PROJECT REPORT

A NOVEL METHOD FOR HANDWRITTEN DIGIT RECOGNITION

submitted by

PNT2022TMID02181

Kumaran BB	-	2116190701105
Kesheka E	-	2116190701093
Hirtih Kumar CV	-	2116190701068
Jaya Krishna Raj	-	2116190701074

TABLE OF CONTENTS

1. INTRODUCTION

Project Overview

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 recognize and comprehend handwritten numbers or letters. Every aspect of life is being digitized to lessen the need for human labor 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 utilized 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 examines this image and the detected result is returned on to UI.

Purpose

In today's society, character recognition is becoming increasingly vital. It facilitates human work and aids with the resolution of more difficult issues. One illustration is handwritten character recognition, which is extensively used worldwide. This technique was created to recognise zip codes or postal codes for use in mail sorting. This can aid people in the difficult-to-read postal code mail sorting process. Researchers have been working on handwriting recognition for more than thirty years. The number of firms participating in handwriting recognition research has steadily expanded over the last several years. Handwriting processing has advanced due to a mix of factors such as improved recognition rates and the usage of complicated systems. We can enter our handwriting into some handwriting recognition systems. Either using a mouse or a third-party drawing tablet, you can accomplish this. We have the option of typing the input or leaving it as an "ink object" with our own handwriting. Additionally, we can manually type the content into any Microsoft Office software file that we want the system to identify. Typing 1s and 0s will allow us to do this. As a Boolean variable, this operates

2. LITERATURE SURVEY

Existing problem

While attempting to address the handwritten recognition problem, many difficulties are encountered. The size, thickness, orientation, and placement of the handwritten digits in relation to the margins are not always the same. The objective is to create a pattern classification approach to identify the user-provided handwritten numbers. The resemblance between the digits like 1 and 7, 5 same digits in many various ways, was the general issue that is faced in this digit categorization challenge. as well as 6, 3 and 8, 8, 8, etc. Last but not least, the distinctiveness and variation of each person's handwriting has an impact on how the numerals form and appear.

References

Paper 1: Novel Deep Neural Network Model for Handwritten Digit Classification and Recognition

Year: 2021

Authors: Ayush Kumar Agrawal and Vineet Kumar Awasthi

An artificial neural network has one hidden layer between the input and output layers, whereas a deep neural network has numerous hidden layers with input and output layers. Deep neural networks use several hidden layers to increase model performance and achieve higher accuracy compared to accuracy of machine learning models. Most researchers do their research in the area of pattern recognition. In the field of pattern recognition, there are many patterns that can be used, including handwritten numbers, characters, pictures, faces, sounds, and speech. This study focuses on the classification and recognition of handwritten digits.1000 were utilized as test samples and 1000 were training samples.10000 picture samples make up the USPS dataset, of which 7291 serve as training samples and 2007 serve as testing samples. We've used the proposed deep neural network technique in this paper to classify and identify data from the ARDIS and USPS datasets. The suggested model consists of six layers with softmax and relu activation functions. After model implementation, accuracy for ARDIS samples reached 98.70% testing and 99.76% training, which is greater than accuracy from prior research. Additionally, using the USPS samples dataset, 98.22% training accuracy and 93.01% testing accuracy were attained. When compared to earlier methodologies, the data show that deep neural networks perform incredibly well.

Paper 2: A Novel Handwritten Digit Classification System Based on Convolutional Neural Network Approach

Year: 2021

Authors: Ali Abdullah Yahya, Jieqing Tan, Min Hu

There have been a tonne of CNN classification algorithms put forth in the literature. However, these algorithms do not take into account the proper filter size selection, data preparation, dataset restrictions, or noise. As a result, few algorithms have been able to significantly increase classification accuracy. The paper makes the following contributions to solve these methods' drawbacks: First, the size

of the effective receptive field (ERF) is determined after taking domain knowledge into account. They choose a typical filter size with the aid of the ERF calculation, improving the classification accuracy of our CNN. Second, excessive data produces inaccurate results, which has a detrimental impact on classification accuracy. Before carrying out the data classification task, data preparation is conducted to ensure that the dataset is devoid of any redundant or irrelevant variables to the goal variable. Thirdly, data augmentation has been suggested as a way to reduce training and validation errors and prevent dataset limitations. Fourthly, the paper suggests adding an additive white Gaussian noise with a threshold of 0.5 to the MNIST dataset in order to imitate the natural factors that can affect image quality in the real world. With a recognition accuracy of 99.98% and 99.40% with 50% noise, our CNN algorithm achieves state-of-the-art performance in handwritten digit recognition.

Paper 3: Handwritten Character Recognition using Neural Network and TensorFlow

Year : 2019

Authors : Megha Agarwal, Shalika, Vinam Tomar, Priyanka Gupta

The offline handwritten character recognition in this study will be carried out using Tensorflow and a convolutional neural network. a process known as using SoftMax Regression, one may assign probabilities to one of the many characters in the handwritten text that offers the range of values from 0 to 1, summed to 1. The objective is to create software that is extremely accurate and that has a minimum level of spatial and temporal complexity. It was determined that strategies for feature extraction like diagonal and direction are significantly better at producing high accuracy. Outcomes in comparison to other conventional vertical and horizontal techniques moreover use the best Neural network tried layers provides the benefit of a higher accurate outcome by having a high noise tolerance. The feed forward model in neural networks is the back-propagation algorithm that was primarily used to classify the characters, recognise them, and receive training continually more. In addition to these, normalizing along with feature extraction, the results were better and more effective. Character recognition is the outcome of accuracy. The paper will describe the best approach to get more than 90% accuracy in the field of Handwritten Character Recognition (HCR).

Paper 4: Improved Handwritten Digit Recognition Using Convolutional Neural Networks (CNN)

Year : 2020

Authors: Savita Ahlawat , Amit Choudhary , Anand Nayyar , Saurabh Singh and Byungun Yoon

Customized features and a vast quantity of past knowledge have been used in traditional handwriting recognition systems. It is difficult to train an optical character recognition (OCR) system based on these conditions. Deep learning approaches have enabled significant performance in the field of handwriting recognition research in recent years. Nonetheless, the increasing increase in the amount of handwritten data, along with the availability of vast computing capacity, necessitates improvements in recognition accuracy and warrants additional exploration. Convolutional neural networks (CNNs) are extremely excellent in perceiving the structure of handwritten characters/words in ways that aid in the automatic extraction of distinguishing features, making CNN the best solution for solving handwriting recognition challenges. The proposed work aims to investigate several design alternatives for CNN-based handwritten digit recognition, such as the number of layers, stride size, receptive field, kernel size, padding, and dilution. Furthermore, we intend to assess the effectiveness of several SGD optimization techniques in enhancing the performance of handwritten digit recognition. Using ensemble architecture

improves the recognition accuracy of a network. In this case, we want to obtain equal accuracy by employing a pure CNN design without ensemble architecture, because ensemble structures increase computational overhead and testing complexity. As a result, a CNN design is developed in order to obtain higher accuracy than ensemble systems while reducing operational complexity and expense. Furthermore, we demonstrate an appropriate combination of learning parameters in the design of a CNN that leads us to a new absolute record in categorizing MNIST handwritten digits. We conducted extensive trials and achieved 99.87% recognition accuracy for an MNIST dataset.

Problem Statement Definition

The following are the limitations that computers encounter while attempting to detect handwritten numbers:

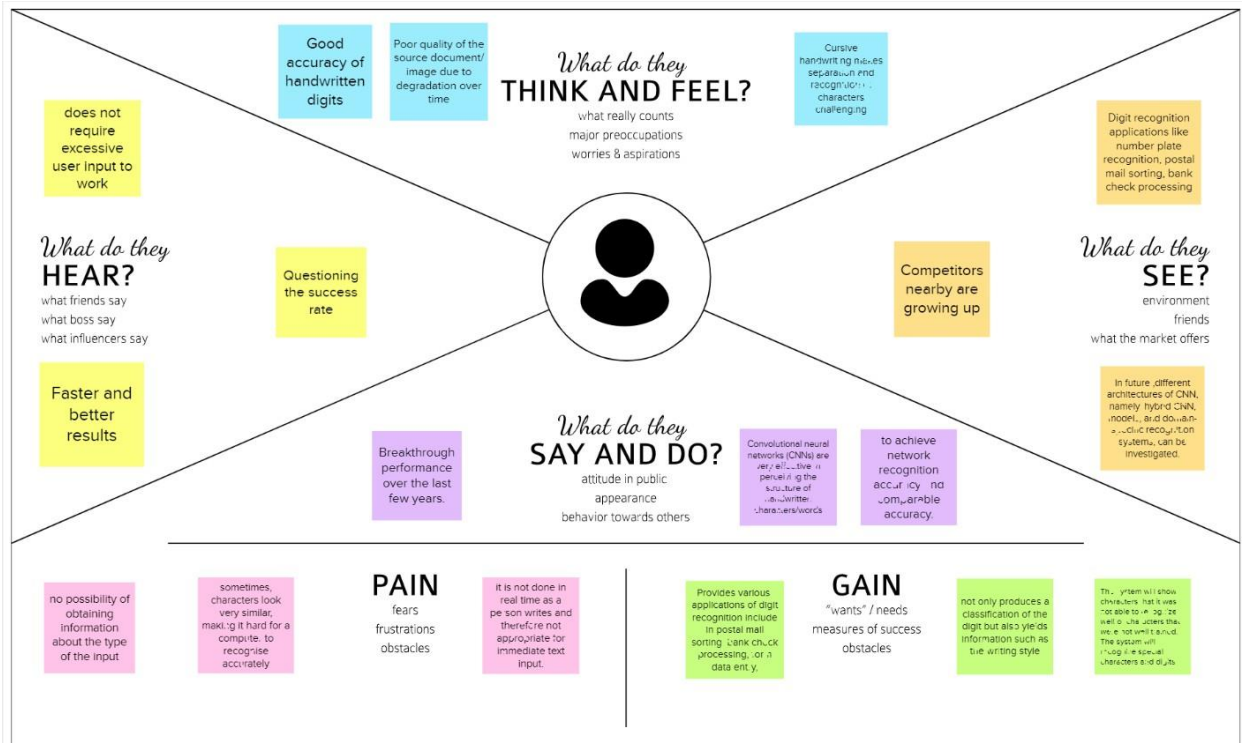
1. Because handwriting varies from person to person, handwritten numbers are not always the same size, width, and margins.

2. The likeness of digits like 1 and 7, which are justified and oriented to 5 and 6, 3 and 8, 2 and 7, etc. Therefore, it is a significant challenge for computers to classify between these values.

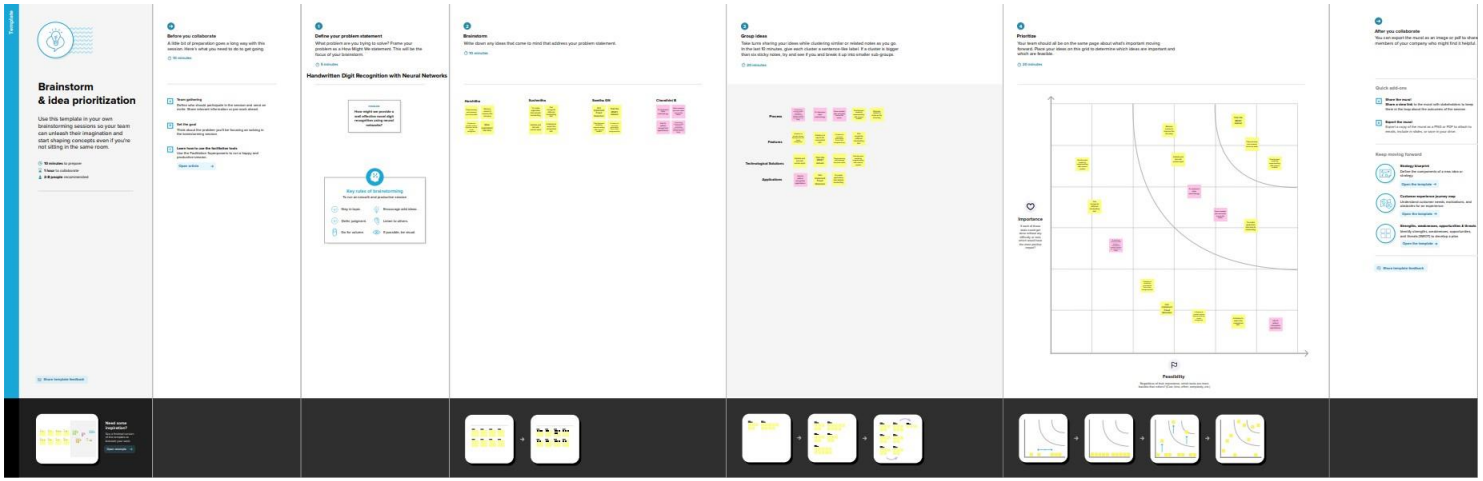
3. The uniqueness and variety of each person's handwriting also has an impact on the way the numerals form and appear.

3. IDEATION & PROPOSED SOLUTION

Empathy Map Canvas



Ideation & Brainstorming



Proposed Solution

1. Problem Statement (Problem to be solved)

In the modern world, digit recognition is crucial. It is capable of solving increasingly difficult problems and making humans' jobs easier. Handwritten digit recognition is one example. This is a worldwide system for recognizing zip codes or postal codes for mail sorting. Handwritten digit recognition can be accomplished using a variety of approaches. The machine has a difficult duty because handwritten digits are not flawless and can be generated with a variety of flavors. The solution to this issue is handwritten digit recognition, which uses an image of a digit and identifies the digit represented in the image.

2. Idea / Solution description

Handwritten digit recognition is performed using the MNIST dataset which contains 60,000 training images of handwritten digits from zero to nine and 10,000 images for testing. So, the MNIST dataset has 10 different classes. In this project, we are going to implement a handwritten digit recognition application trained using the Convolutional Neural Networks model. In the end, a GUI is built where the user gives the handwritten digit as input where it is recognized and the result is displayed immediately.

3. Novelty / Uniqueness

This project introduces an operative strategy for dealing with novelty in the handwritten visual recognition domain. A perfect transcription agent would be able to distinguish known and unknown characters in a picture, as well as determine any aesthetic variations that may occur inside or between texts. The existence of novelty has shown to be a major stumbling block for even the most robust machine learning-based algorithms for these activities. Novelty in handwritten papers might include, among other things, a change in the writer, character properties, writing attributes, or overall document appearance. Instead of examining each element separately, we believe that an integrated agent capable of processing known characters and novelties concurrently is a superior technique. The handwritten digit recognition problem can be seen as a subtask of the optical character recognition (OCR) problem.

4. Social Impact / Customer Satisfaction

There are many benefits associated with the handwriting recognition system. In addition to reading postal addresses and bank check amounts, it is also useful for reading forms. Furthermore, it's used in fraud detection because it makes it easy to compare two texts and determine which one is a copy. As a result, this system fulfills customers' expectations, as it is a novel method for recognizing handwritten digits, ensuring high accuracy for the model and meeting all customer expectations. Users will save a lot of time and effort if the system provides various synonyms for the words recognized. Due to the fact that the users in rural areas will be

using their own regional language, this proposed system should be able to detect those digits as well. As the system is being used in socially crowded places such as banks to check amounts, it should be fast and reliable. As it is designed to solve real-world problems, it should be highly reliable and trustworthy in every way, and users throughout the world should be able to use it effectively.

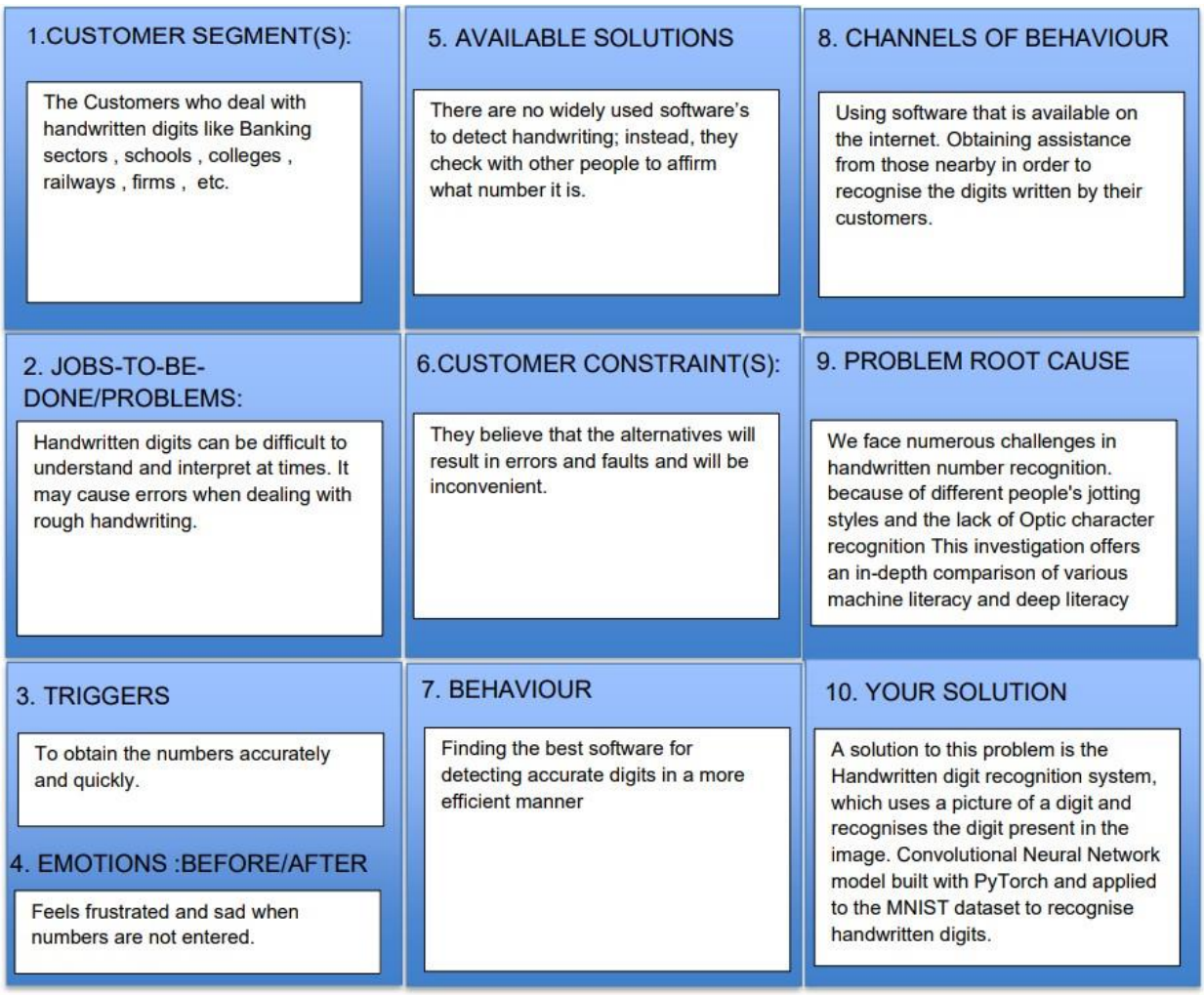
5. Business Model (Revenue Model)

A revenue model means understanding how a startup can make money. Our major revenue sources consist of sales, government funds, and public donations. The introduction of novel ideas increases revenue streams, such as introducing gesture or touch features , voice read out recognized digits, etc..

6. Scalability of the Solution

One of the approaches to making the handwritten digit recognition system scalable is to make use of cloud-native methods. For example, one of the cloud solutions for making AI scalable is IBM Cloud. IBM Cloud Build helps run and manage AI models, optimize decisions at scale across any cloud. The advantage of using cloud to make solutions scalable is that we can deploy our AI application on the specific cloud environment that best supports our business needs. We can take advantage of built-in security capabilities and AI model monitoring. We can Automate AI lifecycles with ModelOps pipelines, deploy and run models through one-click integration and also prepare and build models visually and programmatically. Looking at these advantages, we can drive better business outcomes by optimizing our decisions and also make our solution scalable using cloud

Problem Solution fit



4. REQUIREMENT ANALYSIS

Functional requirement

FR No.	Sub Requirement (Story / Sub-Task)
FR-1	Image Data: Handwritten digit recognition refers to a computer's capacity to identify human handwritten digits from a variety of sources, such as photographs, documents, touch screens, etc., and categorise them into ten established classifications (0-9). In the realm of deep learning, this has been the subject of countless studies.
FR-2	Website: Web hosting makes the code, graphics, and other items that make up a website accessible online. A server hosts every website you've ever visited. The type of hosting determines how much space is allotted to a website on a server. Shared, dedicated, VPS, and reseller hosting are the four basic varieties.
FR-3	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-4	Cloud: The cloud offers a range of IT services, including virtual storage, networking, servers, databases, and applications. In plain English, cloud computing is described as a virtual platform that enables unlimited storage and access to your data over the internet.
FR-5	Modified National Institute of Standards and Technology dataset: The abbreviation MNIST stands for the MNIST dataset. It is a collection of 60,000 tiny square grayscale photographs, each measuring 28 by 28, comprising handwritten single digits between 0 and 9.

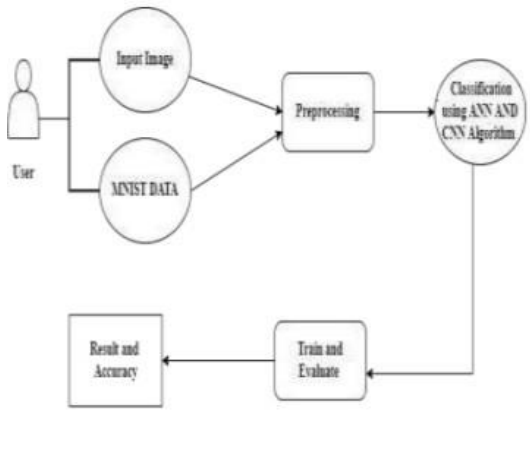
Non-Functional requirements

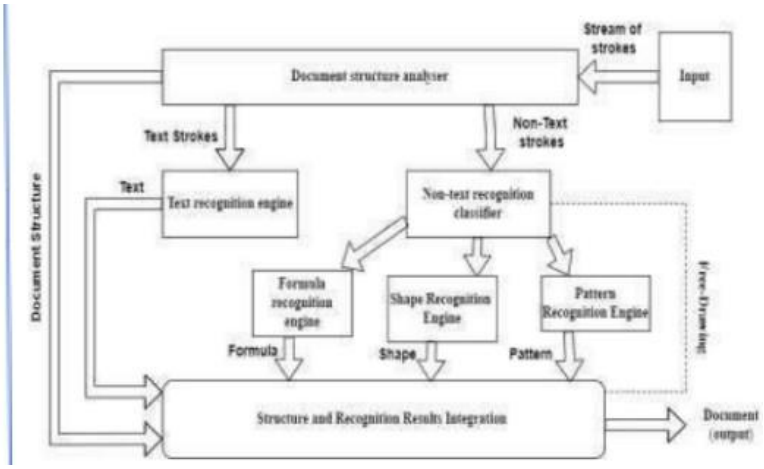
FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	One of the very significant problems in pattern recognition applications is the recognition of handwritten characters. Applications for digit recognition include filling out forms, processing bank checks, and sorting mail.
NFR-2	**Security**	1) The system generates a thorough description of the instantiation parameters, which might

		reveal information like the writing style, in addition to a categorization of the digit. 2) The generative models are capable of segmentation driven by recognition. 3) The procedure uses a relatively.
NFR-3	**Reliability**	The samples are used by the neural network to automatically deduce rules for reading handwritten digits. Furthermore, the network may learn more about handwriting and hence enhance its accuracy by increasing the quantity of training instances. Numerous techniques and algorithms, such as Deep Learning/CNN, SVM, Gaussian Naive Bayes, KNN, Decision Trees, Random Forests, etc., can be used to recognise handwritten numbers.
NFR-4	**Accuracy**	With typed text in high-quality photos, optical character recognition (OCR) technology offers accuracy rates of greater than 99%. However, variances in spacing, abnormalities in handwriting, and the variety of human writing styles result in less precise character identification.
NFR-5	**Availability**	

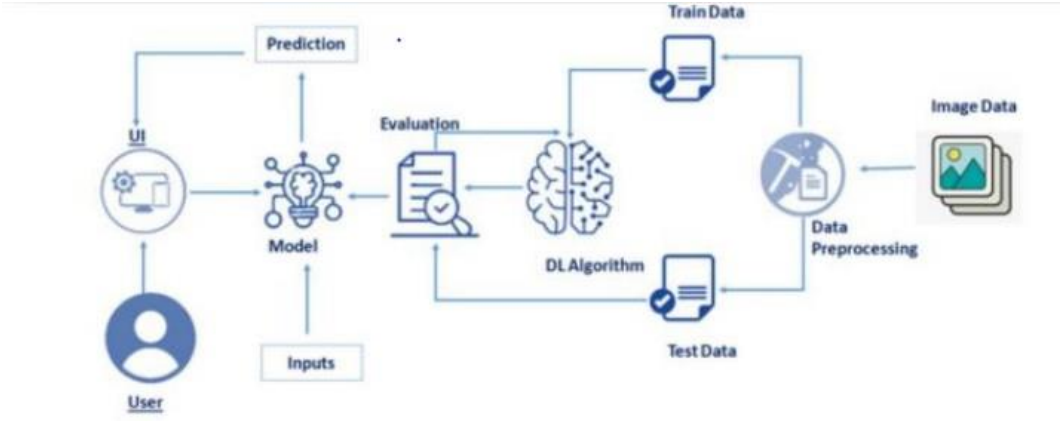
5. PROJECT DESIGN

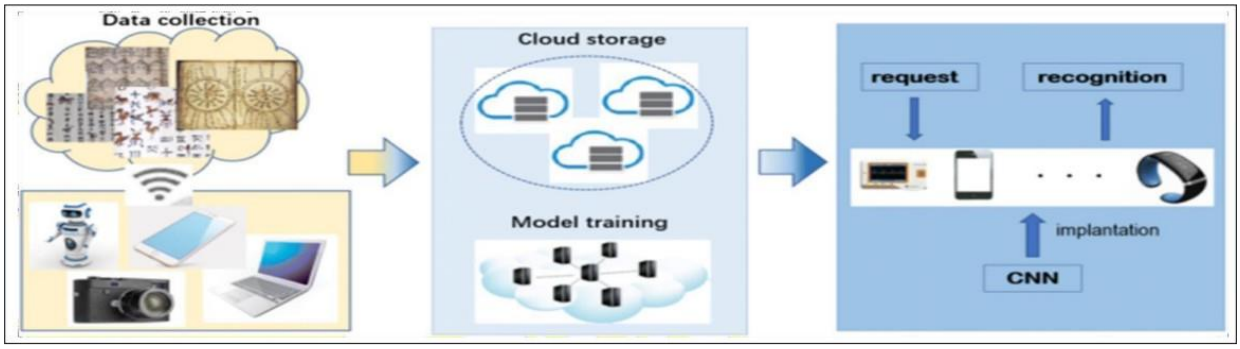
Data Flow Diagrams





Solution & Technical Architecture





User Stories

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobile user)	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	I can access my account / dashboard	High	Sprint-1
		USN-2	As a user, I will receive confirmation email once I have registered for the application	I can receive confirmation email & click confirm	High	Sprint-1
		USN-3	As a user, I can register for the application through Facebook	I can register & access the dashboard with Facebook Login	Low	Sprint-2
		USN-4	As a user, I can register for the application through Gmail	I can register the application with Gmail	Medium	Sprint-2
	Login	USN-5	As a user, I can log into the application by entering email & password	I can login to the application	High	Sprint-1
	Home	USN-6	As a user, I can view the application's home page where I can read the instructions to use this application	I can read instructions also and the home page is user-friendly.	Low	Sprint-1
	Upload Image	USN-7	As a user, I can able to input the images of digital documents to the application	As a user, I can able to input the images of digital documents to the application	High	Sprint-3
	Predict	USN-8	As a user I can able to get the recognised digit as output from the images of digital documents or images	I can access the recognized digits from digital document or images	High	Sprint-3
		USN-9	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.	Medium	Sprint-4
Customer (Web user)	Accessibility	USN-10	As a user, I can use the web application virtually anywhere.	I can use the application in any device with a browser	Medium	Sprint-4

6. PROJECT PLANNING & SCHEDULING

Sprint Planning & Estimation

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Data Collection	USN-1	As a user, I can collect the dataset from various resources with different handwritings.	10	Low	Jaya Krishna Raj Hirtih Kumar
Sprint-1	Data Preprocessing	USN-2	As a user, I can load the dataset, handling the missing data, scaling and split data into train and test.	10	Medium	Kumaran Kesheka
Sprint-2	Model Building	USN-3	As a user, I will get an application with ML model which provides high accuracy of recognized handwritten digit.	5	High	Jaya Krishna Raj Kumaran
Sprint-2	Add CNN layers	USN-4	Creating the model and adding the input, hidden, and output layersto it.	5	High	Hirtih Kumar Kesheka

Sprint-2	Compiling the model	USN-5	With both the training data defined and model defined, it's time to configure the learning process.	2	Medium	Kumaran Hirtih Kumar
Sprint-2	Train & test the model	USN-6	As a user, let us train our model with our image dataset.	6	Medium	Jaya Krishna Raj Kesheka
Sprint-2	Save the model	USN-7	As a user, the model is saved & integrated with an android application or web application in order to predict something.	2	Low	Kumaran Kesheka
Sprint-3	Building UI Application	USN-8	As a user, I will upload the handwritten digit image to the application by clicking a upload button.	5	High	Jaya Krishna Raj Hirtih Kumar
Sprint-3		USN-9	As a user, I can know the details of the fundamental usage of the application.	5	Low	Jaya Krishna Raj Kesheka

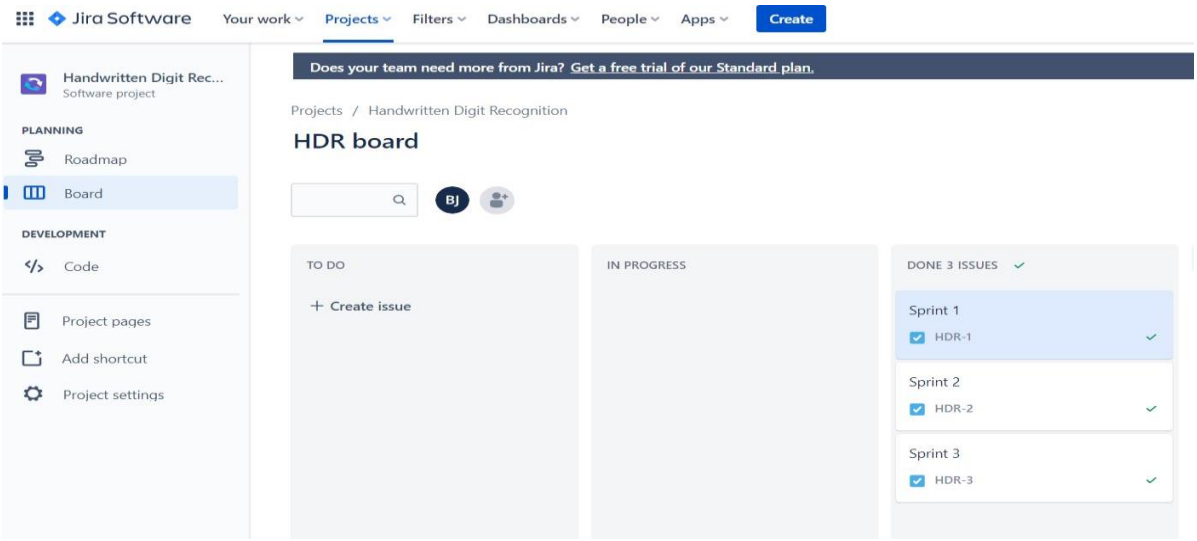
Sprint-3		USN-10	As a user, I can see the predicted / recognized digits in the application.	5	Medium	Kumaran Hirtih Kumar
Sprint-4	Train the model on IBM	USN-11	As a user, I train the model on IBM and integrate flask/Django with scoring end point.	10	High	Jaya Krishna Raj Kumaran
Sprint-4	Cloud Deployment	USN-12	As a user, I can access the web application and make the use of the product from anywhere.	10	High	Hirtih Kumar Kesheka E

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

Title	Description	Date
Literature Survey and Information Gathering	Gathering Information by referring to the technical papers, research publications ,etc.,	20 OCTOBER 2022
Prepare Empathy Map	To capture user pain and gains Prepare a List of Problem Statement.	20 OCTOBER 2022
Ideation	Prioritise a top 3 ideas based on feasibility and Importance	20 OCTOBER 2022
Proposed Solution	Solution include novelty, feasibility, business model, social impact and scalability of solution.	20 OCTOBER 2022
Problem Solution Fit	Solution fit document	20 OCTOBER 2022
Solution Architecture	Solution Architecture	20 OCTOBER 2022
Customer Journey	To Understand User Interactions and experiences with application.	21 OCTOBER 2022
Functional Requirement	Prepare functional Requirement	20 OCTOBER 2022
Data flow Diagrams	Data flow diagram	20 OCTOBER 2022
Technology Architecture	Technology Architecture diagram	20 OCTOBER 2022
Milestone & sprint delivery plan	Activity what we done &further plans	31 October 2022
Project Development Delivery of sprint 1,2,3 & 4	Develop and submit the developed code by testing it	26 October 2022 – 19 November 2022

Reports from JIRA



7. CODING & SOLUTIONING (Explain the features added in the project along with code)

Feature 1: Model Building

Using CNN Model in Our Project: CNN is a model known as Convolutional NeuralNetwork, and it has recently gained a lot of popularity due to its utility. CNN performs computational tasks by employing multilayer perceptrons. When compared to other image classification techniques, CNN requires very little pre-processing. This means that the network learns through filters that were hand-engineered in traditional algorithms. As a result, CNNs are the ideal alternative for image processing tasks.

```
from keras.models import Sequential
from keras.layers import Flatten,Dense
model=Sequential()
input_layer= Flatten(input_shape=(28,28))
model.add(input_layer)
hidden_layer1=Dense(512,activation='relu')
model.add(hidden_layer1)
hidden_layer2=Dense(512,activation='relu')
model.add(hidden_layer2)
output_layer=Dense(10,activation='softmax')
model.add(output_layer)
```

```
#compiling the sequential model
model.compile(optimizer = 'adam',
              loss = 'sparse_categorical_crossentropy',
              metrics=['accuracy'])
```

```
model.fit(train_img,train_lab,epochs=100)
```

```
plt.imshow(test_img[1],cmap='gray_r')
plt.title('Actual Value: {}'.format(test_lab[1]))
prediction=model.predict(test_img)
plt.axis('off')
print('Predicted Value: ',np.argmax(prediction[1]))
if(test_lab[1]==(np.argmax(prediction[1]))):
  print('Successful prediction')
else:
  print('Unsuccessful prediction')
```

```
Predicted Value:  2
Successful prediction
```

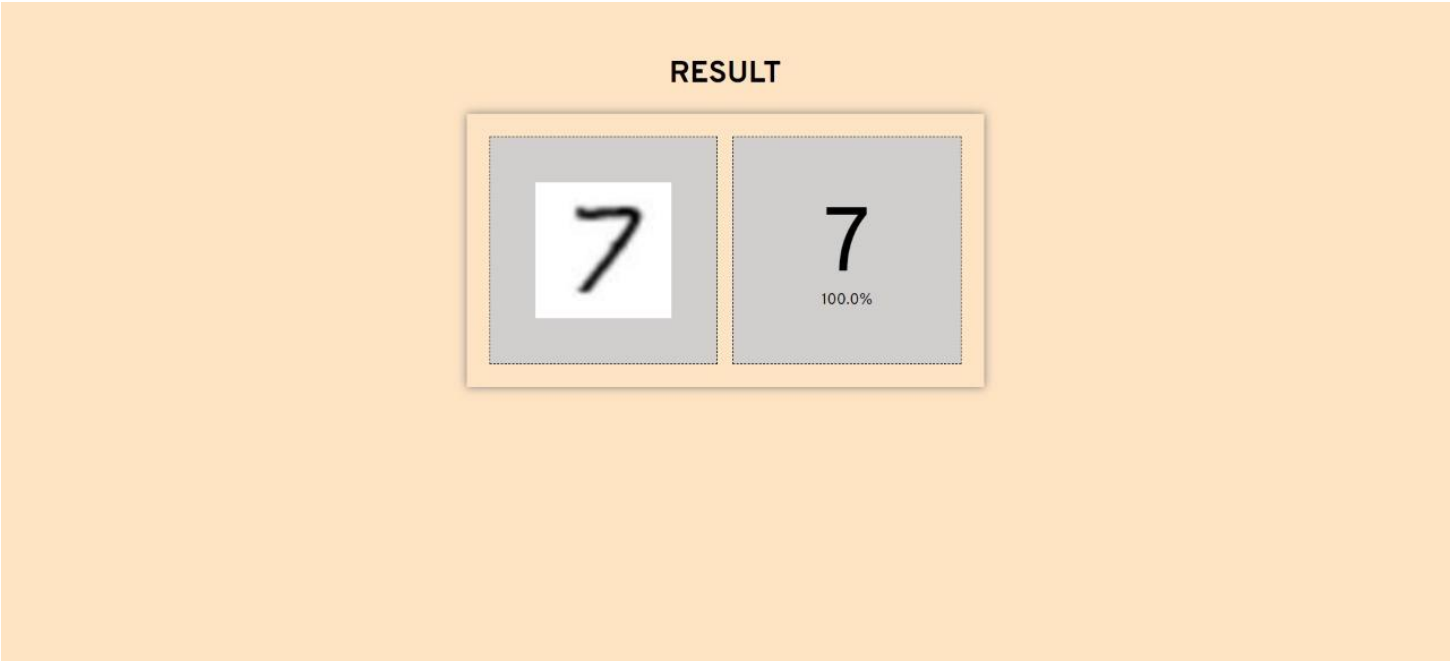
Feature 2: Application Building

In our project, we are use the Flask application: Flask is a Python-based micro web framework. It is characterised as a microframework because it does not necessitate the usage of any specific tools or libraries. It lacks a database abstraction layer, form validation, and other components where third-party libraries provide common functionalities. Flask, on the other hand, enables extensions that can add application functionalities as if they were built into Flask itself. There are extensions for object-relational mappers, form validation, upload handling, different open authentication protocols, and other framework-related tools.

```
model = load_model("Project Development Phase\Sprint 2\models\project.h5")

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


@app.route("/web",methods = ['GET','POST'])
def web():
    if request.method == "POST":
        f = request.files["image"]
        filepath = secure_filename(f.filename)
        f.save(os.path.join(app.config['UPLOAD_FOLDER'], filepath))

        upload_img = os.path.join(UPLOAD_FOLDER, filepath)
        img = Image.open(upload_img).convert("L")  # convert image to monochrome
        img = img.resize((28, 28))  # resizing of input image

        im2arr = np.array(img)  # converting to image
        im2arr = im2arr.reshape(1, 28, 28, 1)  # reshaping according to our requirement

        pred = model.predict(im2arr)

        num = np.argmax(pred, axis=1)  # printing our Labels

        return render_template('web.html',num=str(num[0]))
    return render_template('web.html')

if __name__ == "__main__":
    app.run(debug = True)
```

Output

Predict Page



Predicted Results

8. TESTING

Test Cases

Test Case ID	Feature Type	Component	Test Scenario	Expected Result	Actual Result	Status
Home_Page_TC_1	UI	Home Page	Check if UI elements in the home page are displayed properly	Home Page must be displayed correctly	Working as expected	PASS
Home_Page_TC_2	Functional	Home Page	Verify user is able to access the homepage	Homepage should be displayed	Working as expected	PASS
Model_TC_1	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
Model_TC_2	Functional	Model	Predicts the digit	The model should predict the number	Working as expected	PASS

Predict_Page_TC_1	Functional	Predict Page	Verify user is able to upload the image	Choose file popup should be displayed	Working as expected	PASS
Predict_Page_TC_2	Functional	Predict Page	Verify user is able to preview the uploaded image	Image should be displayed in the preview window	Working as expected	PASS
Predict_Page_TC_3	Functional	Predict Page	Verify user is able to view the predicted result	User should be able to view the predicted result	Working as expected	PASS

Unit Testing

When testing occurs for a specific group or set of related units, this is referred to as unit testing. It is frequently used by programmers to test the portion of the programme that they have implemented.

If unit testing is successful, it means that all modules have been tested and the process can continue.

Functional Testing

This type of testing is performed to determine whether or not the functional components or functionality required from the system have been obtained. It truly falls under the purview of Software Engineering's Black Box testing. This section includes feeding inputs into the system or project and determining whether or not the system or project is returning the expected value; if not, calculate the error and look for more. This project's functional testing consists primarily of the following items. All of these are successfully tested, and errors are calculated.
i)Checking the input image ii)Checking the work flow iii)Correct recognition and error calculation

Testing for Integration:

Many groupings of components are added or summed up in the objective of the project query in a comprehensive project or system. Integration testing involves examining the interplay of several project or system modules. This module also provides the project's hardware and software requirements.

All of the separate modules are combined and evaluated. All of the best and worst instances in which the modules interact or do not interact are correctly tested and passed, and mistakes are calculated for the deep learning platforms.

System Testing:

This type of testing is intended for the system or project, as well as the platform, integrated softwares and tools, and technologies. The goal or purpose of system testing is to check all of the requirements that the system will provide.This project's application, as well as the tools and technologies, have been tested in both Windows and Linux. It passed with flying colors.

User Acceptance Testing

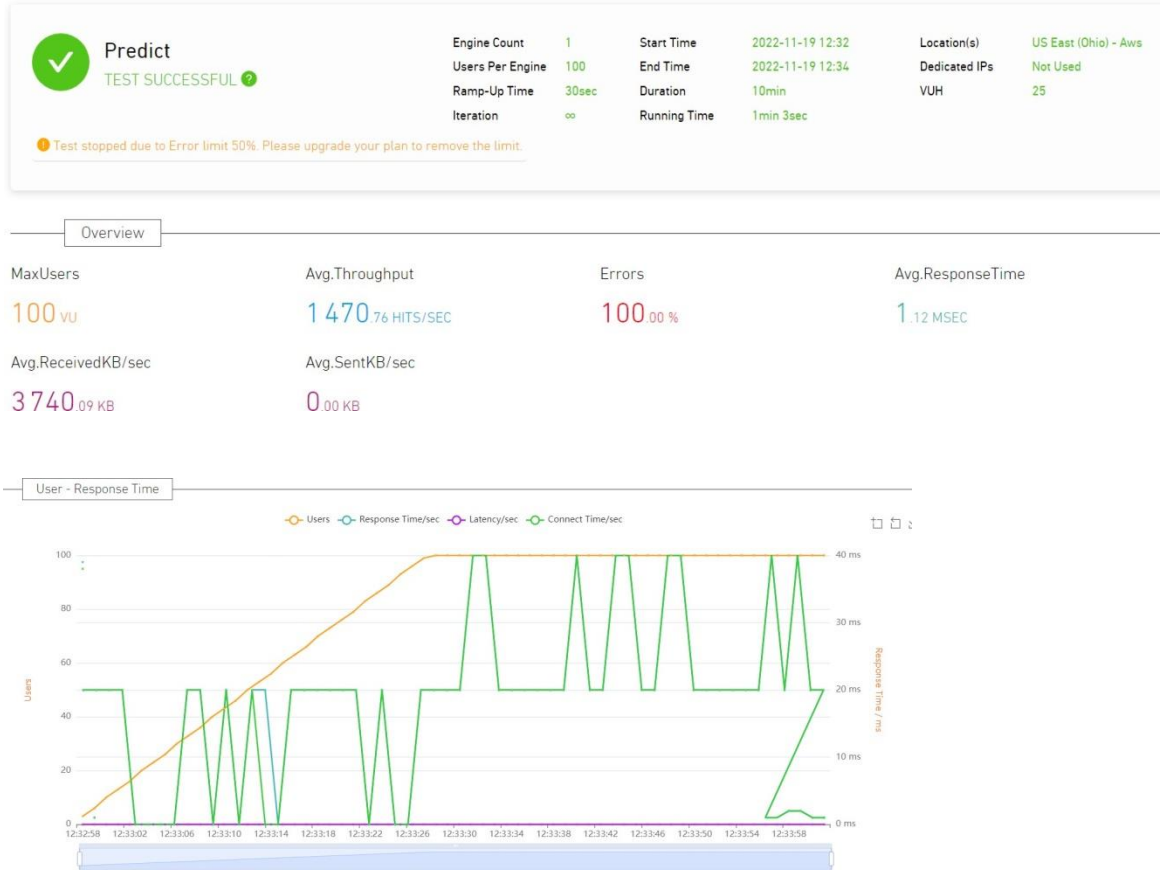
Defect Analysis

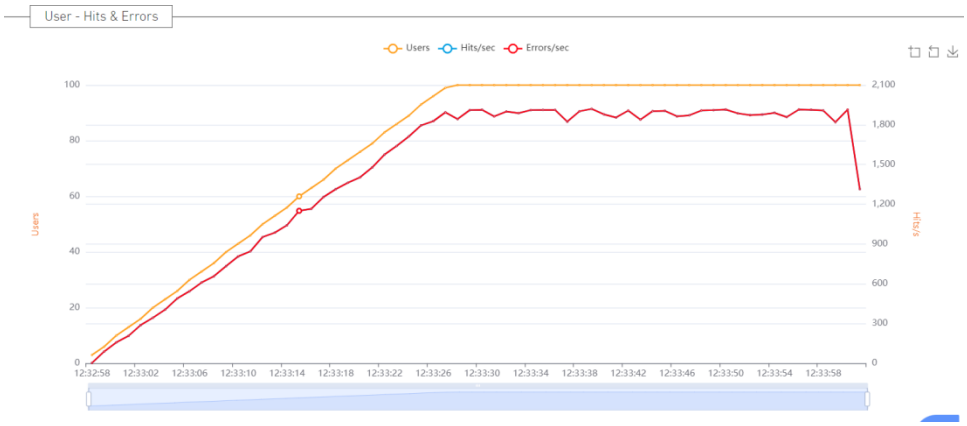
Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Total
By Design	1	0	0	0	1
Duplicate	0	0	0	0	0
External	0	0	0	0	0
Fixed	6	0	0	0	6
Not Reproduced	0	0	0	0	0
Skipped	0	0	0	0	0
Won’t Fix	0	0	0	0	0
Total	0	0	0	0	0

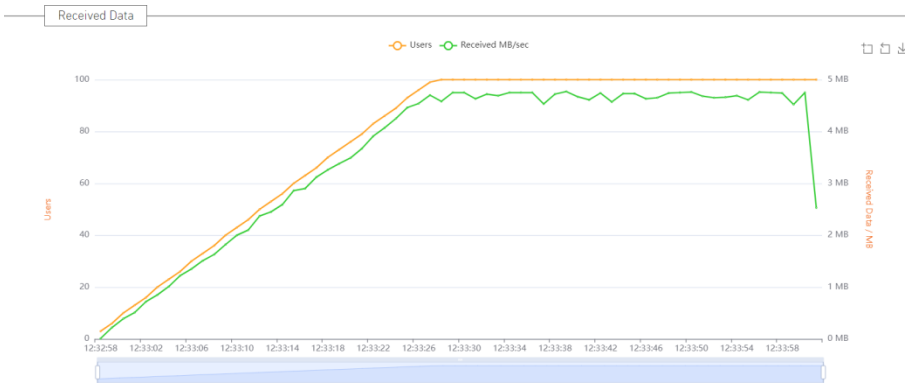
Test Case Analysis

Section	Total Cases	Not Tested	Fail	Pass
Client Application	5	0	0	5
Security	0	0	0	0
Performance	2	0	0	2
Exception Reporting	0	0	0	0

User Acceptance Testing Execution using Loadium



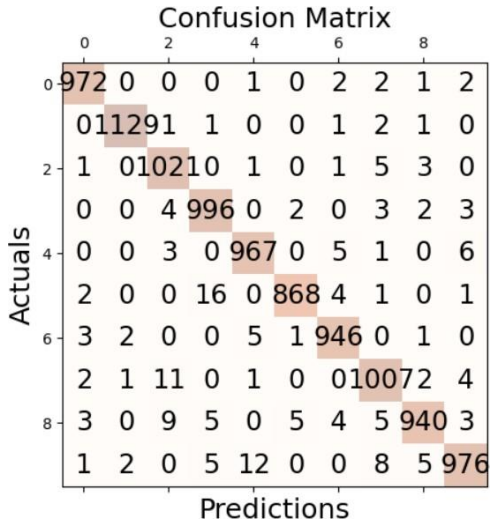




9. RESULTS

Performance Metrics

Confusion Matrix



Precision

```
Precision: 0.982
```

Recall

```
Recall: 0.982
```

Accuracy

```
Accuracy: 0.982
```

F1 measure

```
F1 Score: 0.982
```

10. ADVANTAGES & DISADVANTAGES

This method has a number of benefits, including the following:

- The system not only generates a classification of the digit but also a rich description of the instantiation parameters, which can yield information such as the writing style;
- The generative models can perform recognition driven segmentation;
- The method involves a relatively small number of parameters, making training relatively simple and quick;
- Unlike many other recognition schemes, it does not rely on some type of pre-processing.

The drawback is that it cannot be used for immediate text input because it is not done in real time as a person writes. Offline handwriting recognition has a wide range of uses, including reading postal addresses, bank check amounts, and forms.

11. CONCLUSION

Using the MNIST dataset, a deep learning model for Handwritten Digit Recognition based on Convolutional Neural Network was successfully trained,tested and implemented. The CNN based digit classifier model was trained using keras and tensorflow libraries.

The model is deployed in a Flask webpage interface which consists of a homepage entailing introduction to handwritten digit recognition system and a prediction page where the user is

prompted to select a digit image for prediction and the predicted result is displayed in the interface.
The accuracy of our model gives around 98.2% which outperforms many other models used for handwritten digit recognition systems. Also our model resulted in good performance metrics as well giving a score of precision, recall and F1 measures also as 98.2%.

12. FUTURE SCOPE

Applications based on deep learning and machine learning algorithms have virtually limitless potential for future growth. In the future, we can work on a denser or hybrid algorithm with more diverse data than the existing set of algorithms to find solutions to a variety of issues. Future applications of these algorithms will range from common users to high-level authorities. For example, we can use these algorithms in hospitals to provide detailed medical diagnoses, treatments, and patient monitoring. We can also use them in surveillance systems. These applications will range from common users to high-level authorities because of the differentiation of the algorithms described above.

13. APPENDIX

a. **Python**: Developed by Guido Van Rossum and initially released in 1991, Python is an interpreted, high-level, general-purpose programming language. Python's design philosophy stresses code Readability through the prominent use of significant White space. Its language constructs and object-oriented methodology are designed to aid programmers in creating clean, comprehensible code for both little and big projects. Python has garbage collection and dynamic typing. Programming paradigms like procedural, object-oriented, and functional programming are all supported.

b. **Keras** is a robust and user-friendly deep learning model development and evaluation Python library that is available for free and open source. It covers Theano and TensorFlow, two efficient frameworks for numerical computation, and enables you to define and train neural network models with just a few lines of code. It makes use of standalone machine learning toolkits, C#, Python, and C++ libraries. Although they are incredibly powerful libraries for building neural networks, Theano and TensorFlow are also challenging to grasp.

c. **TensorFlow:**

Google developed and released TensorFlow, a Python library for quick numerical computations. It is a foundation library that may be used to build Deep Learning models directly or indirectly using wrapper libraries created on top of TensorFlow to make the process easier. The TensorFlow tutorial is intended for both novices and experts. Deep neural networks, image processing, and sentiment analysis are just a few of the deep learning techniques covered in this tutorial. TensorFlow is one of the most well-known deep learning frameworks, and it was created by the Google Team. This tutorial is set up so that we can easily implement a deep learning project using a free and open source software library that was created in the Python programming language.

d. **JupyterLab,** a web-based interactive development environment for Jupyter notebooks, code, and data, provides a complement to Jupyter. JupyterLab can be configured and organized to accommodate a variety of data science, scientific computing, and machine learning workflows. JupyterLab is modular and expandable; you may create plugins that connect new components with those already in use.

e. **Machine learning** is a technique for data analysis that automates the creation of analytical models. It is a subfield of artificial intelligence founded on the notion that machines are capable of learning from data, spotting patterns, and making judgments with little assistance from humans.

f. **Deep Learning:** Deep learning is a feature of artificial intelligence (AI) that mimics how the human brain processes information and builds patterns to aid in decision-making. In artificial intelligence, deep learning is a subclass of machine learning that enables networks to learn unsupervised from unstructured or unlabeled data. also referred to as a deep neural network or deep learning.

g. **Neural Networks**: Using a technique that resembles how the human brain functions, a neural network is a collection of algorithms that aims to identify underlying relationships in a set of data. In this context, neural networks are systems of neurons that can be either organic or synthetic in origin.

SOURCE CODE

home.html

```
    <!DOCTYPE html>
<html>
    <head>
        <meta name="viewport" content="width=device-width, initial-scale=1.0" />
        <title>Handwritten Digit Recognition</title>
        <link rel="icon" type="image/svg" sizes="32x32"
href="{{url_for('static',filename='images/icon.svg')}}" />
        <link rel="stylesheet" href="{{url_for('static',filename='css/main.css')}}" />
        <script src="https://unpkg.com/feather-icons"></script>
        <script defer src="{{url_for('static',filename='js/script.js')}}"></script>
    </head>
    <body>
        <div class="container">
            <div class="heading">
                <h1 class="heading__main">Handwritten Digit Recognizer</h1>
                <h2 class="heading__sub">Easily analyze and detect handwritten
digits</h2>
            </div>
            <div class="upload-container">
                <div class="form-wrapper">
                    <form class="upload" action="/predict" method="post"
enctype="multipart/form-data">
                        <label id="label" for="upload-image"><i data-feather="file-
plus"></i>Select File</label>
                        <input type="file" name="photo" id="upload-image" hidden />
                        <button type="submit" id="up_btn"></button>
                    </form>
                    <img id="loading"
src="{{url_for('static',filename='images/loading.gif')}}">
                </div>
            </div>
        </div>
    </body>
</html>
```

predict.html

```
<html>

  <head>

        <title>Prediction | Handwritten Digit Recognition</title>

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

        <link rel="icon" type="image/svg" sizes="32x32"
href="{{url_for('static',filename='images/icon.svg')}}" />
```

```
        <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    </head>
    <body>
        <div class="container">
            <h1>Prediction</h1>
            <div class="result-wrapper">
                <div class="input-image-container">
                    <img
src="{{url_for('static',filename='data/')}}{{img_name}}" />
                </div>
                <div class="result-container">
                    <div class="value">{{best.0}}</div>
                    <div class="accuracy">{{best.1}}%</div>
                </div>
            </div>
            <h1>Other Predictions</h1>
            <div class="other_predictions">
                {% for x in others %}
                <div class="value">
                    <h2>{{x.0}}</h2>
                    <div class="accuracy">{{x.1}}%</div>
                </div>
                {% endfor %}
            </div>
        </div>
    </body>
</html>
```

app.py

```
from flask import Flask, render_template, request
from recognizer import recognize
app = Flask(__name__)
@app.route("/")
def main():
    return render_template("home.html")
@app.route("/predict", methods=["POST"])
def predict():
    if request.method == "POST":
        image = request.files.get("photo", "")
        best, others, img_name = recognize(image)
        return render_template(
            "predict.html", best=best, others=others, img_name=img_nam
```

```
  )
if __name__ == "__main__":
    app.run()
```

GitHub & Project Demo Link :

https://github.com/IBM-EPBL/IBM-Project-19296-1659695653

https://drive.google.com/file/d/1BuniBklfr6jVd0sfMCwo804Zek9OfmLl/view?usp=sharing