A NOVEL METHOD FOR HANDWRITTEN DIGIT RECOGNITION SYSTEM

PROJECT REPORT

SUBMITTED BY

TEAM ID: PNT2022TMID28204

ASHWIN VENKAT (312419106011)

BALAN K (312419106013)

BHARATHRAJ B (312419106015)

DHANUSH V R (312419106027)

BACHELOR OF ENGINEERING

IN

ELECTRONICS AND COMMUNICATION ENGINEERING

St. JOSEPH'S INSTITUTE OF TECHNOLOGY,
(An Autonomous Instutution)

Old Mamallapuram Road, Semmancheri,
Chennai, Tamil Nadu 600119

ABSTRACT

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

Handwritten Digit Recognition is the capability of a computer to fete the mortal handwritten digits or characters from different sources and classify them into 10 predefined classes (0-9). This has been a content of bottomless exploration in the field of deep literacy. Number recognition has numerous operations like number plate recognition, postal correspondence sorting, bank check processing, etc. In Handwritten number recognition, we face numerous challenges because of different styles of jotting of different peoples as it is not an Optic character recognition.

This exploration provides a comprehensive comparison between different machine literacy and deep literacy algorithms for the purpose of handwritten number recognition. For this, we have used Convolutional Neural Network. The Neural network is used to train and identify written digits. 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 UI.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 Project Overview

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 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 backpropagation algorithm.

1.2 Purpose

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 Hand writing recognition on computer tablets or system, recognize number plates of Vehicles , processing bank cheque amounts, numeric entries informs filled up by hand and so on.

CHAPTER 2

LITERATURE SURVEY

2.1 EXISTING PROBLEM

The fundamental problem with handwritten digit recognition is that Hand written 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

Paper 1: Improving Offline Handwritten Text Recognition with Hybrid HMM/ANN Models

Year: 2011

Authors: Salvador Espan˜a-Boquera, Maria Jose Castro-Bleda, Jorge Gorbe-Moya, and Francisco Zamora-Martinez

The use of hybrid Hidden Markov Model (HMM)/Artificial Neural Network (ANN).

PROS: It also presents new techniques to remove slope and slant from handwritten text and to normalize the size of text images with supervised learning methods.

CONS: The recognition is based on hybrid optical HMM/ANN models, where an MLP is used to estimate the emission probabilities.

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

Year: 2021
Authors: Ali Abdullah Yahya, Jieqing Tan, Min Hu

There have been a tons of CNN classification algorithms put forth in the literature. However, these algorithms do not take into account the proper filter size selection, data preparation, dataset restrictions, or noise. As a result, few algorithms have been able to significantly increase classification accuracy. The paper makes the following contributions to solve these methods' drawbacks: First, the size of the effective receptive field (ERF) is determined after taking domain knowledge into account. They choose a typical filter size with the aid of the ERF calculation, improving the classification accuracy of our CNN. Second, excessive data produces inaccurate results, which has a detrimental impact on classification accuracy.

Before carrying out the data classification task, data preparation is conducted to ensure that the dataset is devoid of any redundant or irrelevant variables to the goal variable. Thirdly, data augmentation has been suggested as a way to reduce training and validation errors and prevent dataset limitations. Fourthly, the paper suggests adding an additive white Gaussian noise with a threshold of 0.5 to the MNIST dataset in order to imitate the natural factors that can affect image quality in the real world. With a recognition accuracy of 99.98% and 99.40% with 50% noise, our CNN algorithm achieves state-of-the-art performance in handwritten digit recognition.

Paper 3: Improved Handwritten Digit Recognition Using Convolutional Neural Networks (CNN)

Year : 2020

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

Customized features and a vast quantity of past knowledge have been used in traditional handwriting recognition systems. It is difficult to train an optical character recognition (OCR) system based on these conditions. Deep learning approaches have enabled significant performance in the field of handwriting recognition research in recent years. Nonetheless, the increasing increase in the amount of handwritten data, along with the availability of vast computing capacity, necessitates improvements in recognition accuracy and warrants additional exploration.

Convolutional neural networks (CNNs) are extremely excellent in perceiving the structure of handwritten characters/words in ways that aid in the automatic extraction of distinguishing features, making CNN the best solution for solving handwriting recognition challenges. The proposed work aims to investigate several design alternatives for CNN-based handwritten digit recognition, such as the number of layers, stride size, receptive field, kernel size, padding, and dilution. Furthermore, we intend to assess the effectiveness of several SGD optimization techniques in enhancing the performance of handwritten digit recognition. Using ensemble architecture improves the recognition accuracy of a network. In this case, we want to obtain equal accuracy by employing a pure CNN design without ensemble architecture, because ensemble structures increase computational overhead and testing complexity. As a result, a CNN design is developed in order to obtain higher accuracy than ensemble systems while reducing operational complexity and expense.

Paper 4: Handwritten Character Recognition using Neural Network and TensorFlow

Year : 2019

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

The offline handwritten character recognition in this study will be carried out using Tensorflow and a convolutional neural network. a process known as using SoftMax Regression, one may assign probabilities to one of the many characters in the handwritten text that offers the range of values from 0 to 1, summed to 1.

The objective is to create software that is extremely accurate and that has a minimum level of spatial and temporal complexity. It was determined that strategies for feature extraction like diagonal and direction are significantly better at producing high accuracy. Outcomes in comparison to other conventional vertical and horizontal techniques moreover use the best Neural network tried layers provides the benefit of a higher accurate outcome by having a high noise tolerance.

The feed forward model in neural networks is the back-propagation algorithm that was primarily used to classify the characters, recognise them, and receive training continually more. In addition to these, normalizing along with feature extraction, the results were better and more effective.

2.3 PROBLEM STATEMENT DEFINITION

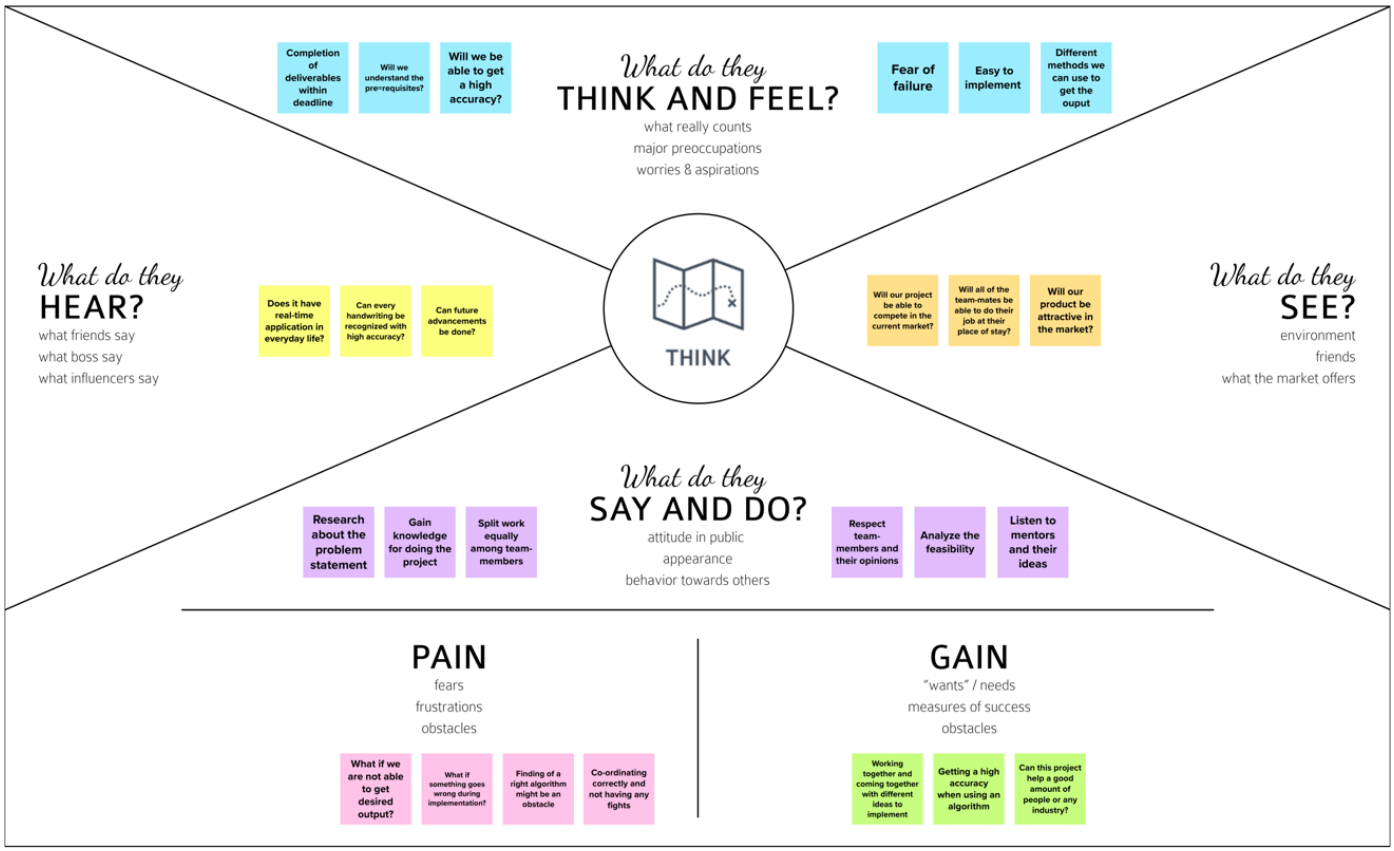
Machines are becoming more smart and intelligent by using machine learning and deep learning techniques so that they can perform tasks similar to humans. With the help of these techniques human effort can be reduced in recognizing, learning, predictions and many other areas. The goal of this project is to correctly identify digits from a dataset of tens of thousands of handwritten digit images and use different machine learning algorithms to learn first-hand what works well and gives the highest accuracy. This algorithm can then be used in various places like in postal services, banking services, etc.

QUESTION	DESCRIPTION
Who Does the Problem Affect?	It affects old aged people, banking services, postal services in understanding the handwritten digits.
What Are the Boundaries of The Problem?	The boundaries are different styles of handwritten digits, high accuracy in the prediction process.
What is the Issue?	Since everyone's handwriting is different, it maybe difficult for some to identify what one has written. Especially in banking and postal services where digits are important.
When does the issue occur?	It occurs when the handwriting of a person is difficult to read or recognize.
Where is the issue occurring?	The issue is occurring in various places from banking sector to industries.
Why is it important that we fix the problem?	It improves the readability and also prevents errors.

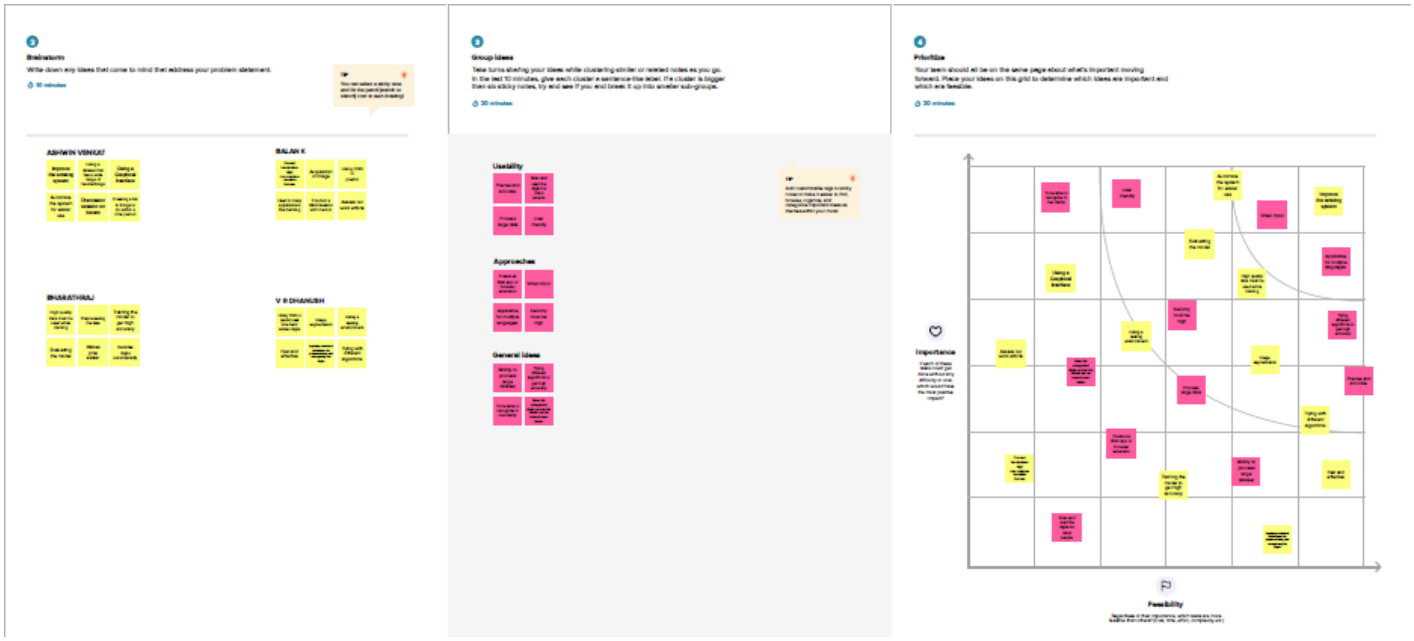
CHAPTER 3

IDEATIONAND PROPOSED SOLUTION

3.1 EMPATHY MAP CANVAS



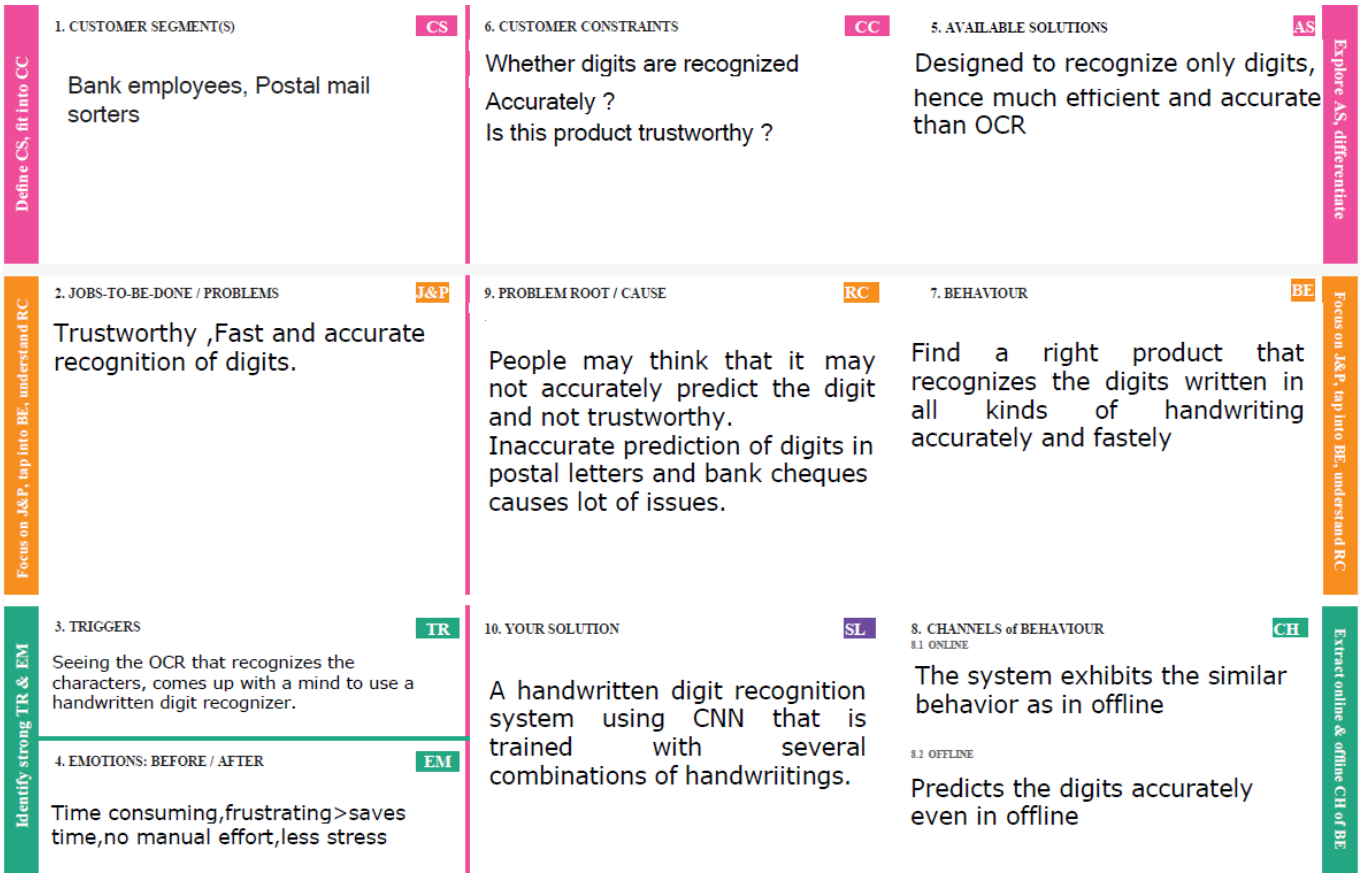
3.2 IDEATION& BRAINSTORMING



3.3 PROPOSED SOLUTION

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	The goal of this project is to correctly identify handwritten digits. It is a hard task because handwritten digits are not always of the same size, width and orientation as they differ from writing of person to person.
2.	Idea / Solution description	The goal of this project is to correctly identify digits from a dataset of tens of thousands of handwritten digit images and use different machine learning algorithms to learn first-hand what works well and gives the highest accuracy.
3.	Novelty / Uniqueness	Using various algorithms to get more accurate results and also classifying the numbers like unique characters like in greek alphabet to get good results.
4.	Social Impact / Customer Satisfaction	This project can then be used in various places like in postal services, banking services, etc. to make sure the digits are recognized properly in everyday use in various places.
5.	Business Model (Revenue Model)	This can be made into a website or app that can help people in their day to day activities where every needed. This can also be made as a government project in surveillance to easily recognize number plates and to maintain a database along with it for necessary use.
6.	Scalability of the Solution	A lot of features can be implemented in this project using Artificial Intelligence and Machine Learning. Use of different algorithms from neural networks also helps in recognition of digits. These features can be implements and can be accessed from several devices so it is very scalable.

3.4 PROPOSED SOLUTION FIT



CHAPTER 4

REQUIREMENTANALYSIS

4.1 FUNCTIONAL REQUIREMENTS

Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through Form Registration through Gmail Registration through Facebook
FR-2	User Confirmation	Confirmation via Email Confirmation via OTP
FR-3	User login	Login through email & password Login through Gmail Login through Facebook
FR-4	Upload handwritten Files	Upload the image from the computer.
FR-5	Recognizing the handwritten digit and displaying.	Recognizing the handwritten digit and displaying.
FR-6	Training and Testing	Train and test the model for better accuracy
FR-7	Displaying output	Display the output with high accuracy

4.2 NON- FUNCTIONAL REQUIREMENTS

Following are the non-functional requirements of the proposed solution.

FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	System design should be easily understood and user friendly to users. Furthermore, users of all skill levels of users should be able to navigate it without problems.
NFR-2	**Security**	The system should automatically be able to authenticate all users with their unique username and password.
NFR-3	**Reliability**	The system will perform its intended function adequately and will operate in a secured environment with atmost accuracy and success rate.
NFR-4	**Performance**	The system will be able to display the output with high accuracy.
NFR-5	**Availability**	The system will be available to anyone with a computer and internet connection.
NFR-6	**Scalability**	The system is open source so it can develop further.

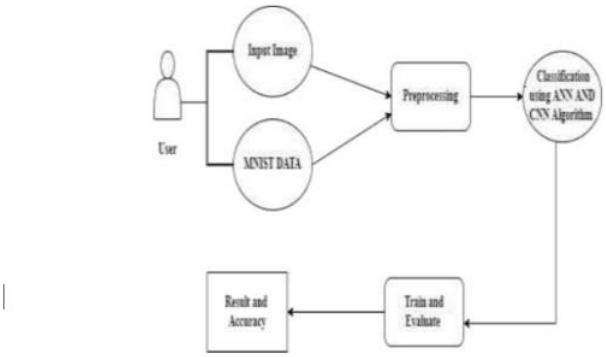
CHAPTER 5

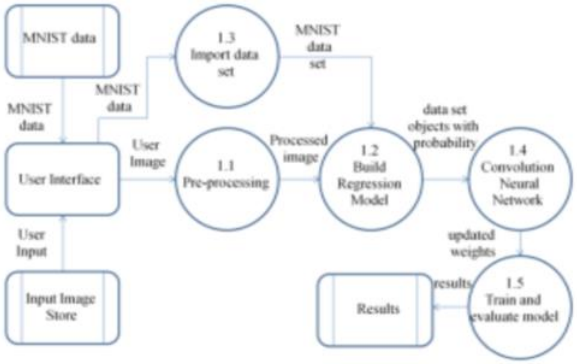
PROJECT DESIGN

5.1 DATA FLOW DIAGRAM

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 enter and leaves the system, what changes the information, and where data is stored.

Example: (Simplified)





5.2 SOLUTION ARCHITECTURE & TECHNOLOGY SLACK

Solution architecture is a complex process – with many sub-processes – that bridges the gap between business problems and technology solutions. Its goals are to:

- Find the best tech solution to solve existing business problems.
- Describe the structure, characteristics, behavior, and other aspects of the software to project stakeholders.
- Define features, development phases, and solution requirements.
- Provide specifications according to which the solution is defined, managed, and delivered.

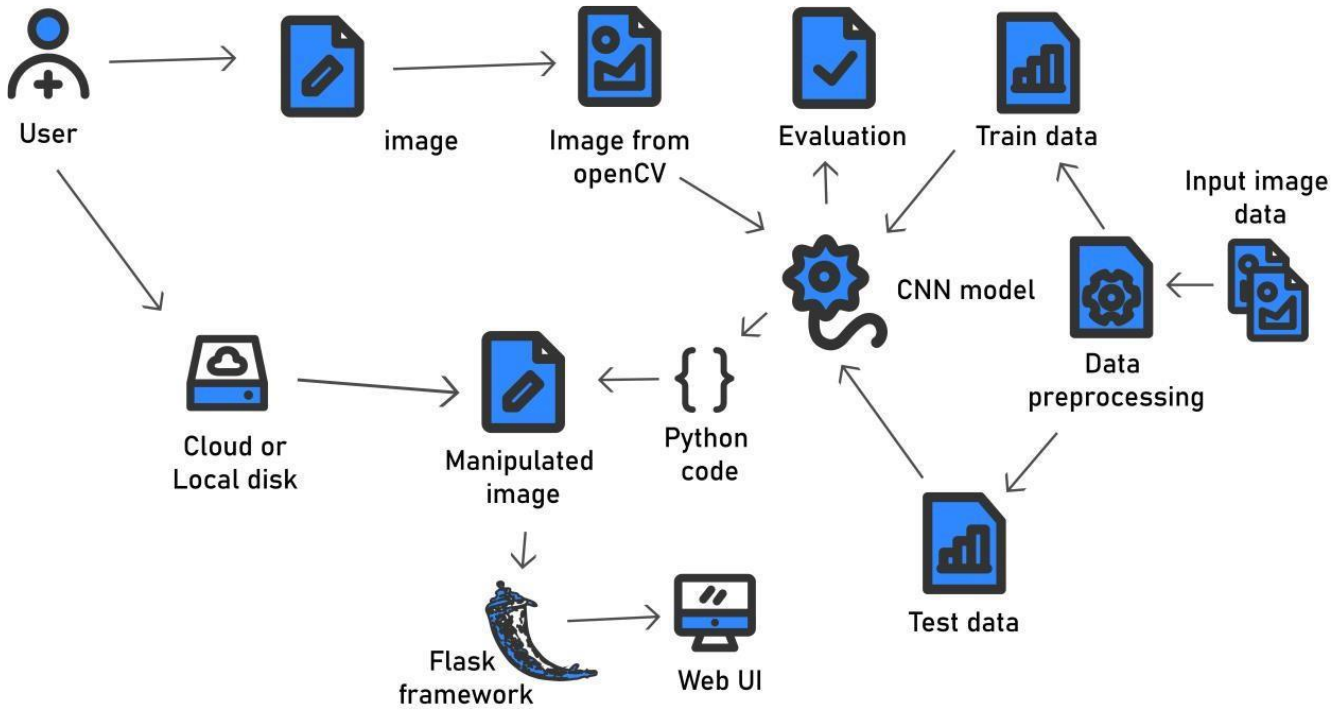


Table-1 : Components & Technologies:

S.No	Component	Description	Technology
1.	User Interface	How user interacts with application e.g. Web UI, Mobile App, Chatbot etc.	HTML, CSS, JavaScript / Angular Js / React Js etc.
2.	Digit Prediction	Here the digit given as input is predicted.	Python
3.	Representation	Skeleton, counters, pixels or others.	Python (CNN)
4.	Segmentation	Task of clustering parts of an image together that belong to the same object class.	Python (CNN)
5.	Database	Data Type, Configurations etc.	MySQL, NoSQL, etc.
6.	Cloud Database	Database Service on Cloud	IBM DB2, IBM Cloudant etc.
7.	File Storage	File storage requirements	IBM Block Storage or Other Storage Service or Local Filesystem
8.	Machine Learning Model	Purpose of Machine Learning Model is to train and test the outcome and improve accuracy.	Object Recognition Model, etc.
9.	Infrastructure (Server / Cloud)	Application Deployment on Local System / Cloud	Local, Cloud Foundry, Kubernetes, etc.

Table-2: Application Characteristics:

S.No	Characteristics	Description	Technology
1.	Open-Source Frameworks	Enables developers to develop complex code and web application quickly	Open source-Jupyter anaconda navigator, flask/django famework.
2.	Security Implementations	The input images are stored in a secure server while being processed and then it gets deleted afterwards.	Encryptions
3.	Availability	This technology will be available for everyone to make use of.	Web servers.
4.	Performance	Work on the Python deep learning project to build a hand written recognition app using MNIST dataset convolutional neural network and a GUI.	Convolutional Neural Networks.

5.3 USER STORIES

Use the below template to list all the user stories for the product.

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobile user)	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	I can access my account / dashboard	High	Sprint-1
		USN-2	As a user, I will receive confirmation email once I have registered for the application	I can receive confirmation email & click confirm	High	Sprint-1
		USN-3	As a user, I can register for the application through Facebook	I can register & access the dashboard with Facebook Login	Low	Sprint-2
		USN-4	As a user, I can register for the application through Gmail	I can register & access the dashboard with Gmail Login	Medium	Sprint-1
	Login	USN-5	As a user, I can log into the application by entering email & password	I can register & access the dashboard with email and password	High	Sprint-1
	Dashboard	USN-6	As a user I can able to read the instruction in the home page	I can read instructions in home page	Low	Sprint-2
Customer (Web user)	Upload	USN-7	As a user I can upload my handwritten digits images to be recognised from the computer.	I can upload images from my computer	High	Sprint-3
Customer Care Executive		USN-8	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
Administrator	Predict	USN-9	As a user I can able to get accurate results based on the training and testing	I can get accurate results	High	Sprint-4
	Accessibility	USN-10	As a user I can use the web application virtually anywhere.	I can use the application portably anywhere.	Medium	Sprint-3
		USN-11	As it is open source ,I can use it free of cost	I can use it without any payment to be paid for it to access.	Low	Sprint-2

CHAPTER 6

PROJECT PLANNING AND SCHEDULING

6.1 SPRINT PLANNING AND ESTIMATION

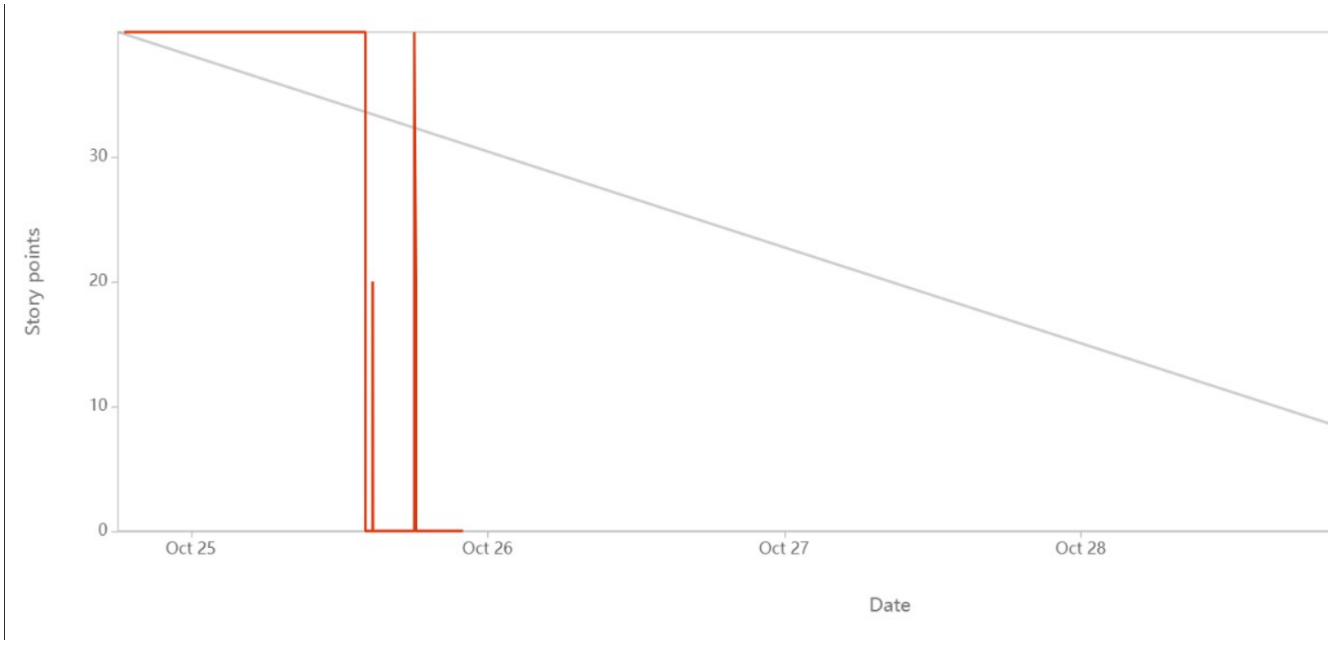
Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Data Collection & pre processing	USN-1	As a user, I can upload any kind of image with the pre-processing step is involved in it.	10	High	Ashwin Venkat, Balan K, Bharath Raj, V R Dhanush
Sprint-1		USN-2	As a user, I can upload the image in any resolution.	10	High	Ashwin Venkat, Balan K, Bharath Raj, V R Dhanush
Sprint-2	Building the Machine learning model	USN-3	As a user, I will get a application with ML model which provides high accuracy of recognized handwritten digit	3	Medium	Bharath Raj, V R Dhanush
Sprint-2		USN-4	As a user, I can pass the handwritten digit image fAor recognizing the digit.	2	Medium	Ashwin Venkat, Balan K,
Sprint-2		USN-5	As a user, I can get the most suitable recognized digit.	10	High	Ashwin Venkat, Balan K, Bharath Raj, V R Dhanush
Sprint-3	Building User Interface Application	USN-6	As a user, I will upload the handwritten digit image to the application by clicking a upload button.	8	Medium	Ashwin Venkat, V R Dhanush
Sprint-3		USN-7	As a user, I can know the details of the fundamental usage of the application.	2	High	Ashwin Venkat, Balan K, Bharath Raj, V R Dhanush
Sprint-3		USN-8	As a user, I can see the predicted / recognized digits in the application	10	Medium	Balan K, Bharath Raj
Sprint-4	Train and deployment of model in IBM Cloud	USN-9	As a user, I can access the web application and make the use of the product from anywhere	20	High	Balan K, V R Dhanush

6.2 SPRINT DELIVERY SCHEDULE

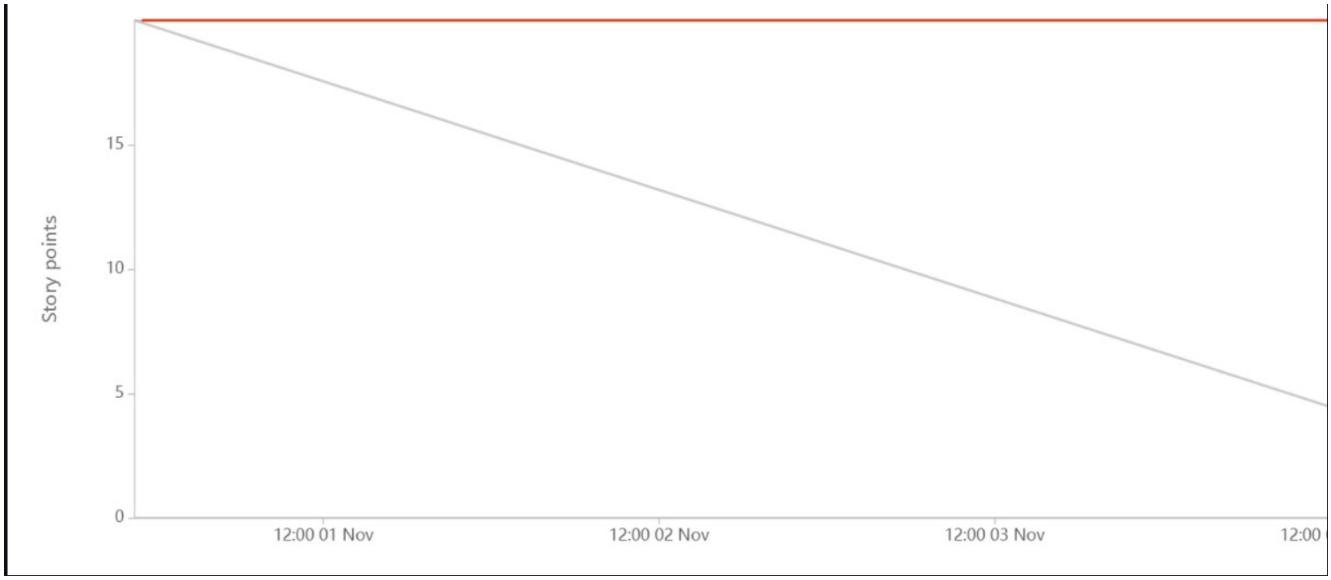
Sprint	Total StoryPoints	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	20	6 Days	24 Oct 202	29 Oct 2022	20	29 Oct 2022
Sprint-2	20	6 Days	31 Oct 202	05 Nov 2022	20	05 Nov 2022
Sprint-3	20	6 Days	07 Nov 202	12 Nov 2022	20	12 Nov 2022
Sprint-4	20	6 Days	14 Nov 202	19 Nov 2022	20	19 Nov 2022

6.3 REPORTS FROM JIRA

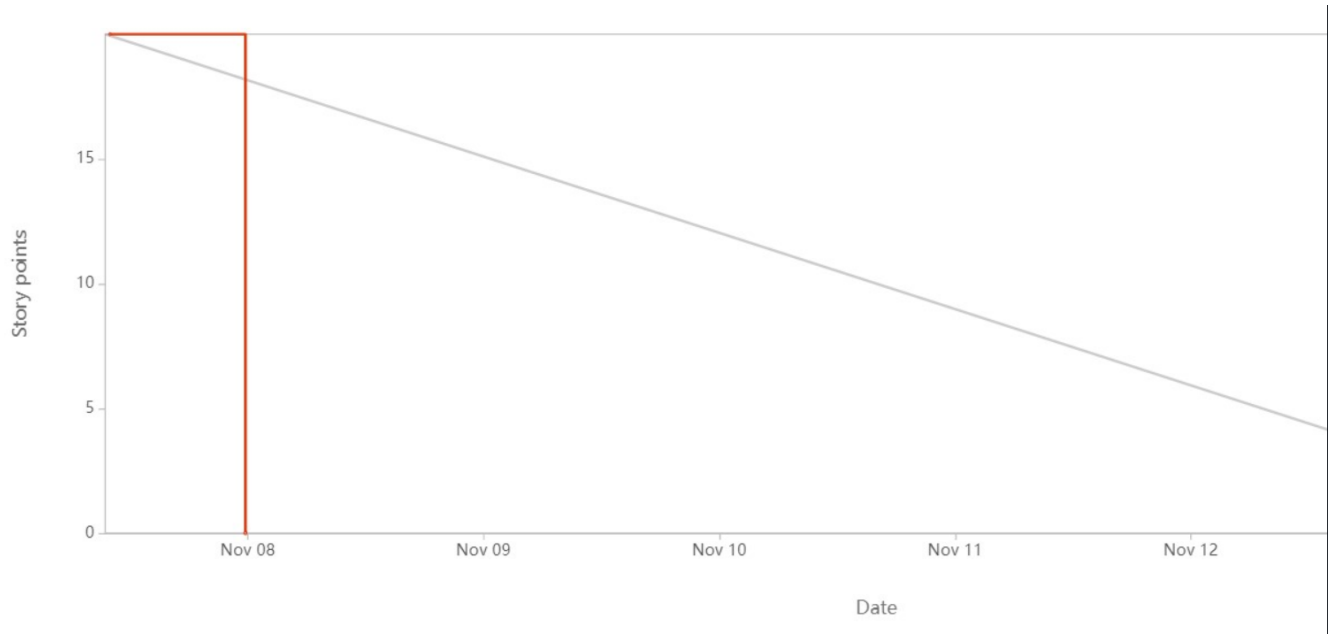
SPRINT 1



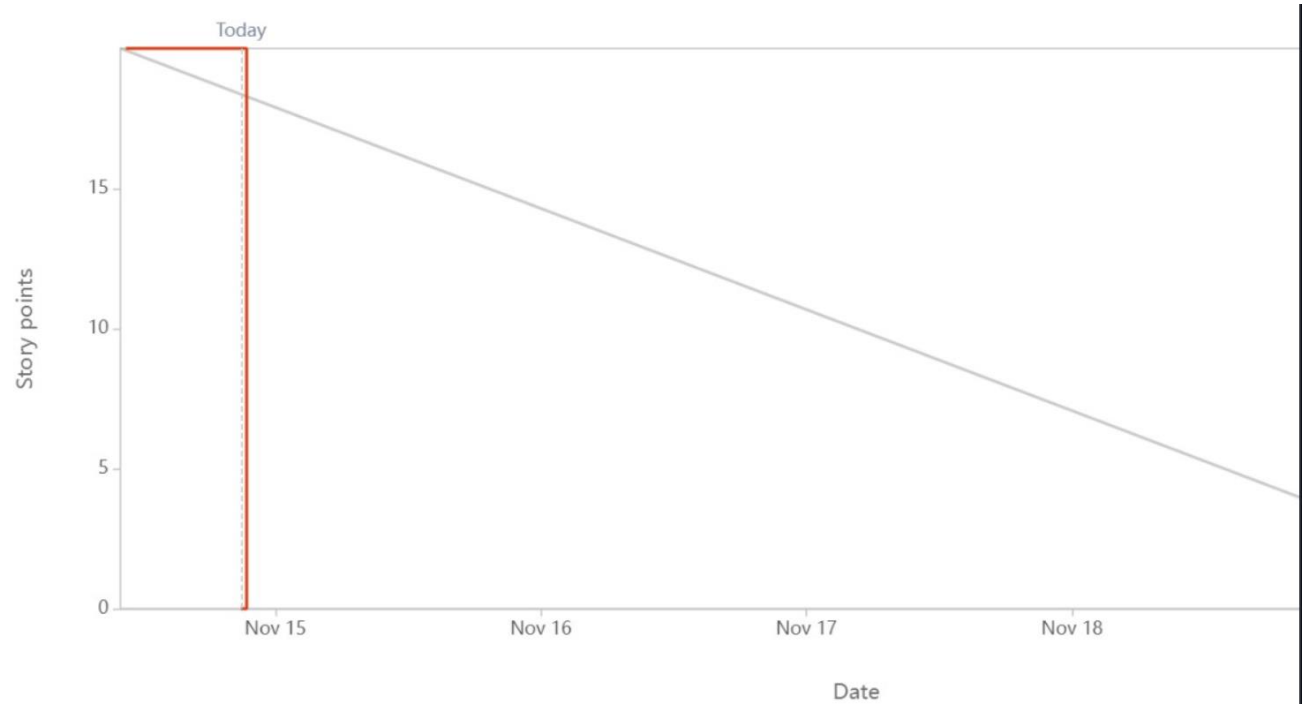
SPRINT 2



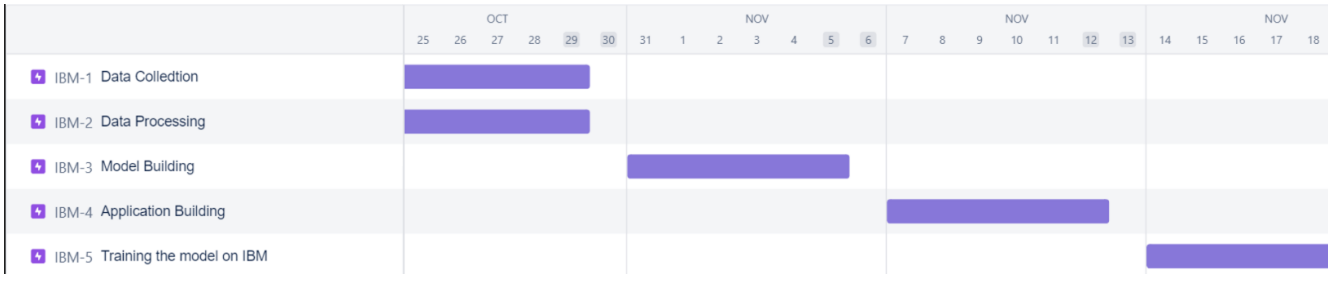
SPRINT 3



SPRINT 4



ROADMAP



CHAPTER 7

CODING & SOLUTIONING

Import the necessary packages

```
[17]
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

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

Data Analysis

```
[3]
print(X_train.shape)
print(X_test.shape)
```

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

```python
X_train[0]
```

[4]

Output exceeds the size limit. Open the full output data in a text editor

```
array([[  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   3,
         18,  18,  18, 126, 136, 175,  26, 166, 255, 247, 127,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,  30,  36,  94, 154, 170,
        253, 253, 253, 253, 253, 225, 172, 253, 242, 195,  64,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,  49, 238, 253, 253, 253, 253,
        253, 253, 253, 253, 251,  93,  82,  82,  56,  39,   0,   0,   0,
          0,   0],
```

```python
y_train[0]
```

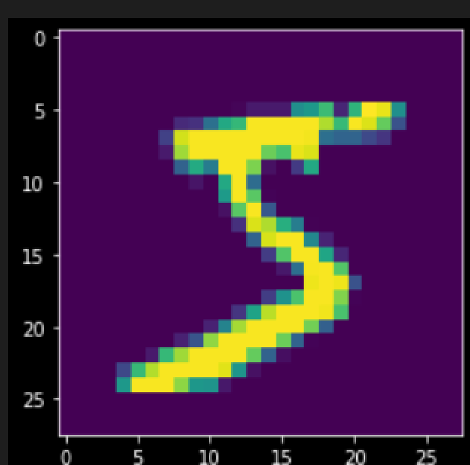
[5]

```
5
```

```python
plt.imshow(X_train[0])
```

[6]

```
<matplotlib.image.AxesImage at 0x1ed310b5930>
```



Data Pre-Processing

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

```
[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)
```

```
[9]
Y_train[0]
```

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

Create model

```
[10]
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

```
[12]
model.fit(X_train, Y_train, batch_size=32, epochs=5, validation_data=(X_test,Y_test))
```

```
Epoch 1/5
1875/1875 [==============================] - 16s 5ms/step - loss: 0.2158 - accuracy: 0.9518 - val_loss: 0.0964 - val_accuracy: 0.9707
Epoch 2/5
1875/1875 [==============================] - 9s 5ms/step - loss: 0.0682 - accuracy: 0.9794 - val_loss: 0.0674 - val_accuracy: 0.9805
Epoch 3/5
1875/1875 [==============================] - 9s 5ms/step - loss: 0.0478 - accuracy: 0.9844 - val_loss: 0.0852 - val_accuracy: 0.9759
Epoch 4/5
1875/1875 [==============================] - 9s 5ms/step - loss: 0.0336 - accuracy: 0.9893 - val_loss: 0.1202 - val_accuracy: 0.9719
Epoch 5/5
1875/1875 [==============================] - 9s 5ms/step - loss: 0.0270 - accuracy: 0.9914 - val_loss: 0.1036 - val_accuracy: 0.9777

<keras.callbacks.History at 0x1ed3324f7f0>
```

Test the model

+ Code + Markdown

```
[13]
metrics = model.evaluate(X_test, Y_test, verbose=0)
print("Metrics (Test Loss & Test Accuracy): ")
print(metrics)
```

```
Metrics (Test Loss & Test Accuracy):
[0.1035672277212143, 0.9776999950408936]
```

```
[14]
prediction = model.predict(X_test[:4])
print(prediction)
```

```
1/1 [==============================] - 0s 177ms/step
[[6.43197941e-15 8.71634543e-21 7.98728167e-11 7.08215517e-12
  2.27718335e-18 1.36703092e-15 2.37176042e-22 1.00000000e+00
  4.51405352e-13 4.25453591e-13]
 [4.56659687e-15 1.54588287e-10 1.00000000e+00 1.20107971e-13
  1.86926159e-19 3.90255250e-20 1.16102319e-11 4.27834925e-23
  7.33884963e-17 1.86307852e-23]
 [1.37352282e-10 9.99961138e-01 3.40877750e-06 1.50240779e-12
  1.99599867e-07 1.10004057e-05 6.72304851e-11 7.78906983e-09
  2.42337919e-05 3.74607870e-13]
 [1.00000000e+00 5.39840355e-16 1.03082355e-10 4.23198737e-17
  8.17481194e-10 2.49619574e-12 1.66041558e-09 5.06253395e-17
  3.02219919e-13 5.55243709e-08]]
```

```
[15]
print(numpy.argmax(prediction, axis=1))
print(Y_test[:4])
```

```
[7 2 1 0]
[[0. 0. 0. 0. 0. 0. 0. 1. 0. 0.]
 [0. 0. 1. 0. 0. 0. 0. 0. 0. 0.]
 [0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
 [1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]]
```

Save the model

```
[16]
model.save("model.h5")
```

Test the saved model

```
[22]
model=load_model("model.h5")
```

```
[23]
img = Image.open("sample.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)
```

```
1/1 [==============================] - 0s 435ms/step
0    8
Name: Label, dtype: int64
```

CHAPTER 8

TESTING

8.1 TEST CASES

Test case ID	Feature Type	Component	Test Scenario	Steps To Execute	Test Data	Expected Result	Actual Result	Status	BUG ID	Executed By
HP_TC_001	UI	Home Page	Verify UI elements in the Home Page	1) Open the page 2) Check if all the UI elements are displayed	127.0.0.1:8000	The Home page must be displayed properly	Working as expected	PASS		Ashwin Venkat Balan K
HP_TC_002	UI	Home Page	Check if the UI elements are displayed properly in different screen sizes	1) Open the page in a specific device 2) Check if all the UI elements are displayed properly 3) Repeat the above steps with different device sizes	-- Screen Sizes -- 2560 x 1801 1440 x 970 1024 x 840 768 x 680 820 x 680	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 680	FAIL	BUG_HP_001	Bharath Raj V R Dhanush
HP_TC_003	Functional	Home Page	Check if user can upload their file	1) Open the page 2) Click on select button 3) Select the input image	Sample 1.png	The input image should be uploaded to the application successfully	Working as expected	PASS		V R Dhanush Balan K
HP_TC_004	Functional	Home Page	Check if user cannot upload unsupported files	1) Open the page 2) Click on select button 3) Select a random input file	Installer.exe	The application should not allow user to select a non image file	User is able to upload any file	FAIL	BUG_HP_002	Ashwin Venkat Bharath Raj
HP_TC_005	Functional	Home Page	Check if the page redirects to the result page once the input is given	1) Open the page 2) Click on select button 3) Select the input image 4) Check if the page redirects	Sample 1.png	The page should redirect to the results page	Working as expected	PASS		Balan K Bharath Raj
BE_TC_001	Functional	Backend	Check if all the routes are working properly	1) Go to Home Page 2) Upload the input image 3) Check the reults page	Sample 1.png	All the routes should properly work	Working as expected	PASS		Balan K Bharath Raj
M_TC_001	Functional	Model	Check if the model can handle various image sizes	1) Open the page in a specific device 2) Upload the input image 3) Repeat the above steps with different input image	Sample 1.png Sample 1 XS.png Sample 1 XL.png	The model should rescale the image and predict the results	Working as expected	PASS		Balan K Bharath Raj
M_TC_002	Functional	Model	Check if the model predicts the digit	1) Open the page 2) Click on select button 3) Select the input image 4) Check the results	Sample 1.png	The model should predict the number	Working as expected	PASS		Balan K Bharath Raj
M_TC_003	Functional	Model	Check if the model can handle complex input image	1) Open the page 2) Click on select button 3) Select the input image 4) Check the results	Complex Sample.png	The model should predict the number in the compex image	The model fails to identify the digit since the model is not built to handle such data	FAIL	BUG_M_001	Ashwin Venkat V R Dhanush
RP_TC_001	UI	Result Page	Verify UI elements in the Result Page	1) Open the page 2) Click on select button 3) Select the input image 4) Checkif all the UI elements are displayed properly	Sample 1.png	The Result page must be displayed properly	Working as expected	PASS		Ashwin Venkat V R Dhanush
RP_TC_002	UI	Result Page	Check if the input image is displayed properly	1) Open the page 2) Click on select button 3) Select the input image 4) Check if the input image are displayed	Sample 1.png	The input image should be displayed properly	The size of the imput image exceeds the display container	FAIL	BUG_RP_001	Ashwin Venkat V R Dhanush
RP_TC_003	UI	Result Page	Check if the result is displayed properly	1) Open the page 2) Click on select button 3) Select the input image 4) Check if the result is displayed	Sample 1.png	The result should be displayed properly	Working as expected	PASS		Ashwin Venkat V R Dhanush
RP_TC_004	UI	Result Page	Check if the other predictions are displayed properly	1) Open the page 2) Click on select button 3) Select the input image 4) Check if all the other predictions are displayed	Sample 1.png	The other predictions should be displayed properly	Working as expected	PASS		Ashwin Venkat V R Dhanush

8.2 USER ACCEPTANCE TESTING

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

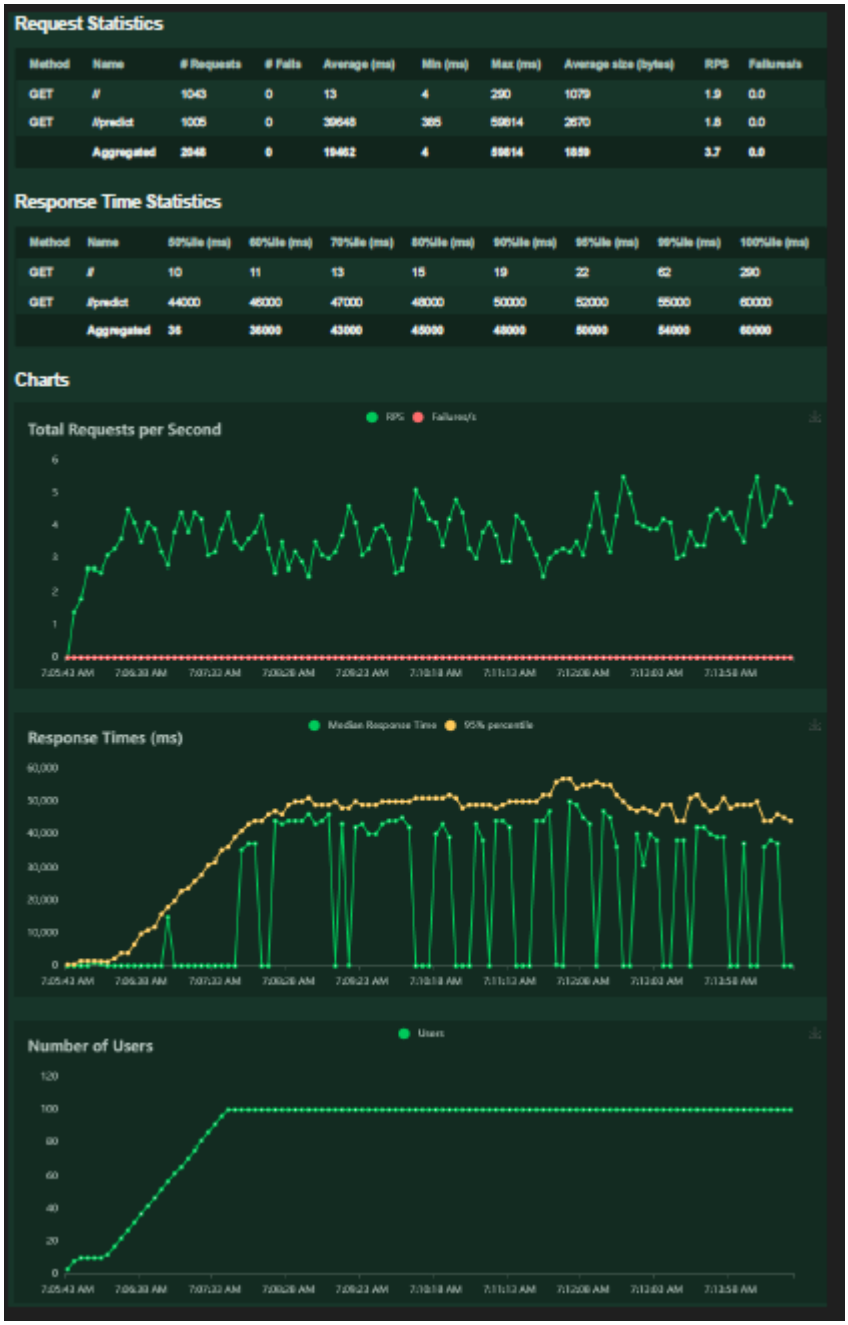
TEST CASE ANALYSIS

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

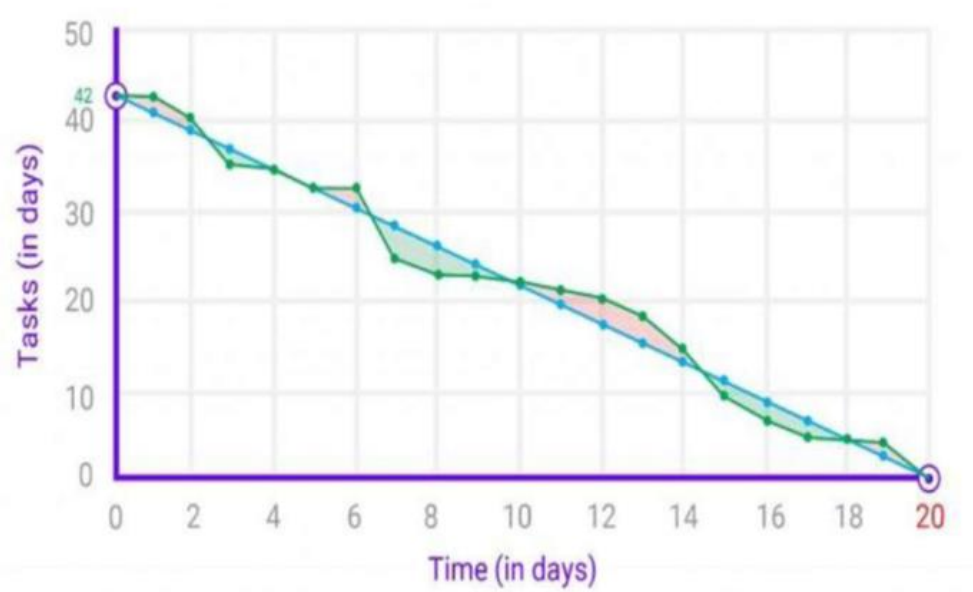
CHAPTER 9

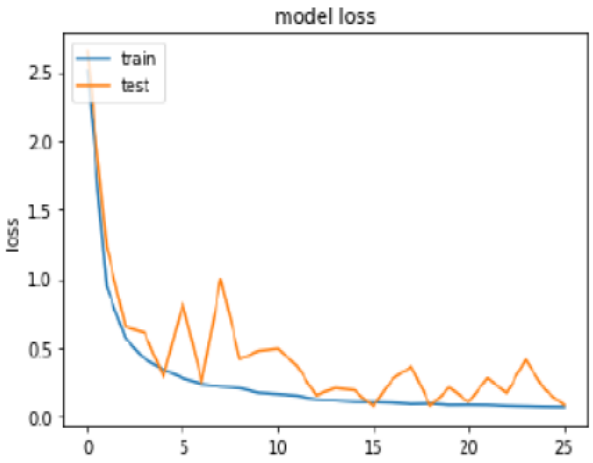
RESULTS

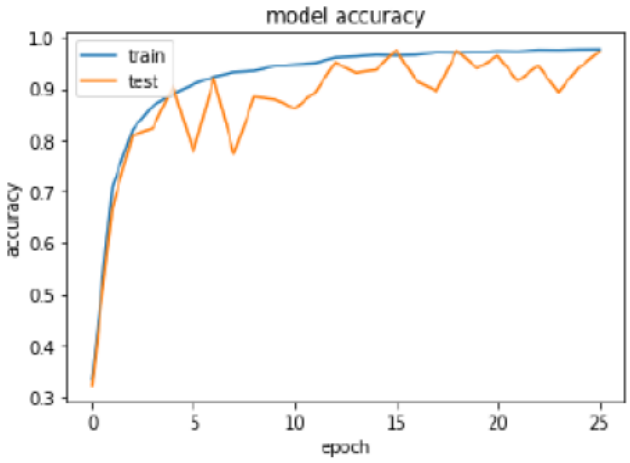
9.1 PERFORMANCE METRICS



BURNDOWN CHART







CHAPTER 10

ADVANTAGES & DISADVANTAGES

ADVANTAGES

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

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

An implementation of Handwritten Digit Recognition using Deep Learning has been implemented in this paper. Additionally, some of the most widely used Machine Learning algorithms i.e. CNN using Tensorflow have been trained and tested on the same data to draw a comparison as to why we require deep learning methods in critical applications like Handwritten Digit Recognition. In this paper, I have shown that that using Deep Learning techniques, a very high amount of accuracy can be achieved. Using the Convolutional Neural Network with Keras and Theano as backend, I am able to get an accuracy of 95.72%. Every tool has its own complexity and accuracy. Although, we see that the complexity of the code and the process is bit more as compared to normal Machine Learning algorithms but looking at the accuracy achieved, it can be said that it is worth it. Also, the current implementation is done only using the CPU Thus we settled on classifying a given handwritten digit image as the required digit using three different algorithms and consequently testing its accuracy.

In future we are planning to further explore the topic to recognize people's handwriting.

CHAPTER 12

FUTURESCOPE

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.

CHAPTER 13

APPENDIX

Understanding the Data

Importing the required libraries

```
Importing Necessary Libraries

import numpy #used for numerical analysis
import tensorflow #open source used for both ML and DL for computation
from tensorflow.keras.datasets import mnist #mnist dataser
from tensorflow.keras.models import Sequential #it is a plain stack of layers
from tensorflow.keras import layers #A layer consists of a tensor-in tensor-out computation function
from tensorflow.keras.layers import Dense, Flatten #Dense-Dense layer is the regular deeply connected n
#Falten-used fot flattening the input or change the dimension
from tensorflow.keras.layers import Conv2D #Convolutional layer
from keras.optimizers import Adam #optimizer
from keras.utils import np_utils #used for one-hot encoding
```

Loading the data

The dataset for this model is imported from the Keras module.

```
load data

(X_train, y_train), (X_test, y_test) = mnist.load_data() #splitting the mnist data into train and test
```

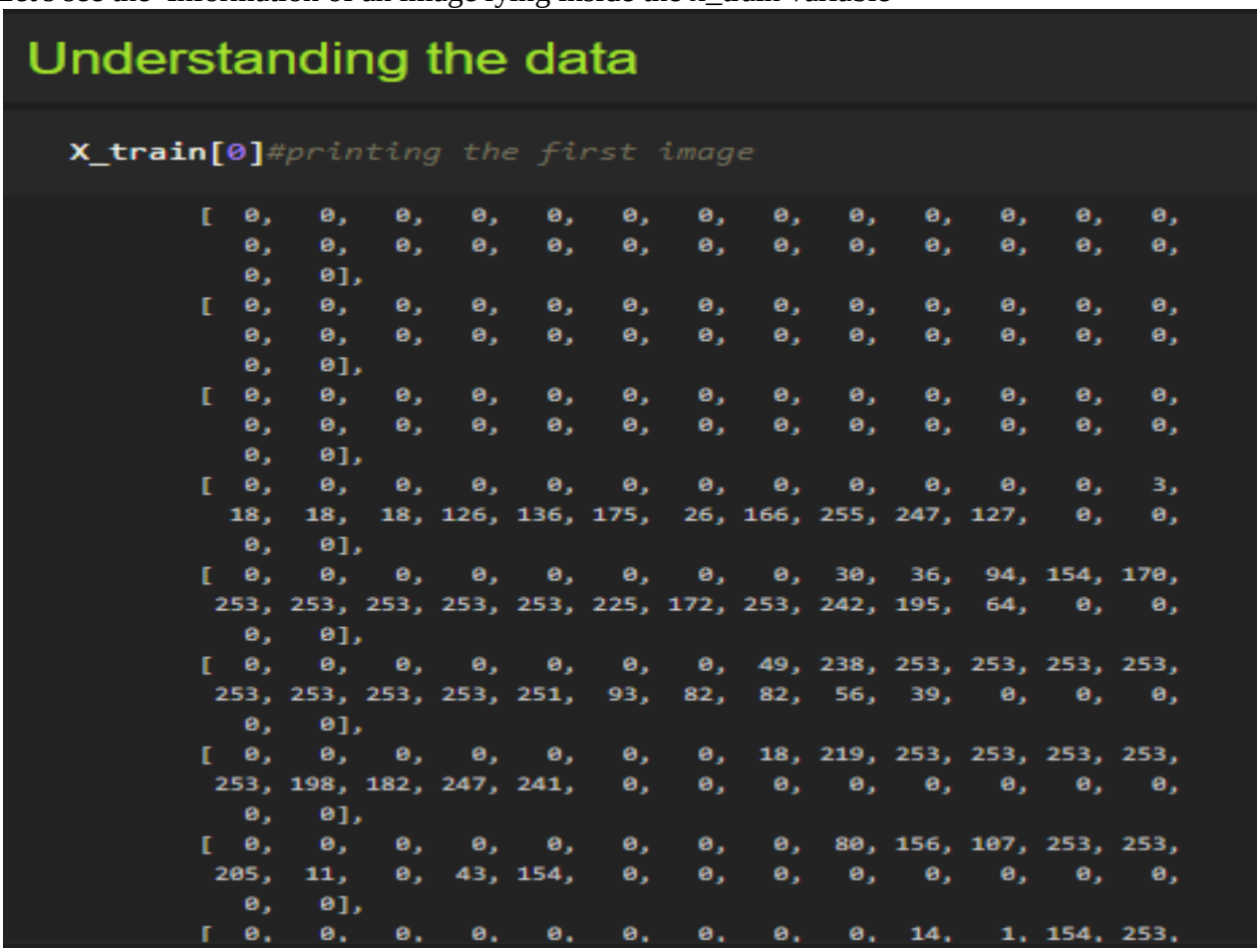
```
print(X_train.shape)#shape is used for give the dimension values #60000-rows 28x28-pixels
print(X_test.shape)

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

We are finding out the shape of X_train and x_test for better understanding. It lists out the dimensions of the data present in it.

In trainset, we have 60000 images, and in the test set we have 10000 images.

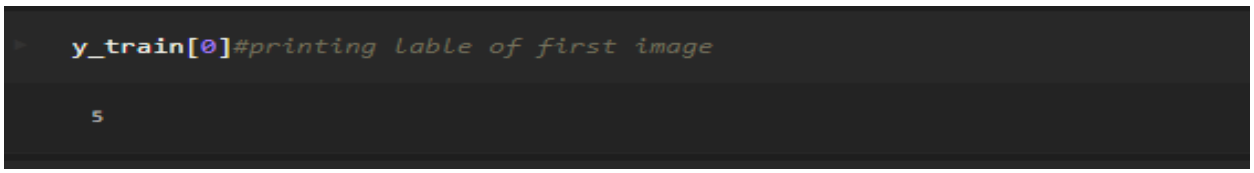
Analyzing the Data

Let's see the Information of an image lying inside the x_train variable



Basically, the pixel values range from 0-255. Here we are printing the first image pixel value which is index [0] of the training data. As you see it is displayed in the output.

With respect to this image, the label of this image will e stored in y_train let's see what is the label of this image by grabbing it from the y_train variable



As we saw in the previous screenshot, we get to know that the pixel values are printed. Now here we are finding to which image the pixel values belong to. From the output displayed we get to know that the image is ‘5’.

```
import matplotlib.pyplot as plt #used for data visualization
plt.imshow(X_train[0]) #ploting the index=0 image
```

```
<matplotlib.image.AxesImage at 0x1c397af32e0>
```

Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. By using the Matplotlib library we are displaying the number ‘5’ in the form of an image for proper understanding.

Reshaping the Data

As we are using Deep learning neural network, the input for this network to get trained on should be of higher dimensional. Our dataset is having three-dimensional images so we have to reshape them too higher dimensions.

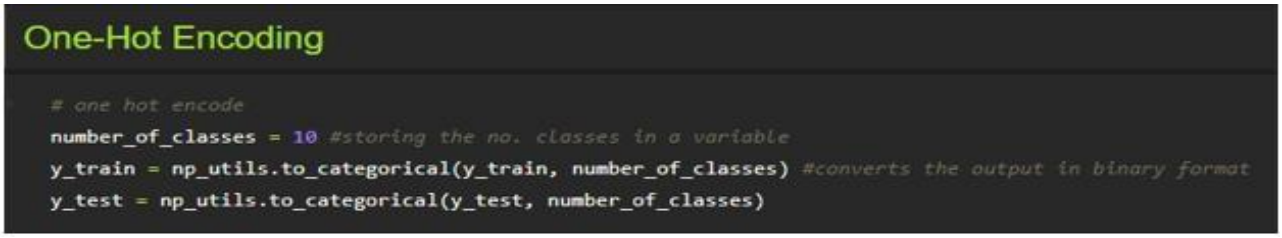
Reshaping Dataset ¶

```
# Reshaping to format which CNN expects (batch, height, width, channels)
X_train = X_train.reshape(60000, 28, 28, 1).astype('float32')
X_test = X_test.reshape(10000, 28, 28, 1).astype('float32')
```

We are reshaping the dataset because we are building the model using CNN. As CNN needs four attributes batch, height, width, and channels we reshape the data.

Applying one hot Encoding

If you see our y_train variable contains Labels representing the images containing in x_train. AS these are numbers usually they can be considered as numerical or continuous data, but with respect to this project these Numbers are representing a set of class so these are to be represented as categorical data, and we need to binaries these categorical data that's why we are applying One Hot encoding for y_train set



One hot encoding is a process by which categorical variables are converted into a form that could be provided to ML algorithms to do a better job in prediction. We apply One-Hot Encoding in order to convert the values into 0's and 1's. For a detailed point of view, look at this link

Model Building

Adding CNN layers

Creating the model and adding the input, hidden, and output layers to it.

Creating the Model

```
# create model
model = Sequential()
# adding model layer
model.add(Conv2D(64, (3, 3), input_shape=(28, 28, 1), activation='relu'))
model.add(Conv2D(32, (3, 3), activation='relu'))
#model.add(Conv2D(32, (3, 3), activation='relu'))
#flatten the dimension of the image
model.add(Flatten())
#output layer with 10 neurons
model.add(Dense(number_of_classes, activation='softmax'))
```

The Sequential model is a linear stack of layers.

Compiling the Model

With both the training data defined and model defined, it's time to configure the learning process. This is accomplished with a call to the compile () method of the Sequential model class. Compilation requires 3 arguments: an optimizer, a loss function, and a list of metrics.

Compiling the model

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

MNIST Dataset Description

Handwritten writing 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 analysed 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 labelled 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 indicates darker pixel). 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 hidden layer to the model to make it Multi-Layer Perceptron.
- Add Dropout to prevent overfitting and check its effect on accuracy.
- Increasing the number of Hidden Layer neuron and check 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 all most mimic the human brain and are a key ingredient in 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.

Training the model

Fitting the model

```
# Fit the model
model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=5, batch_size=32)
```

```
Epoch 1/5
1875/1875 [==============================] - 184s 98ms/step - loss: 0.2451 - accuracy: 0.9497 - val_loss: 0.0966 - val_accuracy: 0.9
715
Epoch 2/5
1875/1875 [==============================] - 183s 98ms/step - loss: 0.0694 - accuracy: 0.9785 - val_loss: 0.0971 - val_accuracy: 0.9
714
Epoch 3/5
1875/1875 [==============================] - 183s 98ms/step - loss: 0.0487 - accuracy: 0.9850 - val_loss: 0.0829 - val_accuracy: 0.9
782
Epoch 4/5
1875/1875 [==============================] - 177s 94ms/step - loss: 0.0382 - accuracy: 0.9877 - val_loss: 0.0881 - val_accuracy: 0.9
769
```

Observing the Metrics

Observing the metrics

```
# Final evaluation of the model
metrics = model.evaluate(X_test, y_test, verbose=0)
print("Metrics(Test loss & Test Accuracy): ")
print(metrics)
```

```
Metrics(Test loss & Test Accuracy):
[0.1097492054104805, 0.9753000140190125]
```

Testing the Model

Predicting the output

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

```
[[5.50544734e-15 7.41999492e-20 5.00876077e-12 1.26642463e-09
  3.52252804e-21 1.54133163e-17 3.15550259e-21 1.00000000e+00
  1.32678888e-13 6.44072333e-14]
 [1.51885260e-08 8.02883537e-09 1.00000000e+00 6.44802788e-13
  6.37117113e-16 3.40490114e-15 2.15804121e-08 2.18907611e-19
  3.38496564e-10 2.07915498e-20]
 [3.14093924e-08 9.99941349e-01 2.01593957e-06 1.45100779e-10
  5.25237965e-06 1.59223120e-07 3.15299786e-08 1.53995302e-07
  5.09846941e-05 1.14552066e-07]
 [1.00000000e+00 1.35018288e-14 2.28308122e-10 1.79766094e-16
  1.28767550e-14 7.12401882e-12 2.92727509e-11 3.52439052e-13
  2.56207252e-12 2.32345068e-12]]
```

```
import numpy as np
print(np.argmax(prediction,axis=1)) #printing our labels from first 4 images
print(y_test[:4]) #printing the actual labels
```

```
[7 2 1 0]
[[0. 0. 0. 0. 0. 0. 0. 1. 0. 0.]
 [0. 0. 1. 0. 0. 0. 0. 0. 0. 0.]
 [0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
 [1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]]
```

Observing the Metrics

Observing the metrics

```
# Final evaluation of the model
metrics = model.evaluate(X_test, y_test, verbose=0)
print("Metrics(Test loss & Test Accuracy): ")
print(metrics)
```

```
Metrics(Test loss & Test Accuracy):
[0.1097492054104805, 0.9753000140190125]
```

SOURCE CODE

Creating an HTML file:

```
<!DOCTYPE html>

<html>

<head>

<title>Home</title>

<style>

Body

{

background:url("https://dm0qx8t0i9gc9.cloudfront.net/thumbnails/video/BXJT8TRDeiymbx54w
/videoblocks-binary-code-black-and-green-background-with-digits-moving-on-
screen_bswlauoflw_thumbnail-1080_01.png");

background-attachment: fixed;

background-color: black;

}

. pd{

padding-bottom:100%;}

.navbar

{

margin: 0px;

padding:20px;

background-color:rgb(219, 213, 213);

opacity:0.6;
```

```
color:black;
font-family:'Roboto',sans-serif;
font-style: italic;
border-radius:20px;
font-size:25px;
}
a
{ss
color:grey;
float:right;
text-decoration:none;
font-style:normal;
padding-right:20px;
}
a:hover{
background-color:black;
color:white;
border-radius:15px;
font-size:30px;
padding-left:10px;
}
P
{
```

```
color:black;

font-style:italic;

font-size:30px;

font-style: normal;

font-weight: 700;

}

</style>

</head>

<body>

<div class="navbar">

<a href="/upload">Recognize</a>

<a href="/home">Home</a>

<br>

</div>

<br>

<center><b class="pd"><font color="black" size="15" font-family="Comic Sans MS" >Handwritten Recognition System</font></b></center>

<div>

<br>

<center>

<p>
</center>

</div>

</body>

</html>
```

Web.html

```
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta http-equiv="X-UA-Compatible" content="ie=edge">
<title>Recognize</title>
<link href="https://cdn.bootcss.com/bootstrap/4.0.0/css/bootstrap.min.css" rel="stylesheet">
<script src="https://cdn.bootcss.com/popper.js/1.12.9/umd/popper.min.js"></script>
<script src="https://cdn.bootcss.com/jquery/3.3.1/jquery.min.js"></script>
<script src="https://cdn.bootcss.com/bootstrap/4.0.0/js/bootstrap.min.js"></script>
<link href="{{ url_for('static', filename='css/main.css') }}" rel="stylesheet">
<style>
body[
background:
url("https://dm0qx8t0i9gc9.cloudfront.net/thumbnails/video/BXJT8TRDeiymbx54w/videoblocks=-binary-code-black-and-green-background-with-digits-moving-on-screen_bswlauoflw_thumbnail-1080_01.png");
background-attachment: fixed;
background-size: cover;
}
.bar
{
margin: 0px;
 padding:20px;
```

```
background-color:rgb(205, 202, 202);
opacity:0.6;
color:black;
font-family:'Roboto',sans-serif;
font-style: italic;
border-radius:20px;
font-size:25px;
}
a
{
color:grey;
float:right;
text-decoration:none;
font-style:normal;
padding-right:20px;
}
a:hover{
background-color:black;
color:white;
border-radius:15px;
font-size:30px;
padding-left:10px;
}
body
{
```

```
        background-color: rgb(251, 251, 250);
    }
    </style>
    </head>
    <body>
    <div class="bar">
    <a href="/upload" >Recognize</a>
    <a href="/home">Home</a>
    <br>
    </div>
        <div class="container">
          <center> <div id="content" style="margin-top:2em">{% block content %}{% endblock %}</div></center>
</div>
</body>
<footer>
        <script src="{{ url_for('static', filename='js/main.js') }}" type="text/javascript"></script>
</footer>
    </html>
```

APP.py

```
import os
 import numpy as np #used for numerical analysis
 from flask import Flask,request,render_template
 # Flask-It is our framework which we are going to use to run/serve our application.
 #request-for accessing file which was uploaded by the user on our application.
 #render_template- used for rendering the html pages
 import  tensorflow as tf
 from tensorflow.keras.models import load_model #to load our trained model
 from tensorflow.keras.preprocessing import image
 from PIL import Image
 app=Flask(_name_)#our flask app
 model=load_model('mnistCNN.h5')#loading the model
 @app.route("/") #default route
 def about():
    render_template("main.html")#rendering html page
 @app.route("/home") #default route
 def home():
    return render_template("main.html")#rendering html page
 @app.route("/upload") #default route
 def test():
    return render_template("index6.html")#rendering html page
 @app.route("/predict",methods=["GET","POST"]) #route for our prediction
 def upload_image_file():
   if request.method == 'POST':
```

```
img = Image.open(request.files['file'].stream).convert("L")
img = img.resize((28,28))
im2arr = np.array(img)
im2arr = im2arr.reshape(1,28,28,1)
y_pred = model.predict(im2arr)
predict = np.argmax(y_pred)
print(predict)
#return "Predicted Number: + str(pred) returning our output
if(predict == 0):
    return render_template("0.html")
elif(predict == 1):
    return render_template("1.html")
elif(predict == 2):
    return render_template("2.html")
elif(predict == 3):
    return render_template("3.html")
elif(predict == 4):
    return render_template("4.html")
elif(predict == 5):
    return render_template("5.html")
elif(predict == 6):
    return render_template("6.html")
elif(predict == 7):
    return render_template("7.html")
elif(predict == 8):
```

```
            return render_template("8.html")
        elif(predict == 9):
            return render_template("9.html")
        else:
            return None
if __name__=="__main__":
    #app.run(debug=False)#running our app
    app.run(host='0.0.0.0', port=8000)
```

GITHUB:

https://github.com/IBM-EPBL/IBM-Project-23569-1659887598

VIDEO LINK:

https://drive.google.com/file/d/1WqK6ZdntdFolNPfXJSTU0vbrlX68keL1/view?usp=sharing