PROJECT REPORT

A Novel Method for Handwritten Digit Recognition System

Submitted By

Team ID-PNT2022TMID39529

Naveenkumar V-510519205011

Arun kumar R-510519205003

Silambarasan B-510519205025

Tharun R K-510519205028

6. PROJECT PLANNING & SCHEDULING

6.1 Sprint Planning & Estimation

6.2 Sprint Delivery Schedule

6.3 Reports from JIRA

7. CODING & SOLUTIONING (Explain the features added in the project along with code)

7.1 Feature

1 7.2 Feature

2 7.3 Database Schema (if Applicable)

8. TESTING

8.1 Test Cases

8.2 User Acceptance Testing

9. RESULTS

9.1 Performance Metrics

10. ADVANTAGES & DISADVANTAGES

11. CONCLUSION

12. FUTURE SCOPE

13. APPENDIX

Source Code

Git Hub & Project Demo Link

1. INTRODUCTION:

Character recognition is a fundamental, but most challenging in the field of pattern recognition with large number of useful applications. It has been an intense field of research since the early days of computer science due to it being a natural way of interactions between computers and humans. More precisely Character recognition is the process of detecting and recognizing characters from the input image and converts it into ASCII or other equivalent machine editable form [1][2].

1.1 PROJECT OVERVIEW:

The technique by which a computer system can recognize characters and other symbols written by hand in natural handwriting is called handwriting recognition system. Handwriting recognition is classified into offline system. Handwriting recognition is classified into offline [3]. If handwriting is scanned and then understood by the computer, it is called offline handwriting recognition. In case, the handwriting is recognized while writing through touch pad using stylus pen, it is called online handwriting recognition. From the classifier perspective, character recognition systems are classified into two main categories (analytic). The segmentation free also known as the holistic approach to recognize the character without segmenting it into subunits or characters. Each word is represented as a set of global features, e.g. ascender, loops, cusp, etc.

1.2 PURPOSE:

Handwritten character processing systems are domain and application specific, like it is not possible to design a generic system which can process all kinds of handwritten scripts and language. Lots of work has been done on European languages and Arabic (Urdu) language. Whereas domestic languages like Hindi, Punjabi, Bangla, Tamil, Gujarati etc.

2 LITERATURE SURVEY :

An early notable attempt in the area of character recognition research is by Grimsdale in 1959. The origin of a great deal of research work in the early sixties was based on an approach known as analysis-by-synthesis method suggested by Eden in 1968. The great importance of Eden's work was that he formally proved that all handwritten characters are formed by a finite number of schematic features, a point that was implicitly included in previous works. This notion was later used in all methods in syntactic (structural) approaches of character recognition. K. Gaurav, Bhatia P. K. [5] Et al, this paper deals with the various pre-processing techniques involved in the character recognition with different kind of images ranges from a simple handwritten form based documents and documents containing colored and complex background and varied intensities. In this, different preprocessing techniques like skew detection and correction, image enhancement techniques of contrast stretching, binarization, noise removal techniques, normalization and segmentation, morphological processing techniques are discussed.

2.1 EXISTING PROBLEM:

Pre-processing is the basic phase of character recognition and it's crucial for good recognition rate. The main objective of pre-processing steps is to normalize strokes and remove variations that would otherwise complicate recognition and reduce the recognition rate. These variations or distortions include the irregular size of text, missing points during pen movement collections, jitter present in text, left or right bend in handwriting and uneven distances of points from neighbouring positions.

2.2 REFERENCES:

[1] R. Jayadevan, S. R. Kolhe, P. M. Patil, and U. Pal, “Automatic processing of handwritten bank cheque images: a survey,” *Int. J. Doc. Anal. Recognit. IJDAR*, vol. 15, no. 4, pp. 267–296, 2012.
[2] N. M. Nasrabadi, “Pattern recognition and machine learning,” *J. Electron. Imaging*, vol. 16, no. 4, p. 049901, 2007.

[3] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *nature*, vol. 521, no. 7553, p. 436, 2015.
[4] Y. Bengio, “Learning deep architectures for AI,” *Found. Trends® Mach. Learn.*, vol. 2, no. 1, pp. 1–127, 2009.
[5] R. Plamondon and S. N. Srihari, “Online and off-line handwriting recognition: a comprehensive survey,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 1, pp. 63–84, 2000.
[6] R. M. Bozinovic and S. N. Srihari, “Off-line cursive script word recognition,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 11, no. 1, pp. 68–83, 1989.
[7] K. Marukawa, M. Koga, Y. Shima, and H. Fujisawa, “A High Speed Word Matching Algorithm for Handwritten Chinese Character Recognition.,” in *MVA*, 1990, pp. 445–450.
[8] R. W. Smith, J. F. McNamara, and D. S. Bradburn, “Pattern Classification Techniques Applied to Character Segmentation,” *TRW Financ. Syst. Berkeley CA*, 1998.
[9] F. Lauer, C. Y. Suen, and G. Bloch, “A trainable feature extractor for handwritten digit recognition,” *Pattern Recognit.*, vol. 40, no. 6, pp. 1816–1824, 2007.
[10] L. Bottou *et al.*, “Comparison of classifier methods: a case study in handwritten digit recognition,” in *Pattern Recognition, 1994. Vol. 2-Conference B: Computer Vision & Image Processing., Proceedings of the 12th IAPR International. Conference on*, 1994, vol. 2, pp. 77–82.
[11] G. S. Lopes, D. C. da Silva, A. W. O. Rodrigues, and P. P. Reboucas Filho, “Recognition of handwritten digits using the signature features and Optimum-Path Forest Classifier,” *IEEE Lat. Am. Trans.*, vol. 14, no. 5, pp. 2455–2460, 2016.
[12] L. S. Oliveira, R. Sabourin, F. Bortolozzi, and C. Y. Suen, “A methodology for feature selection using multiobjective genetic algorithms for handwritten digit string recognition,” *Int. J. Pattern Recognit. Artif. Intell.*, vol. 17, no. 06, pp. 903–929, 2003.
[13] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.
[14] N. Arica and F. T. Yarman-Vural, “An overview of character recognition focused on off-line handwriting,” *IEEE Trans. Syst. Man Cybern. Part C Appl. Rev.*, vol. 31, no. 2, pp. 216–233, 2001.
[15] H. Murase, M. Shinya, T. Wakahara, and K. Odaka, “Segmentation and recognition of handwritten character string using linguistic information,” *Trans Inst. Electron. Inf. Commun. Eng.*, vol. 69, no. 9, pp. 1292–1301, 1986.
[16] D. Lu and Q. Weng, “A survey of image classification methods and techniques for improving classification performance,” *Int. J. Remote Sens.*, vol. 28, no. 5, pp. 823–870, 2007.
[17] U. Pal, A. Belaıd, and C. Choisy, “Touching numeral segmentation using water reservoir concept,” *Pattern Recognit. Lett.*, vol. 24, no. 1–3, pp. 261–272, 2003.
[18] “Machine learning,” *Wikipedia*. 14-Jan-2019.
[19] A. Talwar and Y. Kumar, “Machine Learning: An artificial intelligence methodology,” *Int. J. Eng. Comput. Sci.*, vol. 2, no. 12, 2013.

[20] “Deep learning,” *Wikipedia*. 14-Jan-2019.
[21] D. C. Cireşan, U. Meier, L. M. Gambardella, and J. Schmidhuber, “Deep, big, simple neural nets for handwritten digit recognition,” *Neural Comput.*, vol. 22, no. 12, pp. 3207–3220, 2010.
[22] N. Dalal and B. Triggs, “Histograms of oriented gradients for human detection,” in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, 2005, vol. 1, pp. 886–893.
[23] N. Otsu, “A threshold selection method from gray-level histograms,” *IEEE Trans. Syst. ManCybern.*, vol.

2.3 PROBLEM STATEMENT:

This is a collection of thousands of handwritten pictures used to train classification models using machine learning Techniques. As a part of this problem statement, we will train a multilayer perceptron using tensor flow.

3. IDEATION & PROPOSED SOLUTION:

3.1 EMPATHY MAP :

An emoathy map is a collaborative tool teams can use to gain a deeper insight into their customers. Much like a user personal, an empathy map can represent a group of users, such as customer segment. The empathy map was originally created by Dave Gray and has gained much popularity within the agile community.

Edit this template
Right click to unlock

Empathy Map Canvas

Gain insight and understanding on solving customer problems.

1

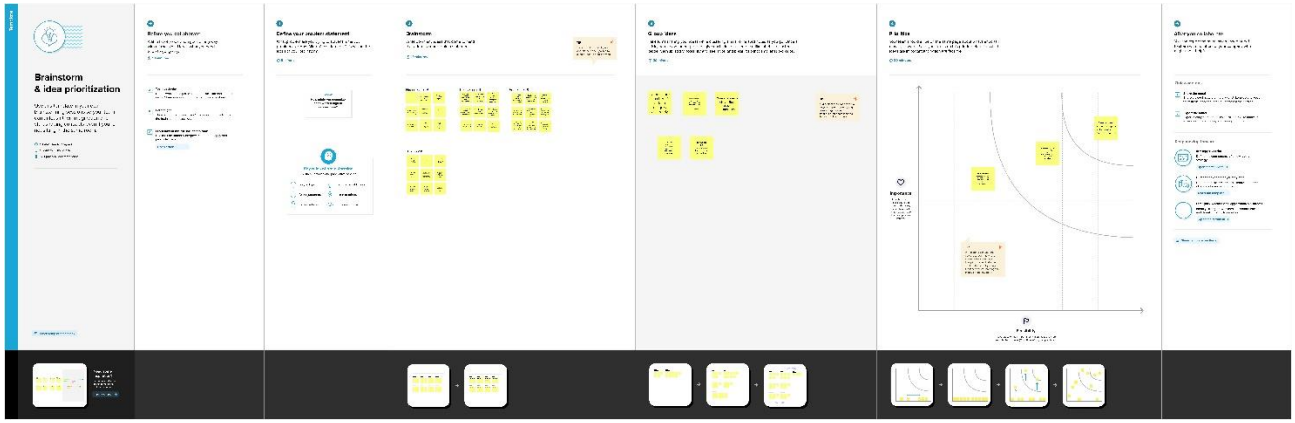
Build empathy and keep your focus on the user by putting yourself in their shoes.

Share your feedback

3.2 BRAIN STROM:

Brainstorming is a group creativity technique by which efforts are made to find a conclusion for a specific problem by gathering by list of ideas spontaneously contributed by its members.

- Pick an appropriate facilitator.
- Set the agenda.
- Holding the session.

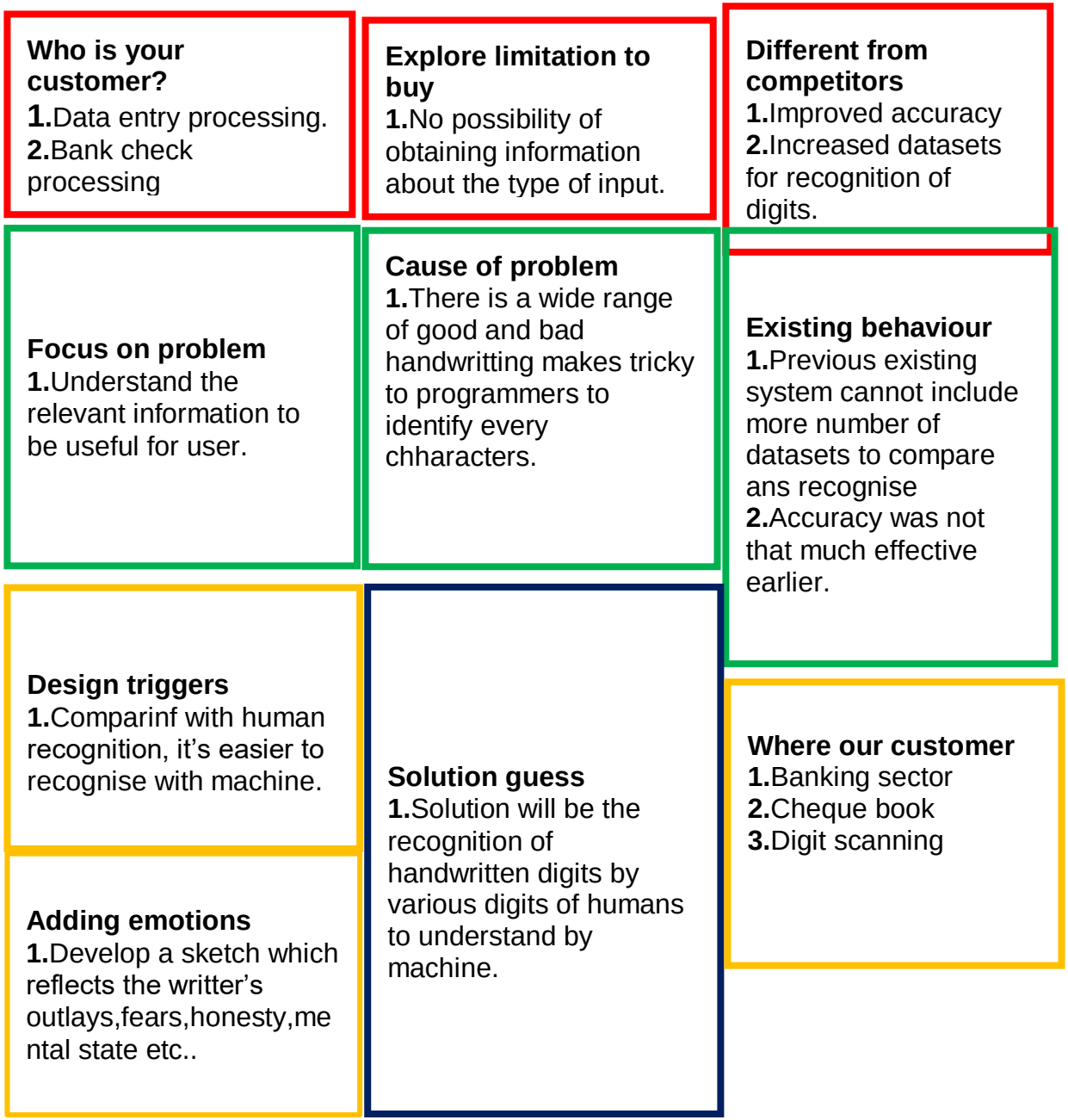


3.3 PROPOSED SOLUTION:

Key rationale toward optical character recognition (OCR) from handwritten image includes features extraction technique supported by a classification algorithm for recognition of characters based on the features. Previously, several algorithms for feature classifications and extraction have been utilized for the purpose of character recognition. But, with the advent of CNN in deep learning, no separate algorithms are required for this purpose. However, in the area of computer vision, deep learning is one of the outstanding performers for both feature extraction and classification

However, DNN architecture consists of many nonlinear hidden layers with a enormous number of connections and parameters. Therefore, to train the network with very less amount of samples is a very difficult task. In CNN, only few set of parameters are needed for training of the system. So, CNN is the key solution capable to map correctly datasets for both input and output by varying the trainable parameters and number of hidden layers with high accuracy [49]. Hence, in this work, CNN architecture with Deeplearning4j (DL4J) framework is considered as best fit for the character recognition from the handwritten digit images. For the experiments and verification of system's performance, the normalized standard MNIST dataset is utilized.

3.4 PROBLEM SOLUTION FIT:



4 REQUIREMENT ANALYSIS:

FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	One of the very significant problems in pattern recognition applications is the recognition of handwritten characters. Applications for digit recognition include filling out forms, processing bank checks, and sorting mail.
NFR-2	**Security**	1) The system generates a thorough 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 categories 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 type of hosting determines how much space is allotted to a website on a server. Shared, dedicated, VPS, and reseller hosting are the four basic varieties.
FR-3	Digit Classifier Model: To train a convolutional network to predict the digit from an image, use the MNIST database of handwritten digits. get the training and validation data first.

FR-4	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.

Non-functional Requirements:

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

		reveal information like the writing style, in addition to a categorization of the digit. 2) The generative models are capable of segmentation driven by recognition. 3) The procedure uses a relatively.
NFR-3	**Reliability**	The samples are used by the neural network to automatically deduce rules for reading handwritten digits. Furthermore, the network may learn more about handwriting and hence enhance its accuracy by increasing the quantity of training instances. Numerous techniques and algorithms, such as Deep Learning/CNN, SVM, Gaussian Naive Bayes, KNN, Decision Trees, Random Forests, etc., can be used to recognise handwritten numbers.

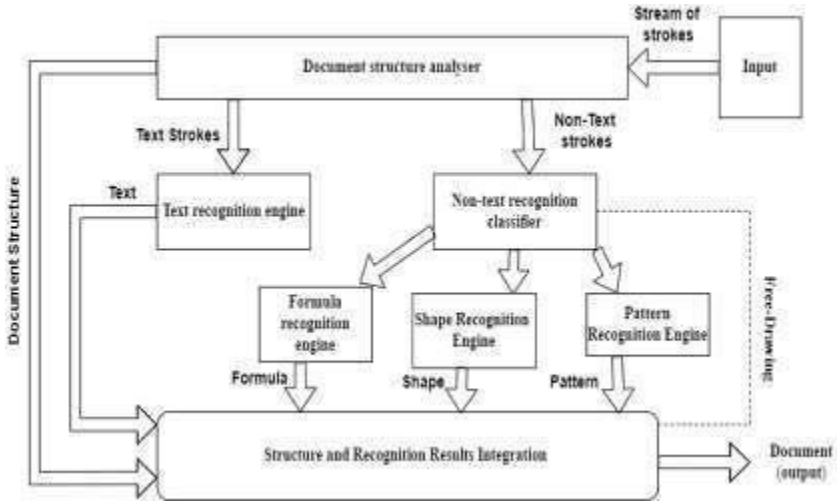
NFR-4	**Accuracy**	With typed text in high-quality photos, optical character recognition (OCR) technology offers accuracy rates of greater than 99%. However, variances in spacing, abnormalities in handwriting, and the variety of human writing styles result in less precise character identification.
NFR-5	**Availability**	

5 PROJECT DESIGN:

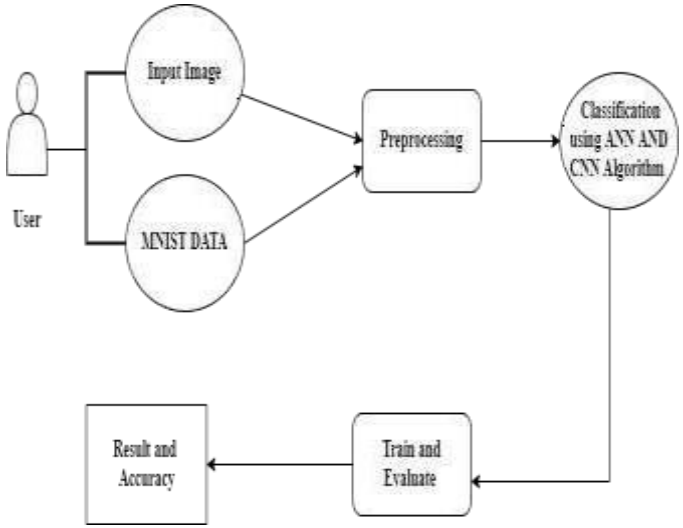
5.1 DATA FLOW DIAGRAM:

It's easy to understand the flow of data through systems with the right data flow diagram software. This guide provides everything you need to know about data flow diagrams, including definitions, history, and symbols and notations. You'll learn the different levels of a DFD, the difference between a logical and a physical DFD and tips for making a DFD.

Example: DFD Level 0 (Industry Standard)


Stream of strokes
Input
Document structure analyser
Text Strokes
Non-Text strokes
Text
Text recognition engine
Non-text recognition classifier
Document Structure
Formula recognition engine
Shape Recognition Engine
Pattern Recognition Engine
Free-Drawing
Formula
Shape
Pattern
Structure and Recognition Results Integration
Document (output)

Simplified diagram:



5.2 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)	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

5.3 SOLUTION ACHITECTURE:

Solution architecture is a complex process – with many sub-processes – that bridgesthe 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 thesoftware to project stakeholders.
- Define features, development phases, and solution requirements.
- Provide specifications according to which the solution is defined, managed,and delivered.

Solution Architecture Diagram:

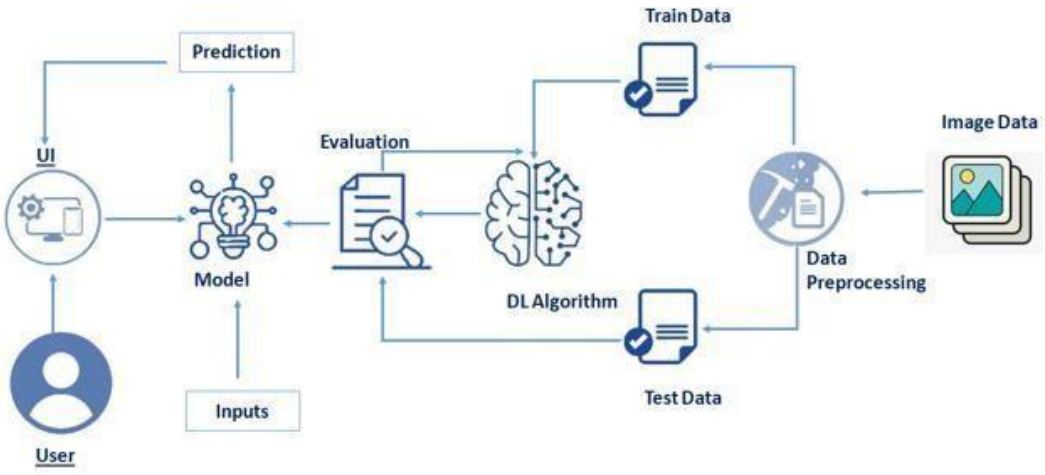


Figure 1: Architecture of the Handwritten digit recognition system.

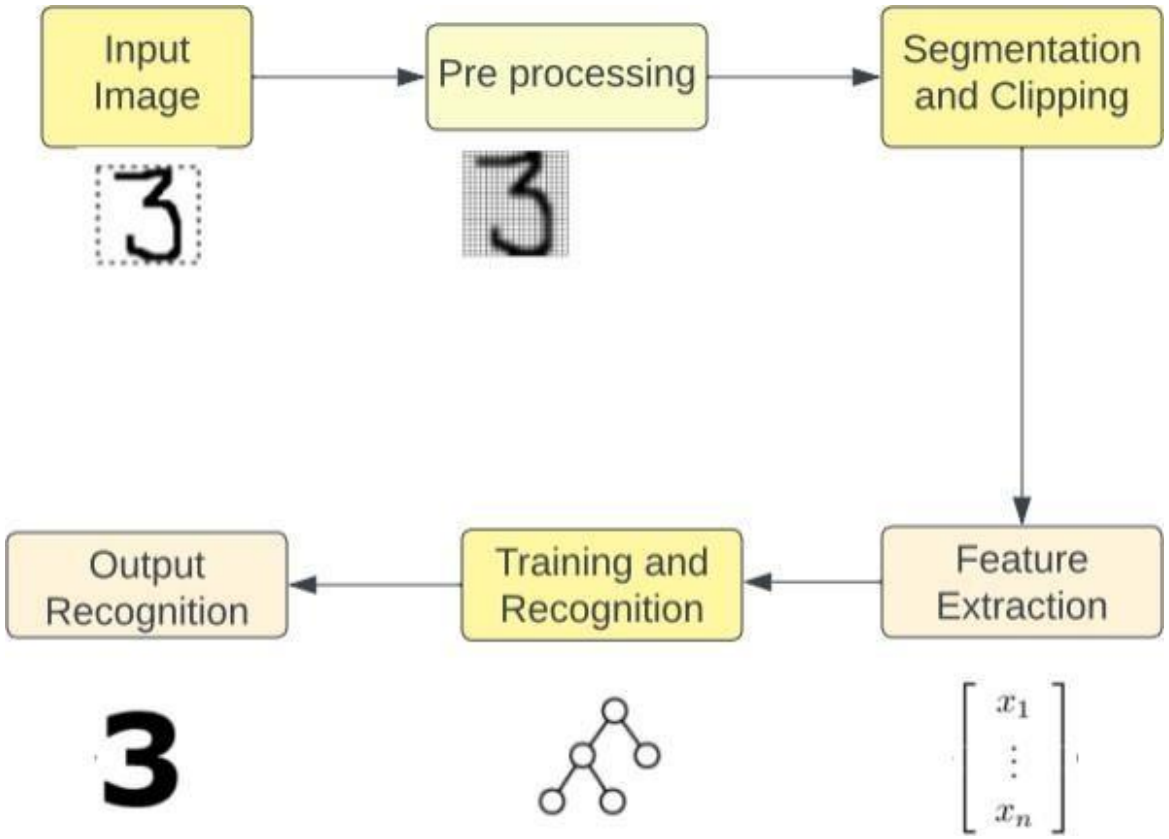


Figure 2: Model Architecture of the Deep Learning model used for Handwritten digit recognition.

6 PROJECT PLANNING:

6.1 SPRINT PLAN:

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members

Sprint-1	Data Collection	USN-1	As a user, I can collect the dataset from various resources with different handwritings.	10	Low	Naveenkumar V
Sprint-1	Data Preprocessing	USN-2	As a user, I can load the dataset, handling the missing data, scaling and split data into train and test.	10	Medium	Arun kumar R
Sprint-2	Model Building	USN-3	As a user, I will get an application with ML model which provides high accuracy of recognized handwritten digit.	5	High	Silambarasan B
Sprint-2	Add CNN layers	USN-4	Creating the model and adding the input, hidden, and output layers to it.	5	High	Tharun R K

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-2	Compiling the model	USN-5	With both the training data defined and model defined, it's time to configure the learning process.	2	Medium	Naveenkumar V
Sprint-2	Train & test the model	USN-6	As a user, let us train our model with our image dataset.	6	Medium	Arun kumar R
Sprint-2	Save the model	USN-7	As a user, the model is saved & integrated with an android application or web application in order to predict something.	2	Low	Silambarasan B
Sprint-3	Building UI Application	USN-8	As a user, I will upload the handwritten digit image to the application by clicking a upload button.	5	High	Tharun R K
Sprint-3		USN-9	As a user, I can know the details of the fundamental usage of the application.	5	Low	Naveenkumar V Silambarasan B
Sprint-3		USN-10	As a user, I can see the predicted / recognized digits in the application.	5	Medium	Arunkumar R Tharun R K
Sprint-4	Train the model on IBM	USN-11	As a user, I train the model on IBM and integrate flask/Django with scoring end point.	10	High	Naveenkumar V Arun kumar R
Sprint-4	Cloud Deployment	USN-12	As a user, I can access the web application and make the use of the product from anywhere.	10	High	Silambarasan B Tharun R K

6.2 SPRINT DELIVERY:

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

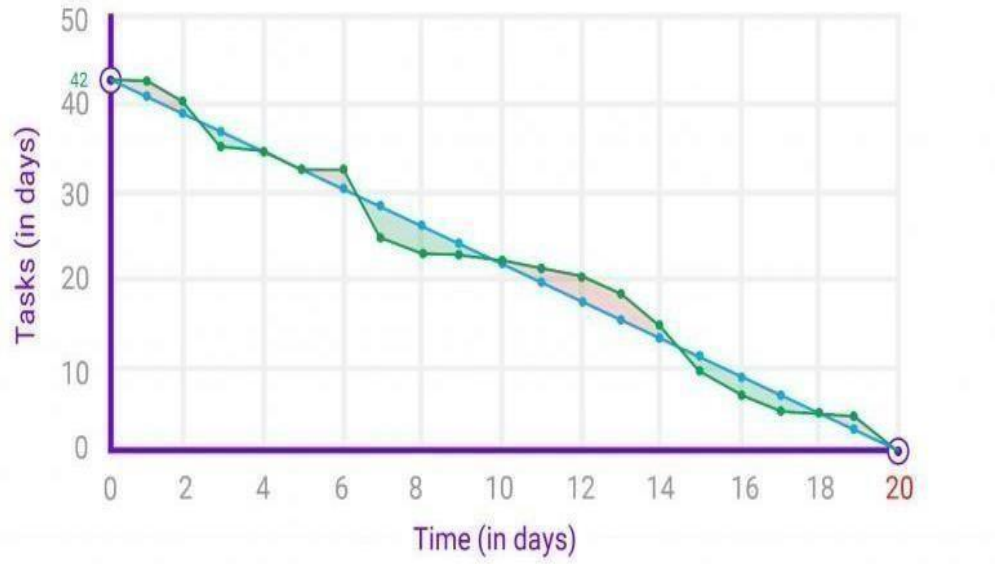
Velocity:

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

Average Velocity = 20 / 6 = 3.33

Burndown Chart:

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



7 CODING AND SOLUTION:

```
{
 "cells": [
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "### Import the necessary packages"
   ]
  },
  {
```

```
 "cell_type": "code",
 "execution_count": 17,
 "metadata": {},
 "outputs": [],
 "source": [
  "import numpy as npm\n",
  "import pandas as pds\n",
  "import matplotlib.pyplot as plt\n",
  "from keras.utils import np_utils\n",
  "from tensorflow.keras.datasets import mnist\n",
  "from tensorflow.keras.models import Sequential\n",
  "from tensorflow.keras.layers import Conv2D, Dense, Flatten\n",
  "from tensorflow.keras.optimizers import Adam\n",
  "from tensorflow.keras.models import load_model\n",
  "from PIL import Image, ImageOps"
 ]
},
{
 "cell_type": "markdown",
 "metadata": {},
 "source": [
  "### Load data"
 ]
},
{
 "cell_type": "code",
 "execution_count": 2,
 "metadata": {},
 "outputs": [],
 "source": [
  "(X_train, y_train), (X_test, y_test) = mnist.load_data()"
 ]
},
{
 "cell_type": "markdown",
 "metadata": {},
 "source": [
```

```
  "### Data Analysis"
 ]
},
{
 "cell_type": "code",
 "execution_count": 3,
 "metadata": {},
 "outputs": [
  {
   "name": "stdout",
   "output_type": "stream",
   "text": [
    "(60000, 28, 28)\n",
    "(10000, 28, 28)\n"
   ]
  }
 ],
 "source": [
  "print(X_train.shape)\n",
  "print(X_test.shape)"
 ]
},
{
 "cell_type": "code",
 "execution_count": 4,
 "metadata": {},
 "outputs": [
  {
   "data": {
    "text/plain": [
     "array([[  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,\n",
     "          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,\n",
     "          0,   0],\n",
     "       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,\n",
     "          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,\n",
     "          0,   0],\n",
     "       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,\n",
```

```
"         0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,\n",
"         0,   0],\n",
"      [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,\n",
"         0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,\n",
"         0,   0],\n",
"      [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,\n",
"         0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,\n",
"         0,   0],\n",
"      [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   3,\n",
"        18,  18,  18, 126, 136, 175,  26, 166, 255, 247, 127,   0,
0,\n",
"         0,   0],\n",
"      [  0,   0,   0,   0,   0,   0,   0,   0,  30,  36,  94, 154, 170,\n",
"       253, 253, 253, 253, 253, 225, 172, 253, 242, 195,  64,   0,
0,\n",
"         0,   0],\n",
"      [  0,   0,   0,   0,   0,   0,   0,  49, 238, 253, 253, 253, 253,\n",
"       253, 253, 253, 253, 251,  93,  82,  82,  56,  39,   0,   0,
0,\n",
"         0,   0],\n",
"      [  0,   0,   0,   0,   0,   0,   0,  18, 219, 253, 253, 253, 253,\n",
"       253, 198, 182, 247, 241,   0,   0,   0,   0,   0,   0,   0,   0,\n",
"         0,   0],\n",
"      [  0,   0,   0,   0,   0,   0,   0,   0,  80, 156, 107, 253, 253,\n",
"       205,  11,   0,  43, 154,   0,   0,   0,   0,   0,   0,   0,   0,\n",
"         0,   0],\n",
"      [  0,   0,   0,   0,   0,   0,   0,   0,   0,  14,   1, 154, 253,\n",
"        90,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,\n",
"         0,   0],\n",
"      [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0, 139, 253,\n",
"       190,   2,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,\n",
"         0,   0],\n",
"      [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  11, 190,\n",
"       253,  70,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,\n",
"         0,   0],\n",
"      [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  35,\n",
"       241, 225, 160, 108,   1,   0,   0,   0,   0,   0,   0,   0,   0,\n",
```

```
    "          0,   0],\n",
    "       [ 0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,\n",
    "         81, 240, 253, 253, 119,  25,   0,   0,   0,   0,   0,   0,   0,\n",
    "          0,   0],\n",
    "       [ 0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,\n",
    "          0,  45, 186, 253, 253, 150,  27,   0,   0,   0,   0,   0,   0,\n",
    "          0,   0],\n",
    "       [ 0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,\n",
    "          0,   0,  16,  93, 252, 253, 187,   0,   0,   0,   0,   0,   0,\n",
    "          0,   0],\n",
    "       [ 0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,\n",
    "          0,   0,   0,   0, 249, 253, 249,  64,   0,   0,   0,   0,   0,\n",
    "          0,   0],\n",
    "       [ 0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,\n",
    "          0,  46, 130, 183, 253, 253, 207,   2,   0,   0,   0,   0,   0,\n",
    "          0,   0],\n",
    "       [ 0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  39,\n",
    "        148, 229, 253, 253, 253, 250, 182,   0,   0,   0,   0,   0,
0,\n",
    "          0,   0],\n",
    "       [ 0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  24, 114, 221,\n",
    "        253, 253, 253, 253, 201,  78,   0,   0,   0,   0,   0,   0,   0,\n",
    "          0,   0],\n",
    "       [ 0,   0,   0,   0,   0,   0,   0,   0,  23,  66, 213, 253, 253,\n",
    "        253, 253, 198,  81,   2,   0,   0,   0,   0,   0,   0,   0,   0,\n",
    "          0,   0],\n",
    "       [ 0,   0,   0,   0,   0,   0,  18, 171, 219, 253, 253, 253, 253,\n",
    "        195,  80,   9,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,\n",
    "          0,   0],\n",
    "       [ 0,   0,   0,   0,  55, 172, 226, 253, 253, 253, 253, 244,
133,\n",
    "         11,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,\n",
    "          0,   0],\n",
    "       [ 0,   0,   0,   0, 136, 253, 253, 253, 212, 135, 132,  16,
0,\n",
    "          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,\n",
    "          0,   0],
```

```
    "       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,\n",
    "          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,\n",
    "          0,   0],\n",
    "       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,\n",
    "          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,\n",
    "          0,   0],\n",
    "       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,\n",
    "          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,\n",
    "          0,   0]], dtype=uint8)"
   ]
  },
  "execution_count": 4,
  "metadata": {},
  "output_type": "execute_result"
 }
],
"source": [
 "X_train[0]"
]
},
{
"cell_type": "code",
"execution_count": 5,
"metadata": {},
"outputs": [
 {
  "data": {
   "text/plain": [
    "5"
   ]
  },
  "execution_count": 5,
  "metadata": {},
  "output_type": "execute_result"
 }
],
"source": [
```

"y_train[0]"
]
},
{
"cell_type": "code",
"execution_count": 6,
"metadata": {},
"outputs": [
{
"data": {
"text/plain": [
"<matplotlib.image.AxesImage at 0x1ed310b5930>"
]
},
"execution_count": 6,
"metadata": {},
"output_type": "execute_result"
},
{
"data": {
"image/png":
"iVBORw0KGgoAAAANSUhEUgAAAPsAAAD4CAYAAAAq5pAIA
AAAOXRFWHRTb2Z0d2FyZQBNYXRwbG90bGliIHZlcnNpb24zLjU
uMiwgaHR0cHM6Ly9tYXRwbG90bGliLm9yZy8qNh9FAAAACXBI
WXMAAAsTAAALEwEAmpwYAAAQtklEQVR4nO3df0zUaX4H8D
cCt1dkFVxw5g4sUE+6rLst1B3qBd1lU6uSXAKmXVe9RHprR7srNba
0hZqm9Jq9BJJ6RPesibPjiYmc5x5HoE2p42KyIXaXG/eGXy7CcguUX
zOgRMS9vcqPp3+YZc+VeQZnvvNDPu9XYiLz5sv3k9l9+x3mmZknC
oACES17K8I9ABGFBstOJATLTiQEy04kBMtOJERMKE92Z3wKnsG
JUJ6SSBRTWjIS1q5eNAuo7Dt27MCJEycQHR2Nd955B1VVVdrv9w
xO4HBueSCnJCKNU7+o9Jr5/TB+xYoVOHXqFAoKCvDcc89h7969y
MrK8vfHEVGQ+V323Nxc9PX1ob+/HzMzM7h48SIKCwuNnI2IDOR3
2VNSUjA0NLTw9fDwMFSUh75PqvVCqfTCafTidXJq/w9HREFKOj
PxttsNlgsFlgsFkxN3A326YjIC7/LPjIygr1i18nZqaipGREUOGIiLj+
V12p9OJDRs2ID09HbGxsdizZw8aGxuNnI2IDOT30tvc3Bxfl
REdH4+zZs/j444+NnI2IDBTQOntTUxMoampqMmoWIgogvlyUSgmUn
EoJlJxKCZScSgmUnEoJlJxKCZScSgmUnEoJlJx

KCZScSgmUnEoJlJxKCZScSgmUnEoJlJxKCZScSgmUnEoJlJxKCZSc
SgmUnEoJlJxKCZScSgmUnEoJlJxKCZScSgmUnEiKgXVwp8kXF6P8
TRycnBfX8PX+X7jWbi5vXHpu2flybx70Zpc3dP/ya1+yXL/5Ue+ytuc+
0+R+/W6rNv/W3H2rzcAio7P39/Ziensbc3BxmZ2dhsViMmouIDBbwlf
2VV17B7du3jZiFiIKIv7MTCRFQ2ZVScDgcuH79OqxW66LfY7Va4X
Q64XQ6sTp5VSCnI6IABPQwfsuWLRgdHUVycjKuXLmCmzdvoqWl
5aHvsdlssNlsAIAeZ18gpyOiAAR0ZR8dHQUATExMoL6+Hrm5uYY
MRUTG87vscXFxiI+PX/j79u3b0dXVZdhgRGQsvx/Gm0wm1NfXP/gh
MTGora3F5cuXDRtsOYnuO2qDN1VOx2nz05QRt/vlm72vCa1br14tb/lC
/3hxOTb9+WptX/WinNm99odZr1j/zufbYSs+favNvtihtHon8Lnt/fz+ys7
MNHIWIgolLb0RCsOxEQrDsREKw7ERCsOxEQvAtrgaYy/8jbf7Dc6e
0eWas97diLmczak6b//Pbf6HNYz7TL399+90Sr9nTI7PaY5+6pV+ai7ve
qs0jEa/sREKw7ERCsOxEQrDsREKw7ERCsOxEQrDsREJwnd0AT/W
MavOPfrNOm2fGeowcx1ClY5u1+af39B9FfW79z7xmU/P6dXLTyf/R5
sH05L2B1Tde2YmEYNmJhGDZiYRg2YmEYNmJhGDZiYRg2YmE4
Dq7AWbH3Nr87apXtfkPduo/7jm6I16bt7/5tjbXeevWH2jzvm1x2nzuzp
g23/ftN71mA0e0hyID7fpvoMfCKzuRECw7kRAsO5EQLDuRECw7kR
AsO5EQLDuREFxnD4E1P/5Amyf/xzPafO72pDbf+PzrXrMbL53VHtt4
5mVtvvZOYO8pj/rA+1p5hv5uIYP5vLLb7XZ4PB50dnYu3AjHw8JCQEc0YiMoDPsp87dw47dz686X15eTmam5uRmZmJ5
uZmlJeXB21fz6nd1kMsHtfvB6cLLfbDZPJ5PV7rVYh48CABYnbzKn9MR
kQEMeTZeKe8fz2ez2WCxWGCxWDA1cdeI0xGRH/wqu9VqhdPpBNBN
zWaMj48bOOhOQRGc+vsjc+wm6kpIStLj0+BBgAUFxcjHMTu3btDPu4cHM
n5+UhKSsLQ0BAqKipw9erV4+4+D7cdeWCCiPVSeBrLJer9n2zbzDMpF7/9OQO4yO7F04Y
5e/piChAfpX99OnTWL63/V6bzrCDT5Z5KUGb4rLNKpBk5xGGsf3NHh4ZhrnZhdV
5ZHlsYyjt/7sTWuW+l3JmmL0MQoQHbZjNobFvqBphWRPZ8ZJhh
UeB8lv3cuXPYuXPYnI7dXV1cjJycHHOTk5aGpqCspwRGQcn2nRdV+5JxQwGsJ/3FurLnDDXN8tqzhyQD0cpBHJlbSRMADWrxWvnn2XnlZ0qIe+uOecw+OrN6OqN7TbxdEKczfb/n3vvf6JNlYAeDACyqJAsmgpRAJV/7Rb8BwS+dKEcq/Xiab/AQCLQ7Yz09NaRiKUKoCtPW5Op7RCXlQYZZ6cfgSkjJgVkvW4sW/+
5e/piChAfpX99OnTWL9+PbKzszE2Nobjx497/V6bzQaLxKQKLxYKpib

t+D0pEgfGr7OPj45ifn4dSCjabDbm5uUbPRUQG86vsZrN54e+7du1CV
1eXYQMRUXD4XGevra1Ffn4+kpKSMDQ0hIqKCuTn5yM7OxtKKQ
wMDODQoUOhmJXCIOpamzb/9Z+v1eaW1/7aa9ZadkJ77M1X3tHm3
03frs2ntmhjcXyWfd++fY/cdvbs2aAMQ0TBw5fLEgnBshMJwbITCcG
yEwnBshMJwbe4UkDmPOPa3HTSe/6bf5jVHhsX9TVtbkv/T23+nV1H
vf/s+lbtscsRr+xEQrDsREKw7ERCsOxEQrDsREKw7ERCsOxEQnCdn
bTmt2Rr81+9+nVt/nz2gNfM1zq6L29P5mjzuIbrAf385YZXdiIhWHYiI
Vh2IiFYdiIhWHYiIVh2IiFYdiIhuM6+zEW9+Lw27z3i4z3jeTXa/KWv
33/smZbq/9SMNv9wMkP/A+bHDJzmyccrO5EQLDuRECw7kRAsO5E
QLDuRECw7kRAsO5EQXGd/AsRkpGnzX33vm16zf3ntovbYP4u/5dd
MRjjmeVGbv39iszZPrPnAyHGWPZ9X9tTUVFy9ehU3btxAV1cXjhw
5AgBITEyEw+FAb28vHA4HEhISgj0rEQXAZ9lnZ2dRWlqKjRs3YvP
mzTh8+DCysrJQXl6O5uZmGeXl5aGYl4LPsbrcbLpcL
AHDv3j10d3cjJSUFhYWFqKl58FLKmpoaFBBUVBXVQIgrMY/3Onpa
WhpycHLS2tsJkMsHtdgN48A+CyWRa9Bir1YqDBw8CAFYnrwpwX
CLy15KfjV+5ciXq6upw9OhRTE9PP5IrpRY9zmazwXy9Nj2b3b1Oh4Bw+XKxwGKxYGrir
v+TElFAllT2mJgY1NXV4cKFC6ivrwcAeDwemM1mAIDZbMb4uH43
TyIKryU9jLfb7ju7kZ1dfXCbY2NjSguLkZVVRWKi4vR0NAQtCGfd
DHpv6vNpzZ9Q5u/9q//rc3/KuHnjz2TUUrH9MtjH/y79+W1Ned+oT02
cZ5La0byWfa8vDzs2bMH/f396OnpAQCUlZWhsrIStXV1OOKXAHaOMvLIB
N69O+jDEpH/fJb92vrVikb34fiJjp07dwb1KtoxFy3N7MRg8mFLsk
ERCsOxEQrDsRELwLa5LFPMNs9uxuxK7bFvzvzt9T+LPaFeTLkgnKKD
RsLPcE0FewRyFLsFs+a3bjAWtMaVyYkkpTQqLR3U
ERCsOxEQrDsREKw7ERCsOxEQrDsREKw7ERCsOxEQrDsR
EL4XGdPTU3F+fPnYTKZZoJTmTNcPLkSVoJTCmTNncPLkSVRUVMBqtWJgLAg2
2cm5qagj6wvzLf0O8F/p03NoVokkdlQj+bL1wrp6XwWfbZ2VUmC
5XIhPj4eH330330Ea5cuQAqK6uxvHjx4M+JBM+JBEFzmfZ3W433G43AGBmZgYd
Xvo7u5GSkpK0K0AcjSpkpK0AcjImM91u/saWlpyMHSoL29HHXa7HXOUrmzsIXZz
QkJCYseY7Va5bNxKO3zsnZAGkpbS8Nw7IrT4ZvaScFfq8xzeycjCuJstNjyurHkHbNDBEFZkl
ltt9vrtT66O7QJYXGRB0UDFnWLRpPesfqe0xMDO
rq6nDhwgXUXU19cDtZUVXDMbHxzE/Pw6ZG3Jzc4tM6KBEFZkllt9vt6O7
uRnV19cJtZvOXu5ru2rULXV363TyJKLx8Kkhx8PkGXl5eH/vHv3oi8
8CDZba9e/ciOzsbiSkpKCjo8Pc6c9LCDbEJTSGzrZvsBSGrYRTZ2Zm7lm7Wy
TyFjV+SNik+FjxKZbm7nuNvOLy7lnxiTSbtMtNNDT6Yg2dR7Mb7vxQdkLpAkv2
ajpH4gQPy6Dz9ADuGADHO5XS6wfH9ZPpwL4ocMG3ZfTWJvgb2oWOkgwtCwSNUT0zy
dSHGgQLTzyLzP4y3dcaNMd0gDHzIaeELpSCGk2H0kiF7NdspUSf3jdPy6F6ksT0
x7E1c4SuyPUzy07Uma0fuZIETBNGprt8jg7pM76MTsI1gWwo3VXpu5Xrgg6nPLR
GFdnxZjsL5h6ju3yCyQKVtT4q9Dfs8Dc4a9S7lJYKSXWJVhpzNzoa4V3zJPNoVT
CR6KmW9tYH8QHyeF03AJhySDxhq7B5ngEf2ZdczuNXLKOhfhRbuvcuNGYiEK7Hz
ELwQMpCwyPHZwzyG+k+GZDDc7Ie6YbgUR1Ge6WCP7yR6kGy1EzHbwuLHbkKWh0H
cZ5b1KzZ0Ehw03s+mqhAxJrSJd2qoZ6C7ZLmb8a7ajqEIL+ACpNG1htnp4KvuVV6
DbOjZS4GhBSwL4ajm1x3uO3qQPv1OjZThPC9Ozj8s9Ynqo8uH+TTL1Aamnx4qvS
GP3oYuA5AGgzBP9zsHLrJ9Bw3GO0OZzHrNnuVCJ5RlBS3QfuD7JawKBGb
mVsRJAKWwqfpVrUXaZOvTMMBJmnyA6s7y48D27MScTqGv2ImUW4kdV7x3tMWd

04kBMtOJATLTiQEy04kBMtOJATLTiREFAAVqpONj49jcHBw4euk
pCTcunUrVKd/LJE6W6TOBXA2fxk5W1paGtauXes1V+H643Q6w3bu
J3W2SJ2Ls0X+bHwYTyQEy04kRFjLfubMmXCeXitSZ4vUuQDO5q9
QzRbSJ+iIKHz4MJ5ICJadSIiwlH3Hjh24efMmPvnkE5SVlYVjBK/6+/s
XPiPf6XSGdRa73Q6Px4POzs6F2xITE+FwONDb2wuHw+F1j71wzFZ
RUYHh4WG4XC64XC4UFBSEZbbU1FRcvXoVN27cQFdXF44cOQI
g/Pedt7lCeb+FdE1xxYoVqq+vT2VkZKjY2FjV1tamsrKywr7W+cWf/v
5+9cwzz4R9DgBq69atKicnR3V2di7cVlVVpcrKyhQAVVZWpiorKyN
mtoqKClVaWhr2+81sNqucnBwFQMXHx6uenh6VlZUV9vvO21yhut9
CfmXPzc1FX18f+vv7MTMzg4sXL6KwsDDUYzwRWlpaMDk5+dBth
YWFqKmpAQDU1NSgqKgoDJMtPlukcLvdC7sX/fY24+G+77zNFSoh
L3tKSgqGhoYWvh4eHo6o/d6VUnA4HLh+/TqsVmu4x3mEyWSC2+0
G8OB/HpPJFOaJHraUbbxD6be3GY+k+86f7c8DxSfovmLLli3YtGkTC
goKcPjwYWzdujXcI2kppcI9woKlbuMdKl/dZvyrwnXf+bv9eaBCXvaR
kRGsW7du4evU1FSMjIyEegyvRkdHAQATExOor6+PuK2oPR7Pwg66
ZrMZ4+PjYZ7oS5G0jfdi24xHwn0Xzu3PQ152p9OJDRs2ID09OJDRs2ID09HbGxsd
izZw8aGxtDPcai4uLiEB8fv/D37du3R9xW1I2NjSguLgYAFBcXo6GhI
cwTfSmStvFebJvxSLjvwr39ecifLS0oKFA9PT2qr69PHTt2LOzP3n7xJy
MjQ7W1tam2tjbV1dUV9tlqa2vV6Oioun//vhoaGlKvv/66WrNmjXrvvfd
Ub2+vunLlikpMTIyY2c6fP686OjpUe3u7amhoUGazOSyz5eXlKaWUa
m9vVy6XS7lcLlVQUBD2+87bXKG63/hyWSIh+AQdkRAsO5EQLDu
RECw7kRAsO5EQLDuRECw7kRD/D2hlIVvotzDnAAAAAElFTkSuQ
mCC",
      "text/plain": [
       "<Figure size 432x288 with 1 Axes>"
      ]
     },
     "metadata": {},
     "output_type": "display_data"
    }
   ],
   "source": [
    "plt.imshow(X_train[0])"
   ]
  },
  {
   "cell_type": "markdown",

```
 "metadata": {},
 "source": [
  "### Data Pre-Processing"
 ]
},
{
 "cell_type": "code",
 "execution_count": 7,
 "metadata": {},
 "outputs": [],
 "source": [
  "X_train = X_train.reshape(60000, 28, 28, 1).astype('float32')\n",
  "X_test = X_test.reshape(10000, 28, 28, 1).astype('float32')"
 ]
},
{
 "cell_type": "code",
 "execution_count": 8,
 "metadata": {},
 "outputs": [],
 "source": [
  "number_of_classes = 10\n",
  "Y_train = np_utils.to_categorical(y_train, number_of_classes)\n",
  "Y_test = np_utils.to_categorical(y_test, number_of_classes)"
 ]
},
{
 "cell_type": "code",
 "execution_count": 9,
 "metadata": {},
 "outputs": [
  {
   "data": {
    "text/plain": [
     "array([0., 0., 0., 0., 0., 1., 0., 0., 0., 0.], dtype=float32)"
    ]
   },
```

```
     "execution_count": 9,
     "metadata": {},
     "output_type": "execute_result"
    }
   ],
   "source": [
    "Y_train[0]"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "### Create model"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 10,
   "metadata": {},
   "outputs": [],
   "source": [
    "model = Sequential()\n",
    "model.add(Conv2D(64, (3, 3), input_shape=(28, 28, 1), activation=\"relu\"))\n",
    "model.add(Conv2D(32, (3, 3), activation=\"relu\"))\n",
    "model.add(Flatten())\n",
    "model.add(Dense(number_of_classes, activation=\"softmax\"))"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 11,
   "metadata": {},
   "outputs": [],
   "source": [
```

```
   "model.compile(loss='categorical_crossentropy',
optimizer=\"Adam\", metrics=[\"accuracy\"])"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "### Train the model"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 12,
   "metadata": {},
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "Epoch 1/5\n",
      "1875/1875 [==============================] - 16s
5ms/step - loss: 0.2158 - accuracy: 0.9518 - val_loss: 0.0964 -
val_accuracy: 0.9707\n",
      "Epoch 2/5\n",
      "1875/1875 [==============================] - 9s
5ms/step - loss: 0.0682 - accuracy: 0.9794 - val_loss: 0.0674 -
val_accuracy: 0.9805\n",
      "Epoch 3/5\n",
      "1875/1875 [==============================] - 9s
5ms/step - loss: 0.0478 - accuracy: 0.9844 - val_loss: 0.0852 -
val_accuracy: 0.9759\n",
      "Epoch 4/5\n",
      "1875/1875 [==============================] - 9s
5ms/step - loss: 0.0336 - accuracy: 0.9893 - val_loss: 0.1202 -
val_accuracy: 0.9719\n",
      "Epoch 5/5\n",
```

```
      "1875/1875 [==============================] - 9s
5ms/step - loss: 0.0270 - accuracy: 0.9914 - val_loss: 0.1036 -
val_accuracy: 0.9777\n"
     ]
    },
    {
     "data": {
      "text/plain": [
       "<keras.callbacks.History at 0x1ed3324f7f0>"
      ]
     },
     "execution_count": 12,
     "metadata": {},
     "output_type": "execute_result"
    }
   ],
   "source": [
    "model.fit(X_train, Y_train, batch_size=32, epochs=5,
validation_data=(X_test,Y_test))"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "### Test the model"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 13,
   "metadata": {},
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
```

```
      "Metrics (Test Loss & Test Accuracy): \n",
      "[0.1035672277212143, 0.9776999950408936]\n"
     ]
    }
   ],
   "source": [
    "metrics = model.evaluate(X_test, Y_test, verbose=0)\n",
    "print(\"Metrics (Test Loss & Test Accuracy): \")\n",
    "print(metrics)"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 14,
   "metadata": {},
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "1/1 [==============================] - 0s
177ms/step\n",
      "[[6.43197941e-15 8.71634543e-21 7.98728167e-11
7.08215517e-12\n",
      "  2.27718335e-18 1.36703092e-15 2.37176042e-22
1.00000000e+00\n",
      "  4.51405352e-13 4.25453591e-13]\n",
      " [4.56659687e-15 1.54588287e-10 1.00000000e+00
1.20107971e-13\n",
      "  1.86926159e-19 3.90255250e-20 1.16102319e-11
4.27834925e-23\n",
      "  7.33884963e-17 1.86307852e-23]\n",
      " [1.37352282e-10 9.99961138e-01 3.40877750e-06
1.50240779e-12\n",
      "  1.99599867e-07 1.10004057e-05 6.72304851e-11
7.78906983e-09\n",
      "  2.42337919e-05 3.74607870e-13]\n",
```

```
      " [1.00000000e+00 5.39840355e-16 1.03082355e-10
4.23198737e-17\n",
      "  8.17481194e-10 2.49619574e-12 1.66041558e-09
5.06253395e-17\n",
      "  3.02219919e-13 5.55243709e-08]]\n"
     ]
    }
   ],
   "source": [
    "prediction = model.predict(X_test[:4])\n",
    "print(prediction)"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 15,
   "metadata": {},
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "[7 2 1 0]\n",
      "[[0. 0. 0. 0. 0. 0. 0. 1. 0. 0.]\n",
      " [0. 0. 1. 0. 0. 0. 0. 0. 0. 0.]\n",
      " [0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]\n",
      " [1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]]\n"
     ]
    }
   ],
   "source": [
    "print(numpy.argmax(prediction, axis=1))\n",
    "print(Y_test[:4])"
   ]
  },
  {
   "cell_type": "markdown",
```

```
 "metadata": {},
 "source": [
  "### Save the model"
 ]
},
{
 "cell_type": "code",
 "execution_count": 16,
 "metadata": {},
 "outputs": [],
 "source": [
  "model.save(\"model.h5\")"
 ]
},
{
 "cell_type": "markdown",
 "metadata": {},
 "source": [
  "### Test the saved model"
 ]
},
{
 "cell_type": "code",
 "execution_count": 22,
 "metadata": {},
 "outputs": [],
 "source": [
  "model=load_model(\"model.h5\")"
 ]
},
{
 "cell_type": "code",
 "execution_count": 23,
 "metadata": {},
 "outputs": [
  {
   "name": "stdout",
```

```
     "output_type": "stream",
     "text": [
      "1/1 [==============================] - 0s
435ms/step\n",
      "0    8\n",
      "Name: Label, dtype: int64\n"
     ]
    }
   ],
   "source": [
    "img = Image.open(\"sample.png\").convert(\"L\")\n",
    "img = img.resize((28, 28))\n",
    "img2arr = npm.array(img)\n",
    "img2arr = img2arr.reshape(1, 28, 28, 1)\n",
    "results  = model.predict(img2arr)\n",
    "results = npm.argmax(results,axis = 1)\n",
    "results = pds.Series(results,name=\"Label\")\n",
    "print(results)"
   ]
  }
 ],
 "metadata": {
  "kernelspec": {
   "display_name": "Python 3.10.8 ('venv': venv)",
   "language": "python",
   "name": "python3"
  },
  "language_info": {
   "codemirror_mode": {
    "name": "ipython",
    "version": 3
   },
   "file_extension": ".py",
   "mimetype": "text/x-python",
   "name": "python",
   "nbconvert_exporter": "python",
   "pygments_lexer": "ipython3",
```

```
   "version": "3.10.8"
  },
  "orig_nbformat": 4,
  "vscode": {
   "interpreter": {
    "hash":
"72cf82f53b15019b5b640600623df8bcf4d62c2c60fee1ea51c8c07b39
5bb5c2"
   }
  }
 },
 "nbformat": 4,
 "nbformat_minor": 2
}
```


8 TESTING:

8.1 TEST CASE:


```
{
 "nbformat": 4,
 "nbformat_minor": 0,
 "metadata": {
  "colab": {
   "provenance": []
  },
  "kernelspec": {
   "name": "python3",
   "display_name": "Python 3"
  },
  "language_info": {
   "name": "python"
  }
 },
 "cells": [
  {
   "cell_type": "markdown",
```

```
   "source": [
    "Sprint-2\n",
    "Team ID-PNT2022TMID39529"
   ],
   "metadata": {
    "id": "G4T0oInf8hgU"
   }
  },
  {
   "cell_type": "markdown",
   "source": [
    "Import necessary package"
   ],
   "metadata": {
    "id": "BU0PScv980Zy"
   }
  },
  {
   "cell_type": "code",
   "execution_count": 1,
   "metadata": {
    "id": "JYsTTpKb8UFP"
   },
   "outputs": [],
   "source": [
    "import numpy\n",
    "import matplotlib.pyplot as plt\n",
    "from keras.utils import np_utils\n",
    "from tensorflow.keras.datasets import mnist\n",
    "from tensorflow.keras.models import Sequential\n",
    "from tensorflow.keras.layers import Conv2D, Dense,
Flatten\n",
    "from tensorflow.keras.optimizers import Adam"
   ]
  },
  {
   "cell_type": "markdown",
```

```
    "source": [
     "Load data"
    ],
    "metadata": {
     "id": "uApZbwMa9Bb4"
    }
   },
   {
    "cell_type": "code",
    "source": [
     "(X_train, y_train), (X_test, y_test) =
mnist.load_data()\n"
    ],
    "metadata": {
     "colab": {
      "base_uri": "https://localhost:8080/"
     },
     "id": "564SXrH088jn",
     "outputId": "2d4a2d41-9b61-4bdd-a4b8-b5133adc9c67"
    },
    "execution_count": 2,
    "outputs": [
     {
      "output_type": "stream",
      "name": "stdout",
      "text": [
       "Downloading data from
https://storage.googleapis.com/tensorflow/tf-keras-
datasets/mnist.npz\n",
       "11490434/11490434
[==============================] - 0s 0us/step\n"
      ]
     }
    ]
   },
   {
    "cell_type": "code",
```

```
  "source": [
   "print(X_train.shape)\n",
   "print(X_test.shape)"
  ],
  "metadata": {
   "colab": {
    "base_uri": "https://localhost:8080/"
   },
   "id": "3Vl-bd-B9IzV",
   "outputId": "844bda4c-b885-4059-9782-a2328fbb2ed8"
  },
  "execution_count": 3,
  "outputs": [
   {
    "output_type": "stream",
    "name": "stdout",
    "text": [
     "(60000, 28, 28)\n",
     "(10000, 28, 28)\n"
    ]
   }
  ]
 },
 {
  "cell_type": "code",
  "source": [
   "X_train[0]"
  ],
  "metadata": {
   "colab": {
    "base_uri": "https://localhost:8080/"
   },
   "id": "D7e8u2qK9Sg8",
   "outputId": "8bb4a4d1-9e73-4218-8014-82972773795b"
  },
  "execution_count": 4,
  "outputs": [
```

{
"output_type": "execute_result",
"data": {
"text/plain": [
"array([[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,\n",
" 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,\n",
" 0, 0],\n",
" [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,\n",
" 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,\n",
" 0, 0],\n",
" [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,\n",
" 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,\n",
" 0, 0],\n",
" [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,\n",
" 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,\n",
" 0, 0],\n",
" [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,\n",
" 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,\n",
" 0, 0],\n",
" [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
3,\n",
" 18, 18, 18, 126, 136, 175, 26, 166, 255, 247,
127, 0, 0,\n",
" 0, 0],\n",
" [0, 0, 0, 0, 0, 0, 0, 0, 30, 36, 94, 154,
170,\n",

```
            "       253, 253, 253, 253, 253, 225, 172, 253, 242,
195,  64,   0,   0,\n",
            "         0,   0],\n",
            "      [  0,   0,   0,   0,   0,   0,   0,  49, 238, 253, 253,
253, 253,\n",
            "       253, 253, 253, 253, 251,  93,  82,  82,  56,  39,
0,   0,   0,\n",
            "         0,   0],\n",
            "      [  0,   0,   0,   0,   0,   0,   0,  18, 219, 253, 253,
253, 253,\n",
            "       253, 198, 182, 247, 241,   0,   0,   0,   0,   0,   0,
0,   0,\n",
            "         0,   0],\n",
            "      [  0,   0,   0,   0,   0,   0,   0,   0,  80, 156, 107,
253, 253,\n",
            "       205,  11,   0,  43, 154,   0,   0,   0,   0,   0,   0,
0,   0,\n",
            "         0,   0],\n",
            "      [  0,   0,   0,   0,   0,   0,   0,   0,   0,  14,   1, 154,
253,\n",
            "        90,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
0,\n",
            "         0,   0],\n",
            "      [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0, 139,
253,\n",
            "       190,   2,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
0,\n",
            "         0,   0],\n",
            "      [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  11,
190,\n",
            "       253,  70,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
0,\n",
            "         0,   0],\n",
            "      [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
35,\n",
            "       241, 225, 160, 108,   1,   0,   0,   0,   0,   0,   0,
0,   0,\n",
```

```
            "          0,   0],\n",
            "       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
0,\n",
            "         81, 240, 253, 253, 119,  25,   0,   0,   0,   0,   0,
0,   0,\n",
            "          0,   0],\n",
            "       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
0,\n",
            "          0,  45, 186, 253, 253, 150,  27,   0,   0,   0,   0,
0,   0,\n",
            "          0,   0],\n",
            "       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
0,\n",
            "          0,   0,  16,  93, 252, 253, 187,   0,   0,   0,   0,
0,   0,\n",
            "          0,   0],\n",
            "       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
0,\n",
            "          0,   0,   0,   0, 249, 253, 249,  64,   0,   0,   0,
0,   0,\n",
            "          0,   0],\n",
            "       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
0,\n",
            "          0,  46, 130, 183, 253, 253, 207,   2,   0,   0,   0,
0,   0,\n",
            "          0,   0],\n",
            "       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
39,\n",
            "        148, 229, 253, 253, 253, 250, 182,   0,   0,   0,
0,   0,   0,\n",
            "          0,   0],\n",
            "       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  24, 114,
221,\n",
            "        253, 253, 253, 253, 201,  78,   0,   0,   0,   0,   0,
0,   0,\n",
            "          0,   0],\n",
```

```
            "       [  0,   0,   0,   0,   0,   0,   0,   0,  23,  66, 213,
253, 253,\n",
            "        253, 253, 198,  81,   2,   0,   0,   0,   0,   0,   0,
0,   0,\n",
            "          0,   0],\n",
            "       [  0,   0,   0,   0,   0,   0,  18, 171, 219, 253, 253,
253, 253,\n",
            "        195,  80,   9,   0,   0,   0,   0,   0,   0,   0,   0,
0,\n",
            "          0,   0],\n",
            "       [  0,   0,   0,   0,  55, 172, 226, 253, 253, 253,
253, 244, 133,\n",
            "         11,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
0,\n",
            "          0,   0],\n",
            "       [  0,   0,   0,   0, 136, 253, 253, 253, 212, 135,
132,  16,   0,\n",
            "          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
0,\n",
            "          0,   0],\n",
            "       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
0,\n",
            "          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
0,\n",
            "          0,   0],\n",
            "       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
0,\n",
            "          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
0,\n",
            "          0,   0],\n",
            "       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
0,\n",
            "          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
0,\n",
            "          0,   0]], dtype=uint8)"
          ]
        },
```

```
      "metadata": {},
      "execution_count": 4
    }
  ]
},
{
  "cell_type": "code",
  "source": [
    "y_train[0]"
  ],
  "metadata": {
    "colab": {
      "base_uri": "https://localhost:8080/"
    },
    "id": "1VsWb5sm9VcB",
    "outputId": "b7521eea-b655-4144-c4b5-7a1772c31270"
  },
  "execution_count": 5,
  "outputs": [
    {
      "output_type": "execute_result",
      "data": {
        "text/plain": [
          "5"
        ]
      },
      "metadata": {},
      "execution_count": 5
    }
  ]
},
{
  "cell_type": "code",
  "source": [
    "plt.imshow(X_train[0])"
  ],
  "metadata": {
```

"colab": {
"base_uri": "https://localhost:8080/",
"height": 282
},
"id": "mXNmeoHL9bYS",
"outputId": "801b729b-8e15-42f6-8943-331e9f8be4ad"
},
"execution_count": 6,
"outputs": [
{
"output_type": "execute_result",
"data": {
"text/plain": [
"<matplotlib.image.AxesImage at 0x7f34394baa90>"
]
},
"metadata": {},
"execution_count": 6
},
{
"output_type": "display_data",
"data": {
"text/plain": [
"<Figure size 432x288 with 1 Axes>"
],
"image/png":
"iVBORw0KGgoAAAANSUhEUgAAAPsAAAD4CAYAAA
AAq5pAIAAAABHNCSVQICAgIfAhkiAAAAAlwSFlzAA
ALEgAACxIB0t1+/AAAADh0RVh0U29mdHdhcmUAbWF0
cGxvdGxpYiB2ZXJzaW9uMy4yLjIsIGh0dHA6Ly9tYXRwb
G90bGliLm9yZy+WH4yJAAAOZ0lEQVR4nO3dbYxc5XnG
8euKbezamMQbB9chLjjgFAg0Jl0ZEBZQobgOqgSoCsSKIk
JpnSY4Ca0rQWlV3IpWbpUQUqRTHExFS+BBIQ/0CTU
QpCowWWhBgwEDMY0NmaNWYENIX5Z3/2w42iBnWe
XmTMv3vv/k1Yzc+45c24NXD5nznNmHkeEAIx/H+p0AwD
ag7ADSRB2IAnCDiRB2IEkJrZZY4d5ckzRtHqlV5V3pbe2
OPR6o1FXbbiyVdJ2mCpH+LiJWl50/RNJ3qc5rZJICC9bGub

q3hw3jbEyTdIOnzkk6UtMT2iY2+HoDWauYz+wJJL0TE5oj
YK+lOSedV0xaAqjUT9qMk/WLY4621Ze9ie6ntPtt9+7Snic0
BaEbLz8ZHxKqI6I2I3kma3OrNAaijmbBvkzRn2ONP1JYB6
ELNhP1RSfNsz7V9mKQvSlpbTVsAqtbw0FtE7Le9TNKPN
DT0tjoinq6sMwCVamqcPSLul3R/Rb0AaCEulwWSIOxAEo
QdSIKwA0kQdiAJwg4kQdiBJAg7kARhB5Ig7EAShB1IgrA
DSRB2IAnCDiRB2IEkCDuQBGEHkiDsQBKEHUiCsANJE
HYgCcIOJEHYgSQIO5AEYQeSIOxAEoQdSIKwA0kQdiCJ
pmZxRffzxPJ/4gkfm9nS7T/3F8fUrQ1OPVBc9+hjdxTrU7/u
Yv3Vaw+rW3u893vFdXcOvl2sn3r38mL9uD9/pFjvhKbCbnu
LpN2SBiXtj4jeKpoCUL0q9uy/FxE7K3gdAC3EZ3YgiWbD
HpJ+bPsx20tHeoLtpbb7bPft054mNwegUc0exi+MiG22j5T0g
O2fR8TDw58QEaskrZKkI9wTTW4PQIOa2rNHxLba7Q5J90
paUEVTAKrXcNhtT7M9/eB9SYskbayqMQDVauYwfpake2
0ffJ3bI+KHlXQ1zkw4YV6xHpMnFeuvnPWRYv2d0+qPCfd
8uDxe/JPPlMebO+k/fzm9WP/Hf1lcrK8/+fa6tZf2vVNcd2X/5
4r1j//k0PtE2nDYI2KzpM9U2AuAFmLoDUiCsANJEHYgCc
IOJEHYgST4imsFBs/+bLF+7S03FOufmlT/q5jj2b4YLNb/5v
qvFOsT3y4Pf51+97K6tenb9hfXnbyzPDQ3tW99sd6N2LMD
SRB2IAnCDiRB2IEkCDuQBGEHkiDsQBKMs1dg8nOvFOu
P/WpOsf6pSf1VtlOp5dtPK9Y3v1X+Kepbjv1+3dqbB8rj5LP
++b+L9VY69L7AOjr27EAShB1IgrADSRB2IAnCDiRB2IEk
CDuQhCPaN6J4hHviVJ/Ttu11i4FLTi/Wdy0u/9zzhCcPL9af+
Pr1H7ing67Z+TvF+qNnlcfRB994s1iP0+v/APGWbxZX1dwl
T5SfgPdZH+u0KwZGnMuaPTuQBGEHkiDsQBKEHUiCsA
NJEHYgCcIOJME4exeYMPOjxfrg6wPF+ku31x8rf/rM1cV1
F/zDN4r1I2/o3HfK8cE1Nc5ue7XtHbY3DlvWY/sB25tqtzOq
bBhA9cZyGH+LpPfOen+lpHURMU/SutpjAF1s1LBHxMOS
3nsceZ6kNbX7aySdX3FfACrW6G/QzYqI7bX7r0qaVe+Jtpd
KWipJUzS1wc0BaFbTZ+Nj6Axf3bN8EbEqInojoneSJje7OQ
ANajTs/bZnS1Ltdkd1LQFohUbDvlbSxbX7F0u6r5p2ALTKq
J/Zbd8h6WxJM21vlXS1pJWS7rJ9qaSXJV3YyibHu8Gdrze1/
r5djc/v/ukvPVOsv3bjhPILHCjPsY7uMWrYI2JJnRJXxwCH
EC6XBZIg7EAShB1IgrADSRB2IAmmbB4HTrji+bq1S04uD
5r8+9HrivWzvnBZsT79e48U6+ge7NmBJAg7kARhB5Ig7E
AShB1IgrADSRB2IAnG2ceB0rTJr3/thOK6/7f2nWL9ymtuL
db/8sILivX43w/Xrc35+58V11Ubf+Y8A/bsQBKEHUiCsANJ

EHYgCcIOJEHYgSQIO5AEUzYnN/BHpxfrt1397WJ97sQp
DW/707cuK9bn3bS9WN+/eUvD2x6vmpqyGcD4QNiBJAg7
kARhB5Ig7EAShB1IgrADSTDOjqI4Y36xfsTKrcX6HZ/8Uc
PbPv7BPy7Wf/tv63+PX5IGN21ueNuHqqbG2W2vtr3D9sZh
y1bY3mZ7Q+3v3CobBlC9sRzG3yJp8QjLvxsR82t/91fbFoCq
jRr2iHhY0kAbegHQQs2coFtm+8naYf6Mek+yvdR2n+2+fdr
TxOYANKPRsN8o6VhJ8yVtl/Sdek+MiFUR0RsRvZM0ucH
NAWhWQ2GPiP6IGIyIA5JukrSg2rYAVK2hsNuePezhBZI2
1nsugO4w6ji77TsknS1ppqR+SVfXHs+XFJK2SPpqRJS/fCz
G2cejCbOOLNZfuei4urX1V1xXXPdDo+yLvvTSomL9zYW
vF+vjUWmcfdRJIiJiyQiLb266KwBtxeWyQBKEHUiCsANJ
EHYgCcIOJMFXXNExd20tT9k81YcV67+MvcX6H3zj8vqvf
e/64rqHKn5KGgBhB7Ig7EAShB1IgrADSRB2IAnCDiQx6rf
ekNuBheWfkn7xC+Upm0+av6VubbRx9NFcP3BKsT71vr6m
Xn+8Yc8OJEHYgSQIO5AEYQeSIOxAEoQdSIKwA0kwzj7
OufekYv35b5bHum86Y02xfuaU8nfKm7En9hXr/7n9hXrjwzMLb/Ag
VF/3TwV9uxAEoQdSIKwA0kQdiAJwg4kQdiBJAg7kATj7I
eAiXOPLtZfvOTjr6zoZ6qcFV/b7H+4fHr6zoZ6qcFV/b7H+0HWn
Fesz1pR/dx7vNuqe3fYc2w/afsb2/VZ/sB25tqtzNa3y6
ARo3lMH6/pOURcaKk0yRdZvtESQskrYyIoZvG0KV
GDXtEbI+Ix2v3d0t6VtJRks6TdPBayjWSm9VkwCa94E+s9
s+RtIpktZLmhURBy8+flXSrDrrLJW0VJKmaGqjfQJo0pjPxt
s+XNIPJF0eEbuG12JodsgRZ4iMiFUR0RsRvZM0ualmATR
uTGG3PUlDQb8tIu6pLe63PbtWny1pR2taBFCFUQ/jbVvSz
ZKejYhrh5XWXSrpY0sra7X0t6XAcmHjMbxXrb/7u7GL9or/7
YbH+px+5p1hvpeXby8NjP/vX+sNrPbf8T3HdGQcYWqvS
WD6znyHpy5Kesr2htuwqDYX8LtuXSnpZ0ZqoWtaRFAFUYN
e0T8VNKIk7tLOqfadgC0CpfLAkkQDiAJwg4kQdiBJAg7kA
RfcR2jibN/s25tYPW04rpfm/tQsb5ken9DPVVh2bbFxfrjN5an
bJ75/Y3Fes9uxq7BXt2IAnCDiRB2IEkCDuQBGEHkiDsQBB
KEHUgizTj73t8v/2zx3j8bKNavOu7/wnQ1VpVf/F2Qz1VpX/wnb
q1M9cuL657/F//vFjveaM8Tn6gWEU3Yc8OJEHYgSQIO5A
EYQeSIOxAEoQdSIKwA0mkGWffcn7n7537XXn+4+4/XrHlzw4kQdiBJ
ji/XrHlpUrHuw3o/7Djn+mpfq1ub1ry+uO1isYjxhzw4kQdiBJ
Ag7kARhB5Ig7EAShB1IgrADSTgiyk+w50VdIsSSFpVUR
cZ3uFpD+R9FrtqVdFRP2v7gMoa+zsnnNJRzX5JRzQmOZn327

pO21+7ttPyvpqFY3BqBaH+gzu+1jJJ0i6eA1mMtsP2l7te0Zd
dZZarvPdt8+7WmqWQCNG3PYbR8u6QeSLo+IXZJulHSsp
Pka2vN/Z6T1ImJVRPRGRO8kTa6gZQCNGFPYbU/SUNBv
i4h7JCki+iNiMCIOSLpJ0oLWtQmgWaOG3bYl3Szp2Yi4dtj
y2cOedoGk8nSeADpqLGfjz5D0ZUlP2d5QW3aVpCW252to
OG6LpK+2pEMAlRjL2fifShpp3K44pg6gu3AFHZAEYQeSI
OxAEoQdSIKwA0kQdiAJwg4kQdiBJAg7kARhB5Ig7EASh
B1IgrADSRB2IIlRf0q60o3Zr0l6ediimZJ2tq2BD6Zbe+vWvi
R6a1SVvR0dER8bqdDWsL9v43ZfRPR2rIGCbu2tW/uS6K1
R7eqNw3ggCcIOJNHpsK/q8P5LurW3bu1LordGdaS3jv7MDqB9On1nB9AmhB1IoiNht32j7Rds77J9Wyc
9A+nd6zA2gTwg4k0ZGw215s+znbL9i+shM91GN7i+2nbG+
w3dfhXlbb3mF747BlPbYfsL2pdjviHHsd6m2F7W21926D7X
M71Nsc2w/afsb207a/VVve0feu0Fdb3re2f2a3PUHS85I+J2mr
pEclLYmIZ9raSB22t0jqjYiOX4Bh+0xJb0m6D0kD
EbGy9g/ljIi4okt6WyHprU5P412brWj28GnGJZ0v6Svq4HtX
6OtCteF968SefYGkFyJic0TslXSnpPM60EfXi4iHJQ28Z/F5k
tbU7q/R0P8sbVent64QEdsj4vHa/d2SDk4z3tH3rtBXW3Qi7E
dJ+sWwx1vVXfO9h6Qf237M9tJONzCWRGxvXb/VUmz
OtnMCEadxrud3jPNeNe8d41Mf94sTtC938KI+Kykz0u6rHa4
2pVi6DNYN42djmka73YZYZrxX+vke9fo9OfN6kTYt0maM
+zxJ2rLukJEbKvd7pB0r7pvpvKur+gzPo1m53dLifX+umabxH
mmZcXfDedXL6806E/VFJ82zPtX2YpC9KWtuBPt7H9tRaH9rTai
RPZniZpkbpvKuq1ki6u3b9Y0n0d7OVdumUa73ejKvD713
Hpz+PiLb/STpXQ2fkX5T0V53orU5fcyX9Z+3v6U73
R3W7dPQuY1LJX1U0jpJmyT9l6SeLurtPyQ9Q9JelJDQVrdod6
W6ihQ/QnJW2o/Z3b6feu0Fdb3jculwWS4AQdkARhB5Ig7E
AShB1IgrADSRB2IAnCDiTx/65XcTNOWsh5AAAAAElFT
kSuQmCC\n"
},
"metadata": {
"needs_background": "light"
}
}
]
},
{
"cell_type": "markdown",

```
      "source": [
        "Data pre processing"
      ],
      "metadata": {
        "id": "ViwBwMJf9nM7"
      }
    },
    {
      "cell_type": "code",
      "source": [
        "X_train = X_train.reshape(60000, 28, 28,
1).astype('float32')\n",
        "X_test = X_test.reshape(10000, 28, 28,
1).astype('float32')"
      ],
      "metadata": {
        "id": "PA7e1C1S9ftz"
      },
      "execution_count": 7,
      "outputs": []
    },
    {
      "cell_type": "code",
      "source": [
        "number_of_classes = 10\n",
        "Y_train = np_utils.to_categorical(y_train,
number_of_classes)\n",
        "Y_test = np_utils.to_categorical(y_test,
number_of_classes)"
      ],
      "metadata": {
        "id": "ps3cxWyZ9rMu"
      },
      "execution_count": 8,
      "outputs": []
    },
    {
```

```
    "cell_type": "code",
    "source": [
     "Y_train[0]"
    ],
    "metadata": {
     "colab": {
      "base_uri": "https://localhost:8080/"
     },
     "id": "lPpmgI5M9vlY",
     "outputId": "24f2b170-c2c1-41a5-f492-7f637fc8ba19"
    },
    "execution_count": 9,
    "outputs": [
     {
      "output_type": "execute_result",
      "data": {
       "text/plain": [
        "array([0., 0., 0., 0., 0., 1., 0., 0., 0., 0.],
dtype=float32)"
       ]
      },
      "metadata": {},
      "execution_count": 9
     }
    ]
   },
   {
    "cell_type": "markdown",
    "source": [
     "Create model"
    ],
    "metadata": {
     "id": "kv6cYS9593Xz"
    }
   },
   {
    "cell_type": "code",
```

```
      "source": [
        "model = Sequential()\n",
        "model.add(Conv2D(64, (3, 3), input_shape=(28, 28, 1),
activation=\"relu\"))\n",
        "model.add(Conv2D(32, (3, 3), activation=\"relu\"))\n",
        "model.add(Flatten())\n",
        "model.add(Dense(number_of_classes,
activation=\"softmax\"))"
      ],
      "metadata": {
        "id": "WkvfmMs-9y0P"
      },
      "execution_count": 10,
      "outputs": []
    },
    {
      "cell_type": "code",
      "source": [
        "model.compile(loss='categorical_crossentropy',
optimizer=\"Adam\", metrics=[\"accuracy\"])"
      ],
      "metadata": {
        "id": "nS6zuQzF96Up"
      },
      "execution_count": 11,
      "outputs": []
    },
    {
      "cell_type": "code",
      "source": [
        "model.fit(X_train, Y_train, batch_size=32, epochs=5,
validation_data=(X_test,Y_test))"
      ],
      "metadata": {
        "colab": {
          "base_uri": "https://localhost:8080/"
        },
```

```
      "id": "r1InLlgH9-2j",
      "outputId": "3a6e787b-6740-461c-c5f2-11fcd9369ade"
     },
     "execution_count": 12,
     "outputs": [
      {
       "output_type": "stream",
       "name": "stdout",
       "text": [
        "Epoch 1/5\n",
        "1875/1875
[==============================] - 202s
107ms/step - loss: 0.2126 - accuracy: 0.9521 - val_loss:
0.0894 - val_accuracy: 0.9732\n",
        "Epoch 2/5\n",
        "1875/1875
[==============================] - 204s
109ms/step - loss: 0.0623 - accuracy: 0.9814 - val_loss:
0.1000 - val_accuracy: 0.9713\n",
        "Epoch 3/5\n",
        "1875/1875
[==============================] - 204s
109ms/step - loss: 0.0425 - accuracy: 0.9864 - val_loss:
0.0851 - val_accuracy: 0.9758\n",
        "Epoch 4/5\n",
        "1875/1875
[==============================] - 198s
105ms/step - loss: 0.0346 - accuracy: 0.9890 - val_loss:
0.1026 - val_accuracy: 0.9763\n",
        "Epoch 5/5\n",
        "1875/1875
[==============================] - 205s
109ms/step - loss: 0.0253 - accuracy: 0.9916 - val_loss:
0.0980 - val_accuracy: 0.9777\n"
       ]
      },
      {
```

```
      "output_type": "execute_result",
      "data": {
        "text/plain": [
          "<keras.callbacks.History at 0x7f3434c97a90>"
        ]
      },
      "metadata": {},
      "execution_count": 12
    }
  ]
 },
 {
  "cell_type": "markdown",
  "source": [
    "Test the model"
  ],
  "metadata": {
    "id": "GajjLCQFCtcV"
  }
 },
 {
  "cell_type": "code",
  "source": [
    "metrics = model.evaluate(X_test, Y_test, verbose=0)\n",
    "print(\"Metrics (Test Loss & Test Accuracy): \")\n",
    "print(metrics)"
  ],
  "metadata": {
    "colab": {
      "base_uri": "https://localhost:8080/"
    },
    "id": "0vwkOGRvB-YJ",
    "outputId": "6106f6ac-25a8-465e-9e3b-d1c8e2c73bdf"
  },
  "execution_count": 13,
  "outputs": [
    {
```

```
        "output_type": "stream",
        "name": "stdout",
        "text": [
          "Metrics (Test Loss & Test Accuracy): \n",
          "[0.09800956398248672, 0.9776999950408936]\n"
        ]
       }
      ]
     },
     {
      "cell_type": "code",
      "source": [
       "prediction = model.predict(X_test[:4])\n",
       "print(prediction)"
      ],
      "metadata": {
       "colab": {
        "base_uri": "https://localhost:8080/"
       },
       "id": "CvzsM8cRCXb0",
       "outputId": "b20715c7-016d-47d6-c7a0-5f0d975cfe41"
      },
      "execution_count": 14,
      "outputs": [
       {
        "output_type": "stream",
        "name": "stdout",
        "text": [
         "1/1 [==============================] - 0s
128ms/step\n",
         "[[2.9166860e-11 7.8674158e-21 6.8197568e-09
9.6392165e-09 1.2284079e-15\n",
         "  2.9200296e-16 1.2543796e-18 1.0000000e+00
6.8977801e-12 8.5674190e-11]\n",
         " [2.2934597e-05 6.7787391e-07 9.9997628e-01
4.7655574e-10 7.8161123e-12\n",
```

```
          "  3.1815811e-15 4.4551697e-08 2.4422626e-14
6.2133083e-08 5.8212260e-17]\n",
          " [4.5624104e-09 9.9988461e-01 3.9433194e-09
3.0301920e-12 9.3741266e-07\n",
          "  1.5071736e-10 1.4758221e-10 3.4230155e-10
1.1441124e-04 2.0948462e-11]\n",
          " [1.0000000e+00 8.1227650e-20 6.9074791e-13
4.1791107e-17 6.7886536e-15\n",
          "  4.3934654e-14 2.0079671e-11 8.7713697e-16
4.4645889e-16 4.8445514e-14]]\n"
         ]
        }
       ]
      },
      {
       "cell_type": "code",
       "source": [
        "print(numpy.argmax(prediction, axis=1))\n",
        "print(Y_test[:4])"
       ],
       "metadata": {
        "colab": {
         "base_uri": "https://localhost:8080/"
        },
        "id": "i-JtdfDqC8fD",
        "outputId": "221ddb4f-97ed-4da2-a80a-b1d5033d5dc3"
       },
       "execution_count": 15,
       "outputs": [
        {
         "output_type": "stream",
         "name": "stdout",
         "text": [
          "[7 2 1 0]\n",
          "[[0. 0. 0. 0. 0. 0. 0. 1. 0. 0.]\n",
          " [0. 0. 1. 0. 0. 0. 0. 0. 0. 0.]\n",
          " [0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]\n",
```

```
            " [1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]]\n"
          ]
         }
        ]
       }
      ]
     }
```


9 RESULT:

we have successfully built a Python deep learning project on handwritten digit recognition app. We have built and trained the Convolutional neural network which is very effective for image classification purposes. Later on, we build the GUI where we draw a digit on the canvas then we classify the digit and show the results.

10 ADVANTAGE AND DISADVANTAGE:

To combat the loss of life due to illegible script, health care facilities are currently implementing digital data strategies, such as electronic health records (EHR). A 2013 survey found that 71 percent of physicians have adopted EHR. Such policies also help facilities lessen their dependence on hard paper, mitigate regulatory violations and compliance risks, as well as forgeries that lead to fraud.

This comes at a time when vehicle infotainment systems are becoming increasingly sophisticated. One technology that helps ensure a safe, enjoyable user experience is handwriting recognition capabilities.

11 CONCLUSIONS:

The paper discusses in detail all advances in the area of handwritten character recognition. The most accurate solution

provided in this area directly or indirectly depends upon the quality as well as the nature of the material to be read. Various techniques have been described in this paper for character recognition in handwriting recognition system A sort comparison is shown between the different methods proposed so far in table 1. From the study done so far, it is analysed that the selection of the classification as well as the feature extraction techniques needs to be proper in order to attain good rate in recognizing the character. Studies in the paper reveals that there is still scope of enhancing the algorithms as well as enhancing the rate of recognition of characters. The paper discusses in detail all advances in the area of handwritten character recognition. The most accurate solution provided in this area directly or indirectly depends upon the quality as well as the nature of the material to be read. Various techniques have been described in this paper for character recognition in handwriting recognition system. A sort comparison is shown between the different methods proposed so far in table 1. From the study done so far, it is analysed that the selection of the classification as well as the feature extraction techniques needs to be proper in order to attain good rate in recognizing the character. Studies in the paper reveals that there is still scope of enhancing the algorithms as well as enhancing the rate of recognition of characters.

Future Scope:

The future development of the applications based on algorithms of deep and machine learning is practically boundless. In the future, we can work on a denser or hybrid algorithm than the

current set of algorithms with more manifold data to achieve the solutions to many problems. In future, the application of these algorithms lies from the public to high-level authorities, as from the differentiation of the algorithms above and with future development we can attain high-level functioning applications which can be used in the classified or government agencies as well as for the common people, we can use these algorithms in hospitals application for detailed medical diagnosis, treatment and monitoring the patients, we can use it in surveillances system to keep tracks of the suspicious activity under the system, in fingerprint and retinal scanners, database filtering applications, Equipment checking for national forces and many more problems of both major and minor category. The advancement in this field can help us create an environment of safety, awareness and comfort by using these algorithms in day-to-day application and high-level application (i.e., corporate level or Government level). Application-based on artificial intelligence and deep learning is the future of the technological world because of their absolute accuracy and advantages over many major problems.

Git hub link:

http://github.com/IBM-EPBL/IBM-Project-35505-1660285385