PROJECT REPORT

A NOVEL METHOD FOR HANDWRITTEN DIGIT RECOGNITION SYSTEM

submitted by

PNT2022TMID09252

Deepakraj R - 310619104023

Eshwanth G M - 310619104032

Jisho J - 310619104046

Kaarthic Varun K - 310619104049

Table of Contents

Chapter 1: Introduction

1.1. Project Overview

Deep learning and machine learning are two of the most important components of artificial intelligence and computer technology. They can help reduce the human effort required in various tasks and activities.

Handwritten Digit recognition is a process that computer systems can perform by recognizing handwritten numbers from various sources, such as documents and images. This project aims to provide users with a more efficient way of performing this task.

1.2. Purpose

Digit recognition systems are able to identify digit from a variety of sources, including emails, bank cheque, papers, images, etc. They can also be used in a variety of real-world situations, such as online handwriting recognition on computer tablets or systems, identifying vehicle license plates, processing bank cheque amounts, and reading numbers from forms that have been filled out by hand (such as tax forms).

Chapter 2: Literature Survey

2.1.Existing Problem

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

2.2.References

A novel method for Handwritten Digit Recognition with Neural Networks MALOTHU NAGU*1, N VIJAY SHANKAR#2, K. ANNAPURNA

Character recognition plays an important role in the modern world. It can solve more complex problems and makes humans' job easier. An example is handwritten character recognition. This is a system widely used in the world to recognize zip code or postal code for mail sorting. There are different techniques that can be used to recognize handwritten characters. Two techniques researched in this paper are Pattern Recognition and Artificial Neural Network (ANN). Both techniques are defined and different methods for each technique is also discussed. Bayesian Decision theory, Nearest Neighbor rule, and Linear Classification or Discrimination is types of methods for Pattern Recognition. Shape recognition, Chinese Character and Handwritten Digit recognition uses Neural Network to recognize them.

Neural Network is used to train and identify written digits. After training and testing, the accuracy rate reached 99%.This accuracy rate is very high.

Hao Y., Shi Y., Zhang D., Zhu X. 2001," An effective result-feedback neural algorithm for handwritten character recognition 'International Journal of Neural Parallel & Science Computations, Vol. 9z No. 2, Pp.139~150

In this paper, a new algorithm of handwritten character recognition based on result-feedback is proposed. It is designed as an effective neural network by adding confidence back-propagation and input modification, thus both pre-processing and recognition operations are closely integrated together. The convergence of the algorithm is proved and many experiments show that the error rate in such a result-feedback neural network (RFNN) can be greatly reduced as well as the robust to environmental noise

Kimura, F. and Shiridhar, M. (1991). Handwritten numerical recognition based on multiple algorithms. Pattern Recognition, no. 10, vol. 24, pp. 969-983

In this paper, the authors developed two algorithms for application to recognition of unconstrained isolated handwritten numerals. While both algorithms yielded very low error rates, the authors combined the two algorithms in different ways to study the best polling strategy and realized significant improvement in performance.

M. Shridhar and A. Badreldin, Recognition of isolated and simply connected handwritten numerals, Pattern Recognition 19, 1-12 (1986).

In this paper the authors describe the results of their investigation into the

development of a recognition algorithm for identifying numerals that may be isolated or connected, broken or continuous. Using a structural classification scheme, the recognition algorithm is derived as a tree classifier. In an extensive test experiment, an accuracy of 99% was realized with isolated numerals. When connected numerals were also included a recognition accuracy of 93% was obtained.

2.3. Problem Statement Definition

The traffic department has been pursuing violators of traffic laws for years. These criminals put not only their own life at peril but also those of others. It is essential to punish these offenders in order to prevent others from following in their footsteps. The inability of the average person to jot down a careless driver's license plate makes it nearly impossible to identify these offenders. In order to reduce traffic offences, the project's objective is to assist the traffic department in identifying these offenders.

Chapter 3: Ideation and Proposed Solution

3.1.Empathy Map Canvas

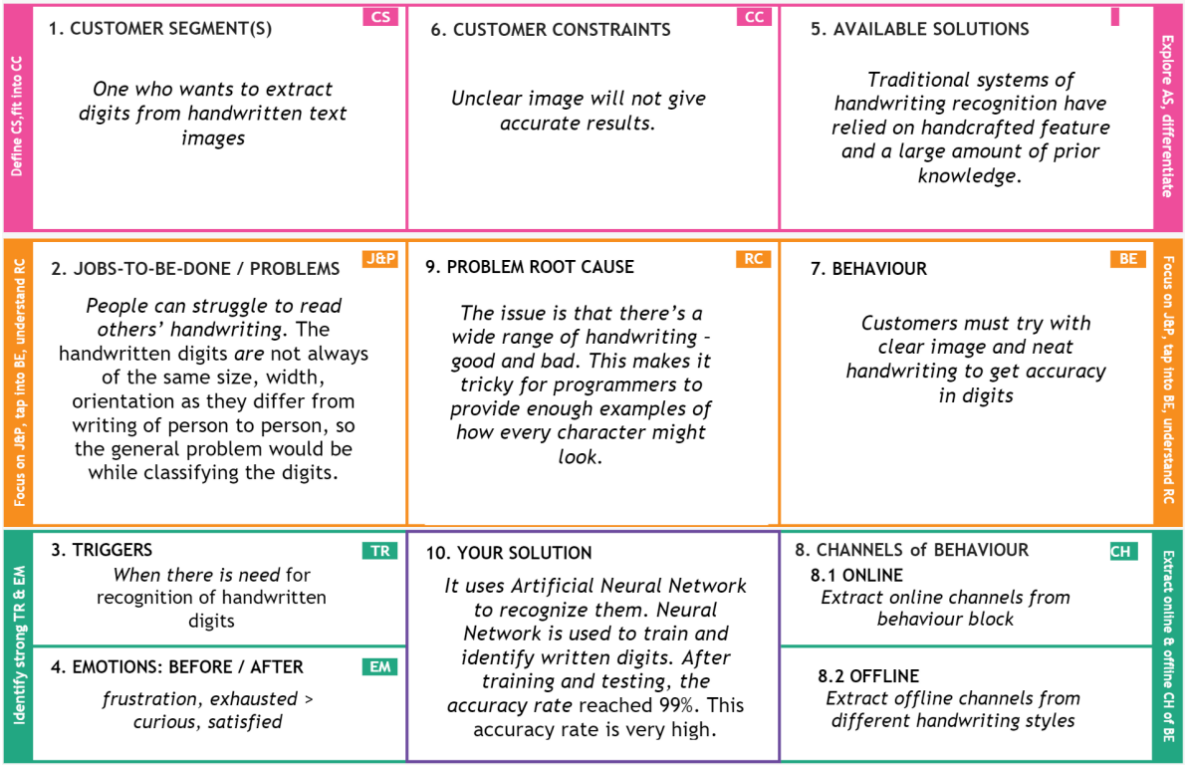
3.2.Ideation and Brainstorming

3.3.Proposed Solution

S.No	Parameter	Description
1	. Problem Statement (Problem to be solved)	The problem statement aims at developing a novel handwritten recognition system using ML. The handwritten digit recognition system is a way to tackle the problem which uses the image of a digit and recognizes the digit present in the image
2	Idea / Solution Description	Developing an AI predictive model to predict the handwritten digits and to construct a neural network with hidden layers and train to detect the digits.
3	Novelty / Uniqueness	The system not only produces a classification of the digit but also a rich description of the instantiation parameters which can yield information such as the writing style.
4	Social Impact / Customer Satisfaction	Handwritten digits can be recognized easily without any strenuous efforts. This reduces time and improves productivity for people.

5	Business Model (Revenue Model)	It is used in the detection of vehicle numbers, banks for reading cheques, post offices for arranging letters, and many other tasks.
6	Scalability of the Solution	To attain higher performances in the domain of character recognition and pattern recognition, due to its excellent feature extraction and working as best classifier characteristics. There is no limit in the number of digits that can be recognized.

3.4. Problem Solution Fit



Chapter 4: Requirement Analysis

4.1.Functional Requirements

Fr.No	Functional Requirements and Description
FR-1	**Image Data**: Handwritten digit recognition is the ability of a computer to recognize the human handwritten digits from different sources like images, papers, touch screens, etc, and classify them into 10 predefined classes (0-9). This has been a topic of boundless research in the field of deep learning. 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	**MNIST dataset**: The MNIST dataset is an acronym that stands for the Modified National Institute of Standards and Technology 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.

FR-5	Databases, software, virtual storage, and networking, among others. In layman's terms, Cloud Computing is defined as a virtual platform that **allows you to store and access your data over the internet without any limitations.**

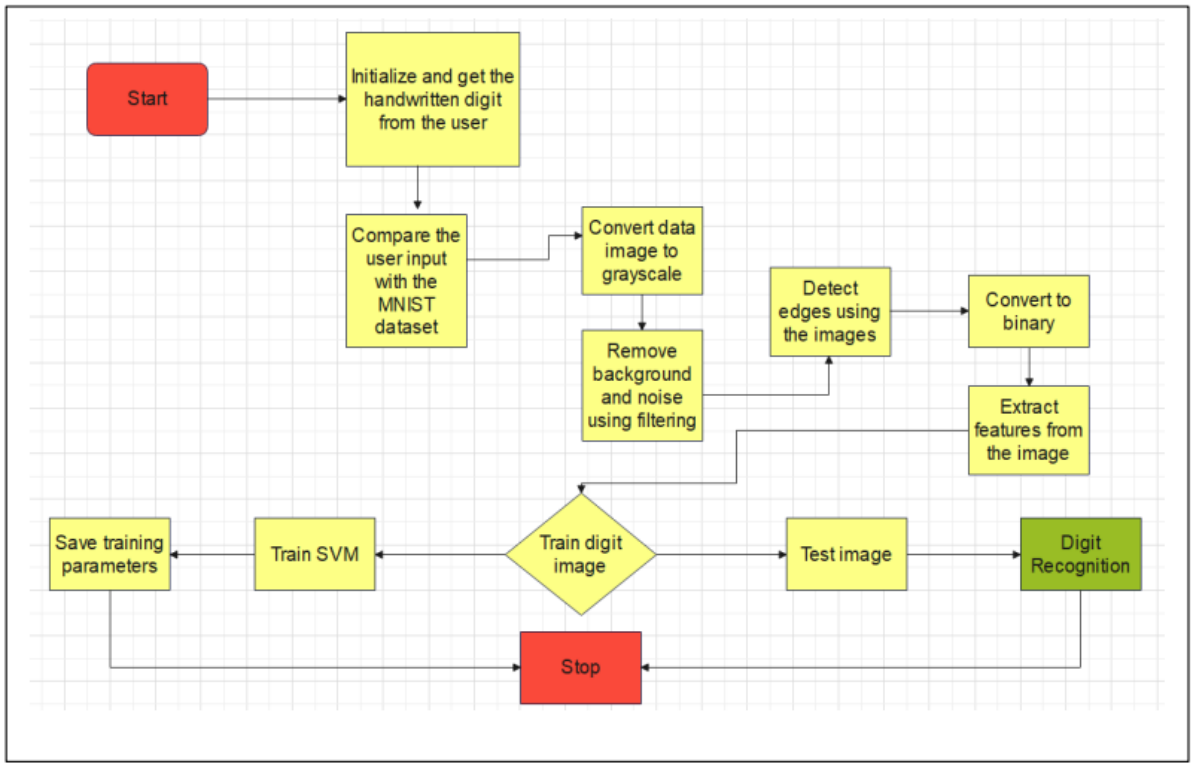
4.2. Non-Functional Requirements

NFR	**NON-FUNCTIONAL REQUIREMENTS**
NFR - 1	**Usability:** 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. 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	**Reliability:** 1) The system not only produces a classification of the digit but also a rich description of the instantiation parameters which can yield information such as the writing style. 2) The generative models can perform recognition driven segmentation. 3) The method involves a relative.

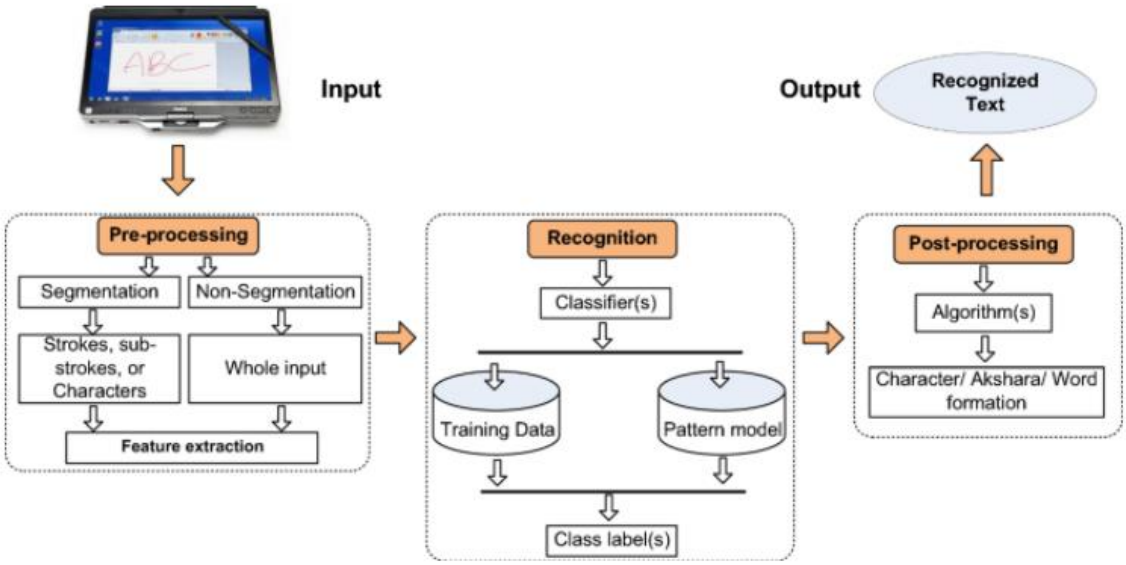
NFR - 3	**Accuracy:** Optical Character Recognition (OCR) technology provides higher than 99% accuracy with typed characters in high quality images. However, the diversity in human writing types, spacing differences, and irregularities of handwriting causes less accurate characterrecognition.
NFR - 4	**Performance:** The neural network uses the examples to automatically infer rules for recognizing handwritten digits. Furthermore, by increasing the number of training examples, the network can learn more about handwriting, and so improve its accuracy. There are a number of ways and algorithms to recognize handwritten digits, including Deep Learning/CNN, SVM, Gaussian Naive Bayes, KNN, Decision Trees,Random Forests, etc.

Chapter 5: Project Design

5.1.Data Flow Diagram



5.2.Technical Architecture



5.3.User Stories

User Type	Functional Requirements	User Story Number	User Story / Task	Acceptance Criteria	Priority	Release
Any common individual		USN-1	Receiving the digital form of the handwritten digits with a very high accuracy	Either write it on the webpage or scan the image of the written digit	High	Sprint-1
Customer (Web user)	Separate Registeration	USN-2	Helps in understanding the amount and account number entered in	Useful in characterizing the digits in banks	High	Sprint-2

			demand draft and cheques in banks			
	Home	USN-3	As a user, I can view the guide to use the web app	I can view the awareness of this application and its limitations.	Low	Sprint-1
		USN-4	As it is a web application , it is installation free	I can use it without the installation of the application or any software	Medium	Sprint-1
Any common person	Login	USN-5	As a user, I can log into the application by entering email & password		Low	Sprint-1

Chapter 6: Project Planning and Scheduling

6.1. Sprint Planning and Estimation

Sprint	Story ID	User Story / Task	Story Points	Priority	Team Members
Sprint – I	USN-1	Get the dataset	3	High	Deepakraj R
	USN-2	Explore the data	2	Medium	Deepakraj R Kaarthic Varun K
	USN-3	Data Pre-Processing	3	High	Jisho J Eshwanth G M
	USN-4	Prepare training and testing data	3	High	Deepakraj R Jisho J
Sprint – II	USN-5	Create the model	3	High	Deepakraj R Kaarthic Varun K
	USN-6	Train the model	3	High	Deepakraj R
	USN-7	Test the model	3	High	Kaarthic Varun K
Sprint – III	USN-8	Improve the model	2	Medium	Kaarthic Varun K Eshwanth G M
	USN-9	Save the model	3	High	Kaarthic Varun K Eshwanth G M
	USN–10	Build the Home Page	3	High	Kaarthic Varun K Jisho J
	USN–11	Setup a database to store input images	2	Medium	Jisho J Eshwanth G M

Sprint - IV	USN-12	Build the results page	3	High	Kaarthic Varun K Jisho J
	USN-13	Integrate the model with the application	3	High	Deepakraj R Jisho J
	USN-14	Test the application	3	High	Deepakraj R Eshwanth G M

6.2.Sprint Delivery Schedule

Sprint	**Total Story Points**	**Duration**	**Sprint Start Date**	**Sprint End Date (Planned)**	**Story Points Completed (As on Planned Date)**	**Sprint Release Date (Actual)**
Sprint – I	11	6 Days	24 Oct 2022	29 Oct 2022	11	24 Oct 2022
Sprint – II	9	6 Days	31 Oct 2022	05 Nov 2022	9	05 Nov 2022
Sprint – III	10	6 Days	07 Nov 2022	12 Nov 2022	10	12 Nov 2022
Sprint – IV	9	6 Days	14 Nov 2022	19 Nov 2022	9	19 Nov 2022

Chapter 7: Coding and Solution

Import the necessary packages

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

Load data

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

Data Analysis

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

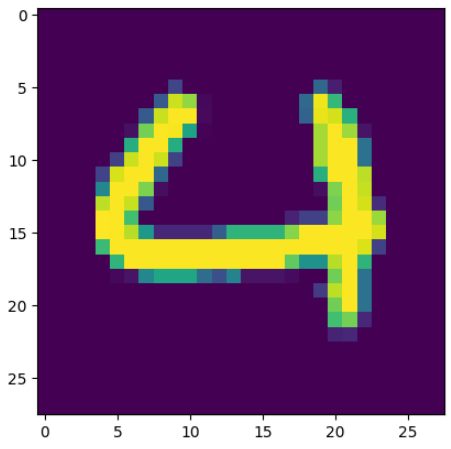
```
(60000, 28, 28)
(10000, 28, 28)
```

```
X_train[60]
```

```
y_train[60]
```

```
4
```

```
plt.imshow(X_train[60])
```



Data Pre-Processing

```
X_train = X_train.reshape(60000, 28, 28, 1).astype('float32')
X_test = X_test.reshape(10000, 28, 28, 1).astype('float32')
```

```
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)
```

```
Y_train[60]
```

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

Create model

```
model = Sequential()
model.add(Conv2D(64, (3, 3), input_shape=(28, 28, 1), activation="relu"))
model.add(Conv2D(32, (3, 3), activation="relu"))
model.add(Flatten())
model.add(Dense(number_of_classes, activation="softmax"))
```

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

Train the model

```
model.fit(X_train, Y_train, batch_size=32, epochs=5, validation_data=(X_test,Y_test))
```

Test the model

```
metrics = model.evaluate(X_test, Y_test, verbose=0)
print("Metrics (Test Loss & Test Accuracy): ")
print(metrics)
```

```
Metrics (Test Loss & Test Accuracy):
[0.11387308686971664, 0.9753000140190125]
```

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

```
print(np.argmax(prediction, axis=1))
print(Y_test[:4])
```

Save the model

```
model.save("model.h5")
```

Test the saved model

```
model=load_model("model.h5")
```

```
img = Image.open("sample_img.png").convert("L")
img = img.resize((28, 28))
img2arr = np.array(img)
img2arr = img2arr.reshape(1, 28, 28, 1)
results  = model.predict(img2arr)
results = np.argmax(results,axis = 1)
results = pd.Series(results,name="Label")
print(results)
```

Chapter 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 screen size 2560 x 1801 and 768 x 630	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 any file	FAIL

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	The size of the input image exceeds the display container	FAIL
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

Chapter 9: Results

9.1.Performance Metrics

1. Model Summary:

```
model.summary()

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

 conv2d_1 (Conv2D)           (None, 24, 24, 32)        18464

 flatten (Flatten)           (None, 18432)             0

 dense (Dense)               (None, 10)                184330

=================================================================
Total params: 203,434
Trainable params: 203,434
Non-trainable params: 0
_________________________________________________________________
```

2. Accuracy:

Content	Value (in percentage)
Training Accuracy	99.14
Training Loss	2.70
Validation Accuracy	97.76
Validation Loss	10.36

3. Classification report:

	precision	recall	f1-score	support
0	1.00	0.97	0.98	980
1	0.99	0.99	0.99	1135
2	0.96	0.99	0.97	1032
3	0.97	1.00	0.98	1010
4	1.00	0.95	0.98	982
5	0.96	1.00	0.98	892
6	0.99	0.96	0.97	958
7	0.99	0.98	0.99	1028
8	0.99	0.99	0.99	974
9	0.97	0.99	0.98	1009
accuracy			0.98	10000
macro avg	0.98	0.98	0.98	10000
weighted avg	0.98	0.98	0.98	10000

Chapter 10: Advantages and Disadvantages

10.1. Advantages

- Reduces manual work
- More accurate than average human
- Capable of handling a lot of data
- Can be used anywhere from any device

10.2. Disadvantages

- Cannot handle complex data
- All the data must be in digital format
- Requires a high-performance server for faster predictions
- Prone to occasional errors

Chapter 11: Conclusion

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

Chapter 12: Future Scope

There is still much work to be done on this project, and it may use a lot of improvement. The following are a few ways this project could be improved:

- Include the ability to save the results of multiple image detection from digits.
- Include a feature to recognize multiple digits.
- Enhance the model to recognize numbers in complicated images
- Include support for more languages to assist users from around the world.

This project has limitless potential and may constantly be improved. Putting this idea into practice will help a variety of sectors and lower the workloads of numerous employees, increasing overall workplace productivity.

Appendix

Source Code

recognizer.py

```
import os
import random
import string
from pathlib import Path
import numpy as np
from tensorflow.keras.models import load_model
from PIL import Image, ImageOps


def random_name_generator(n):
        return ''.join(random.choices(string.ascii_uppercase + string.digits, k=n))

def recognize(image):
        model=load_model(Path("./model/model.h5"))

        img = Image.open(image).convert("L")
        img_name = random_name_generator(10) + '.jpg'

        if not os.path.exists(f"./static/data/"):
                os.mkdir(os.path.join('./static/', 'data'))
        img.save(Path(f"./static/data/{img_name}"))

        img = ImageOps.grayscale(img)
        img = ImageOps.invert(img)
        img = img.resize((28, 28))

        img2arr = np.array(img)
        img2arr = img2arr / 255.0
        img2arr = img2arr.reshape(1, 28, 28, 1)

        results  = model.predict(img2arr)
        best = np.argmax(results,axis = 1)[0]

        pred = list(map(lambda x: round(x*100, 2), results[0]))

        values = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
        others = list(zip(values, pred))

        best = others.pop(best)

        return best, others, img_name
```

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_name)


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

Github

https://github.com/IBM-EPBL/IBM-Project-35185-1660282266

Project Demo

https://drive.google.com/file/d/1hfQzL6ZGESYE1_lw729in8HM5qKhhCbd/view?usp=share_link