PROJECT REPORT

A NOVEL METHOD FOR HANDWRITTEN DIGIT RECOGNITION

submitted by

PNT2022TMID25375

Premkumar G
Ramani S
Simon S
Sankar U
Stebin S

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1. PROJECT OVERVIEW

Machine learning and deep learning play an important role in computer technology and artificial intelligence. With the use of deep learning and machine learning, human effort can be reduced in recognizing, learning, predictions and in many more areas.

Handwritten Digit Recognition is the ability of computer systems to recognise handwritten digits from various sources, such as images, documents, and so on. This project aims to let users take advantage of machine learning to reduce manual tasks in recognizing digits.

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.

CHAPTER 2

LITERATURE SURVEY

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. REFERENCES

Recognition of Handwritten Digit using Convolutional Neural Network in Python with Tensorflow and Comparison of Performance for Various Hidden Layers(2019)

F. Siddique, S. Sakib and M. A. B. Siddique

The increase of Artificial Neural Network (ANN), deep learning has brought a dramatic twist in the field of machine learning by making it more artificially intelligent. Deep learning is remarkably used in vast ranges of fields because of its diverse range of applications such as surveillance, health, medicine, sports, robotics, drones, etc. In deep learning, Convolutional Neural Network (CNN) is at the center of spectacular advances that mixes Artificial Neural Network (ANN) and up to date deep learning strategies. It has been used broadly in pattern recognition, sentence classification, speech

recognition, face recognition, text categorization, document analysis, scene, and handwritten digit recognition. The goal of this paper is to observe the variation of accuracies of CNN to classify handwritten digits using various numbers of hidden layers and epochs and to make the comparison between the accuracies. For this performance evaluation of CNN, we performed our experiment using Modified National Institute of Standards and Technology (MNIST) dataset. Further, the network is trained using stochastic gradient descent and the back-propagation algorithm.

An Efficient And Improved Scheme For Handwritten Digit Recognition Based On Convolutional Neural Network (2019)

Ali, Saqib and Shaukat, Zeeshan and Azeem, Muhammad and Sakhawat, Zareen and Mahmood, Tariq and others

This study uses rectified linear units (ReLU) activation and a convolutional neural network (CNN) that incorporates the Deeplearning4j (DL4J) architecture to recognize handwritten digits. The proposed CNN framework has all the necessary parameters for a high level of MNIST digit classification accuracy. The system's training takes into account the time factor as well. The system is also tested by altering the number of CNN layers for additional accuracy verification. It is important to note that the CNN architecture consists of two convolutional layers, the first with 32 filters and a 5x5 window size and the second with 64 filters and a 7x7 window size. In comparison to earlier proposed systems, the experimental findings show that the proposed CNN architecture for the MNIST dataset demonstrates great performance in terms of time and accuracy. As a result, handwritten numbers are detected with a recognition rate of 99.89% and high precision (99.21%) in a short amount of time.

Handwritten Digit Recognition using Convolutional Neural Network in Python with Tensorflow and Observe the Variation of Accuracies for Various Hidden Layers (2019)

Fathma Siddique, Shadman Sakib, Md. Abu Bakr Siddique

In recent times, with the increase of Artificial Neural Network (ANN), deep learning has brought a dramatic twist in the field of machine learning by making it more Artificial Intelligence (AI). Deep learning is remarkably used in vast ranges of fields because of its diverse range of applications such as surveillance, health, medicine, sports, robotics, drones etc. In deep learning, Convolutional Neural Network (CNN) is at the center of spectacular advances that mixes Artificial Neural Network (ANN) and up to date deep learning strategies. It has been used broadly in pattern recognition, sentence classification, speech recognition, face recognition, text categorization, document analysis, scene, and handwritten digit recognition. The goal of this paper is to observe the variation of accuracies of CNN to classify handwritten digits using various numbers of hidden layers and epochs and to make the comparison between the accuracies. For this performance evaluation of CNN, we performed our experiment using the Modified National Institute of Standards and Technology (MNIST) dataset. Further, the network is trained using stochastic gradient descent and the backpropagation algorithm.

Handwritten Digit Recognition Using Machine And Deep Learning Algorithms (2021)

Pashine, Samay and Dixit, Ritik and Kushwah, Rishika

In this study, they developed three deep and machine learning-based models for handwritten digit recognition using MNIST datasets. To determine which model was the most accurate, they compared them based on their individual properties.

Support vector machines are among the simplest classifiers, making them faster than other algorithms and providing the highest training accuracy rate in this situation. However, due to their simplicity, SVMs cannot categorize

complicated and ambiguous images as accurately as MLP and CNN algorithms can. In their research, they discovered that CNN produced the most precise outcomes for handwritten digit recognition. This led them to the conclusion that CNN is the most effective

solution for all types of prediction issues, including those using picture data. Next, by comparing the execution times of the algorithms, they determined that increasing the number of epochs without changing the configuration of the algorithm is pointless due to the limitation of a certain model, and they discovered that beyond a certain number of epochs, the model begins over-fitting the dataset and provides biased predictions.

An Enhanced Handwritten Digit Recognition Using Convolutional Neural Network(2021)

M. S, C. N. Vanitha, N. Narayan, R. Kumar and G. R

Handwritten digit recognition has a great impact in the applications of deep learning. Convolutional Neural Network in deep learning has become one of the major methods and one of the important factors in the various success in recent times and deep learning is used majorly in the area of object recognition. In the paper work, the speech output feature is integrated along with the text output. Convolutional Neural Network model is applied in the image classification. The dataset used to train and test is the MNIST dataset. There are various applications of handwritten digit recognition in real time. It is applied in detection of vehicle number, reading of bank cheques, the arrangement of letters in the post office.

3. PROBLEM STATEMENT DEFINITION

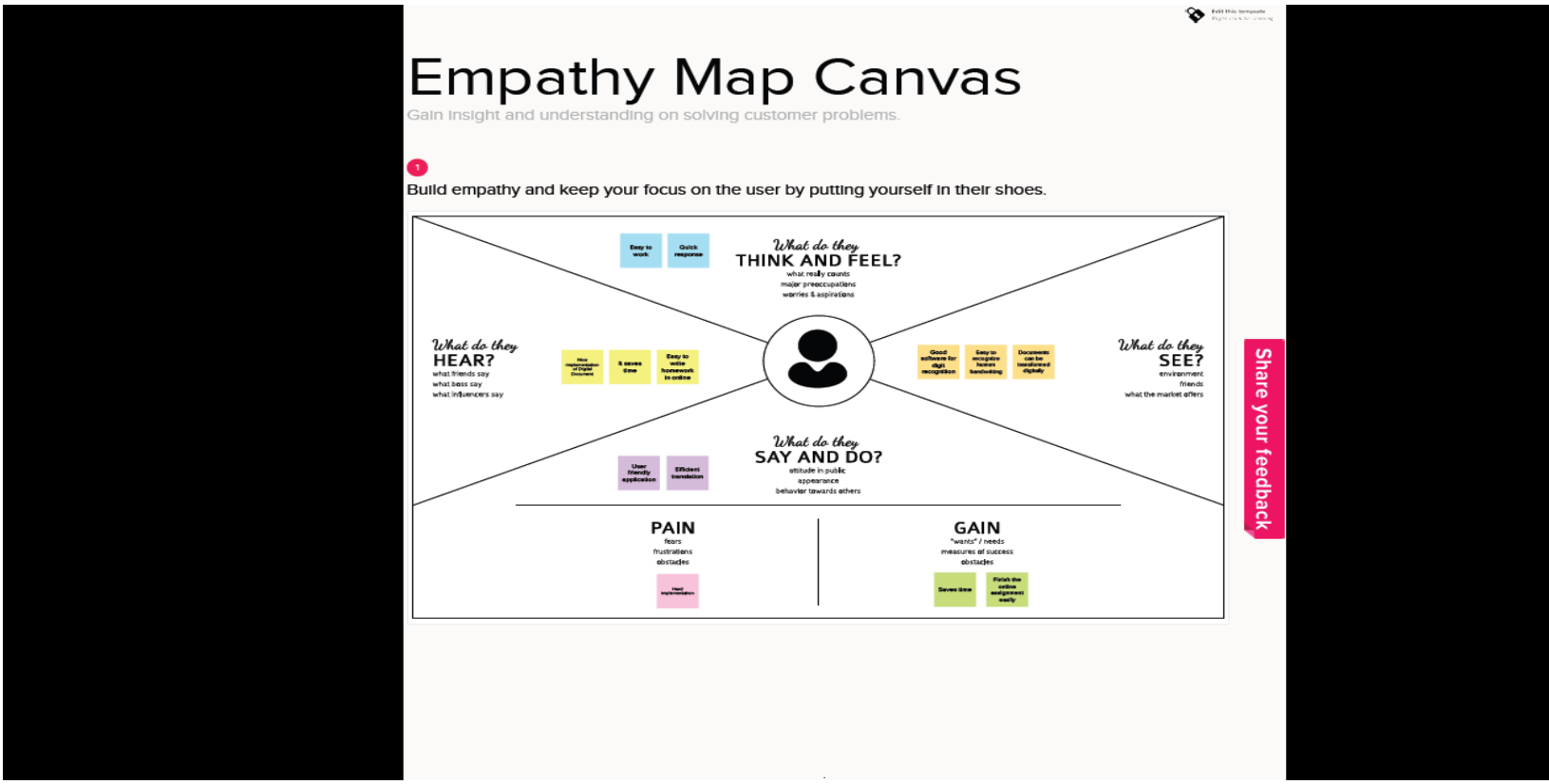
Handwriting recognition is one of the compelling research works going on because every individual in this world has their own style of writing. It is the capability of the computer to identify and understand handwritten digits or characters automatically. Because of the progress in the field of science and technology, everything is being digitized to reduce

human effort. Hence, there comes a need for handwritten digit recognition in many real-time applications. The 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 to the UI. MNIST ("Modified National Institute of Standards and Technology") is considered an unofficial computer vision "hello-world" dataset. This is a collection of thousands of handwritten pictures used to train classification models using Machine Learning techniques.

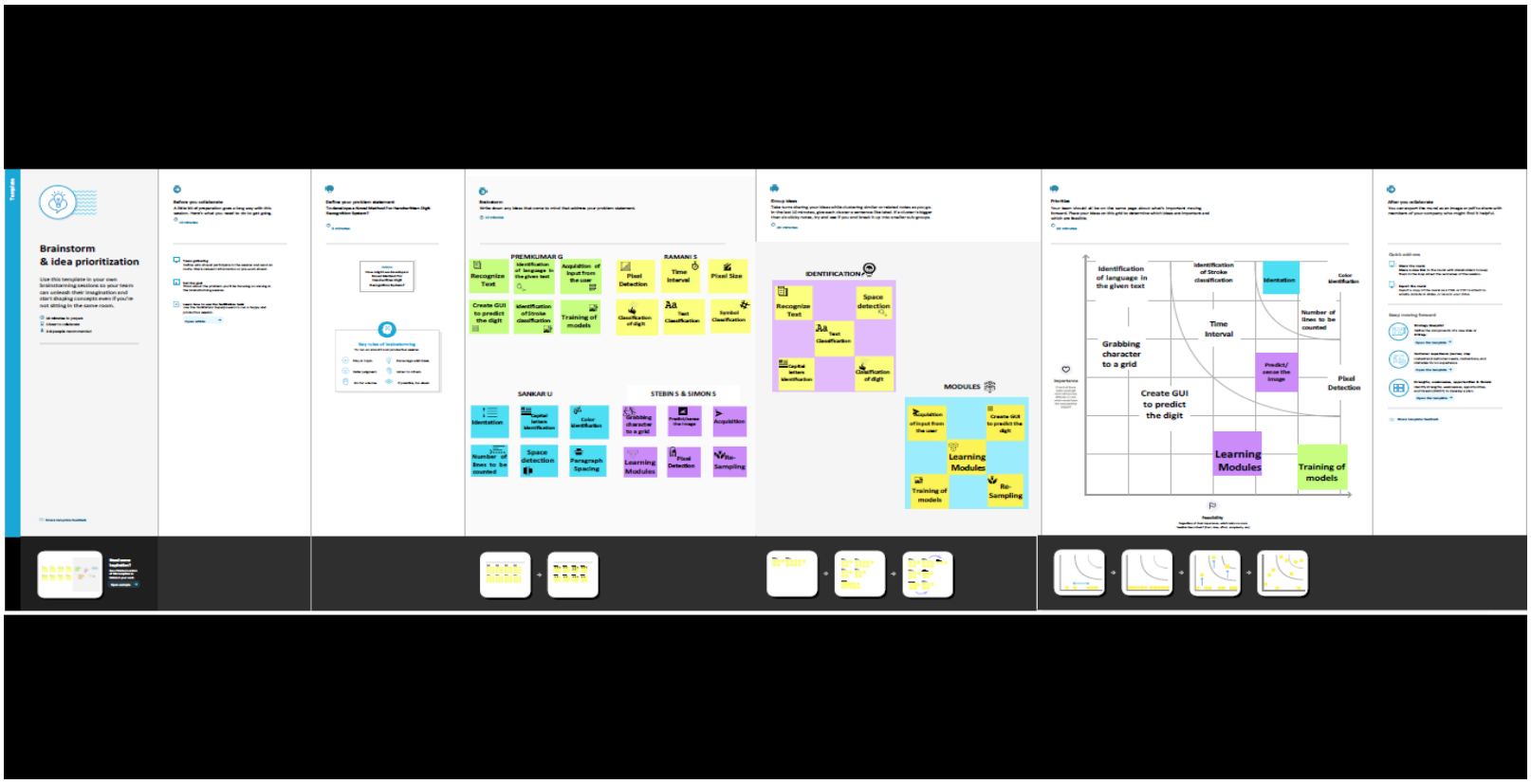
CHAPTER 3

IDEATION AND PROPOSED SOLUTION

1. EMPATHY MAP



2. IDEATION & BRAINSTORMING



3. PROPOSED SOLUTION

Proposed Solution:

S.NO	Parameter	Description
1.	Problem Statement (Problem to be solved)	**Statement-** Handwritten digit recognition is the capability of computer applications to recognize 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. A computer can fete the mortal handwritten integers from different sources like images, papers, and tough defenses. 1. It allows the user 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 recognize digits in more noisy environments. • There is no limit to the number of digits it can be recognized.

4. PROBLEM SOLUTION FIT

MNIST Dataset Description

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 digitized to reduce human effort. Hence, there comes a need for handwritten digit recognition in many real-time applications. The 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

The MNIST Handwritten Digit Recognition Dataset contains 60,000 training and 10,000 testing labeled handwritten digit pictures.

Each picture is 28 pixels in height and 28 pixels wide, for a total of 784 (28×28) pixels. Each pixel has a single pixel value associated with it. It indicates how bright or dark that pixel is (larger numbers indicate darker pixels). This pixel value is an integer ranging from 0 to 255.

PROCEDURE

- Install the latest TensorFlow library.
- Prepare the dataset for the model.
- Develop Single Layer Perceptron model for classifying the handwritten digits.
- Plot the change in accuracy per epochs.

- Evaluate the model on the testing data.
- Analyze the model summary.
- Add a hidden layer to the model to make it a Multi-Layer Perceptron.
- Add Dropout to prevent overfitting and check its effect on accuracy.
- Increasing the number of Hidden Layer neurons and checking its effect on accuracy.
- Use different optimizers and check its effect on accuracy.
- Increase the hidden layers and check its effect on accuracy.
- Manipulate the batch size and epochs and check its effect on accuracy.

MNIST is a dataset which is widely used for handwritten digit recognition. The dataset consists of 60,000 training images and 10,000 test images. The artificial neural networks can almost mimic the human brain and are a key ingredient in the image processing field.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.

APPROACH:

We will approach this project by using a three-layered Neural Network.

- The input layer: It distributes the features of our examples to the next layer for calculation of activations of the next layer.
- The hidden layer: They are made of hidden units called activations providing nonlinear ties for the network. A number of hidden layers can vary according to our requirements.

- The output layer: The nodes here are called output units. It provides us with the final prediction of the Neural Network on the basis of which final predictions can be made.

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.

METHODOLOGY:

We have implemented a Neural Network with 1 hidden layer having 100 activation units (excluding bias units). The data is loaded from a .mat file, features(X) and labels(y) were extracted. Then features are divided by 255 to rescale them into a range of [0,1] to avoid overflow during computation. Data is split up into 60,000 training and 10,000 testing examples. Feedforward is performed with the training set for calculating the hypothesis and then backpropagation is done in order to reduce the error between the layers. The regularization parameter lambda is set to 0.1 to address the problem of overfitting. Optimizer is run for 70 iterations to find the best fit model

ALGORITHM:

Forward Propagation Architecture:

It is a small workflow of how CNN module will extract the features and classify the image based on it. The architecture shows the input layer, hidden layers and output layer of the

network. There are many layers involved in the feature extraction phase of the network which involves convolution and subsampling .

EXPLANATION OF THE PROPOSED SYSTEM

- The first layer of the architecture is the User layer. User layer will comprise of the people who interacts with the app and for the required results.

- The next three layers is the frontend architecture of the application.

The application will be developed using which is the open-source platform for HTML, CSS and JavaScript. The application is deployed in the localhost which is shown on the browser. Through the app, the user will be able to upload pictures of the handwritten digits and convert it into the digitalized form. • The one in between the database and view layer is the business layer which is the logical calculations on the basis of the request from the client side. It also has a service interface. • The backend layer consists of two datasets: Training Data and Test Data. The MNIST database has been used for that which is already divided into a training set of 60,000 examples and a test of 10,000 examples. • The training algorithm used is Convolution Neural Network. This will prepare the trained model which will be used to classify the digits present in the test data. Thus, we can classify the digits present in the images as: Class 0,1,2,3,4,5,6,7,8,9.

WORKING:

- Neural Networks receive an input and transform it through a series of hidden layers.

- Each hidden layer is made up of a set of neurons, where each neuron is fully connected to all neurons in the previous layer.

• Neurons in a single layer function completely independently. • The last fully connected layer is called the "output layer“.

Convolution Layer: The Convolutional layer is the core building block of a CNN. The layer's parameters consist of a set of learnable filters (or kernels), which have a small receptive field, but extend through the full depth of the input volume.

During the forward pass, each filter is convolved across the width and height of the input volume, computing the dot product between the entries of the filter and the input and producing a 2- dimensional activation map of that filter.

As a result, the network learns filters that activate when they see some specific type of feature at some spatial position in the input..

Feature Extraction:

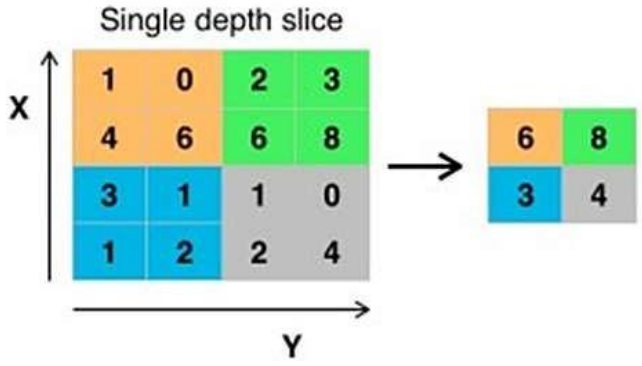
All neurons in a feature share the same weights .In this way all neurons detect the same feature at different positions in the input image. Reduce the number of free parameters.

Subsampling Layer: Subsampling, or down sampling, refers to reducing the overall size of a signal .The subsampling layers reduce the spatial resolution of each feature map. Reduce the effect of noises and shift or distortion invariance is achieved.

Pooling layer: It is common to periodically insert a Pooling layer in-between successive Conv layer in a Convent architecture. Its function is to progressively reduce the spatial size of the representation to reduce the number of parameters and computation in the network, and hence to also control overfitting. The Pooling Layer operates independently on every depth slice of the input and resizes it spatially, using the MAX operation.

TensorFlow: TensorFlow is an open-source machine learning library for research and production. TensorFlow offers APIs for beginners and experts to develop for desktop,

mobile, web, and cloud. See the sections below to get started. By scanning the numerical digit and convert into png format using python3 command in terminal we can get text output and sound output.



Pooling layer

CHAPTER 4

REQUIREMENT ANALYSIS

1. FUNCTIONAL AND NON-FUNCTIONAL REQUIREMENTS

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

Functional Requirements:

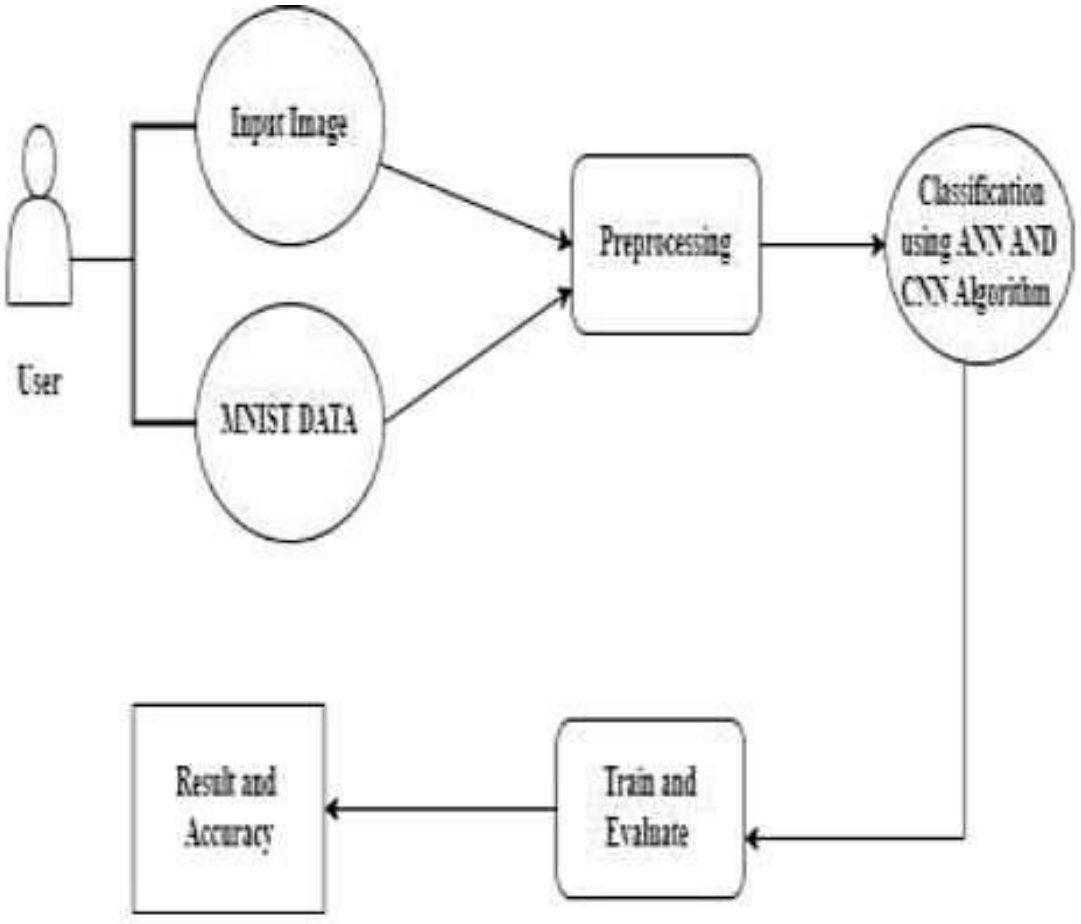
Following are the functional requirements of the proposed solution.

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

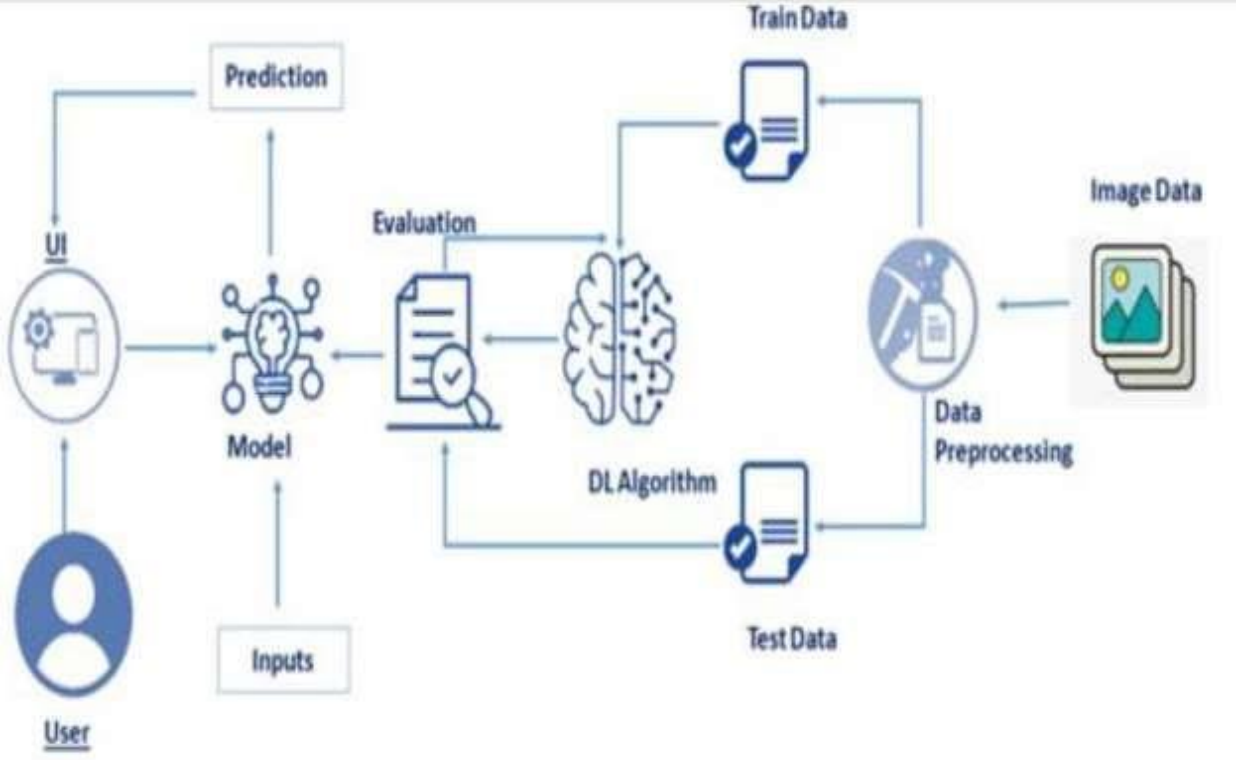
CHAPTER 5

PROJECT DESIGN

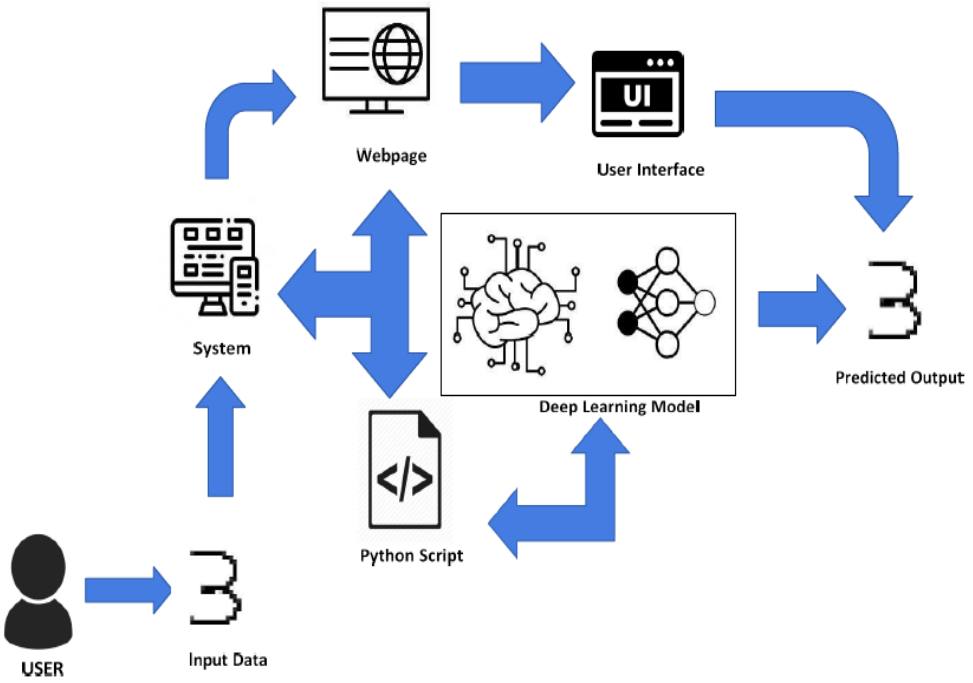
1. DATA FLOW DIAGRAM



2. SOLUTION & TECHNICAL ARCHITECTURE



Technical Architecture
Team ID : PNT2022TMID25375



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 and awareness to use 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 can read the instructions to use this application.	I can read instructions also to use it in a user-friendly method.	Low	Sprint-2
	Recognize	USN-4	As a user, In this prediction page I get to choose the image.	I can choose the image from our local system and predict the output.	High	Sprint-2
	Predict	USN-6	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
		USN-7	As a user, I will train and test the input to get the maximum accuracy of output.	I can able to train and test the application until it gets maximum accuracy of the result.	High	Sprint-4
		USN-8	As a user, I can 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 can view the guide to use the web app.	I can view the awareness of 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 can view the guide and awareness to use this application.	I can view the awareness to use this application and its limitations.	Low	Sprint-1

CHAPTER 6

PROJECT PLANNING AND SCHEDULING

1. SPRINT PLANNING AND ESTIMATION

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

Sprint-3	Upload Image of Handwritten document	USN-5	As a user, I can able to input the images of the handwritten documents or images to the application	2	High	Shajida M
Sprint-3	Recognize text	USN-6	As a user, I can able to choose the font of the text to be displayed	1	Medium	Sakthiyadharshini NP
Sprint-4	Recognize digit	USN-7	As a user I can able to get the recognised digit as output from the images of digital documents or images	1	Medium	Swetha N
Sprint-4	Recognize digit	USN-8	As a user I can able to get the recognised digit as output from the images of handwritten documents or images	2	High	Udhaya P

2. SPRINT DELIVERY SCHEDULE

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

CHAPTER 7

CODING & SOLUTIONING

FEATURE 1

```
import pandas as pd
import seaborn as sns
import numpy as np
from matplotlib import pyplot as plt
%matplotlib inline
```

```
import csv
import tensorflow as tf
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords
STOPWORDS = set(stopwords.words('english'))
```

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow.keras.datasets import mnist
from tensorflow.keras.models import Sequential
from tensorflow.keras import layers
from tensorflow.keras.layers import Dense,Flatten,MaxPooling2D
from tensorflow.keras.layers import Conv2D
from keras.optimizers import Adam
from keras.utils import np_utils
from tensorflow.keras.models import load_model
```

```
reverse_word_index = dict([(value, key) for (key, value) in word_index.items()])

def decode_article(text):
    return ' '.join([reverse_word_index.get(i, '?') for i in text])
print(decode_article(train_padded[10]))
print('---')
print(train_articles[10])
```

FEATURE 2

```
vocab_size = 5000
embedding_dim = 64
max_length = 200
trunc_type = 'post'
padding_type = 'post'
oov_tok = ''
training_portion = .8
```

```
articles = []
labels = []

with open("spam.csv", 'r', encoding = "ISO-8859-1") as dataset:
    reader = csv.reader(dataset, delimiter=',')
    next(reader)
    for row in reader:
        labels.append(row[0])
        article = row[1]
        for word in STOPWORDS:
            token = ' ' + word + ' '
            article = article.replace(token, ' ')
            article = article.replace('  ', ' ')
        articles.append(article)
print(len(labels))
print(len(articles))
```

```
train_size = int(len(articles) * training_portion)

train_articles = articles[0: train_size]
train_labels = labels[0: train_size]

validation_articles = articles[train_size:]
validation_labels = labels[train_size:]

print(train_size)
print(len(train_articles))
print(len(train_labels))
print(len(validation_articles))
print(len(validation_labels))
```

```
train_padded = pad_sequences(train_sequences, maxlen=max_length, padding=padding_type, truncating=trunc_type)
print(len(train_sequences[0]))
print(len(train_padded[0]))

print(len(train_sequences[1]))
print(len(train_padded[1]))

print(len(train_sequences[10]))
print(len(train_padded[10]))
```

```
model.add(Conv2D(64,(3,3),input_shape=(28,28,1),activation="relu"))
model.add(Conv2D(32,(3,3),activation="relu"))
model.add(MaxPooling2D((2,2)))
model.add(Flatten())
model.add(Dense(number_of_classes,activation="softmax"))
```

CHAPTER 8

TESTING

1. TEST CASES

st 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 2353 x 1651 and 758 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

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
AC_TC_001	Functional	Accuracy	check if the model can provide the output with accuracy	The model should predict the image with accuracy from the dataset.	working as expected	PASS

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

2. USER ACCEPTANCE TESTING

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

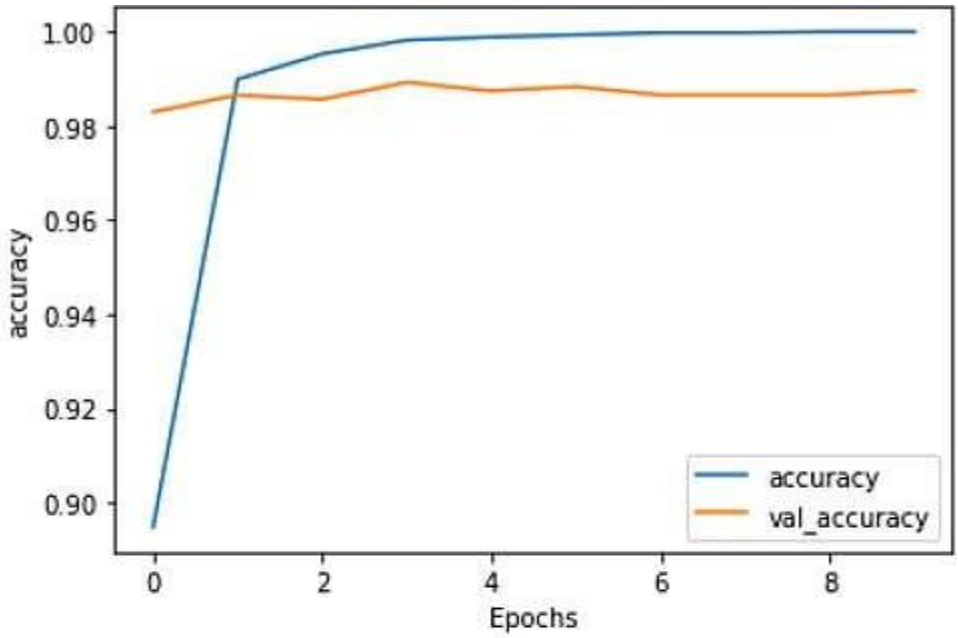
2. TEST CASE ANALYSIS

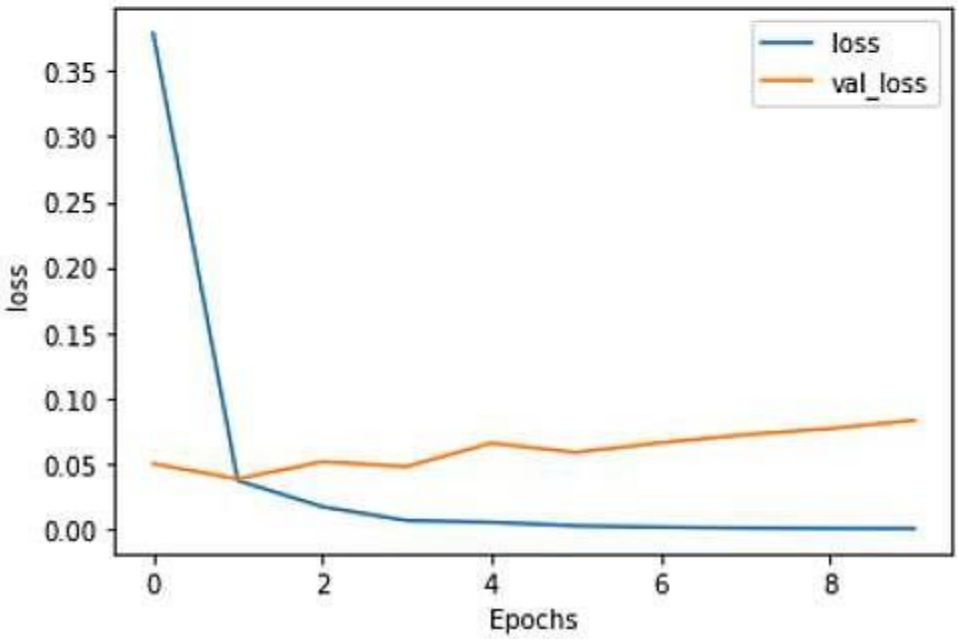
Section	Total Cases	Not Tested	Fail	Pass
Client Application	10	0	3	7
Security	2	0	1	1
Performance	3	0	1	2
Exception Reporting	2	0	0	2

CHAPTER 9

RESULTS

1. PERFORMANCE METRICS





CHAPTER 10

ADVANTAGES & 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

CHAPTER 11
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.

CHAPTER 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 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

MODEL CREATION

Import necessary package

```
import numpy
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
```

Load data

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

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11490434/11490434 [==============================] - 0s 0us/step
```

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

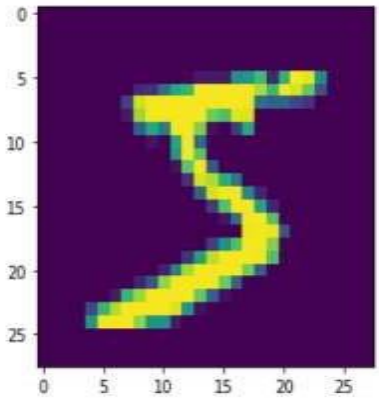
```
(60000, 28, 28)
(10000, 28, 28)
```

In [5]:
```
y_train[0]
```

Out[5]: 5

In [6]:
```
plt.imshow(X_train[0])
```

Out[6]:



Data pre processing

In [7]:
```
X_train = X_train.reshape(60000, 28, 28, 1).astype('float32')
X_test = X_test.reshape(10000, 28, 28, 1).astype('float32')
```

In [8]:
```
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)
```

```python
import tensorflow as tf

from keras.models import Sequential
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten
from keras.layers import Conv2D, MaxPooling2D

def init():
    num_classes = 10
    img_rows, img_cols = 28, 28
    input_shape = (img_rows, img_cols, 1)
    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(MaxPooling2D(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(num_classes, activation='softmax'))

    #load weights into new model
    model.load_weights("weights.h5")
    print("Loaded Model from disk")

    #compile and evaluate loaded model
    model.compile(loss=keras.losses.categorical_crossentropy, optimizer=keras.optimizers.Adadelta(), metrics=['accuracy'])
    #loss,accuracy = model.evaluate(X_test,y_test)
    #print('loss:', loss)
    #print('accuracy:', accuracy)
    graph = tf.get_default_graph()

    return model,
```

FLASK APP

```
from flask import Flask, render_template, request
from scipy.misc import imsave, imread, imresize
import numpy as np
import keras.models
import re
import base64

import sys
import os
sys.path.append(os.path.abspath("./model"))
from load import *

app = Flask(__name__)
global model, graph
model, graph = init()

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

@app.route('/predict/', methods=['GET','POST'])
```

HOME PAGE (HTML)

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

TRAIN THE MODEL

```
import keras
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten
from keras.layers import Conv2D, MaxPooling2D
from keras import backend as K
import json

batch_size = 32
num_classes = 10
epochs = 60

# input image dimensions
img_rows, img_cols = 28, 28

# the data, shuffled and split between train and test sets
(x_train, y_train), (x_test, y_test) = mnist.load_data()

if K.image_data_format() == 'channels_first':
    x_train = x_train.reshape(x_train.shape[0], 1, img_rows, img_cols)
    x_test = x_test.reshape(x_test.shape[0], 1, img_rows, img_cols)
    input_shape = (1, img_rows, img_cols)
else:
    x_train = x_train.reshape(x_train.shape[0], img_rows, img_cols, 1)
    x_test = x_test.reshape(x_test.shape[0], img_rows, img_cols, 1)
    input_shape = (img_rows, img_cols, 1)
```

```
x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
x_train /= 255
x_test /= 255
print('x_train shape:', x_train.shape)
print(x_train.shape[0], 'train samples')
print(x_test.shape[0], 'test samples')

# convert class vectors to binary class matrices
y_train = keras.utils.to_categorical(y_train, num_classes)
y_test = keras.utils.to_categorical(y_test, num_classes)

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(MaxPooling2D(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(num_classes, activation='softmax'))

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

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

model.fit(x_train, y_train,
          batch_size=batch_size,
          epochs=epochs,
          verbose=1,
          validation_data=(x_test, y_test))
score = model.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])


with open('model.json', 'w') as outfile:
    json.dump(model.to_json(), outfile)
model.save_weights('weights2.h5')
```

PREDICT PAGE (HTML)

```
<html>
	<head>
		<title>Prediction | Handwritten Digit Recognition</title>
		<link rel="stylesheet" href="{{url_for('static',filename='css/predict.css')}}" />
		<link rel="icon" type="image/svg" sizes="32x32" href="{{url_for('static',filename='images/icon.svg')}}" />
		<meta name="viewport" content="width=device-width, initial-scale=1.0" />
	</head>
	<body>
		<div class="container">
			<h1>Prediction</h1>
			<div class="result-wrapper">
				<div class="input-image-container">
					<img src="{{url_for('static',filename='data/')}}{{img_name}}" />
				</div>
				<div class="result-container">
					<div class="value">{{best.0}}</div>
					<div class="accuracy">{{best.1}}%</div>
				</div>
			</div>
			<h1>Other Predictions</h1>
			<div class="other_predictions">
				{% for x in others %}
				<div class="value">
					<h2>{{x.0}}</h2>
					<div class="accuracy">{{x.1}}%</div>
				</div>
				{% endfor %}
			</div>
		</div>
	</body>
</html>
```

 **GITHUB**

https://github.com/IBM-EPBL/IBM-Project-42867-1660710548

 **PROJECT DEMO**

https://drive.google.com/file/d/15fx8_LGlvdt_7zHoOdil9m--gboQcNtq/view?usp=drivesdk