

**A NOVEL METHOD FOR HANDWRITTEN DIGIT
RECOGNITION SYSTEM**

IBM-Project-41956-1660646536

*Nalaiya Thiran project based learning on professional readlines for innovation,
employment and entrepreneurship.*

A PROJECT REPORT BY

Murugan P	612719104042
Gowtham R	612719104025
Mohan P	612719104039
Tamilarasan T	612719104068

**BACHELOR OF ENGINEERING IN COMPUTER SCIENCE AND
ENGINEERING**

**THE KAVERY ENGINEERING COLLEGE
SALEM,MECHERI - 636 453**

SI. NO	CONTENTS	PAGE NO
1	INTRODUCTION	3
2	LITERATURE SURVEY	6
3	IDEATION & PROPOSED SOLUTION	9
4	REQUIREMENT ANALYSIS	16
5	PROJECT DESIGN	18
6	PROJECT PLANNING & SCHEDULING	22
7	CODING & SOLUTIONING	23
8	TESTING	25
9	RESULTS	28
10	ADVANTAGES & DISADVANTAGES	30
11	CONCLUSION	32
12	FUTURE SCOPE	33
13	APPENDIX	34

1. INTRODUCTION

Recognition is identifying or distinguishing a thing or an individual from the past experiences or learning. Similarly, Digit Recognition is nothing but recognizing or identifying the digits in any document. Digit recognition framework is simply the working of a machine to prepare itself or interpret the digits. Handwritten Digit Recognition is the capacity of a computer to interpret the manually written digits from various sources like messages, bank cheques, papers, pictures, and so forth and in various situations for web based handwriting recognition on PC tablets, identifying number plates of vehicles, handling bank cheques, digits entered in any forms etc.

Handwriting recognition of characters has been around since the 1980s. The task of handwritten digit recognition, using a classifier, has extraordinary significance and use such as – online digit recognition on PC tablets, recognizing zip codes on mail, processing bank check amounts, numeric sections in structures filled up by hand (for example - tax forms) and so on. There are diverse challenges faced while attempting to solve this problem. The handwritten digits are not always of the same size, thickness, or orientation and position relative to the margins. The main objective was to actualize a pattern characterization method to perceive the handwritten digits provided in the MINIST dataset of images of handwritten digits (0-9).

1.1 PROJECT OVERVIEW

The Handwritten digit recognition is the ability of computers to recognize human handwritten digits. It is a hard task for the machine because handwritten digits are not perfect and can be made with many different flavors. 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. Nevertheless, in these algorithms, appropriate filter size selection, data preparation, limitations in datasets, and noise have not been taken into consideration. As a consequence, most of the algorithms have failed to make a noticeable improvement in classification accuracy. To address the shortcomings of these algorithms, our paper presents the following contributions:

- Firstly, after taking the domain knowledge into consideration, the size of the effective receptive field (ERF) is calculated. Calculating the size of the ERF helps us to select a typical filter size which leads to enhancing the classification accuracy of our CNN.
- Secondly, unnecessary data leads to misleading results and this, in turn, negatively affects classification accuracy. To guarantee the dataset is free from any redundant or irrelevant variables to the target variable, data preparation is applied before implementing the data classification mission.
- Thirdly, to decrease the errors of training and validation, and avoid the limitation of datasets, data augmentation has been proposed.
- Fourthly, to simulate the real-world natural influences that can affect image quality, we propose to add an additive white Gaussian noise with $\sigma = 0.5$ to the MNIST dataset.

As a result, our CNN algorithm achieves state-of-the-art results in handwritten digit recognition, with a recognition accuracy of 99.98%, and 99.40% with 50% noise. A neural network is a model inspired by how the brain works. It consists

of multiple layers having many activations, this activation resembles neurons of our brain. A neural network tries to learn a set of parameters in a set of data which could help to recognize the underlying relationships. Neural networks can adapt to changing input; so the network generates the best possible result without needing to redesign the output criteria.

1.2 PURPOSE

Handwritten character recognition is one of the practically important issues in pattern recognition applications. The applications of digit recognition include postal mail sorting, bank check processing, form data entry, etc. The applications of digit recognition include postal mail sorting, bank check processing, form data entry, etc.

This paper presents an efficient handwritten digit recognition system based on CONVOLUTIONAL NEURAL NETWORKS(CNN).These features are very simple to implement compared to other methods. Pattern recognition is a data analysis method that uses machine learning algorithms to automatically recognize patterns and regularities in data.Handwritten digit recognition using MNIST dataset is a major project made with the help of Neural Network. It basically detects the scanned images of handwritten digits.

We have taken this a step further where our handwritten digit recognition system not only detects scanned images of handwritten digits but also allows writing digits on the screen with the help of an integrated GUI for recognition.

2. LITERATURE SURVEY

2.1 EXISTING PROBLEM

The handwritten digit recognition is the capability of computer applications to recognize the human handwritten digits. It is a hard task for the machine because handwritten digits are not perfect and can be made with many different shapes and sizes. The handwritten digit recognition system is a way to tackle this problem which uses the image of a digit and recognizes the digit present in the image. Convolutional Neural Network model created using Tensorflow library over the MNIST dataset to recognize handwritten digits.

Handwritten Digit Recognition is the capability of a computer to fetch the mortal handwritten integers from different sources like images, papers, touch defences, etc, and classify them into 10 predefined classes (0-9). This has been a Content of bottomless- exploration in the field of deep literacy. Number recognition has numerous operations like number plate recognition, postal correspondence sorting, bank check processing, etc .

In Handwritten number recognition, we face numerous challenges . because of different styles of jotting of different peoples is not an optic character recognition. The main existing problem is, the model can not recognize multiple digit numbers. This exploration provides a comprehensive comparison between different machine literacy and deep literacy algorithms for the purpose of handwritten number recognition. For this, we've used Support, Multilayer Perceptron, and Convolutional Neural Network. The comparison between these algorithms is carried out on the base of their delicacy, crimes, and testing training time corroborated by plots and maps that have been constructed using matplotlib for visualization.

2.2 REFERENCES

1. Watada, J., & Pedrycz, W. A fuzzy regression approach to acquisition of linguistic rules. *Handbook of Granular Computing*, 719-732, 2008.
2. Seewald, A. K. (2011). On the brittleness of handwritten digit recognition models. *ISRN Machine Vision*, 2012.
3. Kloesgen, W., & Zytchow, J. *Handbook of Knowledge Discovery and Data Mining*.
4. Das, N., Sarkar, R., Basu, S., Kundu, M., Nasipuri, M., & Basu, D. K. A genetic algorithm based region sampling for selection of local features in handwritten digit recognition application. *Applied Soft Computing*, 12(5), 1592-1606, 2012.
5. Plamondon, R., & Srihari, S. N. Online and off-line handwriting recognition: a comprehensive survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(1), 63-84.

2.3 PROBLEM STATEMENT DEFINITION

The deep learning method is used for CNN to derive accuracy and performance. The results obtained from the above methods are compared and a method with high accuracy and high performance is considered as the best method for handwritten digits CNN is playing an important role in many sectors like image processing. It has a powerful impact on many fields. Even in nano-technologies like manufacturing semiconductors, CNN is used for fault detection and classification. Handwritten digit recognition has become an issue of interest among researchers. There are a large number of papers and articles being published these days about this topic. In research, it is shown that Deep Learning algorithm like multilayer CNN using Keras with Theano and Tensorflow gives the highest accuracy in comparison with the most widely used machine learning algorithms like SVM, KNN & RFC. Because of its highest accuracy, Convolutional Neural Network (CNN) is being used on a large scale in image classification, video analysis, etc.

3. IDEATION & PROPOSED SOLUTION

3.1 EMPATHY MAP CANVAS



3.2 IDEATION AND BRAINSTORMING

Template



Brainstorm & idea prioritization

Use this template in your own brainstorming sessions so your team can unleash their imagination and start shaping concepts even if you're not sitting in the same room.

🕒 10 minutes to prepare
🕒 1 hour to collaborate
👤 2-8 people recommended

[Share template feedback](#)

➔

Before you collaborate

A little bit of preparation goes a long way with this session. Here's what you need to do to get going.

🕒 10 minutes

A

Team gathering

Define who should participate in the session and send an invite. Share relevant information or pre-work ahead.

B

Set the goal

Think about the problem you'll be focusing on solving in the brainstorming session.

C

Learn how to use the facilitation tools

Use the Facilitation Superpowers to run a happy and productive session.

[Open article](#) ➔

1

Define your problem statement

What problem are you trying to solve? Frame your problem as a How Might We statement. This will be the focus of your brainstorm.

🕒 5 minutes

A Novel Method for Handwritten Digit Recognition System

PROBLEM

How might we make the computer capable of identifying and understanding handwritten digits or characters automatically

Key rules of brainstorming

To run a smooth and productive session

🗣️ Stay in topic.

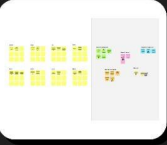
💡 Encourage wild ideas.

⏸️ Defer judgment.

👂 Listen to others.

🗣️ Go for volume.

👁️ If possible, be visual.



Need some inspiration?

See a finished version of this template to kickstart your work.

[Open example](#) ➔

10

2

Brainstorm

Write down any ideas that come to mind that address your problem statement.

🕒 10 minutes

Rishika V

- Train the model with handwriting from different users
- To implement Agile methodology.
- The data has to be preprocessed and cleaned
- Modify dataset once good accuracy is achieved

Swetha M

- Use a labelled dataset to train and validate
- Can implement Fraud Detection
- Can recognize different handwriting
- The application has to be real time

CS Tharunraj

- Must be able to export and share recognized text
- The system should be able to identify grammatical errors
- The system should be able to identify grammatical errors
- Any noise in the data has to be removed to improve accuracy
- Train Using MNIST dataset

Gokulrajan N

- Use the accuracy score to quantify the accuracy of the model
- Provide a good sensor to identify presence of text
- Find the best model by experimenting with various models
- Use for pattern recognition applications

3

Group ideas

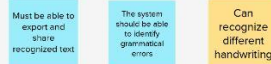
Take turns sharing your ideas while clustering similar or related notes as you go. In the last 10 minutes, give each cluster a sentence-like label. If a cluster is bigger than six sticky notes, try and see if you and break it up into smaller sub-groups.

🕒 20 minutes

Process



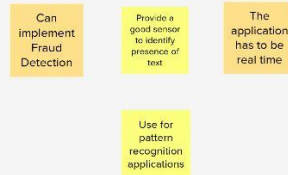
Features



Technological Solutions



Applications

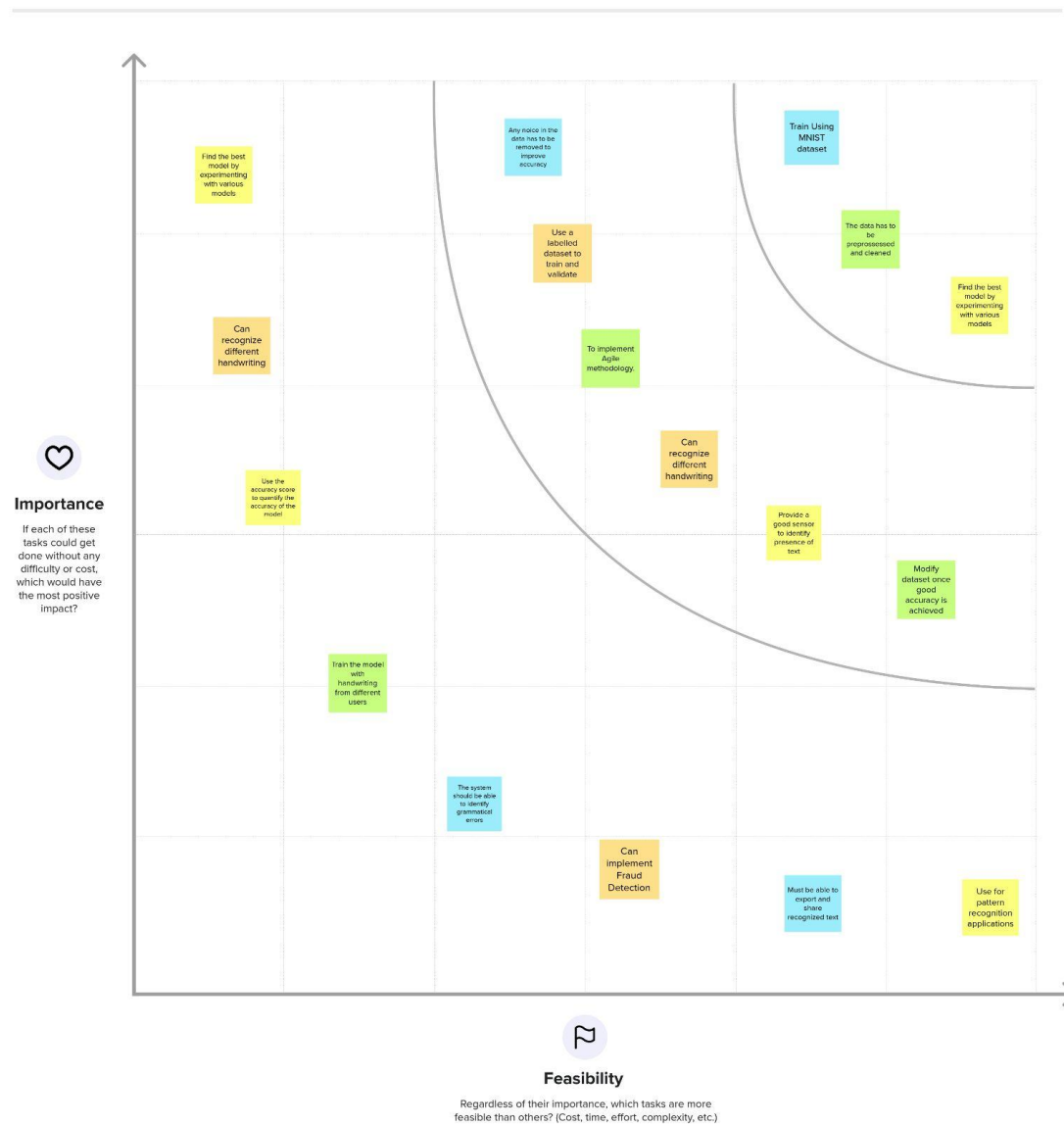


4

Prioritize

Your team should all be on the same page about what's important moving forward. Place your ideas on this grid to determine which ideas are important and which are feasible.

🕒 20 minutes



3.3 PROPOSED SOLUTION

S.No	Parameter	Description
1.	Problem Statement (Problem to be solved)	<ul style="list-style-type: none">● Mohan is a cashier, he needs a way to quickly enter the account details which are written by account holders in the challan so that account holders don't have to wait long.
2.	Idea / Solution description	<ul style="list-style-type: none">● Using MNIST dataset and Convolutional Neural Network to perform digits recognition.● All of those digits can be converted into electronic words in a text document format, and this data only needs a fraction of the physical storage space of the physical copies.
3.	Novelty / Uniqueness	<ul style="list-style-type: none">● Unlike OCR, which recognizes all the characters, it can accurately recognize the digits
4.	Social Impact / Customer Satisfaction	<ul style="list-style-type: none">● This system saves time and workload in sectors which use this technology.
5.	Business Model (Revenue Model)	<ul style="list-style-type: none">● It is very useful in the banking sectors, In the banking sector, numerical detail in cheque can be easily recognized.● Pincode details are easily obtained in the postal system.
6.	Scalability of the Solution	<ul style="list-style-type: none">● This system has no restriction on the number of digits to be recognized.

3.4 PROBLEM SOLUTION FIT

Define CS, fit into CC	1. CUSTOMER SEGMENT(S) CS The Bank Employee who makes the transaction through the Cheque.	6. CUSTOMER CC External dependencies are quite expensive and it is not offered by the people, so this process overcome the problem through their installation in mobile.	5. AVAILABLE SOLUTIONS AS → Automatic digit recognition. → In past, people identify the digits to their analysis sometimes it causes wrong transactions. → By using this application, they could easily identify the digits.	Explore AS, differentiate
	2. JOBS-TO-BE-DONE / PROBLEMS J&P Every sign has their own style of writing which could not recognize by the computer.	9. PROBLEM ROOT CAUSE RC Every single has their own style of writing which could not recognize by the computer.	7. BEHAVIOUR BE To classify the digits in correct way, they could make the transactions easier without any doubtfulness.	
Focus on J&P, fit into BE, understand RC	3. TRIGGERS TR Feel free to make transaction without any fear about their style of writing	10. YOUR SOLUTION SL → CNN model could be used to provide very high accuracy in image recognition problems and also reduces the high dimensionality of the images, without losing its information. → It can be used to convert the handwritten digits to machine readable format.	8. CHANNELS of BEHAVIOUR CH ONLINE: Promotion this application through the mobiles, the transaction could be done at any place without the presence in bank. OFFLINE: The identification of the digits which is in the handwritten from directly captured by using mobile application and that could be used to convert the those digits into machine readable forms.	Focus on J&P, fit into BE, understand RC
	4. EMOTIONS: BEFORE / AFTER EM If the person faces a problem regarding the transactions they could confidently handle the situation by using handwritten digit recognition system			
Identify strong TR & EM				Extend online & offline CH of BE

4. REQUIREMENT ANALYSIS

4.1 SYSTEM REQUIREMENT

This project needs the help of hardware and software requirements to be fit in the computer or the laptop. The user and the toolkits and hardware and software requirements are required also.

4.2 FUNCTIONAL REQUIREMENT

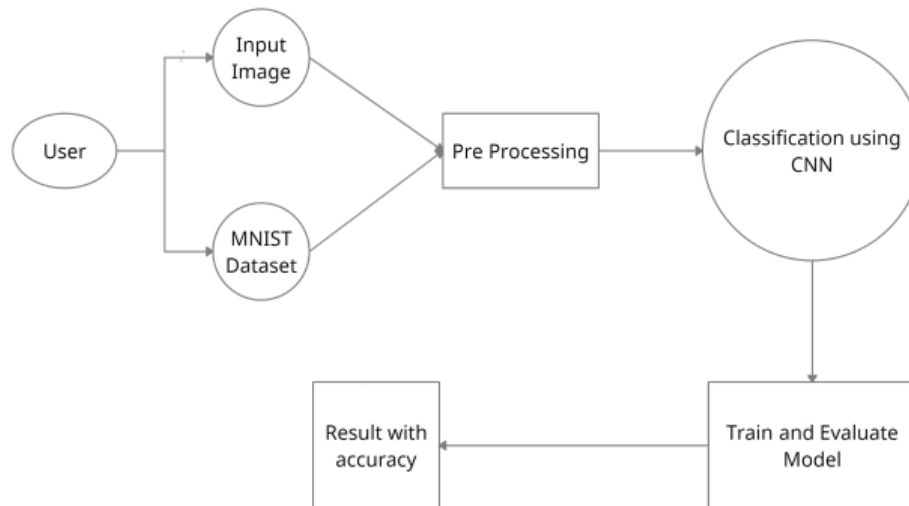
SI. NO	Functional requirements
1.	Image Data: <ul style="list-style-type: none">Handwritten digit recognition refers to a computer's capacity to identify human handwritten digits from a variety of sources, such as photographs, documents, touch screens, etc., and categorise them into ten established classifications (0-9). In the realm of deep learning, this has been the subject of countless studies.
2.	Website: <ul style="list-style-type: none">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.
3.	Modified National Institute of Standards and Technology dataset: <ul style="list-style-type: none">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.
4.	Digit Classifier Model: <ul style="list-style-type: none">To train a convolutional network to predict the digit from an image, use the MNIST database of handwritten digits and get the training and validation data first
5.	Cloud: <ul style="list-style-type: none">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.

4.2 NON FUNCTIONAL REQUIREMENT

SI. NO	Non Functional Requirements
1.	Usability: <ul style="list-style-type: none">• 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.
2.	Security: <ul style="list-style-type: none">• 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.• The generative models are capable of segmentation driven by recognition..
3.	Reliability: <ul style="list-style-type: none">• 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 recognize handwritten numbers.
4.	Accuracy: <ul style="list-style-type: none">• 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.
5.	Availability: <ul style="list-style-type: none">• The features for handwritten digit recognition have been Acquainted. These features are based on shape analysis of the digit image and extract slant or slope information. They are effective in obtaining good recognition of accuracy.
6.	Scalability: <ul style="list-style-type: none">• The scalability in the task of handwritten digit recognition, using a classifier, has great importance and it makes use of online handwriting recognition on computer tablets, recognizing zip codes on mail for postal mail sorting, processing bank check amounts, and numeric entries in forms filled up manually.

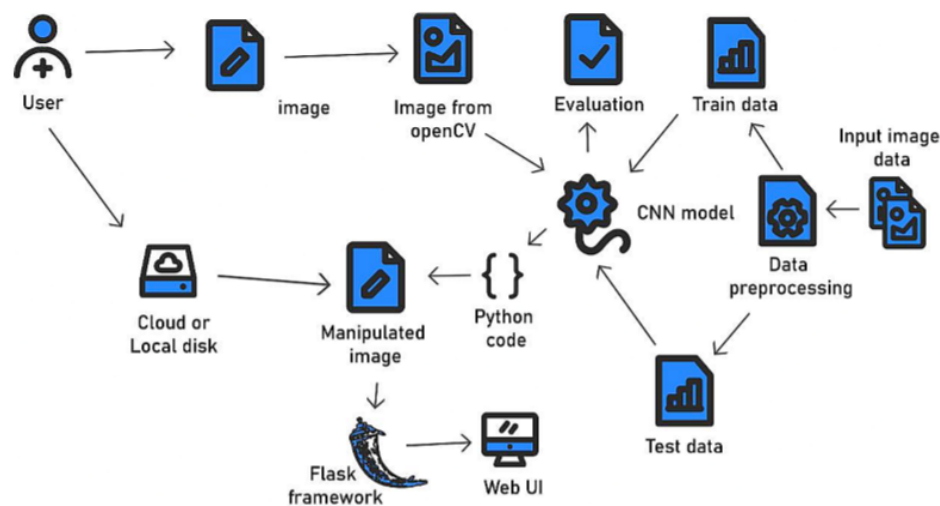
5. PROJECT DESIGN

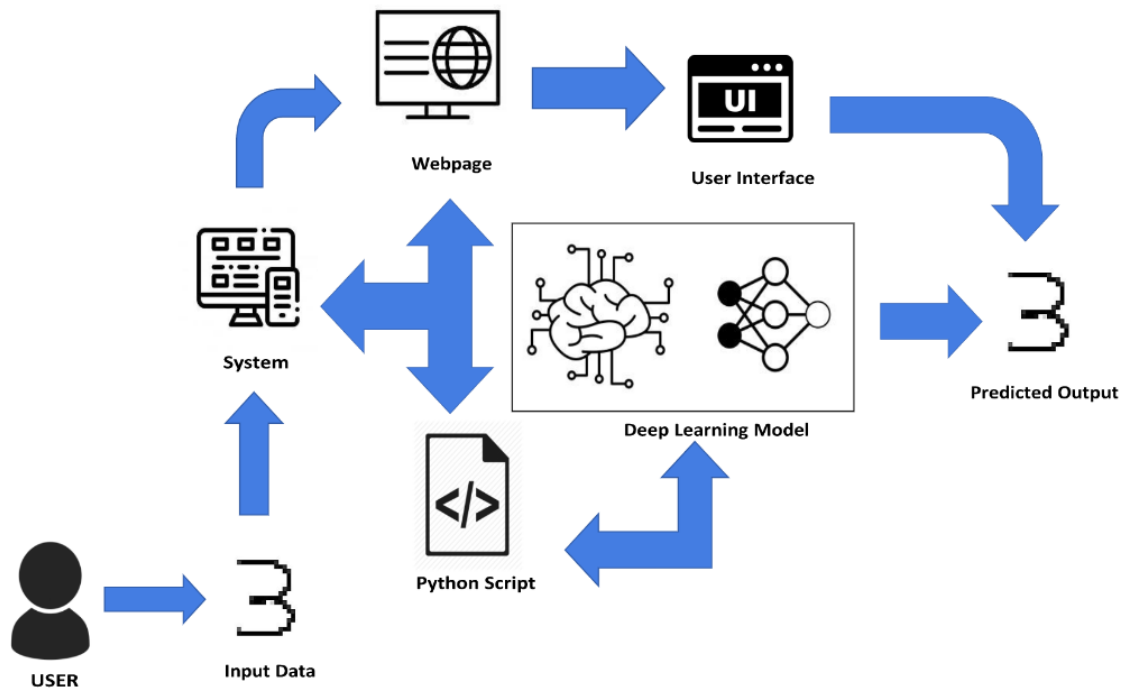
5.1 DATA FLOW DIAGRAMS



5.2 SOLUTION & TECHNICAL ARCHITECTURE

Solution architecture is a complex process – with many sub-processes – that bridges the gap between business problems and technology solutions. Its goals are to find the best tech solution to solve existing business problems. Describe the structure, characteristics, behaviour, and other aspects of the software to project stakeholders. Define features, development phases, and solution requirements. Provide specifications according to which the solution is defined, managed, and delivered.





5.3 USER STORIES

User Type	Functional Requirements (EPIC)	User Story Number	Number User Story / Task	Acceptance Criteria	Priority	Release
Customer	Application	USN-1	As a user, I can easily open an application	I can download the application	High	Sprint-1
		USN-2	As a user, I will be given access to upload the handwritten number image file.	I can upload the image	High	Sprint-2
		USN-3	As a user, I can upload the different types of image files.	I can upload different types of image files	Medium	Sprint-3

6. PROJECT PLANNING AND SCHEDULING

6.1 SPRINT PLANNING & ESTIMATION

Sprint	Functional Requirement	Task
Sprint-1	Image Data	As a User, you must acquire Image Data of Hand Written Images in order to train the model.
Sprint-2	Dash Board or Website	We created a dynamic Webpage to host our model using the Python Flask Framework (UI)
Sprint-3	Classifier Model	Image Classification Using the CNN Model
Sprint-4	Cloud	The Organised application is hosted on a cloud platform.

6.2 SPRINT DELIVERY SCHEDULE

Sprint	Duration	Sprint Start Date	Sprint End Date
Sprint-1	5 Days	31 Oct 2022	04 Nov 2022
Sprint-2	5 Days	05 Oct 2022	09 Nov 2022
Sprint-3	5 Days	10 Nov 2022	14 Nov 2022
Sprint-4	5 Days	5 Nov 2022	19 Nov 2022

7. CODING & SOLUTIONING

Recognizer.py

```
# importing required libraries
import cv2
import numpy as np
import matplotlib.pyplot as plt
from scipy import ndimage
import math
from keras.models import load_model

# loading pre trained model
model = load_model("models/digit_classifier.h5")

# predicting the digit
def predict_digit(img):
    test_image = img.reshape(-1, 28, 28, 1)
    return np.argmax(model.predict(test_image))

# refining each digit
def image_refiner(gray):
    org_size = 22
    img_size = 28
    rows, cols = gray.shape

    if rows > cols:
        factor = org_size / rows
        rows = org_size
        cols = int(round(cols * factor))
    else:
        factor = org_size / cols
        cols = org_size
        rows = int(round(rows * factor))
    gray = cv2.resize(gray, (cols, rows))

# get padding
colsPadding = (
    int(math.ceil((img_size - cols) / 2.0)),
    int(math.floor((img_size - cols) / 2.0)),
)
rowsPadding = (
```

```

        int(math.ceil((img_size - rows) / 2.0)),
        int(math.floor((img_size - rows) / 2.0)),
    )
    # apply padding
    gray = np.lib.pad(gray, (rowsPadding, colsPadding), "constant")
    return gray

# getting the predicted image
def get_output(path):
    img = cv2.imread(path, 2)
    img_org = cv2.imread(path)

    ret, thresh = cv2.threshold(img, 127, 255, 0)
    contours, hierarchy = cv2.findContours(
        thresh, cv2.RETR_CCOMP, cv2.CHAIN_APPROX_SIMPLE
    )

    predicted_values = []
    for j, cnt in enumerate(contours):
        epsilon = 0.01 * cv2.arcLength(cnt, True)
        approx = cv2.approxPolyDP(cnt, epsilon, True)

        hull = cv2.convexHull(cnt)
        k = cv2.isContourConvex(cnt)
        x, y, w, h = cv2.boundingRect(cnt)

        if hierarchy[0][j][3] != -1 and w > 10 and h > 10:
            # cropping each image and process
            roi = img[y : y + h, x : x + w]
            roi = cv2.bitwise_not(roi)
            roi = image_refiner(roi)
            th, fnl = cv2.threshold(roi, 127, 255,
cv2.THRESH_BINARY)

            # getting prediction of cropped image
            pred = predict_digit(roi)
            predicted_values.append(pred)
    return predicted_values

if __name__ == "__main__":
    recognized_digits = get_output(path="static/images/1.png")
    print(recognized_digits)

```


8. TESTING

8.1 TEST CASES

Test Case ID	Feature Type	Component	Test Scenario	Expected Result	Actual Result	Status
HP_TC_001	UI	Home Page	Verify UI elements in the Home Page	The Home page must be displayed properly	Working as expected	PASS
HP_TC_002	UI	Home Page	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 mobile screens	FAIL
HP_TC_003	Functional	Home Page	Check if user can upload their file	The input image should be uploaded to the application successfully	Working as expected	PASS
HP_TC_004	Functional	Home Page	Check if user cannot upload unsupported files	The application should not allow user to select a non image file	User is able to upload only JPEG, JPG, PNG, and SVG	PASS
HP_TC_005	Functional	Home Page	Check if the page redirects to the result page once the input is given	The page should redirect to the results page	Working as expected	PASS
BE_TC_001	Functional	Backend	Check if all the routes are working properly	All the routes should properly work	Working as expected	PASS

M_TC_001	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
M_TC_002	Functional	Model	Check if the model predicts the digit	The model should predict the number	Working as expected	PASS
M_TC_003	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
RP_TC_001	UI	Result Page	Verify UI elements in the Result Page	The Result page must be displayed properly	Working as expected	PASS
RP_TC_002	UI	Result Page	Check if the input image is displayed properly	The input image should be displayed properly	Working as expected	PASS
RP_TC_003	UI	Result Page	Check if the result is displayed properly	The result should be displayed properly	Working as expected	PASS
RP_TC_004	UI	Result Page	Check if the other predictions are displayed properly	The other predictions should be displayed properly	Working as expected	PASS

8.2 USER ACCEPTANCE TESTING

8.2.1 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

8.2.2 TEST CASE ANALYSIS

Section	Total Cases	Not Tested	Fail	Pass
Client Application	10	0	3	7
Security	2	0	1	1
Performance	3	0	1	2
Exception Reporting	2	0	0	2

9. RESULTS

9.1 PERFORMANCE METRICS

9.1.1 MODEL SUMMARY

```
Model: "sequential"
```

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 26, 26, 32)	320
conv2d_1 (Conv2D)	(None, 24, 24, 64)	18496
max_pooling2d (MaxPooling2D)	(None, 12, 12, 64)	0
dropout (Dropout)	(None, 12, 12, 64)	0
flatten (Flatten)	(None, 9216)	0
dense (Dense)	(None, 128)	1179776
dropout_1 (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 10)	1290

```
=====  
Total params: 1,199,882  
Trainable params: 1,199,882  
Non-trainable params: 0  
=====
```

9.1.2 ACCURACY

Accuracy	0.9861000180244446
Loss	0.049150239676237106

9.1.3 CONFUSION MATRIX

0	0	1	2	3	4	5	6	7	8	9
0	975	0	1	0	0	0	2	0	1	1
1	0	1138	1	0	0	0	0	2	0	0
2	1	1	1022	0	1	0	0	5	2	0
3	0	0	0	1006	0	3	0	0	0	1
4	0	0	0	0	980	0	0	0	1	1
5	0	0	0	2	0	889	1	0	0	0
6	2	2	0	0	1	1	952	0	0	0
7	0	2	1	1	0	0	0	1021	1	2
8	3	0	2	1	0	2	1	2	962	1
9	1	2	0	0	10	5	0	7	3	981

10. ADVANTAGES & DISADVANTAGES

10.1 ADVANTAGES

The main motivation behind using convolutional layers is that it is typically true of images that pixels in close proximity are more related with each other than with pixels that are a greater distance away. Thus, compared to fully connected layers, convolutional layers give a better indication of general features that appear in an image by taking advantage of this spatial structure of images.

Shift Invariance

- A major shortcoming of fully interconnected networks is their dependence on position of a feature in an image. Such a network would recognize an image, but not its slightly shifted self. Training shift invariance in a fully connected network is possible, and involves extensive expansion of training data, but it's significantly more efficient to use convolution, which naturally has this property.
- Convolutional layers detect a given feature, regardless of its position on an image. Because the MNIST data set is centred and normalized, a fully connected network can still work, but a network with convolutional layers is able to handle data that is not properly centred or normalized.

Computationally Efficient

- Another consequence of using convolutional networks is that there are fewer parameters involved, making the network more computationally efficient to train.
- For any given neuron in a fully connected hidden layer, there is a weight and a bias associated with each neuron in the previous layer, and as such, the number of parameters scale as the number of neurons squared, assuming a similar number of neurons in each interconnected layer.
- Electronic data storage.

- More organized files.
- Easier data retrieval.
- Power full high level feature.

10.2 DISADVANTAGES

- Not always accurate.
- Spacing of letter of words.
- Different languages.
- May lack robustness when writing variations are large.
- Requires large amounts of training data.
- Does not model temporal variations very well.

11.CONCLUSION

In this paper, the Handwritten Digit Recognition using Deep learning methods has been implemented. The most widely used Machine learning algorithms, KNN, SVM, RFC and CNN have been trained and tested on the same data in order to acquire the comparison between the classifiers. Utilising these deep learning techniques, a high amount of accuracy can be obtained. Compared to other research methods, this method focuses on which classifier works better by improving the accuracy of classification models by more than 99%. Using Keras as backend and Tensorflow as the software, a CNN model is able to give accuracy of about 98.72%. In this initial experiment, CNN gives an accuracy of 98.72%, while KNN gives an accuracy of 96.67%, while RFC and SVM are not that outstanding.

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:

- Add support to detect digits from multiple images and save the results.
- 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.

13. APPENDIX

13.1 SOURCE CODE

Requirements.txt

1. Flask==2.2.2
2. keras==2.10.0
3. matplotlib==3.5.3
4. numpy==1.21.6
5. opencv-python==4.6.0.66
6. Pillow==9.3.0
7. scipy==1.7.3
8. tensorflow==2.10.0

Create_model.py

```
# importing required libraries
import numpy as np
import cv2
import matplotlib.pyplot as plt
from keras.models import Sequential
from keras.layers.core import Dense, Dropout, Activation, Flatten
from keras.layers import Conv2D, MaxPool2D
from keras import optimizers
from keras.datasets import mnist
from keras.utils import to_categorical
import keras

# loading mist hand written dataset
(X_train, y_train), (X_test, y_test) = mnist.load_data()
print(X_train.shape, y_train.shape, X_test.shape, y_test.shape)

# Applying threshold for removing noise
_, X_train_th = cv2.threshold(X_train, 127, 255, cv2.THRESH_BINARY)
_, X_test_th = cv2.threshold(X_test, 127, 255, cv2.THRESH_BINARY)

# Reshaping
X_train = X_train_th.reshape(-1, 28, 28, 1)
X_test = X_test_th.reshape(-1, 28, 28, 1)

# Creating categorical output from 0 to 9
y_train = to_categorical(y_train, num_classes = 10)
```

```

y_test = to_categorical(y_test, num_classes = 10)

# cross checking shape of input and output
print(X_train.shape, y_train.shape)
print(X_test.shape, y_test.shape)

# Creating CNN model
input_shape = (28,28,1)
number_of_classes = 10

# Applying CNN layers
model = Sequential()
model.add(Conv2D(32, kernel_size=(3,
3),activation='relu',input_shape=input_shape))

model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPool2D(pool_size=(2, 2)))

model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(128, activation='relu'))

model.add(Dropout(0.5))
model.add(Dense(number_of_classes, activation='softmax'))

# compiling the model
model.compile(loss=keras.losses.categorical_crossentropy,

optimizer=keras.optimizers.Adadelta(),metrics=[ 'accuracy'])

model.summary()

# fitting the model
history = model.fit(X_train, y_train,epochs=10, shuffle=True,
                    batch_size = 200,validation_data= (X_test,
y_test))

# saving the model
model.save("models\digit_classifier.h5")

```

TEMPLATES

Index.html

```
<!-- HOME PAGE -->
<!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, shrink-to-fit=no">
  <title>Home - HandWritten Digit Recognition System</title>
  <!-- Bootstrap CDN - CSS -->
  <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@4.1.3/dist/css/bootstrap.min.css"
integrity="sha384-MCw98/SFnGE8fJT3GXwEOngsV7Zt27NXFoaoApmYm81iuXoPkF
OJwJ8ERdknLPM0" crossorigin="anonymous">
  <style>
    .container {
      height: 100vh;
      display: flex;
      justify-content: center;
      align-items: center;
    }

    .display-4 {
      font-weight: 350;
    }

    .lead {
      /* font-weight: normal; */
      text-align: justify;
    }
  </style>
</head>

<body>
```

```

<div class="container">
  <div class="jumbotron">
    <!-- Project Name -->
    <h1 class="display-4">
      A Novel Method for Handwritten Digit Recognition
System
    </h1>
    <!-- Project Description -->
    <p class="lead">
      Handwritten digit recognition system is a technology
that is much needed in this world as of today. This
      system is used to recognize the digits from the
images that we provide as a input. Before proper
      implementation of this technology we have relied on
writing digits with our own hands which can result
      in errors. It is difficult to store and access
physical data with efficiency. This project presents
      recognizing the handwritten digits from the famous
MNIST dataset. Here, we will be using artificial
      neural network / convolutional neural network.
    </p>
    <!-- Routes to recognize page -->
    <a href="/recognize" class="btn btn-primary
btn-lg">Recognize</a>
  </div>
</div>
<!-- Bootstrap CDN - JavaScript -->
<script src="https://code.jquery.com/jquery-3.3.1.slim.min.js"
integrity="sha384-q8i/X+965Dz00rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+
8abtTE1Pi6jizo"
crossorigin="anonymous"></script>
<script
src="https://cdn.jsdelivr.net/npm/popper.js@1.14.3/dist/umd/popper.m
in.js"
integrity="sha384-ZMP7rVo3mIykV+2+9J3UJ46jBk0WLaUAdn689aCwoqbBJiSnjA
K/18WvCWPIpM49"
crossorigin="anonymous"></script>
<script
src="https://cdn.jsdelivr.net/npm/bootstrap@4.1.3/dist/js/bootstrap.

```

```
min.js"
```

```
integrity="sha384-ChfqquxuZUCnJSK3+MXmPNIyE6ZbWh2IMqE241rYiqJxyMiZ6OW  
/JmZQ5stwEULTy"
```

```
crossorigin="anonymous"></script>
```

```
</body>
```

```
</html>
```

Recognize.html

```
<!-- RECOGNIZE PAGE -->
```

```
<!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, shrink-to-fit=no">
```

```
  <title>Recognize - HandWritten Digit Recognition System</title>
```

```
  <!-- Bootstrap CDN - CSS -->
```

```
  <link rel="stylesheet"
```

```
href="https://cdn.jsdelivr.net/npm/bootstrap@4.1.3/dist/css/bootstra  
p.min.css"
```

```
integrity="sha384-MCw98/SFnGE8fJT3GXwEOngsV7Zt27NXFoaoApmYm81iuXoPkF  
0JwJ8ERdknLPM0" crossorigin="anonymous">
```

```
  <style>
```

```
    .container {  
      height: 100vh;  
      display: flex;  
      justify-content: center;  
      align-items: center;  
    }
```

```
    .display-4 {  
      font-weight: 350;  
    }
```

```
    .lead {
```

```

        /* font-weight: normal; */
        text-align: justify;
    }
</style>
</head>

<body>

    <!-- Back Button -->
    <nav class="navbar navbar-light bg-light">
        <a class="navbar-brand" href="/" style="font-size: 23px;">
            <img src={{back_url}} width="30" height="30" alt="">
        </a>
    </nav>

    <div class="container">
        <div class="jumbotron">
            <h1 class="display-4">Handwritten digit recognizer </h1>
            <p class="lead">This is the page where you can get the
result by uploading image from your local storage.
            </p>

            <!-- Gets image input -->
            <form method="POST" enctype="multipart/form-data">
                <input type="file" name="file" class="file"
required>
                <button class="btn btn-success
active">Recognize</button>
            </form>
            <br>
            <!-- Displays result -->
            {% if display_output %}
                
                <br>
                <div class="alert alert-{{alert_box_color}}
alert-dismissible fade show" role="alert">
                    <strong>{{display_output}}</strong>
                    <button type="button" class="close"
data-dismiss="alert" aria-label="Close">
                        <span aria-hidden="true">&times;</span>

```

```

        </button>
    </div>
{% endif %}
</div>
</div>
<!-- Bootstrap CDN - JavaScript -->
<script src="https://code.jquery.com/jquery-3.3.1.slim.min.js"
integrity="sha384-q8i/X+965Dz00rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+
8abtTE1Pi6jizo"
    crossorigin="anonymous"></script>
<script
src="https://cdn.jsdelivr.net/npm/popper.js@1.14.3/dist/umd/popper.m
in.js"
integrity="sha384-ZMP7rVo3mIykV+2+9J3UJ46jBk0WLaUAdn689aCwoqbBJiSnjA
K/18WvCWPIpM49"
    crossorigin="anonymous"></script>
<script
src="https://cdn.jsdelivr.net/npm/bootstrap@4.1.3/dist/js/bootstrap.
min.js"
integrity="sha384-ChfqqxuZUCnJSK3+MXmPNIyE6ZbWh2IMqE241rYiqJxyMiZ6OW
/JmZQ5stwEULTy"
    crossorigin="anonymous"></script>
</body>

</html>

```

Recognizer.py

```

# importing required libraries
import cv2
import numpy as np
import matplotlib.pyplot as plt
from scipy import ndimage
import math
from keras.models import load_model

# loading pre trained model
model = load_model("models/digit_classifier.h5")

```



```

# predicting the digit
def predict_digit(img):
    test_image = img.reshape(-1, 28, 28, 1)
    return np.argmax(model.predict(test_image))

# refining each digit
def image_refiner(gray):
    org_size = 22
    img_size = 28
    rows, cols = gray.shape

    if rows > cols:
        factor = org_size / rows
        rows = org_size
        cols = int(round(cols * factor))
    else:
        factor = org_size / cols
        cols = org_size
        rows = int(round(rows * factor))
    gray = cv2.resize(gray, (cols, rows))

    # get padding
    colsPadding = (
        int(math.ceil((img_size - cols) / 2.0)),
        int(math.floor((img_size - cols) / 2.0)),
    )
    rowsPadding = (
        int(math.ceil((img_size - rows) / 2.0)),
        int(math.floor((img_size - rows) / 2.0)),
    )

    # apply padding
    gray = np.lib.pad(gray, (rowsPadding, colsPadding), "constant")
    return gray

# getting the predicted image
def get_output(path):
    img = cv2.imread(path, 2)
    img_org = cv2.imread(path)

```

```

ret, thresh = cv2.threshold(img, 127, 255, 0)
contours, hierarchy = cv2.findContours(
    thresh, cv2.RETR_CCOMP, cv2.CHAIN_APPROX_SIMPLE
)

predicted_values = []

for j, cnt in enumerate(contours):
    epsilon = 0.01 * cv2.arcLength(cnt, True)
    approx = cv2.approxPolyDP(cnt, epsilon, True)

    hull = cv2.convexHull(cnt)
    k = cv2.isContourConvex(cnt)
    x, y, w, h = cv2.boundingRect(cnt)

    if hierarchy[0][j][3] != -1 and w > 10 and h > 10:
        # cropping each image and process
        roi = img[y : y + h, x : x + w]
        roi = cv2.bitwise_not(roi)
        roi = image_refiner(roi)
        th, fnl = cv2.threshold(roi, 127, 255,
cv2.THRESH_BINARY)

        # getting prediction of cropped image
        pred = predict_digit(roi)
        predicted_values.append(pred)

return predicted_values

if __name__ == "__main__":
    recognized_digits = get_output(path="static/images/1.png")
    print(recognized_digits)

```

App.py

```

# importing required libraries
from flask import Flask, render_template, url_for, request
from werkzeug.utils import secure_filename
import os
from recognizer import get_output

```

```

# instantiating the flask application
app = Flask(__name__)

# allowing only specified image files
ALLOWED_EXTENSIONS = ["jpg", "jpeg", "png", "svg"]
# folder path to store the uploaded image
UPLOAD_FOLDER = "static/images/"

# configuring application
app.secret_key = os.urandom(24)
app.config["UPLOAD_FOLDER"] = UPLOAD_FOLDER

def is_allowed_file(filename: str) -> bool:
    """
    Checking the upload file is valid or not
    """
    return ( "." in filename ) and (
        filename.rsplit(".", 1)[1].lower() in ALLOWED_EXTENSIONS
    )

# routes to home page
@app.route("/")
def index():
    return render_template("index.html")

# routes to recognize page
@app.route("/recognize", methods=["GET", "POST"])
def recognize():
    alert_box_color = "dark"
    display_output = None
    uploaded_img_path = None
    back_url = url_for("static", filename="icons/back.svg")

    if request.method == "POST":
        file = request.files["file"]
        if file and is_allowed_file(file.filename):
            filename = secure_filename(file.filename)
            file.save(os.path.join(app.config["UPLOAD_FOLDER"],
filename))

            image_url = url_for("static",

```

```

filename=f"images/{file.filename}")
    output = get_output(path=image_url[1:])

    if len(output) > 1:
        output = list(map(str, output))
        display_output = f"Recognized digits are {'',
'.join(output)}"
        uploaded_img_path = image_url[1:]
    else:
        display_output = f"Recognized digit is {output[0]}"
        uploaded_img_path = image_url[1:]

    else:
        alert_box_color = "danger"
        display_output = f"Chosen file format is not supported.
Choose only {'', '.join(ALLOWED_EXTENSIONS)} images."

    return render_template(
        "recognize.html",
        back_url=back_url,
        alert_box_color=alert_box_color,
        display_output=display_output,
        uploaded_img_path=uploaded_img_path,
    )

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

```

GITHUB & PROJECT DEMO LINK

