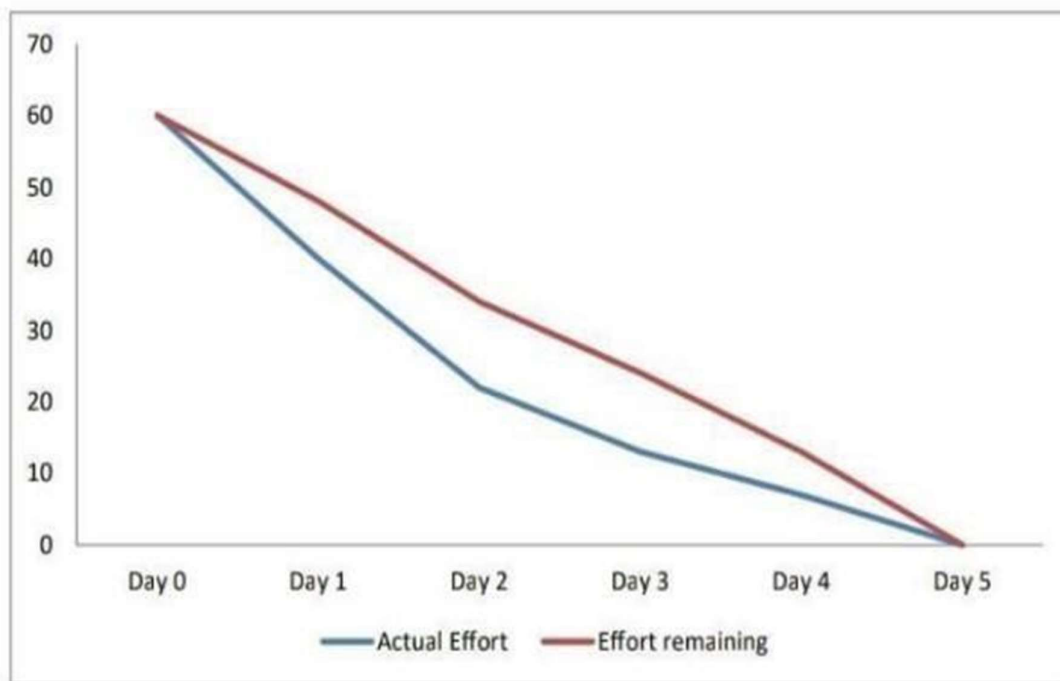


Fig 6.3: Burndown Chart

CHAPTER 7

CODING & SOLUTIONING

7.1 CNN Model

The Convolutional Neural Network is used in the machine learning model for handwritten digit recognition (CNN). Convolution entails looking at every match a feature in an image might have. This convolutional procedure is referred as as filtering. It includes these things

1. Line up the features and image patch
2. Multiply each image pixel by corresponding feature pixel
3. Add them up and divide by total number of pixels in the feature

The 28x28 images in the MNIST datasets are filtered using 32 features to create a stack of 32 filtered images with a 26x26 dimension.

After the initial Conv2D layer comes the pooling layer. The dimensions of the filtered image are reduced while the details are kept. The steps listed below are used to do this.

1. Choosing a window of size 2 or 3
2. Walk the window through the filtered image
3. Pick the maximum value from each window

After the application of pooling, the dimension of the filtered images becomes half

```
model.add(Conv2D(32, (3,3),input_shape=(28, 28, 1), activation = 'relu'))  
model.add(MaxPooling2D(pool_size=(2, 2)))
```

Each layer is followed by normalisation, where all negative values are converted to zero. This stops the computer from reading undesirable properties.

The data is flattened and thick layers are added before the neural link reaches the output layer. The output of this dense layer is an array with a length of 10. To

increase accuracy, the dense layer may also be stacked recursively. The prediction is then generated as a length 10-element array representing 0–9. The predicted digit will be determined by the index with the highest value.

To create hidden layers, the aforementioned layers can be continually piled in a particular order. The accuracy of the model improves as the number of hidden layers increases.

```
model.add(Conv2D(64,(3,3), activation = 'relu'))  
model.add(Conv2D(64,(3,3), activation = 'relu'))
```

```
model.add(Conv2D(32,(3,3), activation = 'relu'))  
model.add(MaxPooling2D(pool_size=(2, 2)))
```

```
model.add(Flatten())  
model.add(Dense(10, activation = 'softmax'))
```

The CNN model utilised in the current project has the following model summary:

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 26, 26, 32)	320
max_pooling2d (MaxPooling2D)	(None, 13, 13, 32)	0
conv2d_1 (Conv2D)	(None, 11, 11, 64)	18496
conv2d_2 (Conv2D)	(None, 9, 9, 64)	36928
conv2d_3 (Conv2D)	(None, 7, 7, 32)	18464
max_pooling2d_1 (MaxPooling2D)	(None, 3, 3, 32)	0
flatten (Flatten)	(None, 288)	0
dense (Dense)	(None, 10)	2890
Total params: 77,098		
Trainable params: 77,098		
Non-trainable params: 0		

Next, this model is trained using the 60,000-dataset provided by MNIST with an epoch of 5 and a batch size of 120.

Adam is an optimizer that is used to build and train the model.

Compiling the Model

```
model.compile(loss=keras.losses.categorical_crossentropy, optimizer=keras.optimizers.Adam(), metrics=['accuracy'])
```

Train the Model

```
model.fit(x_train, y_train, validation_data = (x_test, y_test), epochs = 5, batch_size = 120)
```

```
Epoch 1/5
500/500 [=====] - 52s 95ms/step - loss: 0.5196 - accuracy: 0.8995 - val_loss: 0.1107 - val_accuracy: 0.9653
Epoch 2/5
500/500 [=====] - 47s 95ms/step - loss: 0.0870 - accuracy: 0.9734 - val_loss: 0.0671 - val_accuracy: 0.9792
Epoch 3/5
500/500 [=====] - 48s 95ms/step - loss: 0.0584 - accuracy: 0.9822 - val_loss: 0.0560 - val_accuracy: 0.9824
Epoch 4/5
500/500 [=====] - 47s 94ms/step - loss: 0.0411 - accuracy: 0.9872 - val_loss: 0.0526 - val_accuracy: 0.9844
Epoch 5/5
500/500 [=====] - 47s 94ms/step - loss: 0.0333 - accuracy: 0.9889 - val_loss: 0.0468 - val_accuracy: 0.9862
```

When the trained model is put to the test, it correctly predicts the digits 98.6% of the time.

```
metrics = model.evaluate(x_test, y_test, verbose=0)
print('METRICS\n Loss: {:.3f}\n Accuracy: {:.3f}'.format(metrics[0],metrics[1]))
```

```
METRICS
Loss: 0.047
Accuracy: 0.986
```

7.2 The User Interface

Using HTML, CSS, and JavaScript, the UI is created. Flask, a Python framework for building backend applications, manages the application's backend. The application is two pages long,

1. main.html
2. index6.html

The home page, main.html, provides a basic overview of handwritten digit recognition.

The users might enter an image including a handwritten digit on the prediction page, index6.html. This page shows the user the anticipated result.

Home page (main.html)

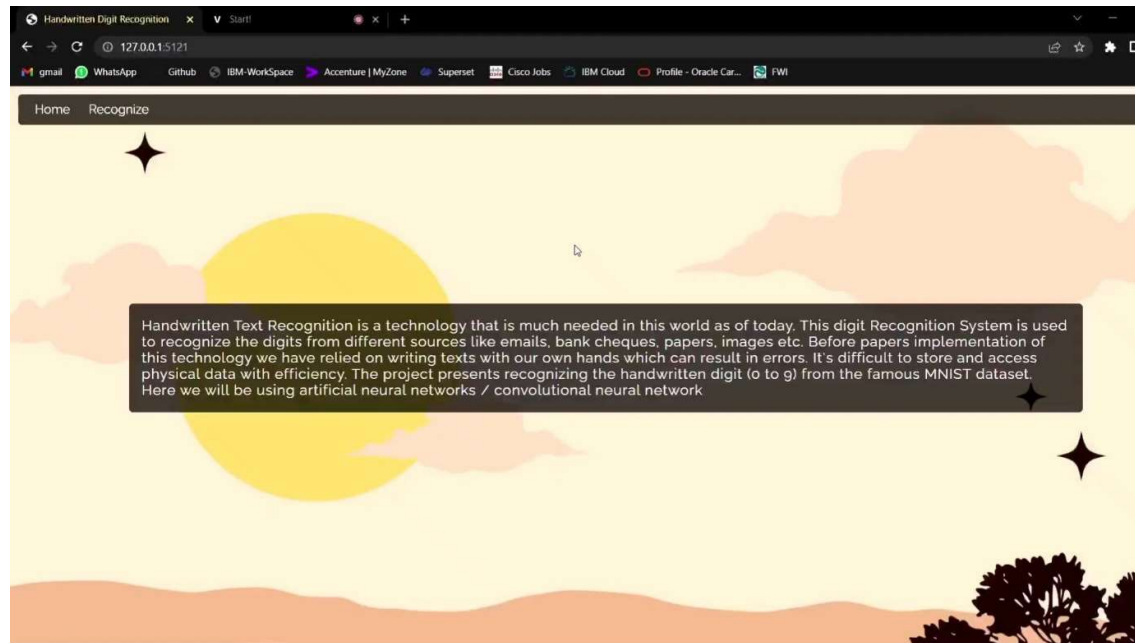


Fig 7.1 Home page(main.html)

Index6.html

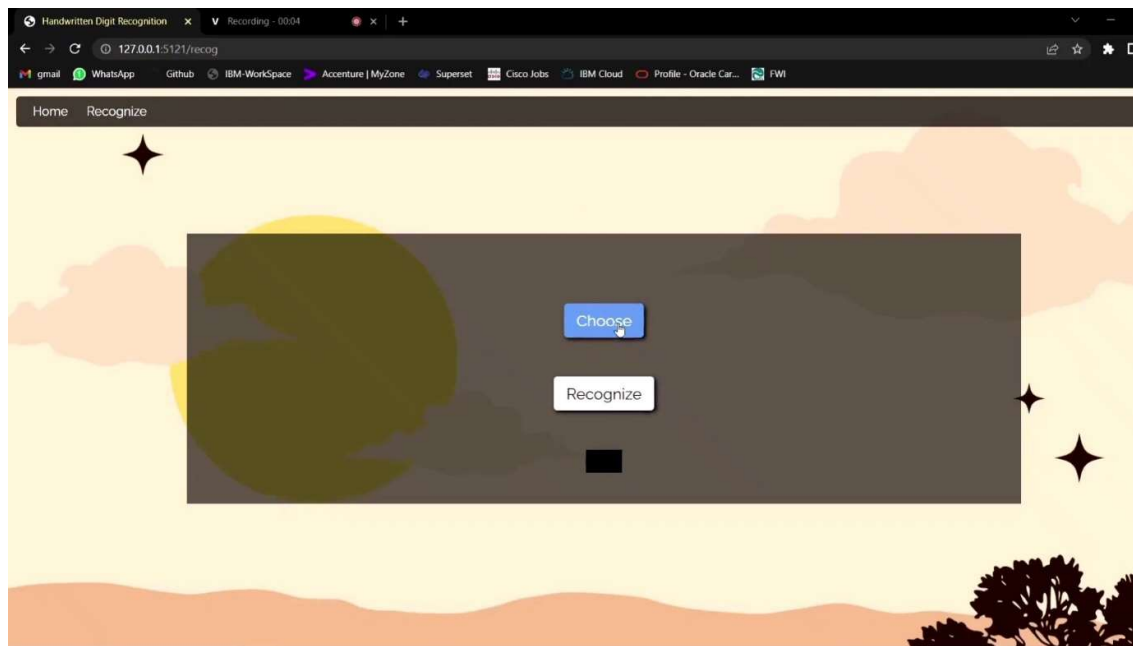


Fig 7.2 Index6.html

Outputs

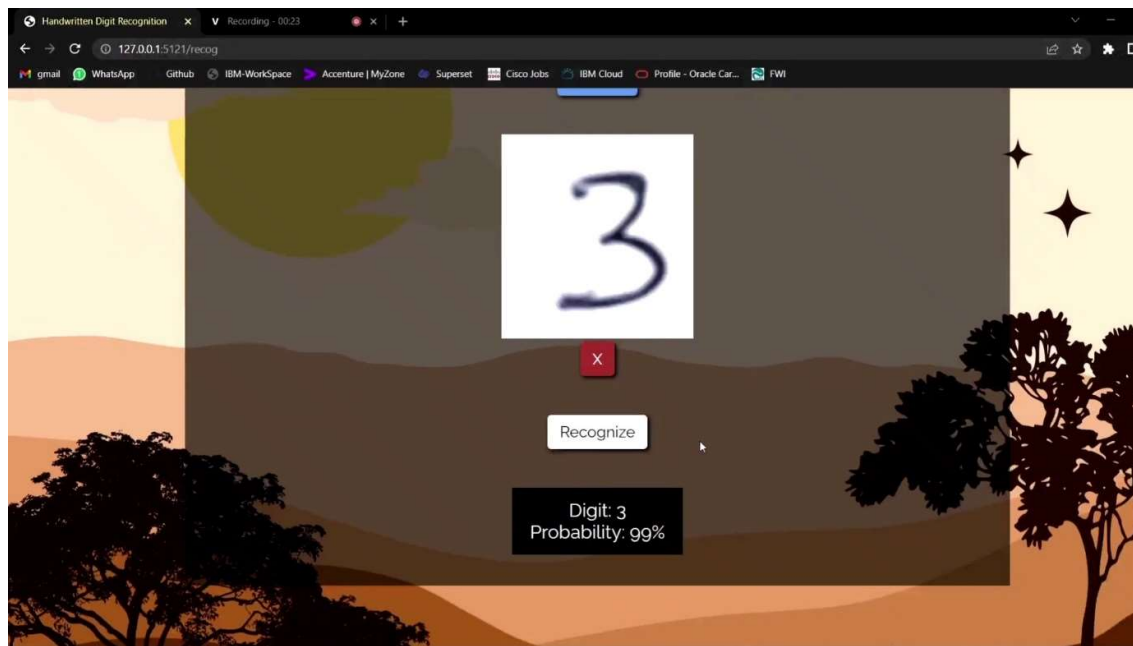


Fig 7.3 Output 1

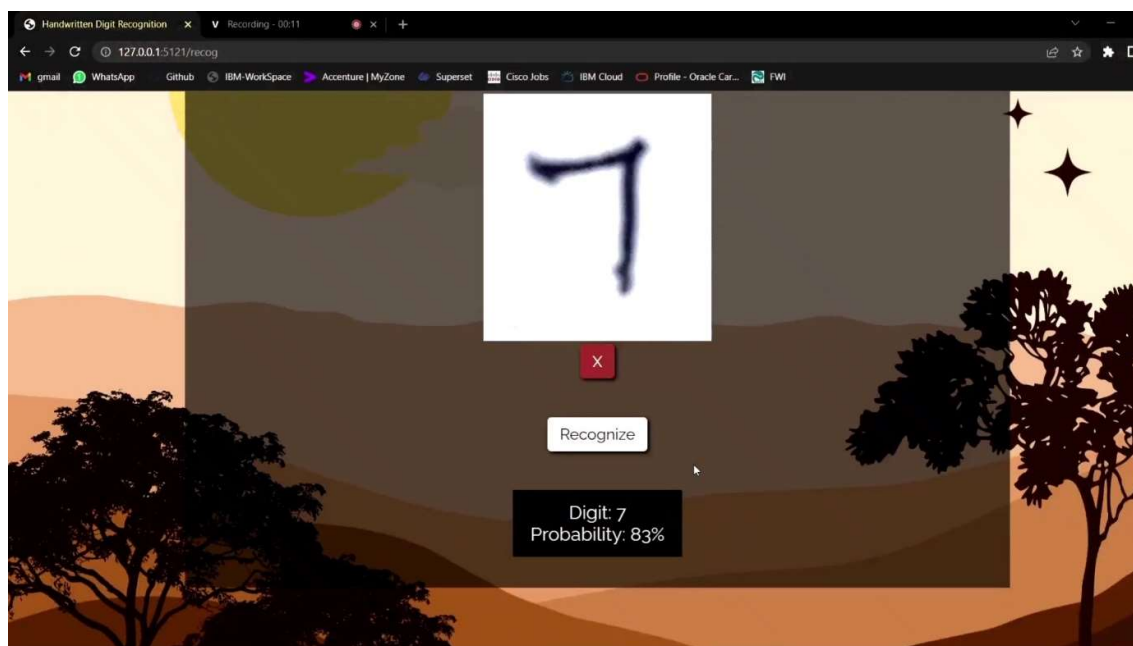


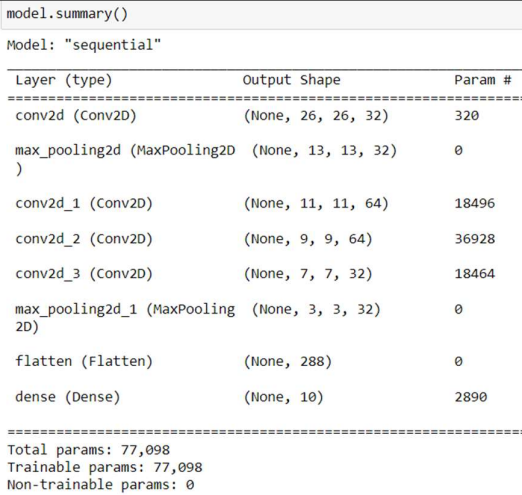
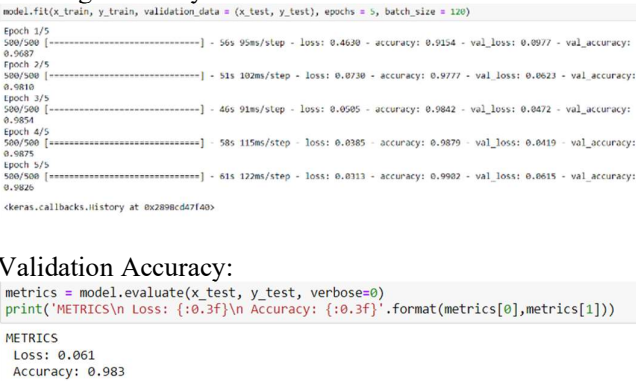
Fig 7.4 Output 2

CHAPTER-8

RESULT

8.1 Performance Metrics

One of the crucial phases in creating a successful machine learning model is evaluating its performance. The performance measures make it clear to us how successfully our model handled the supplied data. By adjusting the hyper-parameters, we can make the model perform better. Performance metrics assist measure how well a machine learning (ML) model generalises on new or previously unexplored data. Accuracy is the primary criterion that should be taken into account when developing the model for handwritten digit recognition. The model created for this suggested system has a 98.6% average accuracy in predicting the digit.

S.No.	Parameter	Values	Screenshot
1.	Model Summary	Layer 1: Conv2D (26,26,32) Layer 2: Pooling (13,13,32) Layer 3: Conv2D (11,11,64) Layer 4: Conv2D (9,9,64) Layer 5: Conv2D (7,7,32) Layer 6: Conv2D (3,3,32) Layer 7: Flatten (288) Layer 8: Dense (10)	 <pre> model.summary() Model: "sequential" Layer (type) Output Shape Param # ===== conv2d (Conv2D) (None, 26, 26, 32) 320 max_pooling2d (MaxPooling2D) (None, 13, 13, 32) 0 conv2d_1 (Conv2D) (None, 11, 11, 64) 18496 conv2d_2 (Conv2D) (None, 9, 9, 64) 36928 conv2d_3 (Conv2D) (None, 7, 7, 32) 18464 max_pooling2d_1 (MaxPooling2D) (None, 3, 3, 32) 0 flatten (Flatten) (None, 288) 0 dense (Dense) (None, 10) 2890 ===== Total params: 77,098 Trainable params: 77,098 Non-trainable params: 0 </pre>
2.	Accuracy	Training Accuracy – 98.26 % Validation Accuracy – 98.3 5	 <pre> model.fit(x_train, y_train, validation_data = (x_test, y_test), epochs = 5, batch_size = 120) Epoch 1/5 500/500 [-----] - 56s 90ms/step - loss: 0.4630 - accuracy: 0.9154 - val_loss: 0.0977 - val_accuracy: 0.9687 Epoch 2/5 500/500 [-----] - 51s 102ms/step - loss: 0.0730 - accuracy: 0.9777 - val_loss: 0.0623 - val_accuracy: 0.9810 Epoch 3/5 500/500 [-----] - 46s 91ms/step - loss: 0.0505 - accuracy: 0.9842 - val_loss: 0.0472 - val_accuracy: 0.9854 Epoch 4/5 500/500 [-----] - 58s 115ms/step - loss: 0.0385 - accuracy: 0.9870 - val_loss: 0.0419 - val_accuracy: 0.9875 Epoch 5/5 500/500 [-----] - 61s 122ms/step - loss: 0.0313 - accuracy: 0.9902 - val_loss: 0.0615 - val_accuracy: 0.9826 <keras.callbacks.History at 0x2898cd67f40> Validation Accuracy: metrics = model.evaluate(x_test, y_test, verbose=0) print('METRICS\n Loss: {:.3f}\n Accuracy: {:.3f}'.format(metrics[0],metrics[1])) METRICS Loss: 0.061 Accuracy: 0.983 </pre>

CHAPTER-9

ADVANTAGES AND DISADVANTAGES

Advantages

The system generates a detailed description of the instantiation parameters, which can reveal details like the writing style, in addition to a classification of the digit. The generative models are capable of segmentation driven by recognition. As a result of the method's few parameters, training is simple and quick. Unlike many other recognition techniques, it also does not require pre-normalization of the input images and can tolerate arbitrary scaling, translations, and a little amount of image rotation. This approach focuses on which classifier performs better by increasing the accuracy of classification models by more than 99% in comparison to previous research approaches. A CNN model can predict with an accuracy of 98.3% or better. Without any human oversight, it automatically recognises the crucial characteristics.

Disadvantages

Despite having a 99.3% overall accuracy rate, the model has a few flaws both as a whole and in each segment. Only single digits can be predicted using handwritten digits; if the number has multiple digits, each digit must be provided as a distinct image, which requires extra effort and repeating of the same operations. Additionally, a few sets of digits have low precision. The model's prediction accuracy is poor since the written forms of the numbers three (3), six (6), and eight (8) are identical. Due to their similar appearance when written by hand, the digits seven (7) and one (1) are frequently confused. Seven (7) is sometimes anticipated incorrectly as one (1) or vice versa.

CHAPTER-10

CONCLUSION

Conclusion

The handwritten digits vary from writer to writer in terms of size, width, orientation, and justification to the margins. This project uses deep learning techniques to implement handwritten digit recognition. On the MNIST dataset, the most popular machine learning algorithm, CNN, has been trained and tested. The description of the solution states that handwritten digits can be recognised using the MNIST dataset. This system's novelty or uniqueness offers authentication to protect users' privacy and allows users to store data. With the help of this deep learning method, a great level of accuracy can be attained. The recognition rate for this model is 98.6% accurate.

For the goal of introducing and exhibiting neural networks to the general audience, the handwritten digit recognition using convolutional neural networks has proven to be of a reasonably good efficiency and usefulness. Customer satisfaction or the Social Impact The postal office and courier firms can quickly locate the written digits. Reading postal addresses, bank check amounts, and forms are just a few of the many uses for handwriting recognition. Banking and the postal sectors both use business models (revenue models). Numerous handwritten numbers are used in the financial industry. Our system lessens human errors. The solution's scalability and high level of accuracy in identifying handwritten numbers.

CHAPTER-11

FUTURE SCOPE

Future work:

The accuracy achieved is 98.6% and the model was trained using the CNN model. In the future, the model will be trained with the goal of achieving the maximum accuracy possible for both the overall model and each individual digit. Additionally, the model must be trained to predict any set of numbers in a single image, regardless of the amount of digits. Future research may take into account utilising the convolution network architecture, which performed best on the MNIST database and is used to create the suggested recognition system on handwritten numbers. Such systems can be created for the recognition of handwritten characters, objects, images, handwriting, and text languages. Future research may also take into account the hardware implementation of an online digit recognition system with increased performance and efficiency as well as real-time results from test scenarios.

CHAPTER-12

APPENDIX

Source Code:

Handwritten Digit Recognition.ipynb

Understanding Data

```
import numpy as np
import tensorflow
import keras
from tensorflow.keras.datasets import mnist
from tensorflow.keras.models import Sequential
from tensorflow.keras import layers
from tensorflow.keras.layers import Dense, Flatten
from tensorflow.keras.layers import Conv2D, MaxPooling2D
from keras.optimizers import Adam
from keras.utils import np_utils
(x_train, y_train), (x_test, y_test) = mnist.load_data()
import matplotlib.pyplot as plt
plt.imshow(x_train[0])
```

Reshaping the data

```
x_train = x_train.reshape(60000, 28, 28, 1).astype('float32')
x_test = x_test.reshape(10000, 28, 28, 1).astype('float32')
```

One Hot Encoding

```
number_of_classes = 10
y_train = np_utils.to_categorical(y_train, number_of_classes)
y_test = np_utils.to_categorical(y_test, number_of_classes)
```

#Model Building

adding cnn layer

```

model = Sequential()
model.add(Conv2D(32, (3,3),input_shape=(28, 28, 1), activation = 'relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(64,(3,3), activation = 'relu'))
model.add(Conv2D(64,(3,3), activation = 'relu'))
model.add(Conv2D(32,(3,3), activation = 'relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())
model.add(Dense(10, activation = 'softmax'))

```

#Compiling and Training

```

model.compile(loss=keras.losses.categorical_crossentropy,optimizer=keras.optimizers.Adam(), metrics=['accuracy'])
model.fit(x_train, y_train, validation_data = (x_test, y_test), epochs = 5,
batch_size = 120)

```

#Testing the Model

```

prediction = model.predict(x_test[:4])
np.argmax(prediction, axis=1)
metrics = model.evaluate(x_test, y_test, verbose=0)
print('METRICS\n Loss: {:.3f}\n Accuracy:
:0.3f}'.format(metrics[0],metrics[1]))

```

#Saving the Model

```

model.save('models/mnistCNN.h5')

```

app.py

```

# Importing the packages
from flask import Flask, jsonify, render_template, request

```

```

from PIL import Image
import numpy as np
import tensorflow as tf
from tensorflow.keras.models import load_model

# Initializing the app
app = Flask(__name__)

def img_preprocess(image):
    image = image.resize((28,28))
    im2arr = np.array(image)
    im2arr = im2arr.reshape(1,28,28,1)
    if (im2arr[0][0][0][0]>=170):
        im2arr = im2arr - 255
    return im2arr

# Routing
@app.route('/')
def home():
    return render_template('main.html')

@app.route('/recog',methods=['GET', 'POST'])
def get_recog():
    if request.method == 'GET':
        return render_template('index6.html')
    elif request.method == 'POST':
        try:
            img = Image.open(request.files['img'].stream).convert("L")

```

```

        img = img_preprocess(img)
        model = load_model("./models/IBM_mnistCNN.h5")
        pred = model.predict(img)
        val = str(np.argmax(pred, axis=1)[0])
        data = {'num': val, 'prob': int(pred[0][int(val)]*100)}
        print(pred)
    except Exception as e:
        print(e)
        val = 'Invalid Input'
        return jsonify(data)
    else:
        return 'Unknown Request'

# Main
if __name__ == '__main__':
    app.jinja_env.auto_reload = True
    app.config['TEMPLATES_AUTO_RELOAD'] = True
    app.run(port=5121, debug=True)

```

main.html

```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />

```

```

<link rel="stylesheet" href="../static/css/style.css" />
<title>Handwritten Digit Recognition</title>
</head>
<style>
</style>
<body>
  <nav>
    <ul>
      <li><a href="/">Home</a></li>
      <li><a href="/recog">Recognize</a></li>
    </ul>
  </nav>
  <section class="des">
    Handwritten Text Recognition is a technology that is much needed in this
    world as of today. This digit Recognition System is used to recognize the
    digits from different sources like emails, bank cheques, papers, images
    etc. Before papers implementation of this technology we have relied on
    writing texts with our own hands which can result in errors. It's
    difficult to store and access physical data with efficiency. The project
    presents recognizing the handwritten digit (0 to 9) from the famous MNIST
    dataset. Here we will be using artificial neural networks / convolutional
    neural network
  </section>
</body>
</html>

```

index6.html

```

<!DOCTYPE html>
<html lang="en">

```



```

<head>
  <meta charset="UTF-8" />
  <meta http-equiv="X-UA-Compatible" content="IE=edge" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <link rel="stylesheet" href="../static/css/style.css" />
  <title>Handwritten Digit Recognition</title>
</head>
<body>
  <nav>
    <ul>
      <li><a href="/">Home</a></li>
      <li><a href="/recog">Recognize</a></li>
    </ul>
  </nav>
  <section class="content">
    <center>
      <form
        action="/recog"
        method="POST"
        id="img_form"
        enctype="multipart/form-data"
      >
        <input
          type="file"
          name="img"
          onchange="clean_res()"
          id="in_image"
          class="input"
        />

```

```

        <label class="inline" for="in_image"> <span
class="btn">Choose</span></label>
        <section id="preview">THIS IS THE PREVIEW</section>
        <input type="button" class="btn hide" id="remove" value="X"
title="Remove Image"/>
        <input type="submit" id="sub" class="btn" value="Recognize" />
    </form>
    <section id="prediction">
        <div class="res" id="res_box"></div>
    </section>
</center>
</section>
</body>
<script
    language="JavaScript"
    type="text/javascript"
    src="//ajax.googleapis.com/ajax/libs/jquery/1.8.3/jquery.min.js"
></script>
<script
    language="javascript"
    type="text/javascript"
    src="../static/js/fn.js"
></script>
</html>

```

fn.js

```

$(function () {
    $("#in_image").change(function () {
        $("#preview").html("");
    });
});

```

```

var regex = /^[a-zA-Z0-9\s_\.\-:]+(.jpg|.jpeg|.gif|.png|.bmp)$/;
if (regex.test($(this).val().toLowerCase())) {
    if ($.browser.msie && parseFloat(jQuery.browser.version) <= 9.0) {
        $("#preview").show();
        $("#preview")[0].filters.item(
            "DXImageTransform.Microsoft.AlphaImageLoader"
        ).src = $(this).val();
    } else {
        if (typeof FileReader != "undefined") {
            $("#preview").show();
            $("#preview").append("<img />");
            var reader = new FileReader();
            reader.onload = function (e) {
                $("#preview img").attr("src", e.target.result);
                $("#remove").show();
            };
            reader.readAsDataURL($(this)[0].files[0]);
        } else {
            alert("This browser does not support FileReader.");
        }
    }
} else {
    alert("Please upload a valid image file.");
}
});
});

$("#remove").on("click", () => {
    document.getElementById("img_form").reset();

```

```
$("#preview").hide();
$("#remove").hide();
clean_res();
});
```

```
$("#img_form").on("submit", function (ev) {
    document.getElementById("sub").value = "Please Wait....";
    ev.preventDefault();
    var formData = new FormData(this);
    $.ajax({
        url: "/recog",
        type: "POST",
        data: formData,
        success: (val) => build_res(val),
        cache: false,
        contentType: false,
        processData: false,
    });
});
```

```
function clean_res() {
    document.getElementById("res_box").innerHTML = "";
}
```

```
function build_res(val) {
    var resBox = document.getElementById("res_box");

    if (val != "Invalid Input") {
        var span = document.createElement("SPAN");
```

```
span.setAttribute("id", "result");
span.innerHTML = "Digit: "+val ['num']+"<br>Probability:
"+val['prob']+"%";
resBox.appendChild(span);
} else {
resBox.innerHTML = val;
}
document.getElementById("sub").value = "Recognize";
}
```

Github: <https://github.com/IBM-EPBL/IBM-Project-11462-1659329967>