**­­­­PROJECT REPORT**

A NOVEL METHOD FOR HANDWRITTEN DIGIT RECOGNITION

**submitted by**

**PNT2022TMID52672**

Zamin Hamid - 1905123

Sanjana S Menon - 1905112

Nishanthini A M - 1905101

Priyanka - 1905108

**TABLE OF CONTENTS**

1. INTRODUCTION 1
   1. 1.1 [PROJECT OVERVIEW 1](#_bookmark0)

1.2 [PURPOSE 1](#_bookmark1)

1. LITERATURE SURVEY 2

2.1 [EXISTING PROBLEM 2](#_bookmark2)

2.2 [REFERENCES 2](#_bookmark3)

2.3 [PROBLEM STATEMENT DEFINITION](#_bookmark4) 5

1. IDEATION AND PROPOSED SOLUTION 6

3.1 [EMPATHY MAP CANVAS](#_bookmark5) 6

3.2 [IDEATION & BRAINSTORMING](#_bookmark6) 8

3.3 [PROPOSED SOLUTION](#_bookmark7) 9

3.4 [PROBLEM SOLUTION FIT](#_bookmark8) 11

1. REQUIREMENT ANALYSIS 12

4.1 [FUNCTIONAL REQUIREMENTS](#_bookmark9) 12

4.2 [NON-FUNCTIONAL REQUIREMENTS 1](#_bookmark10)3

1. PROJECT DESIGN 15
   1. [DATA FLOW DIAGRAM 1](#_bookmark11)5

5.2 [SOLUTION & TECHNICAL ARCHITECTURE 15](#_bookmark12)

5.3 COMPOMENTS AND TECHNOLOGIES 19

5.4 USER STORIES 19

1. PROJECT PLANNING AND SCHEDULING 21

6.1 [SPRINT PLANNING AND ESTIMATION](#_bookmark14) 21

6.2 [SPRINT DELIVERY SCHEDULE 21](#_bookmark15)

6.3 REPORT FROM JIRA 22

1. CODING & SOLUTIONING 24
2. RESULTS 28

9.1 OUTPUT [2](#_bookmark20)8

1. ADVANTAGES & DISADVANTAGES 30

[ADVANTAGES 30](#_bookmark21)

[DISADVANTAGES 30](#_bookmark22)

1. CONCLUSION 31
2. FUTURE SCOPE 32

[APPENDIX 33](#_bookmark23)

[SOURCE CODE 33](#_bookmark24)

SCREENSHOTS 43

GITHUB 43

# CHAPTER 1

## INTRODUCTION

### 1.1 PROJECT OVERVIEW

Handwritten Digit Recognition is the capacity of a computer to interpret the manually written digits from various sources like messages, bank cheques, papers, pictures, and so forth and in various situations for web-based handwriting recognition on PC tablets, identifying number plates of vehicles, handling bank cheques, digits entered in any forms etc. Machine Learning provides various methods through which human efforts can be reduced in recognizing the manually written digits.

Deep Learning is a machine learning method that trains computers to do what easily falls into place for people: learning through examples. With the utilization of deep learning methods, human attempts can be diminished in perceiving, learning, recognizing and in a lot more regions. Using deep learning, the computer learns to carry out classification works from pictures or contents from any document. Deep Learning models can accomplish state-of-art accuracy, beyond the human level performance. The digit recognition model uses large datasets in order to recognize digits from distinctive sources.

### PURPOSE

### The main objective was to actualize a pattern characterization method to perceive the handwritten digits provided in the MINIST data set of images of handwritten digits (0‐9). The goal of our work is to create a model that will be able to recognize and classify the handwritten digits from images by using concepts of Convolution Neural Network. Though the goal of our research is to create a model for digit recognition and classification, it can also be extended to letters and an individual’s handwriting. With high accuracy rates, the model can solve a lot of real life problems.

### The main applications are vehicle license-plate recognition, postal letter-sorting services, Cheque truncation system (CTS) scanning and historical document preservation in archaeology departments, old documents automation in libraries and banks, etc. All these areas deal with large databases and hence demand high recognition accuracy, lesser computational complexity and consistent performance of the recognition system.

# CHAPTER 2

## LITERATURE SURVEY

### 2.1EXISTING PROBLEM

The fundamental problem with handwritten digit recognition is that handwritten digits do not always have the same size, width, orientation, and margins since they vary from person to person. People can struggle to read others’ handwriting. The handwritten digits are not always of the same size, width, orientation as they differ from writing of person to person, so the general problem would be while classifying the digits.

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.

### 2.2REFERENCES

### Effective Handwritten Digit Recognition using Deep Convolution Neural Network(2021)

Yellapragada SS Bharadwaj1 , Rajaram P 2 , Sriram V.P3 , Sudhakar S4 , Kolla Bhanu Prakash5

This paper proposed a simple neural network approach towards handwritten digit

recognition using convolution. With machine learning algorithms like KNN,SVM/SOM,

recognizing digits is considered as one of the unsolvable tasks due to its distinctiveness in

the style of writing. In this paper, Convolution Neural Networks are implemented with an

MNIST dataset of 70000 digits with 250 distinct forms of writings. The proposed method

achieved 98.51% accuracy for real-world handwritten digit prediction with less than 0.1

% loss on training with 60000 digits while 10000 under validation.

# A Comparative Analysis on Bangla Handwritten Digit Recognition with Data Augmentation and Non-Augmentation Process(2020)

# Md. Abdullah Al Nasim , Refat E Ferdous , Mahim Anzum Haque Pantho and Atiqul Islam Chowdhury

Determination of Bangla handwritten digit is a momentous image classification task. Though object recognition technology is getting smarter day by day, still Bangla handwritten digit recognition remains inconclusive. Researchers are becoming more concerned about handwritten digit recognition for it's educational and advantageous importance. But it is a matter of trouble that the improvement in Bangla handwritten digit recognition is significantly less as compared to the other languages. To improve the performance of the Bangla handwritten digit recognition system, we have designed a model, in which all basic Bangla digits have been classified. Furthermore, we have also demonstrated Densenet121 architecture in our system. For recognizing Bangla handwriting digits, we proposed CNN (Convolution Neural Network) model. Our system has been experimented on the NumtaDB dataset for recognizing Bangla digit both with augmentation and non-augmentation.

**Bangla Handwritten Digit Recognition And Generation(2021)**

Md Fahim Sikder

Handwritten digit or numeral recognition is one of the classical issues in the area of pattern recognition and has seen tremendous advancement because of the recent wide availability of computing resources. Plentiful works have already done on English, Arabic, Chinese, Japanese handwritten script. Some work on Bangla also have been done but there is space for development. From that angle, in this paper, an architecture has been implemented which achieved the validation accuracy of 99.44% on BHAND dataset and outperforms Alexnet and Inception V3 architecture. Beside digit recognition, digit generation is another field which has recently caught the attention of the researchers though not many works have been done in this field especially on Bangla. In this paper, a Semi-Supervised Generative Adversarial Network or SGAN has been applied to generate Bangla handwritten numerals and it successfully generated Bangla digits.

**Handwritten Digit Recognition using OpenCV and CNN(2021)**

Swetha, Hithaishi, L. Tejaswini, Parthasaradhi,V. Venkateswara Rao

Handwritten Digit Recognition (HDR) is the process of converting images of handwritten digit into digital format. A lot of money is wasted on converting the information that is in paper to digital format. This problem can be solved by using HDR. The heart of our project lies within the ability to develop an efficient algorithm that can recognize the handwritten digits which are scanned and sent as input by the user. The goal of this paper is to observe the variation of different algorithms that can classify the handwritten digits using different hidden layers, various number of epochs and to make a comparison based on the accuracy. This experiment is performed using the Modified National Institute of Standards and Technology (MNIST) dataset.

**A Novel Handwritten Digit Classification System Based on Convolutional Neural Network Approach(2021)**

Ali Abdullah Yahya , Jieqing Tan 2 and Min Hu 2

An enormous number of CNN classification algorithms have been proposed in the literature. Nevertheless, in these algorithms, appropriate filter size selection, data preparation, limitations in datasets, and noise have not been taken into consideration. As a consequence, most of the algorithms have failed to make a noticeable improvement in classification accuracy. To address the shortcomings of these algorithms, our paper presents the following contributions: Firstly, after taking the domain knowledge into consideration, the size of the effective receptive field (ERF) is calculated. Calculating the size of the ERF helps us to select a typical filter size which leads to enhancing the classification accuracy of our CNN. Secondly, unnecessary data leads to misleading results and this, in turn, negatively affects classification accuracy. To guarantee the dataset is free from any redundant or irrelevant variables to the target variable, data preparation is applied before implementing the data classification mission. Thirdly, to decrease the errors of training and validation, and avoid the limitation of datasets, data augmentation has been proposed. Fourthly, to simulate the real-world natural influences that can affect image quality, we propose to add an additive white Gaussian noise with σ = 0.5 to the MNIST dataset. As a result, our CNN algorithm achieves state-of-the-art results in handwritten digit recognition, with a recognition accuracy of 99.98%, and 99.40% with 50% noise.

### PROBLEM STATEMENT DEFINITION

The problem statement is to classify handwritten digits. The goal is to take an image of a handwritten digit and determine what that digit and character is. It is easy for the human to perform a task accurately by practicing it repeatedly and memorizing it for the next time. Human brain can process and analyze images easily. Also, recognize the different elements present in the images.

The handwritten digit recognition is the capability of computer applications to recognize the 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 Python library over the MNIST dataset to recognize handwritten digits. Handwriting number recognition is a challenging problem researchers had been research into this area for so long especially in the recent years.

# CHAPTER 3

## IDEATION AND PROPOSED SOLUTION

**­­­­**

### 

### 3.1 EMPATHY MAP CANVAS

### 

### 

­­­­

### 3.2IDEATION & BRAINSTORMING

### 



**3.3 PROPOSED SOLUTION:**

|  |  |  |
| --- | --- | --- |
| **SNO** | **Parameter** | **Description** |
|  | Problem Statement (Problem to be solved) | * The handwritten digit recognition is the capability of computer applications to recognize the 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. |
|  | Idea / Solution description | * It is the capability of a computer to fetch the mortal handwritten integers from different sources like images, papers, touch defences. * It allows user to translate all those signature and notes into electronic words in a text document format and this data only requires far less physical space than the storage of the physical copies. |
|  | Novelty / Uniqueness | Accurately recognize the digits rather than recognizing all the characters like OCR. |
|  | Social Impact / Customer Satisfaction | * AI developed the app called Handwritten digit Recognizer. * It converts the written word into digital approximations to identify characters before churning out a digital approximation. * As it is designed to solve real-world problems, it should be highly reliable and trustworthy in every way, and users throughout the world should be able to use it effectively. |
|  | Business Model (Revenue Model) | * This system can be integrated with traffic surveillance cameras to recognize the vehicle’s number plates for effective traffic management. * Can be integrated with Postal system to identify and recognize the address and the pin-code details easily. |
|  | Scalability of the Solution | There is no limit in the number of digits it can be recognized. |

### PROBLEM SOLUTION FIT

### 

### 

### 

# CHAPTER 4

## REQUIREMENT ANALYSIS

## 

## 4.1 FUNCTIONAL REQUIREMENTS:

|  |  |  |
| --- | --- | --- |
| **FR No.** | **Functional Requirement (Epic)** | **Sub Requirement (Story / Sub-Task)** |
| FR-1 | System Design | Image Data:  Handwritten digit recognition is the ability of a computer to recognise human handwritten digits from a wide range of sources, including pictures, papers, touch screens, etc., and classify them into ten predetermined categories (0-9). This has been the focus of innumerable studies in the field of deep learning. |
| FR-2 | Hosting | Website:  Web hosting enables online access to the HTML, graphics, and other components of a website. Every website you've ever visited is hosted by a server. The amount of server space provided to a website depends on the hosting type. The four primary types of hosting are shared, dedicated, VPS, and reseller. |
| FR-3 | Training | Digit Classifier Model:  Use the MNIST database of handwritten digits to train a neural network to predict the digit from a picture. Assemble the data for training and validation first. |
| FR-4 | Deployment | Cloud:  Cloud computing is defined as an internet-based virtual platform that allows for limitless data storage and access. The cloud provides a variety of IT services, such as server, database, virtual storage, networking, and servers. |
| FR-5 | Dataset | MNIST dataset:  Modified National Institute of Standards and Technology the shorthand for The MNIST dataset.  It consists of 60,000 minuscule square grayscale photos, each measuring 28 by 28, with handwritten single numerals from 0 to 9. |
|  |  |  |

### 

### 

### 4.2 NON-FUNCTIONAL REQUIREMENTS

|  |  |  |
| --- | --- | --- |
| **FR No.** | **Non-Functional Requirement** | **Description** |
| NFR-1 | **Usability** | The conversion of handwritten compositions on paper into legible computer text documents is made possible by handwriting recognition software.  The recognition of handwritten characters is one of the major issues with pattern recognition applications. Filling out forms, processing bank checks, and sorting mail are examples of applications using digit recognition. |
| NFR-2 | **Security** | In addition to categorising the digit, the algorithm also creates an extensive description of the instantiation parameters, which could reveal details like the writing style. The generative models have segmentation powered by recognition capabilities. |
| NFR-3 | **Reliability** | The handwritten digit recognition system uses neural network concepts. This neural network model is trained used various samples so it can automatically deduce the rules for reading handwritten digits. By increasing the number of training instances, the network may also learn more about handwriting and hence improve its accuracy.  To recognise handwritten numbers, a variety of methods and algorithms can be employed, including Deep Learning/CNN, SVM, Gaussian Naive Bayes, KNN, Decision Trees, Random Forests, etc. |
| NFR-4 | **Performance** | Written textual content in Graphical User Interface (GUI) technology offers accuracy rates of more than or equal to 75%. However, spacing discrepancies, handwriting anomalies, and human typeface variability reduce the accuracy of character identification. |

# 

# 

# 

# CHAPTER 5

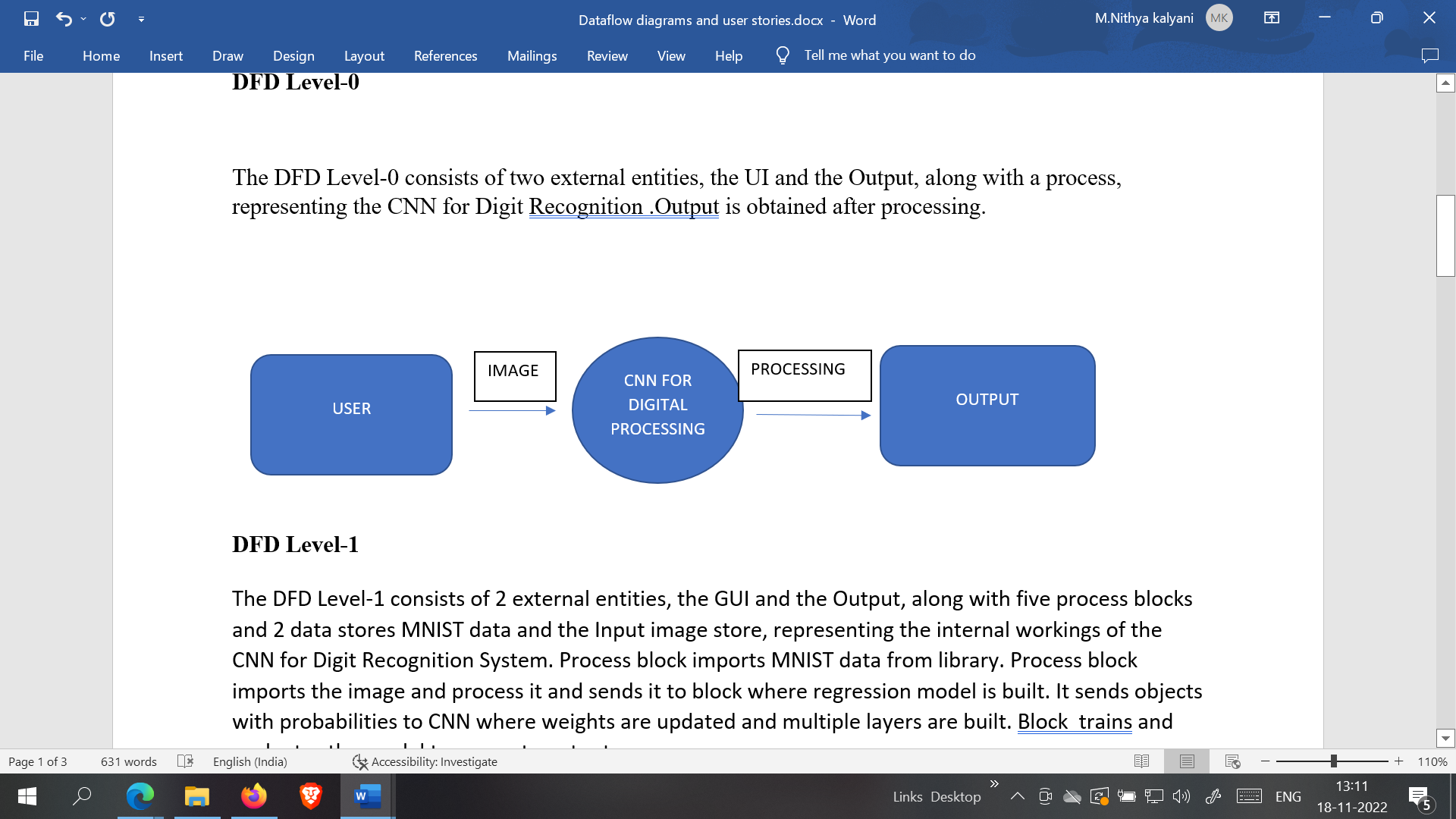
## PROJECT DESIGN

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

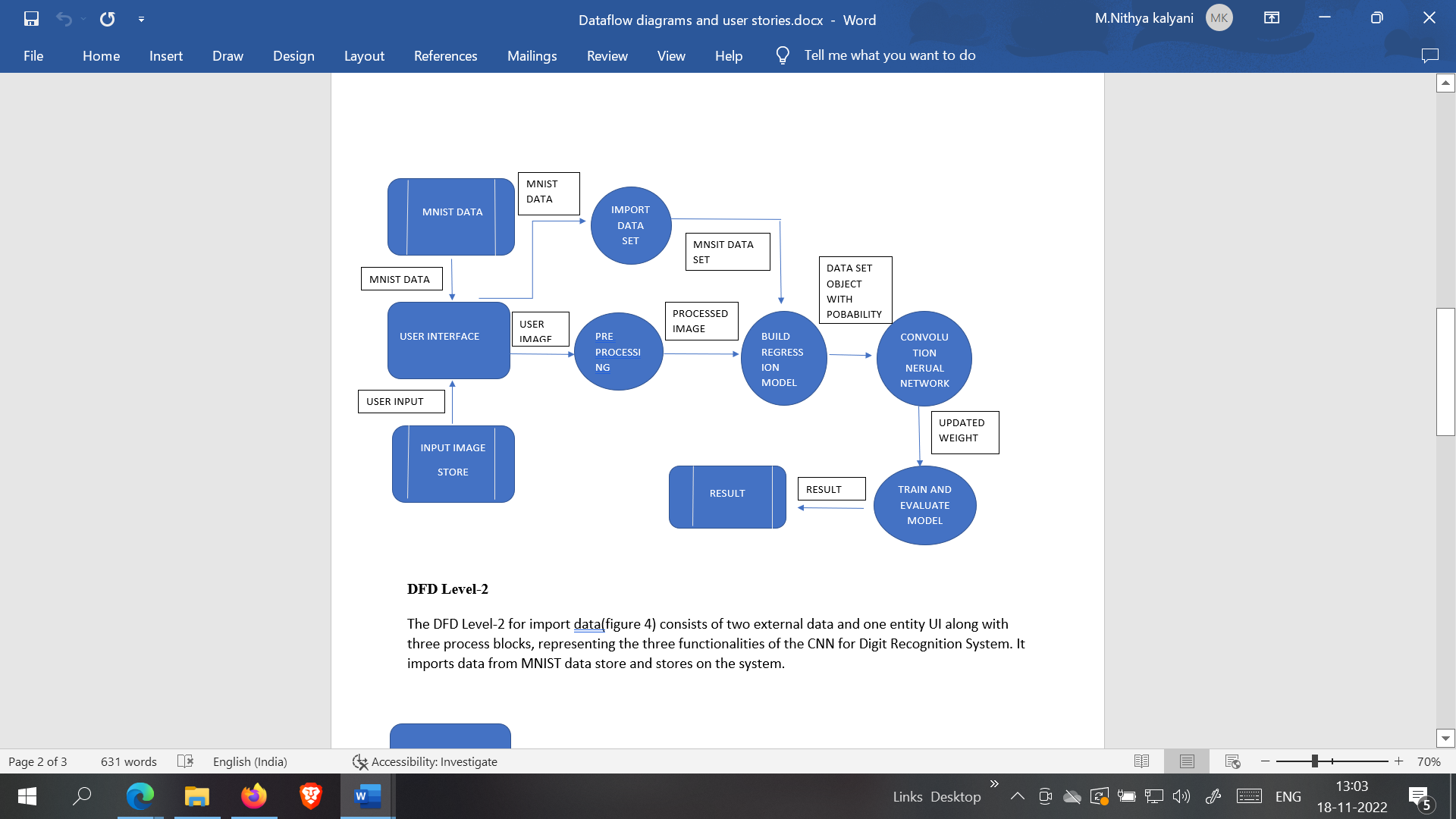
**DFD Level-0**

The DFD Level-0 consists of two external entities, the UI and the Output, along with a process, representing the CNN for Digit Recognition .Output is obtained after processing.



**DFD Level-1**

The DFD Level-1 consists of 2 external entities, the GUI and the Output, along with five process blocks and 2 data stores MNIST data and the Input image store, representing the internal workings of the CNN for Digit Recognition System. Process block imports MNIST data from library. Process block imports the image and process it and sends it to block where regression model is built. It sends objects with probabilities to CNN where weights are updated and multiple layers are built. Block trains and evaluates the model to generate output.

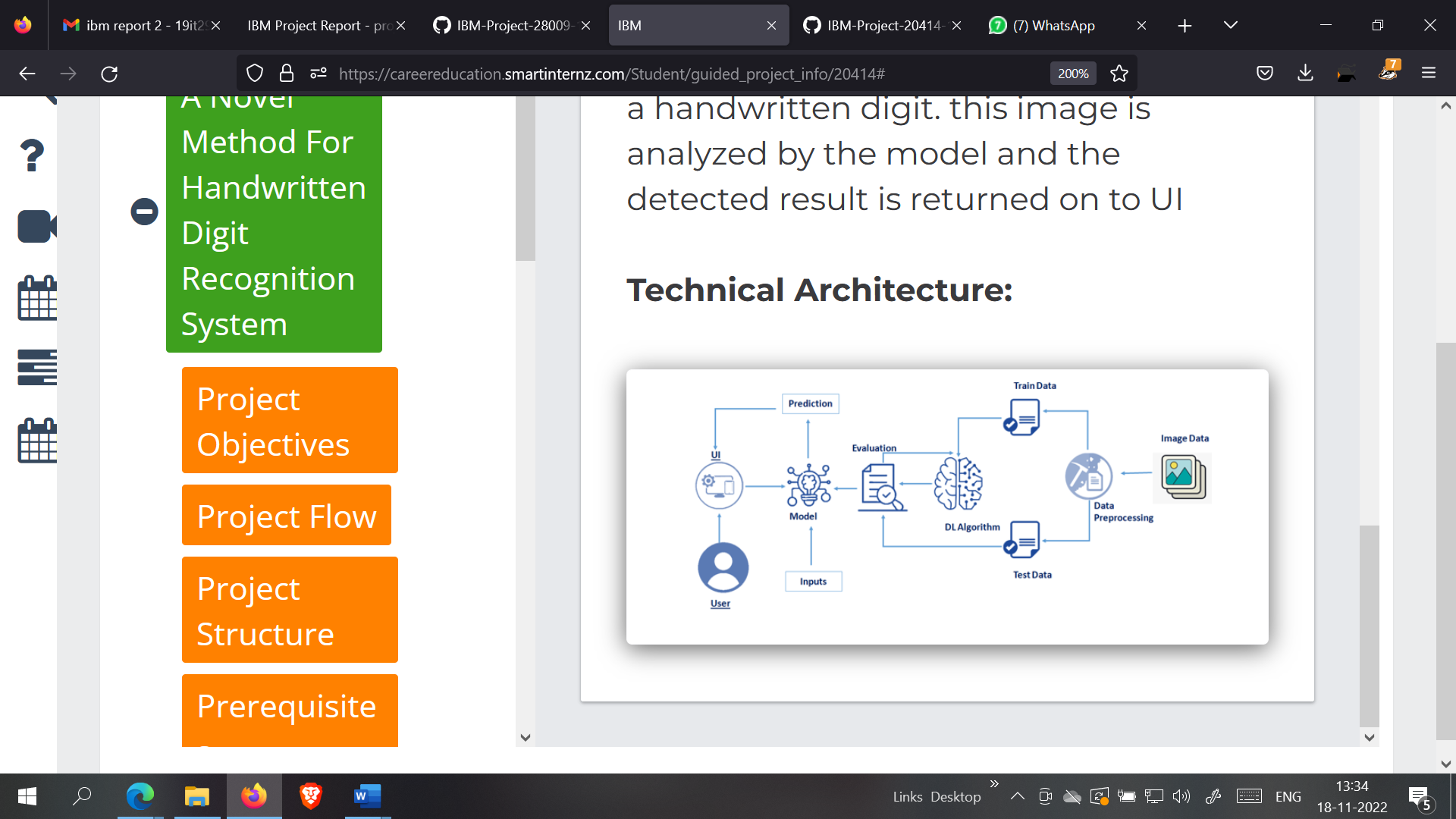
****

**DFD Level-2**

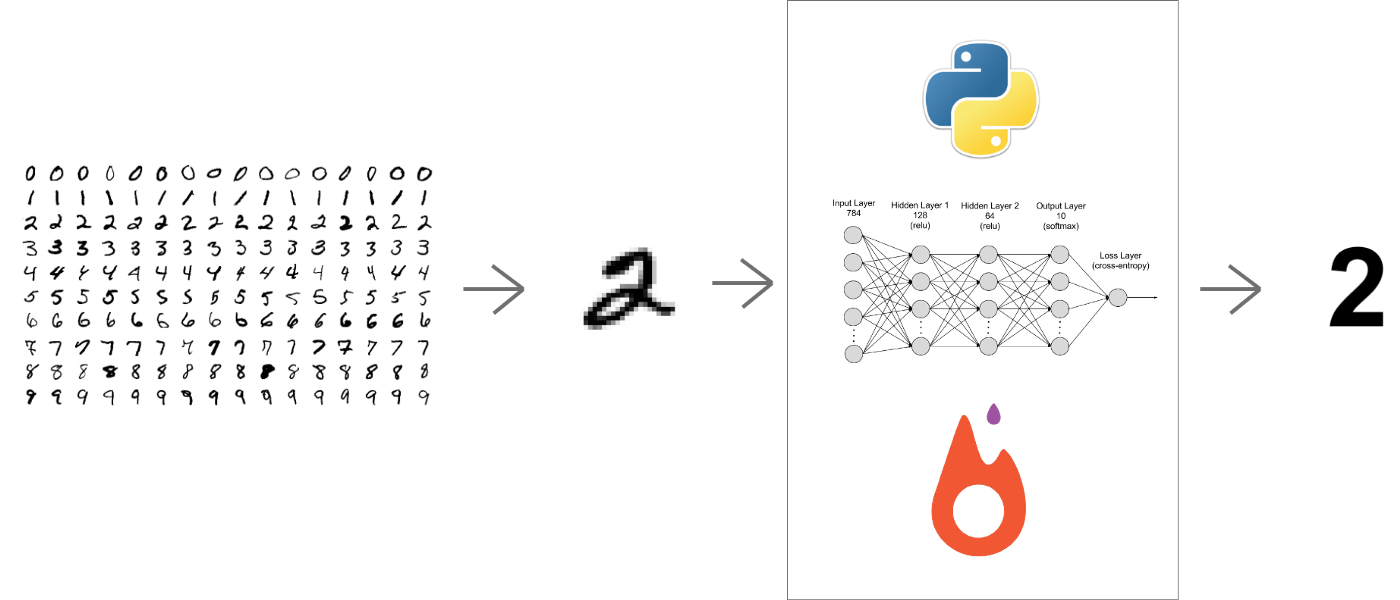
The DFD Level-2 for import data(figure 4) consists of two external data and one entity UI along with three process blocks, representing the three functionalities of the CNN for Digit Recognition System. It imports data from MNIST data store and stores on the system.

### 5.2 SOLUTION & TECHNICAL ARCHITECTURE

### 



**MNIST DATASET PROCESSING WITH PYTHON**



### 

### 

**5.3 COMPONENTS & TECHNOLOGIES:**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Component** | **Description** | **Technology** |
|  | User Interface | How user interacts with the system-image and digital writing options | HTML + CSS |
|  | Machine learning model | The goal of developing the machine learning model | To train the model to recognize the written digit |
|  | Programming language | Language used to build the model | Python |
|  | Database | Data Type, Configurations to store the data | MySQL, NoSQL, etc. |
|  | Cloud Database | Maintaining the database in the cloud | IBM DB2, IBM Cloudant |
|  | File Storage | File storage requirements | IBM Block Storage |
|  | External API | To integrate the application with other applications | IBM API, Aadhar AI |
|  | Infrastructure (Server / Cloud) | Resource to run and train the model | Local servers and cloud services |

### USER STORIES

| **User Type** | **Functional Requirement (Epic)** | **User Story Number** | **User Story / Task** | **Acceptance criteria** | **Priority** | **Release** |
| --- | --- | --- | --- | --- | --- | --- |
| Customer (Mobile user) | Home | USN-1 | As a user, I can view the guide and awareness to use the 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’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 |
|  | 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-5 | 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-6 | As a user, I will train and test the input to get the maximum accuracy of output. | I can train and test the application until it gets maximum accuracy of the result. | High | Sprint-4 |
|  |  | USN-7 | As a user I can access the MNIST dataset | I can access the MNIST dataset and produce accurate results | Medium | Sprint-3 |
| Customer (Web user) | Home | USN-8 | 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 |
| Customer  (PC user) | Home | USN-9 | 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-10 | 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-11 | As it is an open source, can use it cost freely. | I can use it without any payment to be paid for it to access. | Medium | Sprint-2 |
|  | Recognize | USN-12 | As a user I’m able to access web application from anywhere virtually. | I can use the application portably anywhere. | High | Sprint-1 |
|  |  | USN-13 | As it is a web application, it is installation free | I can use it without the installation of the application or any software. | Medium | Sprint-3 |
|  | Predict | USN-14 | 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 |

# 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 | USN-1 | As a user, I can collect the dataset from various resources with different handwritings. | 10 | Low | Sanjana  Priyanka |
| Sprint-1 | Data Preprocessing | USN-2 | As a user, I can load the dataset, handling the missing data, scaling and split data into training and testing | 10 | Medium | Priyanka  Nishanthini |
| 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 | Priyanka  Sanjana  Nishanthini |
| Sprint-2 | Add CNN layers | USN-4 | Creating the model and adding the input, hidden and output layers to it. | 5 | High | Nishanthini  Sanjana |
| 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 | Priyanka |
| Sprint-2 | Train and test the model | USN-6 | As a user, let us train our model with our image dataset | 6 | Medium | Nishanthini  zamin  sanjana |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Sprint-2 | Train and test the model | USN-6 | As a user, let us train our model with our image dataset | 6 | Medium | Nishanthini zamin sanjana |
| 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 | Priyanka  Zamin |
| 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 | Nishanthini Sanjana  Priyanka |
| Sprint-3 |  | USN-9 | As a user, I can know the details of the fundamental usage of the application. | 5 | Low | Zamin |
| Sprint-3 |  | USN-10 | As a user, I can see the predicted / recognized digits in the application. | 5 | Medium | Sanjana |
| 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 | Sanjana  Nishanthini |
| 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 | Sanjana  Priyanka |

**6.2 SPRINT DELIVERY SCHEDULE:**

| **Sprint** | **Total Story Points** | **Duration** | **Sprint Start Date** | **Sprint End Date (Planned)** | **Story Points Completed (as on Planned End Date)** | **Sprint Release Date (Actual)** |
| --- | --- | --- | --- | --- | --- | --- |
| Sprint-1 | 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 |
|  |  |  |  |  |  |  |

**6.3 REPORT FROM JIRA**

**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](https://www.visual-paradigm.com/scrum/what-is-agile-software-development/) methodologies such as [Scrum](https://www.visual-paradigm.com/scrum/scrum-in-3-minutes/). However, burn down charts can be applied to any project containing measurable progress over time.

**CHAPTER 7**

**CODING & SOLUTION**

**7 FEATURE – FLASK FILE UPLOADING**

Handling file upload in Flask is very easy. It needs an HTML form with its enctype attribute set to ‘multipart/form-data’, posting the file to a URL. The URL handler fetches file from request.files[] object and saves it to the upload folder.

import torch

import base64

import config

import matplotlib

import numpy as np

from PIL import Image

from io import BytesIO

from train import MnistModel

import matplotlib.pyplot as plt

from flask import Flask, request, render\_template, jsonify

matplotlib.use('Agg')

MODEL = None

DEVICE = torch.device('cuda:0' if torch.cuda.is\_available() else 'cpu')

app = Flask(\_\_name\_\_)

class SaveOutput:

    def \_\_init\_\_(self):

        self.outputs = []

    def \_\_call\_\_(self, module, module\_in, module\_out):

        self.outputs.append(module\_out)

    def clear(self):

        self.outputs = []

def register\_hook():

    save\_output = SaveOutput()

    hook\_handles = []

    for layer in MODEL.modules():

        if isinstance(layer, torch.nn.modules.conv.Conv2d):

            handle = layer.register\_forward\_hook(save\_output)

            hook\_handles.append(handle)

    return save\_output

def module\_output\_to\_numpy(tensor):

    return tensor.detach().to('cpu').numpy()

def autolabel(rects, ax):

    """Attach a text label above each bar in \*rects\*, displaying its height."""

    for rect in rects:

        height = rect.get\_height()

        ax.annotate('{0:.2f}'.format(height),

                    xy=(rect.get\_x() + rect.get\_width() / 2, height),

                    xytext=(0, 3),  # 3 points vertical offset

                    textcoords="offset points",

                    ha='center', va='bottom')

def prob\_img(probs):

    fig, ax = plt.subplots()

    rects = ax.bar(range(len(probs)), probs)

    ax.set\_xticks(range(len(probs)), (0, 1, 2, 3, 4, 5, 6, 7, 8, 9))

    ax.set\_ylim(0, 110)

    ax.set\_title('Probability % of Digit by Model')

    autolabel(rects, ax)

    probimg = BytesIO()

    fig.savefig(probimg, format='png')

    probencoded = base64.b64encode(probimg.getvalue()).decode('utf-8')

    return probencoded

def interpretability\_img(save\_output):

    images = module\_output\_to\_numpy(save\_output.outputs[0])

    with plt.style.context("seaborn-white"):

        fig, \_ = plt.subplots(figsize=(20, 20))

        plt.suptitle("Interpretability by Model", fontsize=50)

        for idx in range(16):

            plt.subplot(4, 4, idx+1)

            plt.imshow(images[0, idx])

        plt.setp(plt.gcf().get\_axes(), xticks=[], yticks=[])

    interpretimg = BytesIO()

    fig.savefig(interpretimg, format='png')

    interpretencoded = base64.b64encode(

        interpretimg.getvalue()).decode('utf-8')

    return interpretencoded

def mnist\_prediction(img):

    save\_output = register\_hook()

    img = img.to(DEVICE, dtype=torch.float)

    outputs = MODEL(x=img)

    probs = torch.exp(outputs.data)[0] \* 100

    probencoded = prob\_img(probs)

    interpretencoded = interpretability\_img(save\_output)

    \_, output = torch.max(outputs.data, 1)

    pred = module\_output\_to\_numpy(output)

    return pred[0], probencoded, interpretencoded

@app.route("/process", methods=["GET", "POST"])

def process():

    data\_url = str(request.get\_data())

    offset = data\_url.index(',')+1

    img\_bytes = base64.b64decode(data\_url[offset:])

    img = Image.open(BytesIO(img\_bytes))

    img = img.convert('L')

    img = img.resize((28, 28))

    # img.save(r'templates\image.png')

    img = np.array(img)

    img = img.reshape((1, 28, 28))

    img = torch.tensor(img, dtype=torch.float).unsqueeze(0)

    data, probencoded, interpretencoded = mnist\_prediction(img)

    response = {

        'data': str(data),

        'probencoded': str(probencoded),

        'interpretencoded': str(interpretencoded),

    }

    return jsonify(response)

@app.route("/", methods=["GET", "POST"])

def start():

    return render\_template("default.html")

if \_\_name\_\_ == "\_\_main\_\_":

    MODEL = MnistModel(classes=10)

    MODEL.load\_state\_dict(torch.load(

        'checkpoint/mnist.pt', map\_location=DEVICE))

    MODEL.to(DEVICE)

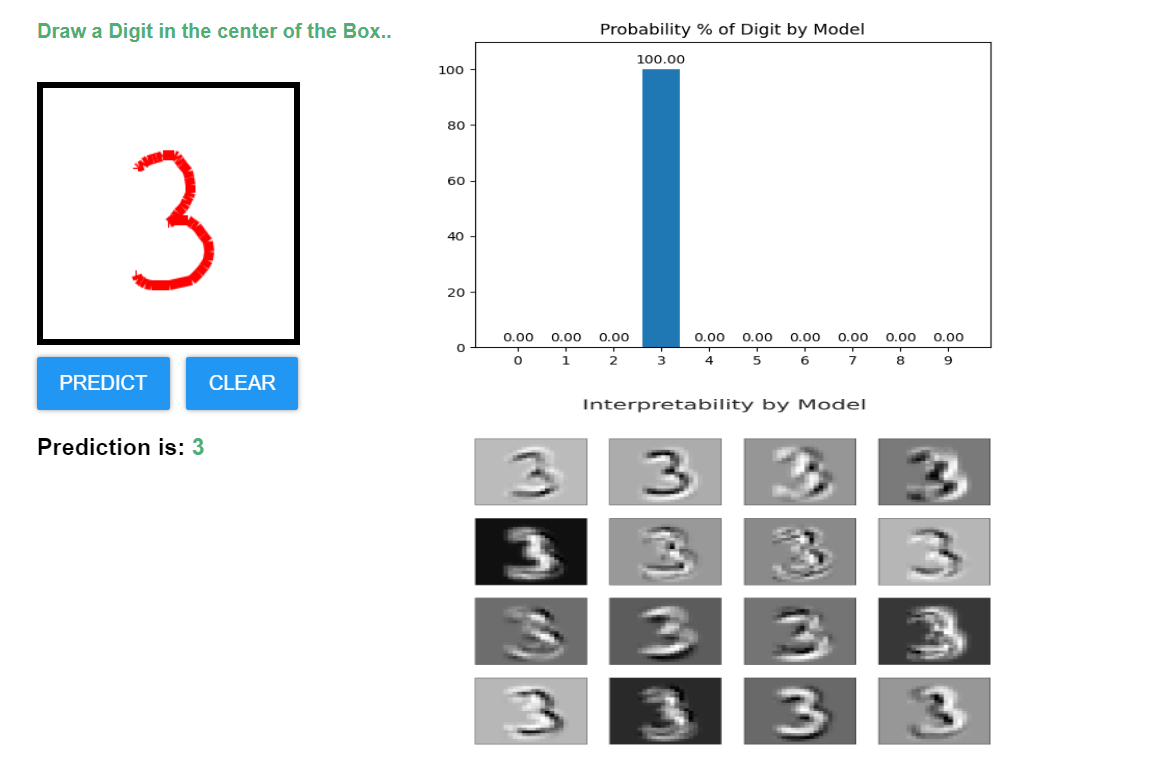
    MODEL.eval()

    app.run(host=config.HOST, port=config.PORT, debug=config.DEBUG\_MODE)

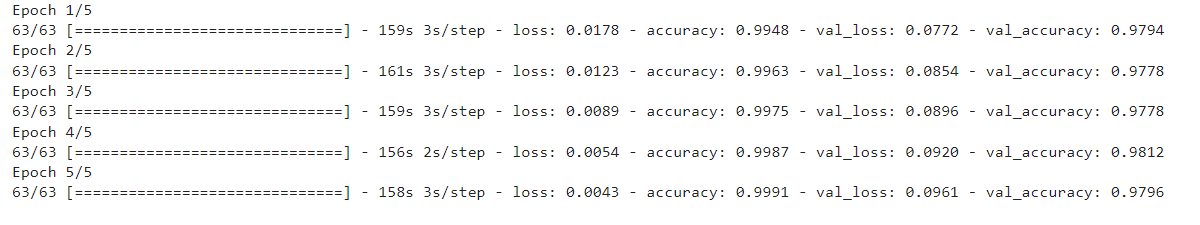
# CHAPTER 8

## RESULTS

### 9.1 OUTPUT:



**EPOCHS:**



# CHAPTER 9

## 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
    - Neural Network is used to train and identify written digits for greater efficiency.
    - The accuracy rate is very high.
    - Speed of data entry
    - It is much easier to dictate the machine than to write
    - Easier data retrieval

### 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
    - There is a wide range of handwriting – good and bad.
    - It is tricky for programmers to provide enough examples of how every character might look.
    - Customers must try with clear image and neat handwriting to get accuracy in digits.
    - Unclear image will not give accurate results.

**CHAPTER 10**

## CONCLUSION

Convolutional Neural Network (CNN) adds its significant improvement to the Manuscript Document Recognition System. This paper tells us the effectiveness of CNN-based classification of data and pre-processing methods. Our model clearly sees handwriting and achieves outgoing predictions of up to 82.16% and accurate predictions of up to 69.16%. However the model can be continuously developed using multiple training samples. This will help the model to learn as well as the generalize better. There are many images in the training set that are completely invisible to the human eye.

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

Through extensive evaluation using a MNIST dataset, the present work suggests the role of various hyper-parameters. Fine tuning of hyper-parameters is essential in improving the performance of CNN architecture. We achieved a recognition rate of 99.89% with the Adam optimizer for the MNIST database, which is better than all previously reported results. The effect of increasing the number of convolutional layers in CNN architecture on the performance of handwritten digit recognition is clearly presented through the experiments.

# CHAPTER 11

## FUTURE SCOPE

This project can be enhanced with a great field of machine learning and artificial intelligence. The world can think of a software which can recognize the text from a picture and can show it to the others, for example a shop name detector. Or this project can be extended to a greater concept of all the character sets in the world. This project has not gone for the total English alphabet because there will be more and many more training sets and testing values that the neural network model will not be enough to detect. Think of a AI modeled car sensor going with a direction modeling in the roadside, user shall give only the destination.

All of these enhancement is an application of the texture analysis where advanced image processing, Neural network model for training and advanced AI concepts will come. These applications can be modeled further .As this project is fully done by free and available resources and packages this can be also a limitation of the project. The fund is very important because all machine learning libraries and advanced packages are not available for free. Unless of those the most of the visualizing platforms like on which developers are doing some works like Watson Studio or Aws. These all are mainly paid platforms where a lot of ML projects are going on.

This project is far from complete and there is a lot of room for improvement.

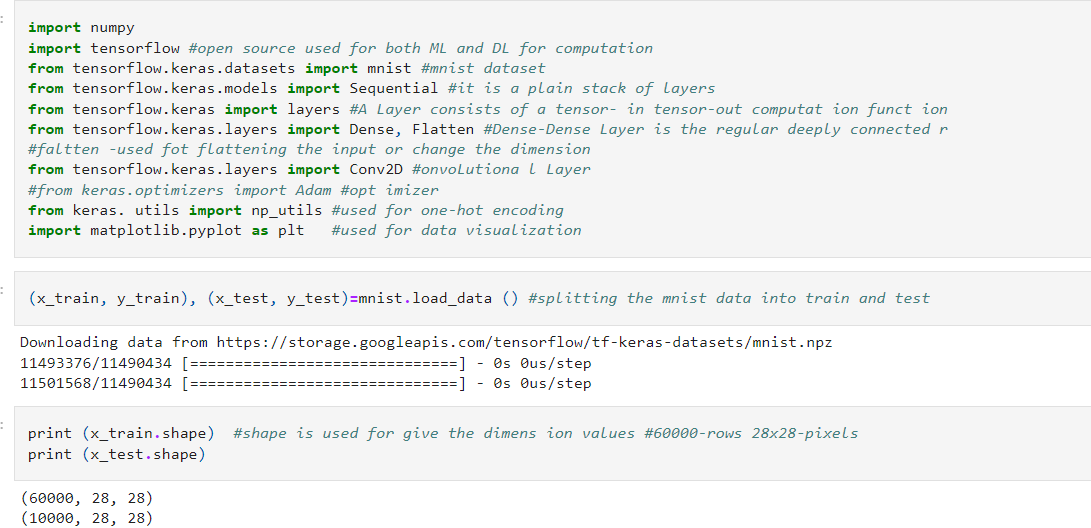
Some of the improvements that can be made to this project are as follows:

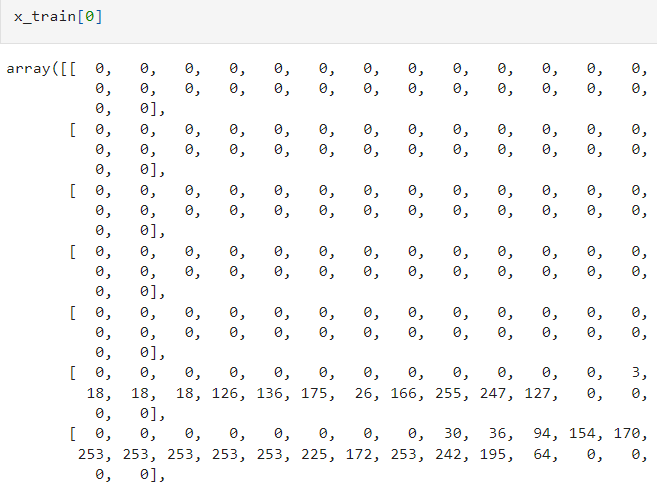
* + - Add support to detect from digits multiple images and save the results
    - Add support to detect multiple digits
    - Improve model to detect digits from complex images
    - Add support to different languages to help users from all over the world

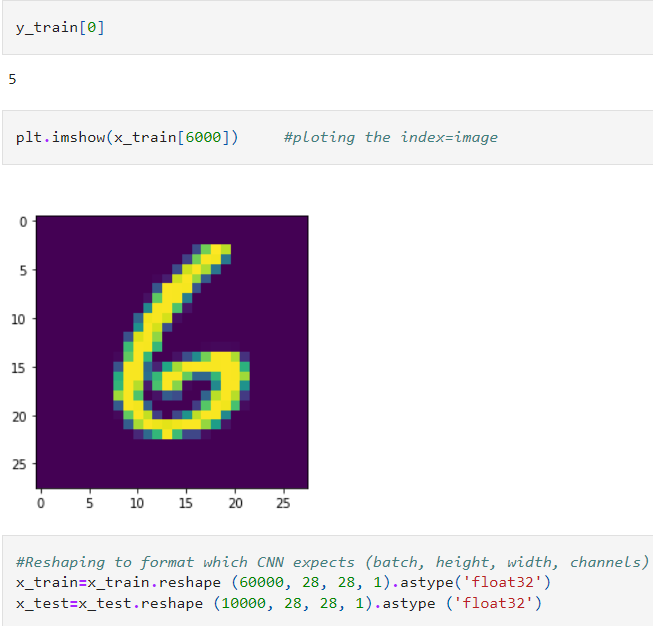
This project has endless potential and can always be enhanced to become better. Implementing this concept in the real world will beneﬁt several industries and reduce the workload on many workers, enhancing overall work efﬁciency.

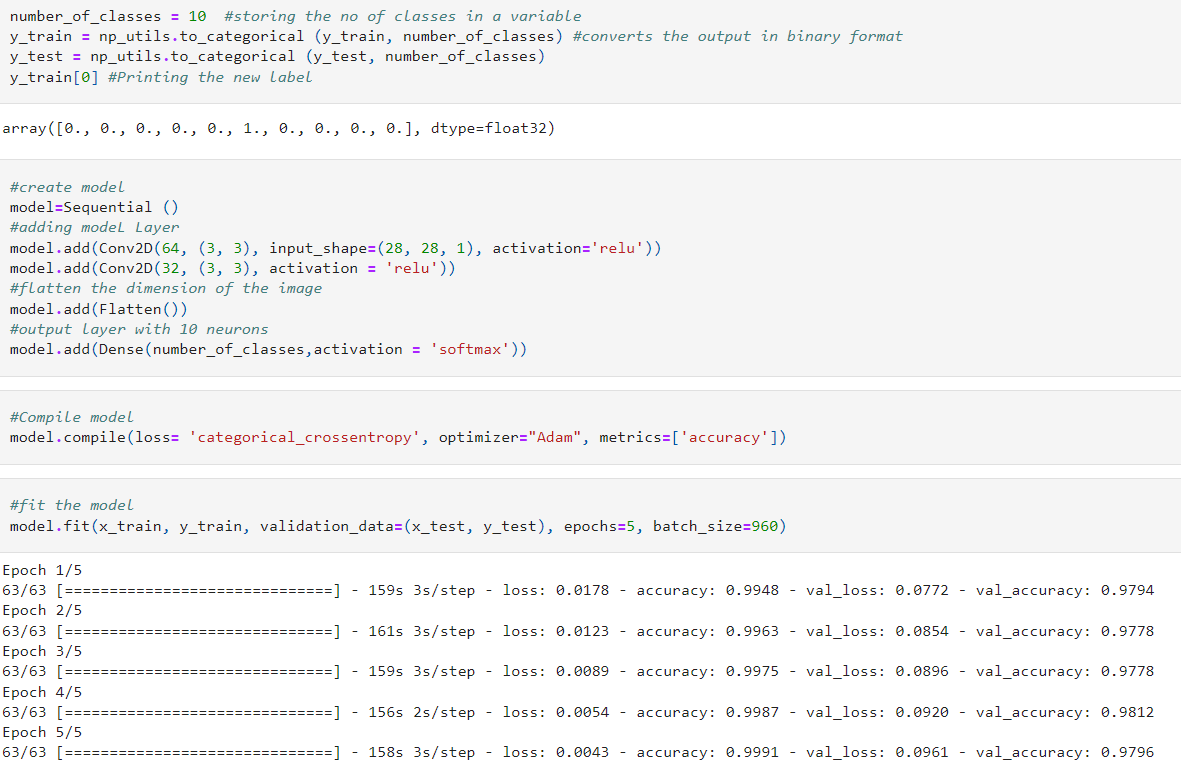
## APPENDIX

### SOURCE CODE











### TRAIN THE MODEL ON IBM:





**HOME PAGE(CSS) – style.css**

.ripple {

background-position: center;

transition: background 0.8s;

}

/\* .ripple:hover {

} \*/

.ripple:active {

background: #25282b radial-gradient(circle, transparent 1%, #47a7f5 1%) center/15000%;

/\* background-color: #6eb9f7; \*/

background-size: 100%;

transition: background 0s;

}

/\* Button style \*/

button {

border: none;

border-radius: 2px;

padding: 12px 18px;

font-size: 16px;

text-transform: uppercase;

cursor: pointer;

color: white;

background-color: #2196f3;

box-shadow: 0 0 4px #999;

outline: none;

}

/\* Span style \*/

#text {

color: #4DAF74;

}

/\* Canvas style \*/

#can {

margin-top: 10;

border:5px solid;

}

/\* canvasimage style \*/

#canvasimg{

position:absolute;

display:none;

}

body{

font-family: sans-serif;

}

#side{

float: left;

padding-left:15%;

}

**FLASK APP - app.py**

import torch

import base64

import config

import matplotlib

import numpy as np

from PIL import Image

from io import BytesIO

from train import MnistModel

import matplotlib.pyplot as plt

from flask import Flask, request, render\_template, jsonify

matplotlib.use('Agg')

MODEL = None

DEVICE = torch.device('cuda:0' if torch.cuda.is\_available() else 'cpu')

app = Flask(\_\_name\_\_)

class SaveOutput:

    def \_\_init\_\_(self):

        self.outputs = []

    def \_\_call\_\_(self, module, module\_in, module\_out):

        self.outputs.append(module\_out)

    def clear(self):

        self.outputs = []

def register\_hook():

    save\_output = SaveOutput()

    hook\_handles = []

    for layer in MODEL.modules():

        if isinstance(layer, torch.nn.modules.conv.Conv2d):

            handle = layer.register\_forward\_hook(save\_output)

            hook\_handles.append(handle)

    return save\_output

def module\_output\_to\_numpy(tensor):

    return tensor.detach().to('cpu').numpy()

def autolabel(rects, ax):

    """Attach a text label above each bar in \*rects\*, displaying its height."""

    for rect in rects:

        height = rect.get\_height()

        ax.annotate('{0:.2f}'.format(height),

                    xy=(rect.get\_x() + rect.get\_width() / 2, height),

                    xytext=(0, 3),  # 3 points vertical offset

                    textcoords="offset points",

                    ha='center', va='bottom')

def prob\_img(probs):

    fig, ax = plt.subplots()

    rects = ax.bar(range(len(probs)), probs)

    ax.set\_xticks(range(len(probs)), (0, 1, 2, 3, 4, 5, 6, 7, 8, 9))

    ax.set\_ylim(0, 110)

    ax.set\_title('Probability % of Digit by Model')

    autolabel(rects, ax)

    probimg = BytesIO()

    fig.savefig(probimg, format='png')

    probencoded = base64.b64encode(probimg.getvalue()).decode('utf-8')

    return probencoded

def interpretability\_img(save\_output):

    images = module\_output\_to\_numpy(save\_output.outputs[0])

    with plt.style.context("seaborn-white"):

        fig, \_ = plt.subplots(figsize=(20, 20))

        plt.suptitle("Interpretability by Model", fontsize=50)

        for idx in range(16):

            plt.subplot(4, 4, idx+1)

            plt.imshow(images[0, idx])

        plt.setp(plt.gcf().get\_axes(), xticks=[], yticks=[])

    interpretimg = BytesIO()

    fig.savefig(interpretimg, format='png')

    interpretencoded = base64.b64encode(

        interpretimg.getvalue()).decode('utf-8')

    return interpretencoded

def mnist\_prediction(img):

    save\_output = register\_hook()

    img = img.to(DEVICE, dtype=torch.float)

    outputs = MODEL(x=img)

    probs = torch.exp(outputs.data)[0] \* 100

    probencoded = prob\_img(probs)

    interpretencoded = interpretability\_img(save\_output)

    \_, output = torch.max(outputs.data, 1)

    pred = module\_output\_to\_numpy(output)

    return pred[0], probencoded, interpretencoded

@app.route("/process", methods=["GET", "POST"])

def process():

    data\_url = str(request.get\_data())

    offset = data\_url.index(',')+1

    img\_bytes = base64.b64decode(data\_url[offset:])

    img = Image.open(BytesIO(img\_bytes))

    img = img.convert('L')

    img = img.resize((28, 28))

    # img.save(r'templates\image.png')

    img = np.array(img)

    img = img.reshape((1, 28, 28))

    img = torch.tensor(img, dtype=torch.float).unsqueeze(0)

    data, probencoded, interpretencoded = mnist\_prediction(img)

    response = {

        'data': str(data),

        'probencoded': str(probencoded),

        'interpretencoded': str(interpretencoded),

    }

    return jsonify(response)

@app.route("/", methods=["GET", "POST"])

def start():

    return render\_template("default.html")

if \_\_name\_\_ == "\_\_main\_\_":

    MODEL = MnistModel(classes=10)

    MODEL.load\_state\_dict(torch.load(

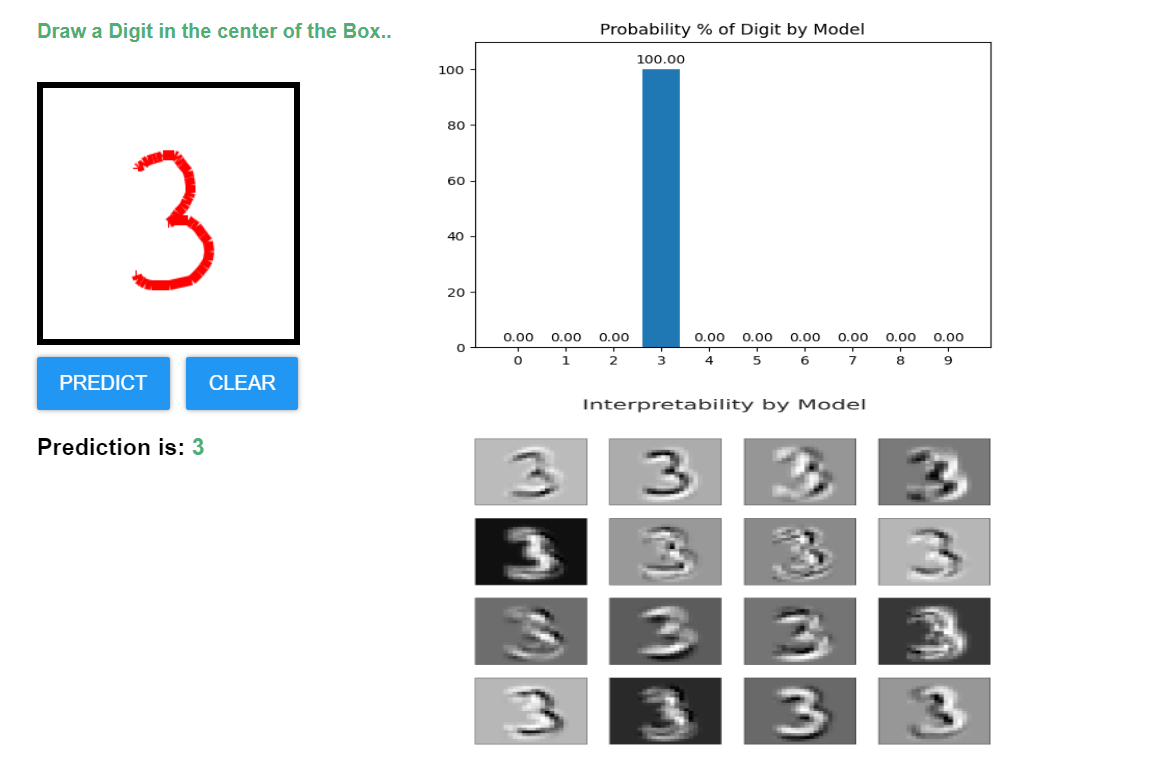
        'checkpoint/mnist.pt', map\_location=DEVICE))

    MODEL.to(DEVICE)

    MODEL.eval()

    app.run(host=config.HOST, port=config.PORT, debug=config.DEBUG\_MODE)

**SCREENSHOTS:**



**GITHUB LINK:**

[**https://github.com/IBM-EPBL/IBM-Project-41411-1660641859**](https://github.com/IBM-EPBL/IBM-Project-41411-1660641859)