PROJECT REPORT

A NOVEL METHOD FOR HANDWRITTEN DIGIT RECOGNITION SYSTEM

Submitted by

PNT2022TMID35498

Srinivasan Babu	**2019105581**
Hariharan C A	**2019105012**
Navaneethakrishnan R	**2019105551**
Rakshith T N	**2019105561**

TABLE OF CONTENTS

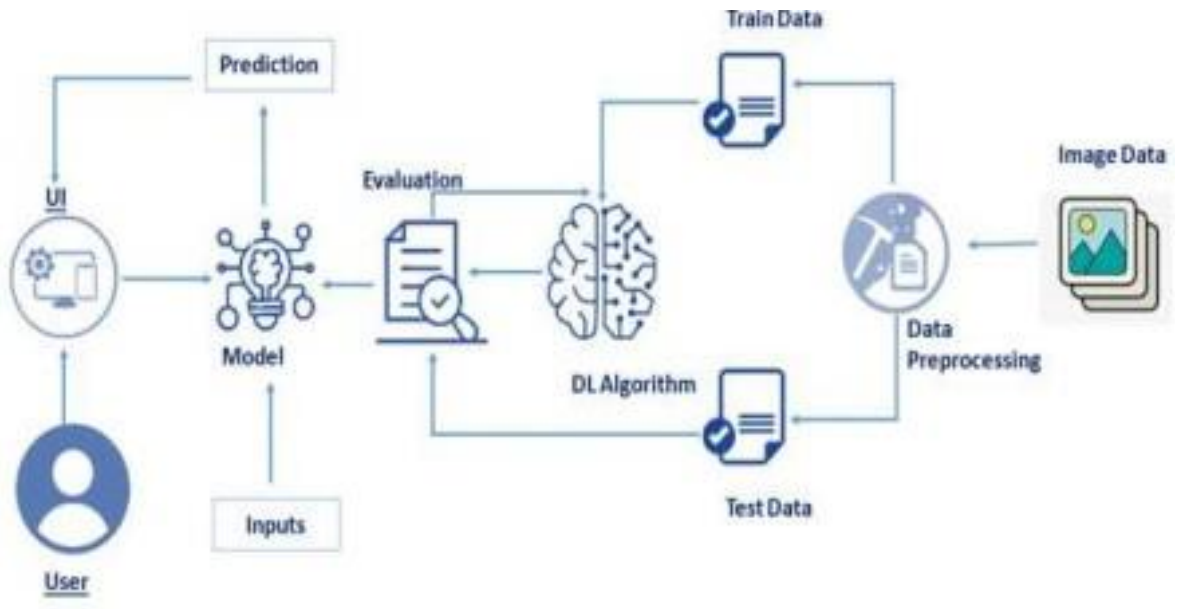
GITHUB & PROJECT DEMO LINK

INTRODUCTION

1.1 PROJECT OVERVIEW

Handwriting recognition is one of the compelling research works going on because every individual in this world has their own style of writing. It is the capability of the computer to identify and understand handwritten digits or characters automatically. Because of the progress in the field of science and technology, everything is being digitalized to reduce human effort. Hence, there comes a need for handwritten digit recognition in many real-time applications. MNIST data set is widely used for this recognition process and it has 70000 handwritten digits. We use Artificial neural networks to train these images and build a deep learning model. Web application is created where the user can upload an image of a handwritten digit. this image is analyzed by the model and the detected result is returned on to UI

Technical Architecture:



1.2 PURPOSE

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

LITERATURE SURVEY

2.1 EXISTING PROBLEM:

The issue is that there's a wide range of handwriting – good and bad. This makes it tricky for programmers to provide enough examples of how every character might look. Sometimes, characters look very similar, making it hard for a computer to recognise accurately. The fundamental problem with handwritten digit recognition is that handwritten digits do not always have the same size, width, orientation, and margins since they vary from person to person. Additionally, there would be issues with identifying the numbers because of similarities between numerals like 1 and 7, 5 and 6, 3 and 8, 2 and 5, 2 and 7, etc. Finally, the individuality and variation of each individual's handwriting influence the structure and appearance of the digits.

2.2 REFERENCES:

R. Bajaj, L. Dey, S. Chaudhari et al, employed three different kinds of features, namely, the density features, moment features and descriptive component features for classification of Devanagari Numerals. They proposed multi classifier connectionist architecture for increasing the recognition reliability and they obtained 89.6% accuracy for handwritten Devanagari numerals.

Salvador España-Boquera et al, in this paper hybrid Hidden Markov Model (HMM)

model is proposed for recognizing unconstrained offline handwritten texts. In this, the structural part of the optical model has been modelled with Markov chains, and a Multilayer Perceptron is used to estimate the emission probabilities. In this paper, different techniques are applied to remove slope and slant from handwritten text and to normalize the size of text images with supervised learning methods. The key features of this recognition system were to develop a system having high accuracy in pre processing and recognition, which are both based on ANNs.

Yoshimasa Kimura presented a work on how to select features for Character Recognition Using Genetic Algorithms. The author proposes a novel method of feature selection for character recognition using genetic algorithms (GA). The proposed method selects only the genes for which the recognition rate of training samples exceeds the predetermined threshold as a candidate of the parent gene and adopts a reduction ratio in the number of features used for recognition as the fitness value.

Renata F. P. Neves has proposed SVM based offline handwritten digit recognition. Authors claim that SVM outperforms the Multilayer perceptron classifier. Experiment is carried out on NIST SD19 standard dataset. Advantage of MLP is that it is able to segment non-linearly separable classes. However, MLP can easily fall into a region of local minimum, where the training will stop assuming it has achieved an optimal point in the error surface. Another hindrance is defining the best network architecture to solve the problem, considering the number of layers and the amount of perceptron in each hidden layer. Because of these disadvantages, a digit recognizer using the MLP structure may not produce the desired low error rate.

M. Hanmandlu, O.V. Ramana Murthy has presented in their study the recognition of handwritten Hindi and English numerals by representing them in the form of exponential membership functions which serve as a fuzzy model. The recognition is carried out by modifying the exponential membership functions fitted to the fuzzy sets. These fuzzy sets are derived from features consisting of normalized distances obtained using the Box approach. The membership function is modified by two structural parameters that are estimated by optimizing the entropy subject to the attainment of membership function to unity. The overall recognition rate is found to be 95% for Hindi numerals and 98.4% for English numerals.

Ragha & Sasikumar describes a system for Kannada characters. In this paper, the moment features are extracted from the Gabor wavelets of pre-processed images of 49 characters. The comparison of moments features of 4 directional images with original images are tested on Multi-Layer Perceptron with Back Propagation Neural Network. The average performance of the system with these two features together is 92%.

Aparna et al, proposed a method to construct a handwritten Tamil character by executing a sequence of strokes. A structure or shape-based representation of a stroke was used in which a stroke was represented as a string of shape features. Using this string representation, an unknown stroke was identified by comparing it with a database of strokes using a flexible string-matching procedure.

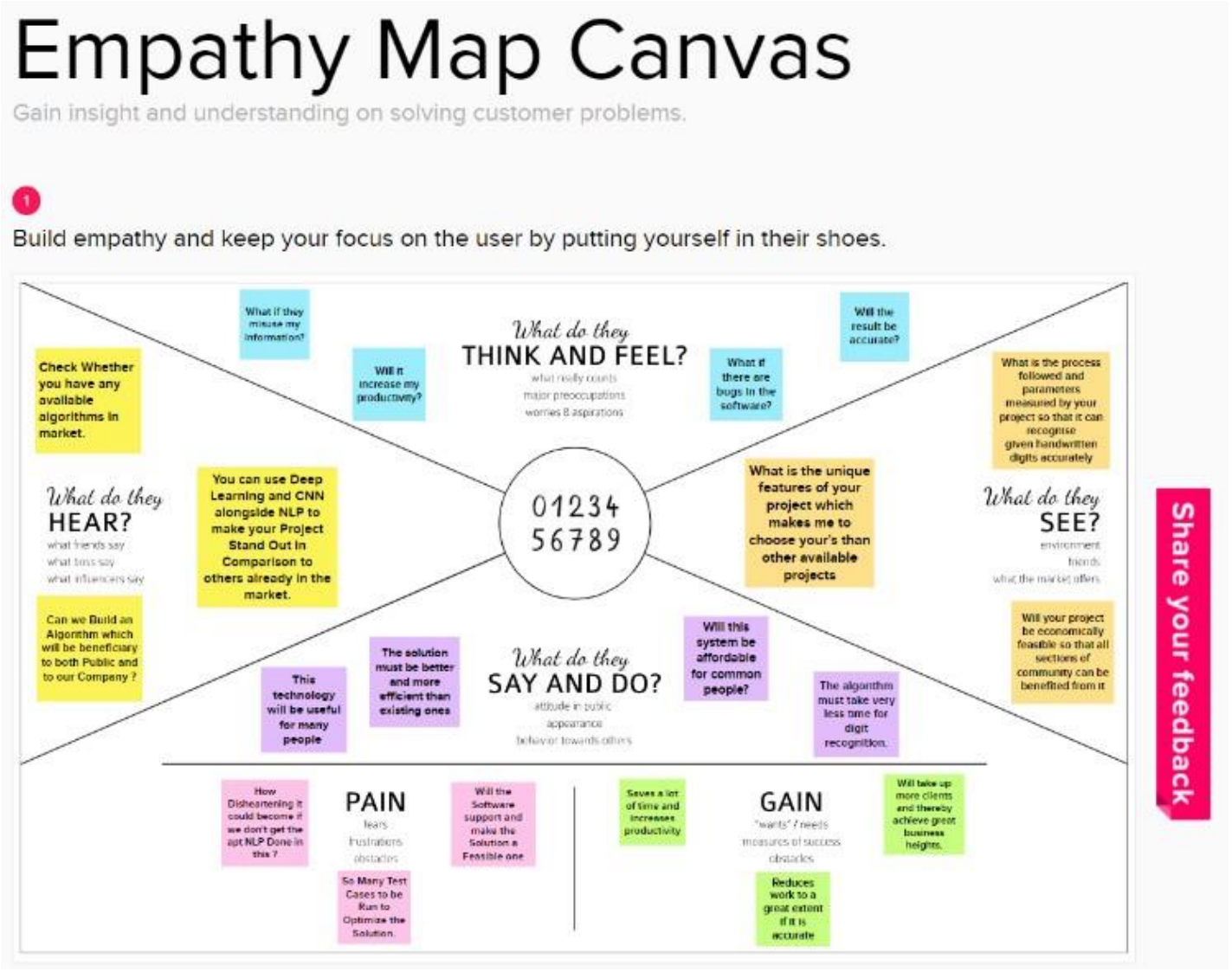
2.3 PROBLEM STATEMENT DEFINITION:

Arun is an employee at the postal office who's job is to segregate mail with different handwritten pin code or postal address quickly and efficiently so that every mail is sent to their correct address. So, a novel method for Handwritten Digit Recognition System needs to be developed.

QUESTION	DESCRIPTION
Who Does the Problem Affect?	It affects from small occupations to big officials in understanding any important code in digits.
What Are the Boundaries of The Problem?	The Boundaries are Size, width, style, orientation of the written digit.
What is the Issue?	Early Accurate Detection of the Important Handwritten digits (Ex: Pin Code) may save huge losses which may be incurred if detected wrong.
When does the issue occur?	It occurs when the handwriting of the writer of the Digit is not understood by common man.
Where is the issue occurring?	It ranges from small occupations to big industries.
Why is it important that we fix the problem?	It improves detection of the digits and unreliable things such as Guesswork, Prediction can be overcome.

IDEATION & PROPOSED SOLUTION

3.1 EMPATHY MAP CANVAS:



3.2 IDEATION & BRAINSTORMING:

3.3 PROPOSED SOLUTION:

S.NO	Parameter	Description
1.	Problem Statement (Problem to be solved)	**Statement**-The handwritten digit recognition is the capability of computer applications to recognize the human handwritten digits. **Description**: It is a hard task for the machine because handwritten digits are not perfect and can be made with many different shapes and sizes.
2.	Idea / Solution description	1. It is the capability of a computer to fete the mortal handwritten integers from different sources like images, papers, touch defenses. 2. It allows users to translate all those signatures and notes into electronic words in a text document format and this data only requires far less physical space than the storage of the physical copies.
3.	Novelty / Uniqueness	Accurately recognize the digits rather than recognizing all the characters like OCR.

4.	Social Impact / Customer Satisfaction	1.Artificial Intelligence developed the app called Handwritten digit Recognizer. 2. It converts the written word into digital approximations and utilizes complex algorithms to identify characters before churning out a digital approximation.
5.	Business Model (Revenue Model)	• This system can be integrated with traffic surveillance cameras to recognize the vehicle's number plates for effective traffic management.
		• Can be integrated with the Postal system to identify and recognize the pin-code details easily.
6.	Scalability of the Solution	• Ability to recognise digits in more noisy environments. • There is no limit in the number of digits it can be recognized.

3.4 PROBLEM SOLUTION FIT:

Team ID: PNT2022TMID35498 PROBLEM - SOLUTION FIT: PROJECT NAME:

A NOVEL METHOD FOR HANDWRITTEN DIGIT RECOGNITION SYSTEM

1.CUSTOMER SEGMENT

MNIST is a dataset which is widely used for handwritten digit recognition system

6. CUSTOMER CONSTRAINT

They believe that the alternatives will result in errors and faults and will be inconvenient.

2.JOBS-TO-BE-DONE/PROBLEMS:

Handwritten digits can be difficult to understand and interpret at times. It may cause errors when dealing with rough handwriting

7. BEHAVIOUR

Find a right product that recognizes the digits written in all kinds of handwriting accurately and fast

3.TRIGGERS TO ACT

The dataset consistof 60,000 trainingimages and 10,000 test images

8.CHANNELS of BEHAVIOR

The ability of a computer to recgnize the human handwritten digits image,paper, touchscreen

4. EMOTIONS: BEFORE / AFTER

Time consuming ,Manual effort ,Irritated

After: Fast, Less manual effort , Happy customers

9.PROBLEM ROOT CAUSE

Wide variety of writing styles used by different people.

5.AVAILABLE SOLUTIONS

The applications of digit recognition include in postal mail sorting,bank check process

10. YOUR SOLUTION: A solution to this problem is the Handwritten digit recognition system, which uses a picture of a digit and recognises the digit present in the image. Convolutional Neural Network model built with PyTorch and applied to the MNIST dataset to

REQUIREMENT ANALYSIS

4.1 FUNCTIONAL REQUIREMENTS:

FR No.	Functional Requirement	Description
FR-1	Website	• A website having a login feature, where each user will have to register and then he/she will be able to login using his/her username and password.
FR-2	Upload Image	• Must be able to take the handwritten inputs in the form of the images. (JPG or PNG)
FR-3	Image correlation	• Image correlation is a technique used to recognize characters from images. Collecting data and prepare it for training and testing.
FR-4	Feature extraction	• Feature extraction is analyzing the images and deriving some characteristics from these images that identify each element.
FR-5	Output	• System should retrieve characters present in the image and display them to the user. • System must be able to display the accurate output in text format.

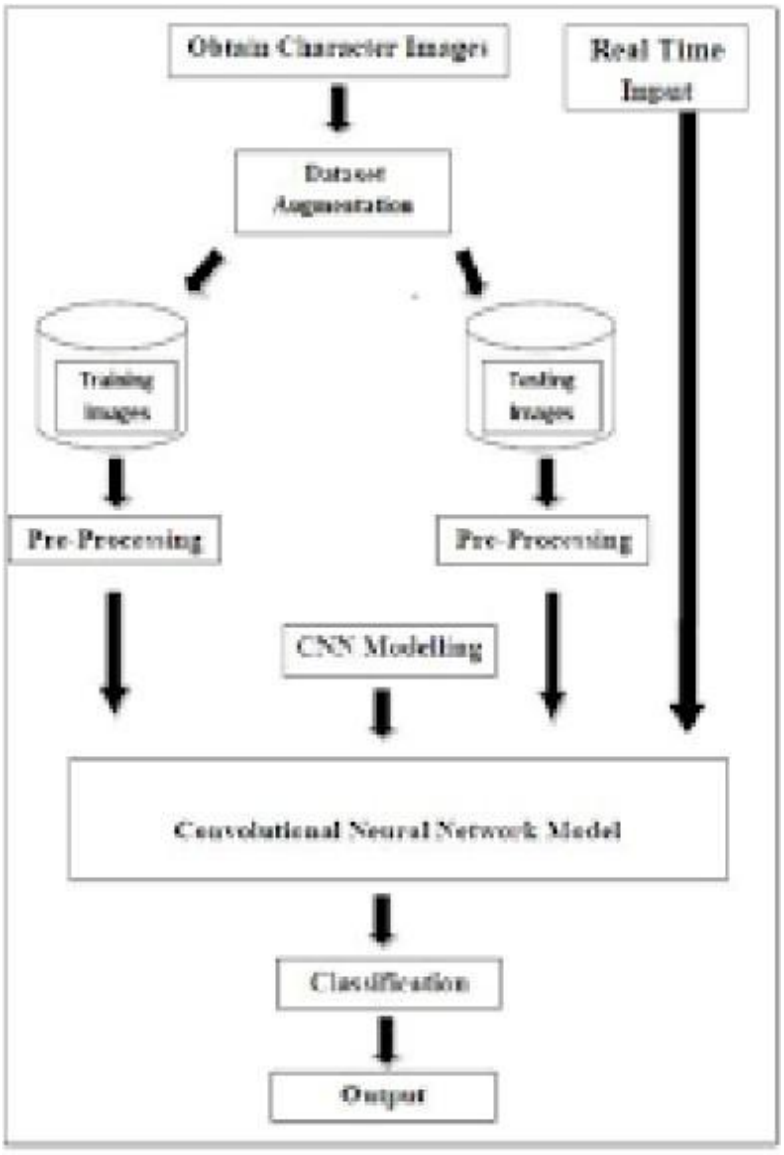
4.2 NON-FUNCTIONAL REQUIREMENTS:

FR No.	Non-Functional Requirement	Description
NFR-1	Usability	• Application for digit recognition include filling out forms, processing bank cheque , and sorting mail. • Should be easy to use for everyone.
NFR-2	Security	• As it will be used in the banking sector, it should be able to store the cheque details securely. • This will be done by authenticating the users using their username and password.
NFR-3	Reliability	• This software should work reliably for low resolution image and should not display any errors.
NFR-4	Performance	• The software should be responsive and provide output quickly even for complex handwriting.
NFR-5	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 inequalities of handwriting causes less accurate character recognition.
NFR-6	Scalability	• Large numbers of users can access the digits at any time without restrictions.

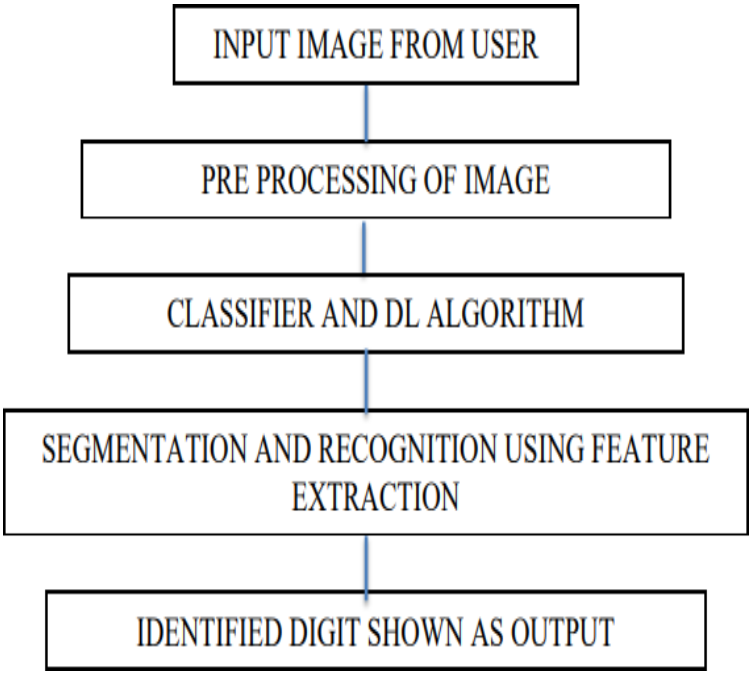
PROJECT DESIGN

5.1 DATA FLOW DIAGRAMS:

A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It shows how data enters and leaves the system, what changes the information, and where data is stored.


Obtain Character Images
Real Time Input
Dataset Augmentation
Training Images
Testing Images
Pre-Processing
Pre-Processing
CNN Modelling
Convolutional Neural Network Model
Classification
Output

5.2 SOLUTION & TECHNICAL ARCHITECTURE:



DATASET:

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 picture. The 60,000 training and 10,000 testing labeled handwritten digit images in the MNIST Handwritten Digit Recognition Dataset.

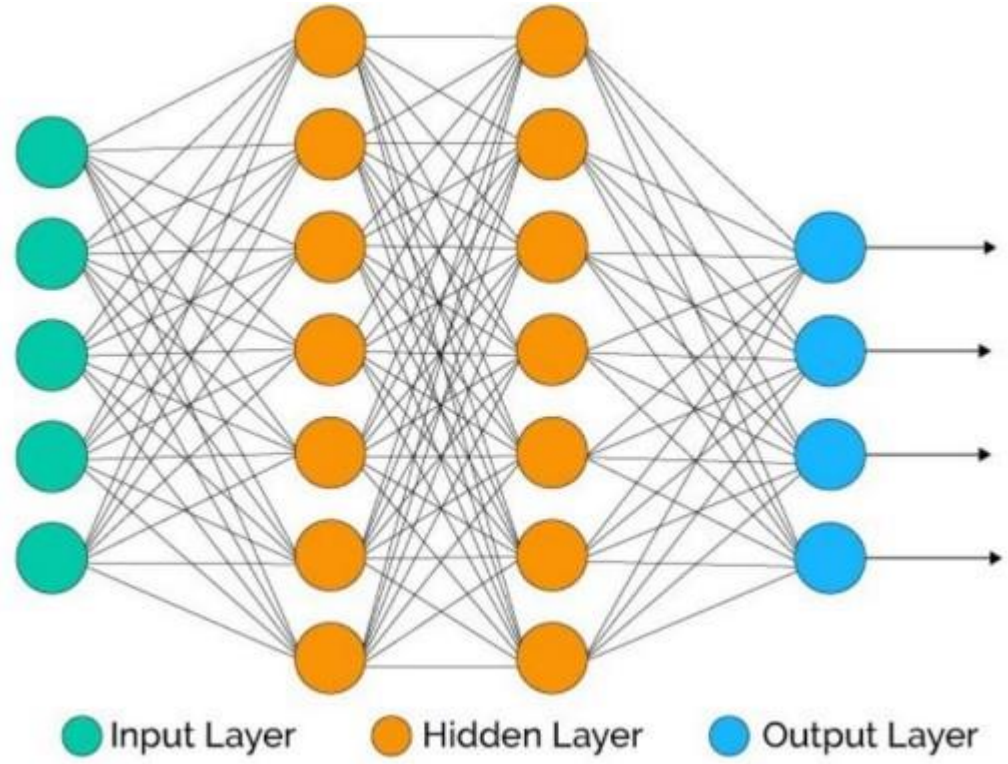
APPROACH:

This project will be approached utilizing a convolutional neural network. (CNN)

- **The convolutional and max-pool layer**: The feature is extracted from the input image and got max pooled to reduce the dimensionality of the image without any changes in the extracted feature.

- **The dense layer**: The flattened output from the max-pool layer is fed to a feed forward neural network and backpropagation applied to every iteration of training.

- **The output layer**: The nodes in this stratum are referred to as output units. It gives us access to the neural network's final prediction, which may be used to make final predictions.



A neural network is a model of the brain's operations. It is made up of numerous layers with a variety of activations; these activations mimic the neurons in our brain. An attempt is made by a neural network to learn a set of parameters from a set of data that might aid in understanding the underlying relationships. Since neural networks are capable of adapting to changing input, the network can produce the best outcome without having to change the output criterion.

METHODOLOGY:

A neural network with one hidden layer and 100 activation units has been put into practice (excluding bias units). The features (X) and labels (Y) were retrieved after

the data was loaded from a mat file. To prevent overflow during computation, features are then scaled into a range of [0,1] by dividing by 255. 10,000 testing cases and 60,000 training examples make up the data. With the training data, feedforward is used to calculate the hypothesis, and backpropagation is then used to lower the error between the layers. To combat overfitting, the regularization parameter lambda is set to 0.1. To identify the model that fits the situation the optimizer runs for 70 times. WORKING: After receiving an input, neural networks change it using a number of hidden layers. Each group of neurons in a hidden layer is completely linked to every other neuron in the layer above it. One layer of neurons have perfect independence from one anotherr. The “output layer” is the final layer to be fully connected. CONVOLUTION

LAYER : The foundational component of a CNN is the convolutional layer. The parameters of the layer are a set of learnable filters (or kernels) that cover the entire depth of the input volume but have a narrow receptive field. Each filter is convolved across the width and height of the input volume during the forward pass, computing the dot product between each filter entry and the input to create a two-dimensional activation map of the filter. As a result, the network picks up filters that turn on when it detects a certain kind of feature at a particular spatial location in the input.

FEATURE EXTRACTION : All neurons in a feature share the same weights .In this way all neurons detect the same feature at different positions in the input image. Reduce the number of free parameters.

TENSORFLOW : An open-source machine learning library for both research and production is called TensorFlow. TensorFlow provides developers of all skill levels with APIs for desktop, mobile, web, and cloud applications. We can achieve text output and sound output by scanning the number digit and converting it to PNG format using the python3 command in the terminal.

5.3 USER STORIES:

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobile user)	Home	USN-1	As a user, I can view the guide touse this application.	I can be aware to use this application and its limitations.	Low	Sprint-1
		USN-2	As a user, I'm allowed to view the demonstration video to use the interface of this application.	I can gain knowledge to use this application by a practical method.	Low	Sprint-1
		USN-3	As a user, I am allowed read the instructions to use thisapplication.	I can read instructions to use it in a user-friendly method.	Low	Sprint-2
	Recognize	USN-4	As a user, in the prediction page I can choose the image.	I can choose the image from the local system and predict the output.	High	Sprint-2
	Predict	USN-6	As a user, I'm able to upload and choose the image to be uploaded.	I can upload and choose the image from the system storage and also in any virtual storage.	Medium	Sprint-3
		USN-7	As a user, I will train and test the input to get the maximum accuracy of output.	I can train and test the application until it gets maximum accuracy of the result.	High	Sprint-4
		USN-8	As a user, I will access the MNIST data set	I can access the MNIST data set to produce the accurate result.	Medium	Sprint-3
Customer (Web user)	Home	USN-9	As a user, I am allowed to view the guide to use the web app.	I can be aware to use this application and its limitations.	Low	Sprint-1

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobile user)	Home	USN-1	As a user, I am able view the guide and awareness touse this application.	I can view the awareness to use this application and its limitations.	Low	Sprint-1
		USN-2	As a user, I'm allowed to view the guided video to use the interface of this application.	I can gain knowledge to use this application by a practical method.	Low	Sprint-1
		USN-3	As a user, I am able read the instructions to use thisapplication.	I can read instructions also to use it in a user-friendly method.	Low	Sprint-2
	Recognize	USN-10	As a user, I am allowed use the web application virtually anywhere.	I can use the application portably anywhere.	High	Sprint-1
		USN-11	As it is an open source, can use it cost freely.	I can use it without any payment to be paid for it to access.	Medium	Sprint-2
		USN-12	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-4
	Predict	USN-13	As a user, I'm allowed to upload and choose the image to be uploaded.	I can upload and choose the image from the system storage and also in any virtual storage.	Medium	Sprint-3

PROJECT PLANNING & SCHEDULING

6.1 SPRINT PLANNING & ESTIMATION:

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Data Collection	USN-1	As a user, I can collect the dataset from various resources with different handwriting.	10	Low	Srinivasan B Hariharan CA
Sprint-1	Data Preprocessing	USN-2	As a user, I can load the dataset, handle the missing data and split data into train and test.	10	Medium	Rakshith TN Navaneethakrishnan R
Sprint-2	Model Building	USN-3	As a user, I will get an application with an ML model which provides high accuracy of recognized handwritten digits.	5	High	Srinivasan B Hariharan CA Navaneethakrishnan R
Sprint-2	Add CNN layers	USN-4	Creating the model and adding the input, hidden and output layers to it.	5	High	Srinivasan B Hariharan CA
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	Hariharan CA
Sprint-2	Train & test the model	USN-6	As a user, we will train our model with our image dataset.	6	Medium	Srinivasan B Hariharan CA Rakshith TN
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	Hariharan CA Navaneethakrishnan R
Sprint-3	Building UI Application	USN-8	As a user, I will upload the handwritten digit image to the application by clicking an upload button.	5	High	Srinivasan B Hariharan CA Navaneethakrishnan R
Sprint-3		USN-9	As a user, I can know the details of the fundamental usage of the application.	5	Low	Rakshith TN
Sprint-3		USN-10	As a user, I can see the predicted / recognized digits in the application.	5	Medium	Hariharan CA Rakshith TN
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 points.	10	High	Srinivasan B Hariharan CA
Sprint-4	Cloud Deployment	USN-12	As a user, I can access the web application and make use of the product from anywhere.	10	High	Navaneethakrishnan R Rakshith TN

Project Tracker, Velocity & Burndown Chart:

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	20	6 Days	24 Oct 2022	29 Oct 2022	20	29 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022	20	05 Nov 2022
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022	20	12 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022	20	19 Nov 2022

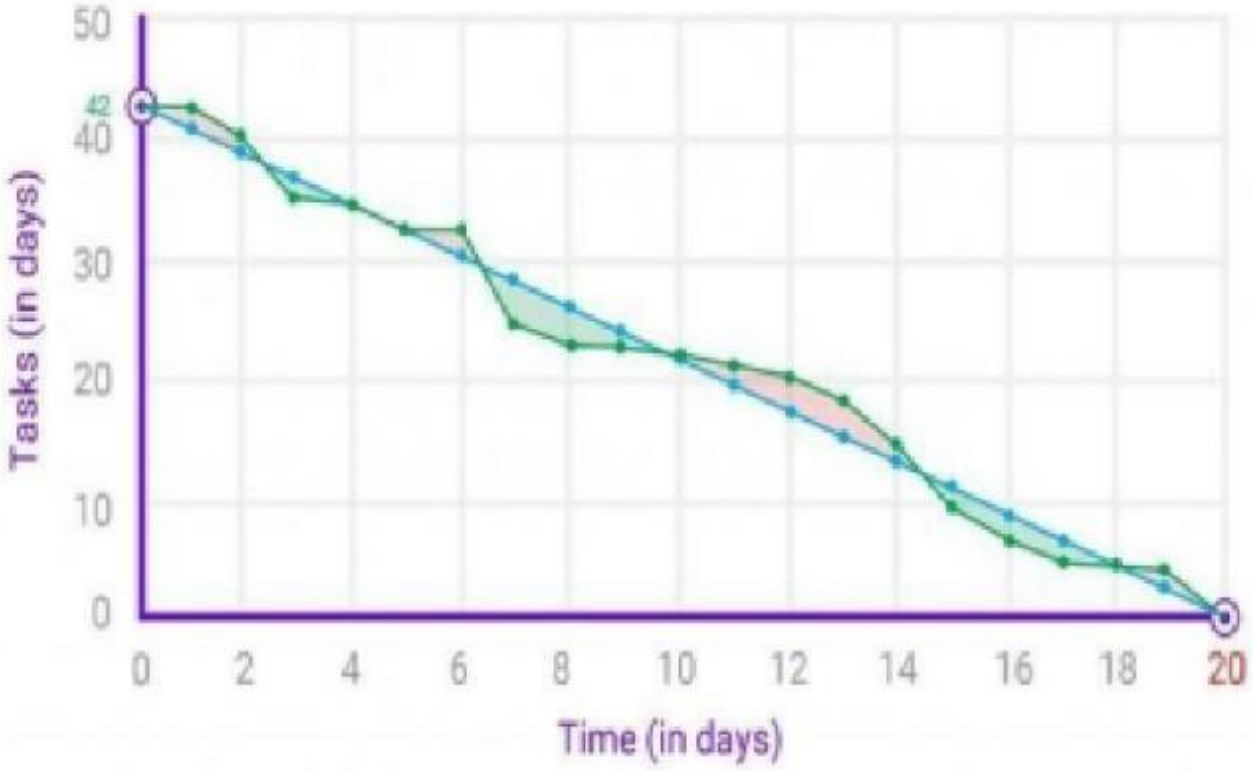
Velocity:

Imagine we have a 10-day sprint duration, and the velocity of the team is 20 (points per sprint). Let's calculate the team's average velocity (AV) per iteration unit (story points per day).

$$\text{Average Velocity} = 20 / 6 = 3.33$$

Burndown Chart:

A burn down chart is a graphical representation of work left to do versus time. It is often used in agile software development methodologies such as Scrum. However, burn down charts can be applied to any project containing measurable progress over time.


50
45
40
30
20
10
0
Tasks (in days)
0
2
4
6
8
10
12
14
16
18
20
Time (in days)

6.2 SPRINT DELIVERY SCHEDULE:

S.No.	Milestones	Activities	Timeline
1	Literature Survey	Literature survey on the handwritten digit recognition system and information gathering.	10 September 2022
2.	Empathy Map	Prepared empathy map canvas to capture the user gains and pains.	10 September 2022
3.	Brainstorming and Ideation	Ideas are listed and top 3 ideas are prioritized based on the feasibility and importance.	17 September 2022
4.	Proposed Solution Document	Proposed solution document is prepared which includes Novelty, feasibility of idea, social impact, scalability of solution, etc.	24 September 2022
5.	Problem Solution Fit	Includes customer segments and customer constraints, the problem root cause and jobs to be done.	1 October 2022
6.	Solution Architecture	From data collection to digit recognition by the web application are represented in architectural diagram.	1 October 2022
7.	Customer Journey Map	Prepare Customer Journey maps to understand user interactions and	8 October 2022

		experiences with the application	
8.	Functional Requirement Document	Functional requirements and non functional requirements alike scalability and accuracy are described.	15 October 2022
9.	Data Flow Diagrams	Data flow diagram and user stories are prepared and four sprint phases are described.	15 October 2022
10.	Technology Architecture	Technical flow graphs are created and the functions of technical stacks are defined.	15 October 2022

CODING AND SOLUTIONING

7.1 FEATURE 1:

Importing Required Libraries:

```
In [1]: import numpy as np
        import pandas as pd
        import random
        import tensorflow as tf
        import keras
        from keras import layers
        keras.backend.set_image_data_format('channels_last')
        import matplotlib.pyplot as plt
        from sklearn.model_selection import train_test_split
        from sklearn.metrics import confusion_matrix
        from keras.utils.np_utils import to_categorical

        print(keras.__version__)
        print(tf.__version__)

        2.10.0
        2.10.0
```

Here, we are importing required libraries like numpy, pandas, tensorflow and keras.

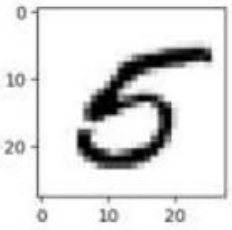
Data Preprocessing:

DATA PREPROCESSING

In [13]:
```
X_train = X_train/255.0
X_test = X_test/255.0

X_train = X_train.values.reshape(-1, 28, 28, 1)
X_test = X_test.values.reshape(-1, 28, 28, 1)


fig, ax = plt.subplots(figsize=(2,2))
plt.imshow(X_train[random.randint(0,len(X_train))], cmap='Greys')
plt.show()
```



TRAIN AND TEST SPLIT

In [14]:
```
validation_size = 0.2
X_train, X_val, Y_train, Y_val = train_test_split(X_train, Y_train, test_size = validation_size)
```

ONE HOT ENCODING

In [15]:
```
Y_train = keras.utils.np_utils.to_categorical(Y_train, num_classes = 10)
Y_val = keras.utils.np_utils.to_categorical(Y_val, num_classes = 10)
```

Model Building:

MODEL BUILDING

In [75]:
```
def Conv_Neural_Net():
    model = keras.Sequential()
    model.add(layers.Conv2D(32, kernel_size=(3,3), input_shape=(28,28,1)))
    model.add(layers.BatchNormalization(name='bn_1'))
    model.add(layers.Activation('relu', name='relu_1'))
    model.add(layers.MaxPooling2D(pool_size=(2, 2)))
    model.add(layers.Conv2D(64, kernel_size=(3,3)))
    model.add(layers.MaxPooling2D(pool_size=(2, 2)))
    model.add(layers.Flatten()) # Flattening the 2D arrays for fully connected layers
    model.add(layers.Dense(128, activation=tf.nn.relu))
    model.add(layers.Dropout(0.2))
    model.add(layers.Dense(10,activation=tf.nn.softmax))
    return model
```

In [76]:
```
model = Conv_Neural_Net()
```

COMPILING THE MODEL

In [77]:
```
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])
```

7.2 FEATURE 2:

Training the Model:

TRAINING THE MODEL

```
In [125]: model.fit(X_train, Y_train, validation_data = (X_val, Y_val), epochs = 50)
```

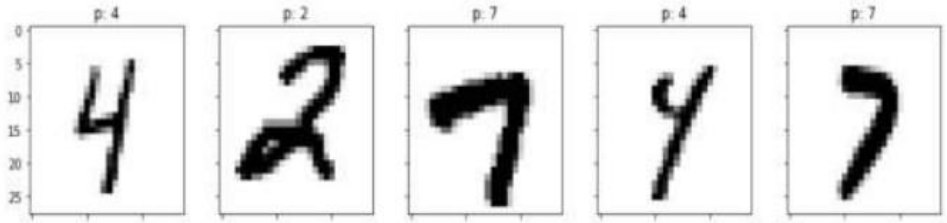
```
Epoch 1/50
1050/1050 [==============================] - 22s 21ms/step - loss: 0.0086 - accuracy: 0.9974 - val_loss: 0.0939 - val_accurac
y: 0.9882
Epoch 2/50
1050/1050 [==============================] - 22s 21ms/step - loss: 0.0069 - accuracy: 0.9978 - val_loss: 0.0968 - val_accurac
y: 0.9875
Epoch 3/50
1050/1050 [==============================] - 22s 21ms/step - loss: 0.0056 - accuracy: 0.9985 - val_loss: 0.0934 - val_accurac
y: 0.9874
Epoch 4/50
1050/1050 [==============================] - 22s 21ms/step - loss: 0.0068 - accuracy: 0.9979 - val_loss: 0.0835 - val_accurac
y: 0.9869
Epoch 5/50
1050/1050 [==============================] - 22s 21ms/step - loss: 0.0055 - accuracy: 0.9982 - val_loss: 0.0982 - val_accurac
y: 0.9885
Epoch 6/50
1050/1050 [==============================] - 22s 21ms/step - loss: 0.0071 - accuracy: 0.9979 - val_loss: 0.1012 - val_accurac
y: 0.9873
Epoch 7/50
```

Evaluating the Model and Predicting:

TESTING THE MODEL

```
In [126]: def predict(model, X, start,end):
              s = int(np.sqrt(end-start))
              fig, ax = plt.subplots(s, s, sharex=True, sharey=True, figsize=(15, 15))
              ax = ax.flatten()
              preds = model.predict(X[start:end])
              for i in range(end-start):
                  y_pred = np.argmax(preds[i])
                  img = X[start+i].reshape(28, 28)
                  ax[i].imshow(img, cmap='Greys', interpolation='nearest')
                  ax[i].set_title(f'p: {y_pred}')
```

```
In [127]: predict(model,X_test,25,50)
```



Cloud Deployment:

```
hine-learning-client) (1.20.3)

In [132]: from ibm_watson_machine_learning import APIClient
          wml_credentials={
              "url":"https://us-south.ml.cloud.ibm.com",
              "apikey":"8SQKHqX1NPhcxGUbqhEx-vm90X2ZKd7-eY0BQ8pqUUUV"
          }
          client=APIClient(wml_credentials)

14/]:
import os, types
import pandas as pd
from botocore.client import Config
import ibm_boto3

def __iter__(self): return 0

# @hidden_cell
# The following code accesses a file in your IBM Cloud Object Storage. It includes your credentials.
# You might want to remove those credentials before you share the notebook.
cos client = ibm boto3.client(service name='s3',

.48]: img = Image.open(streaming_body_3).convert("L")
      img = img.resize( (28,28) )

.49]: img

.49]:
```

```
.50]: im2arr = np.array(img) #converting to image
      im2arr = im2arr.reshape(1, 28, 28, 1)


      pred = model.predict(im2arr)
      print(np.argmax(pred))

      3
```

TESTING

8.1 USER ACCEPTANCE TESTING:

Date	18 November 2022
Team ID	PNT2022TMID35498
Project Name	Project – A novel method for handwritten digit recognition
Maximum Marks	4 Marks

1. Purpose of Document

The purpose of this document is to briefly explain the test coverage and open issues of the **A novel method for handwritten digit recognition** project at the time of the release to User Acceptance Testing (UAT).

2. Defect Analysis

This report shows the number of resolved or closed bugs at each severity level, and how they were resolved

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
By Design	13	4	2	5	24
Duplicate	1	0	0	2	3
External	2	3	1	1	7
Fixed	4	6	4	10	24
Not Reproduced	1	1	1	1	4
Skipped	1	2	0	1	4
Won't Fix	0	5	2	4	11

Totals 22 21 10 24 77

8.2 TEST CASE ANALYSIS:

This report shows the number of test cases that have passed, failed, and untested

Section	Total Cases	Not Tested	Fail	Pass
Print Engine	7	0	0	7
Security	2	0	0	2
Performance	3	0	0	3
Exception Reporting	9	0	0	9

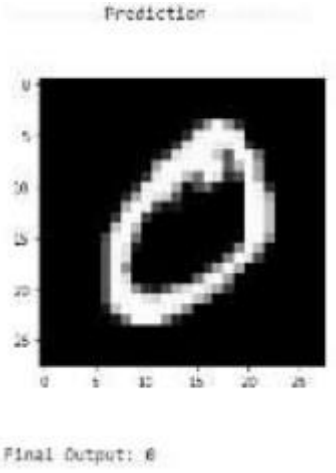
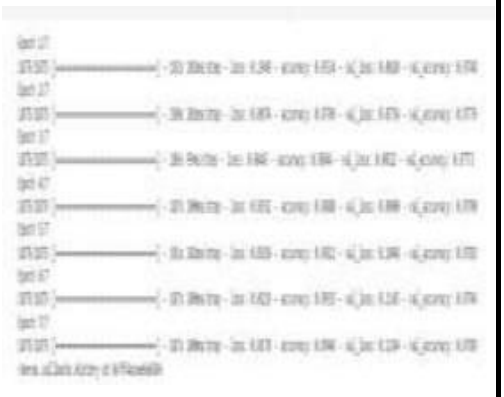
RESULTS

9.1 PERFORMANCE METRICS:

Date	18 November 2022
Team ID	PNT2022TMID35498
Project Name	Project – A novel method for handwritten digit recognition
Maximum Marks	10 Marks

Model Performance Testing:

Project team shall fill the following information in the model performance testing template.

S.No.	Parameter	Values	Screenshot
1.	Model Summary	Loss: 0.0181 Val_loss:0.1560	
2.	Accuracy	Training Accuracy - 97.87 Validation Accuracy -95.48	

ADVANTAGES AND DISADVANTAGES

ADVANTAGES:

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

DISADVANTAGES:

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

CONCLUSION

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

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 from digits multiple images and save the

results ● Add support to detect multiple digits

- Improve model to detect digits from complex images

- Add support to different languages to help users from all over the world This project has endless potential and can always be enhanced to become better. Implementing this concept in the real world will benefit several industries and reduce the workload on many workers, enhancing overall work efficiency.

APPENDIX

SOURCE CODE:

Importing Required Libraries

In [1]:
```
import numpy as np
from numpy import array
import matplotlib.pyplot as plt
import tensorflow as tf
import pandas as pd
from keras.datasets import mnist
from keras.preprocessing import image
```

Loading Data

In [2]:
```
(X_train, y_train), (X_test, y_test) = mnist.load_data()


print("Training Images:", len(X_train))
print("Testing Images:", len(X_test))
print("Shape:", X_train[0].shape)
```

```
Training Images: 60000
Testing Images: 10000
Shape: (28, 28)
```

Data Preprocessing

In [3]:
```
import random
import matplotlib.pyplot as plt
random_image = random.choice(X_train)
plt.imshow(random_image, cmap="gray")
```

Out[3]:
```
<matplotlib.image.AxesImage at 0x20c3e3f2c10>
```

In [4]:
```
X_train = X_train.reshape(X_train.shape[0], 28, 28, 1)
X_test = X_test.reshape(X_test.shape[0], 28, 28, 1)
X_train.shape

X_train = X_train / 255.
X_test = X_test / 255.


X_train = X_train.astype(np.float32)
X_test = X_test.astype(np.float32)

input_shape = X_train[0].shape
```

Building a Model

In [5]:
```
# Importing the required Keras modules containing model and layers
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Conv2D, Dropout, Flatten, MaxPooling2D
# Creating a Sequential Model and adding the layers
model = Sequential()
model.add(Conv2D(28, kernel_size=(3,3), input_shape=input_shape))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten()) # Flattening the 2D arrays for fully connected layers
model.add(Dense(128, activation=tf.nn.relu))
model.add(Dropout(0.2))
model.add(Dense(10,activation=tf.nn.softmax))
```

Compile the Model

In [6]:
```
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
```

In [7]:
```
model.fit(x=X_train,y=y_train, epochs=10)
model.save('Digit_Recog.h5')
```

```
Epoch 1/10
1875/1875 [==============================] - 22s 11ms/step - loss: 0.2113 - accuracy: 0.9372
Epoch 2/10
1875/1875 [==============================] - 22s 12ms/step - loss: 0.0862 - accuracy: 0.9739
Epoch 3/10
1875/1875 [==============================] - 21s 11ms/step - loss: 0.0602 - accuracy: 0.9812
Epoch 4/10
1875/1875 [==============================] - 21s 11ms/step - loss: 0.0451 - accuracy: 0.9857
Epoch 5/10
1875/1875 [==============================] - 21s 11ms/step - loss: 0.0355 - accuracy: 0.9884
Epoch 6/10
1875/1875 [==============================] - 22s 12ms/step - loss: 0.0293 - accuracy: 0.9901
Epoch 7/10
1875/1875 [==============================] - 20s 10ms/step - loss: 0.0243 - accuracy: 0.9919
Epoch 8/10
1875/1875 [==============================] - 21s 11ms/step - loss: 0.0230 - accuracy: 0.9921
Epoch 9/10
1875/1875 [==============================] - 21s 11ms/step - loss: 0.0193 - accuracy: 0.9937
Epoch 10/10
1875/1875 [==============================] - 20s 11ms/step - loss: 0.0191 - accuracy: 0.9934
```

Evaluate and predict the model

In [20]:
```
# network evaluation
model.evaluate(X_test, y_test)

# test sample data
image_index = 980
plt.imshow(X_test[image_index].reshape(28, 28),cmap='Greys')
pred = model.predict(X_test[image_index].reshape(1, 28, 28, 1))
print(pred.argmax())
```

```
313/313 [==============================] - 2s 5ms/step - loss: 0.0629 - accuracy: 0.9857
1/1 [==============================] - 0s 39ms/step
2
```

GITHUB:

https://github.com/IBM-EPBL/IBM-Project-6947-1658844189

PROJECT DEMO LINK:

https://drive.google.com/file/d/1eWlih-NfrbQ-NXaiB41iZYvIvqg7FfxW/view?usp=share_link