PROJECT REPORT

A NOVEL METHOD FOR HANDWRITTEN DIGIT RECOGNITION SYSTEM

Submitted By

PNT2022TMID02012

Poomani A
Ragavi T
Sowmiya P V
Mahakrishna Moorthy M

TABLE OF CONTENTS

1. INTRODUCTION

1.1 Project Overview

Machine learning and deep learning play an important role in computer technology and Artificial Intelligence. With the use of Deep Learning and Machine learning, human effort can be reduced in recognizing, learning, predictions and in many more areas.

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

1.2 Purpose

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

2. LITERATURE SURVEY

2.1 Existing problem

- ✓ The different architectures of CNN, hybrid CNN,CNN - RNN and CNNHMM models, and domain - specific recognition system, are not thoroughly inquired and evolutionary algorithms are not clearly explored for optimizing CNN learning parameters ,the number of layers, learning rate and kernel sizes of convolutional filters.

- ✓ The fluctuation of accuracies for handwritten digits was observed for 15 epochs by varying the hidden layers. There is no clear explanation given for observing variation in the overall classification accuracy by varying the number of hidden layers and batch size.

2.2 References

S.NO	Author Name	Paper Title	Journal/ Conference title	Page No/ Volume No	Year of Publication	Description
	Savita Ahlawat, Amit Choudhary, Anand Nayyar, Saurabh Singh and Byungun Yoon.	Improved Handwritten Digit Recognition Using Convolutional Neural Networks (CNN)	IEEE Sensors Journal		2020	In this paper, with the aim of improving the performance of handwritten digit recognition, they valuated variants of a convolutional neural network to avoid complex preprocessing, costly feature extraction and a complex ensemble (classifier combination) approach

						of a traditional recognition system.
	Vijayalaxmi R Rudraswamimath, Bhavanishankar and Channasandra.	Handwritten Digit Recognition using CNN	International Journal of Innovative Science and Research Technology	Volume -4 Issue-6	2019	In this paper, the most widely used Machine learning algorithms, KNN, SVM, RFC and CNN have been trained and tested on the same data in order acquire the comparison between the classifiers
	Fathma Siddique, Shadman Sakib and Md. Abu Bakr Siddique.	Recognition of Handwritten Digit using Convolutional Neural Network in Python with Tensorflow and Comparison of Performance for Various Hidden Layers	5th International Conference on Advances in Electrical Engineering (ICAEE)		2019	In this paper, they observed the variation of accuracies of CNN to classify handwritten digits for 15 epochs using various numbers of hidden layers and epochs and

						to make the comparison between the accuracies. For this performance evaluation of CNN, they performed the experiment using Modified National Institute of Standards and Technology(MN IST) dataset.
	Akanksha Gupta, Ravindra Pratap Narwaria and Madhav Singh	Review on Deep Learning Handwritten Digit Recognition using Convolutional Neural Network	International Journal of Recent Technology and Engineering (IJRTE)	Volume -9 Issue-5	2021	In this paper, Object Character Recognition (OCR) is used on printed or documented letters to convert them into text. The database has training image database of 60,000 images and

						testing image database of 10,000 images. The KNN algorithm describes categorical value by making use of majority of votes of K - nearest neighbors, the K value used to differ here.
	Md. Anwar Hossain and Md. Mohon Ali	Recognition of Handwritten Digit using Convolutional Neural Network (CNN)	Global Journal of Computer Science and Technology: D Neural & Artificial Intelligence	Volume 19 Issue2	2019	The goal of this work will be to create a model that will be able to identify and determine the handwritten digit from its image with better accuracy using using the concepts of Convolutional Neural Network and MNIST

						dataset. Later it can be extended for character recognition and real-time person's handwriting. The results can be made more accurate with more convolution layers and more number of hidden neurons.

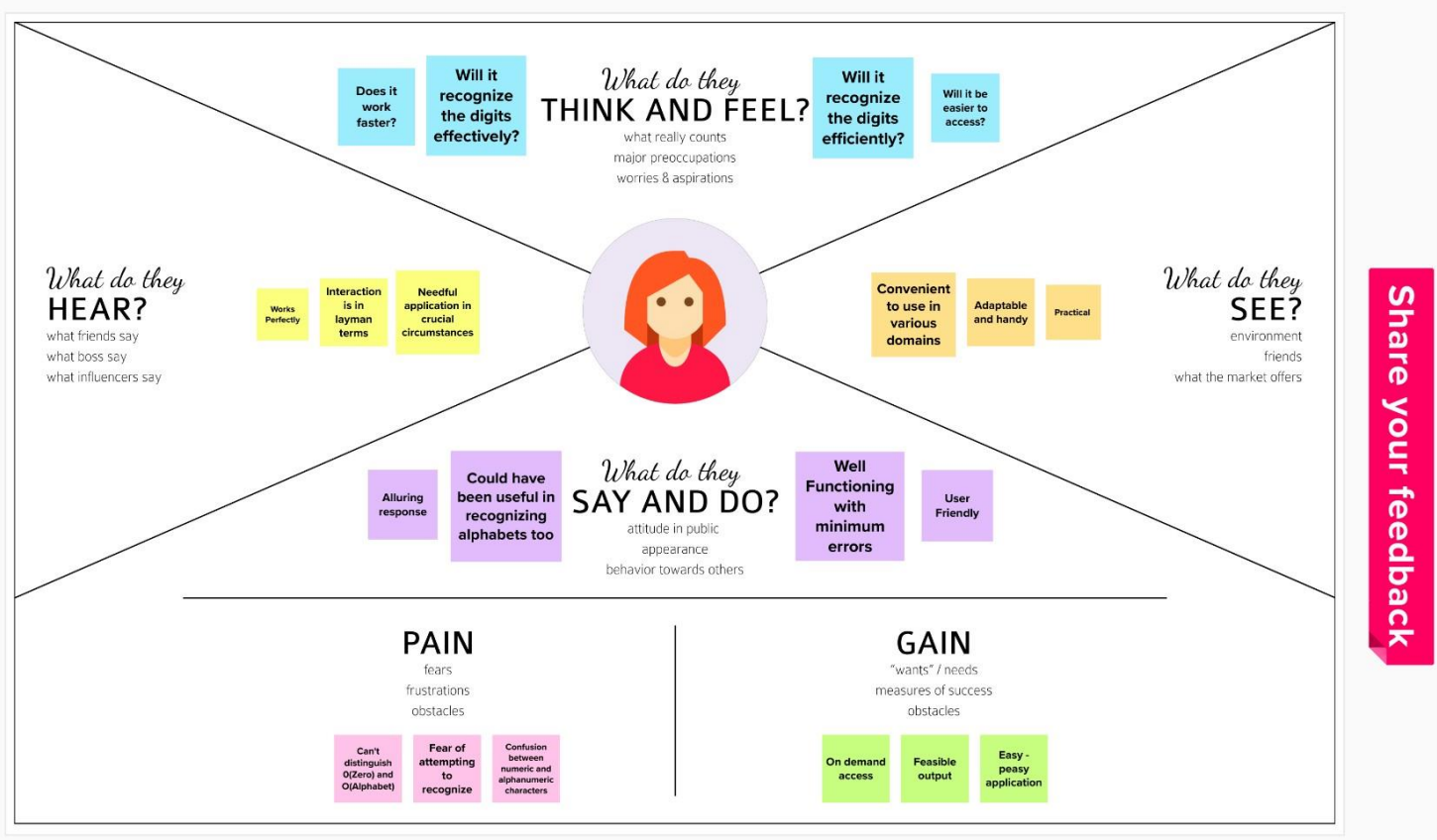
2.3 Problem Statement Definition

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

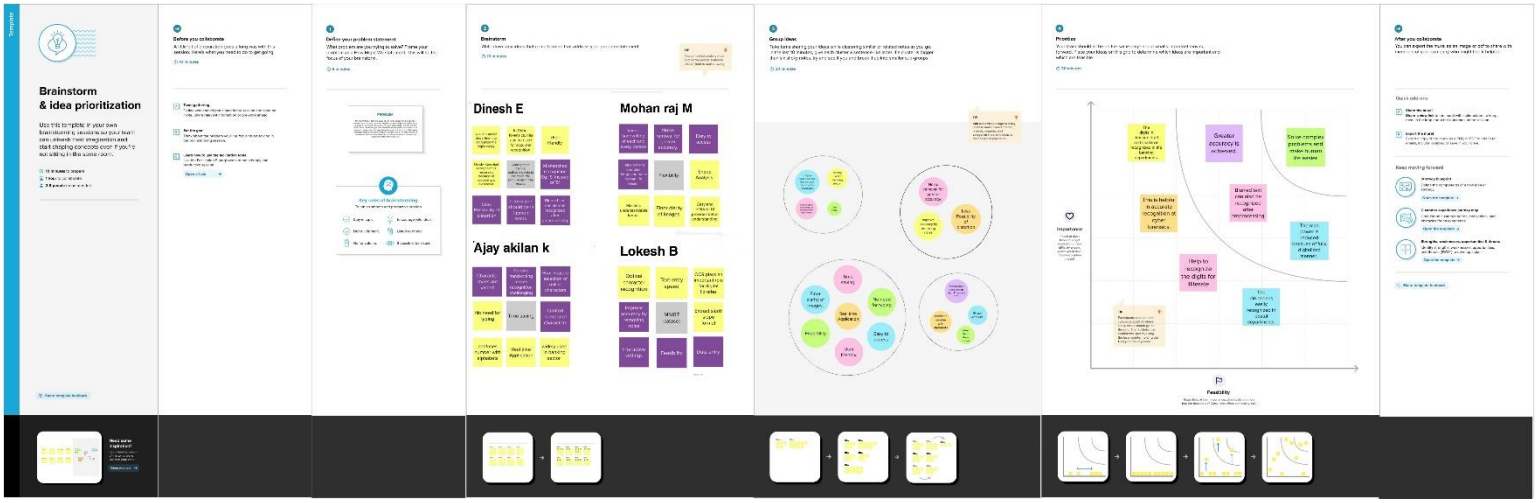
Hence, there comes a need for handwritten digit recognition in many real time applications. MNIST data set is widely used for this recognition process and it has 70000 handwritten digits. We use Artificial neural networks to train these images and build a deep learning model. Web application is created where the user can upload an image of a handwritten digit. this image is analyzed by the model and the detected result is returned on to UI(User Interface).

3. IDEATION & PROPOSED SOLUTION

3.1 Empathy Map Canvas



3.2 Ideation & Brainstorming



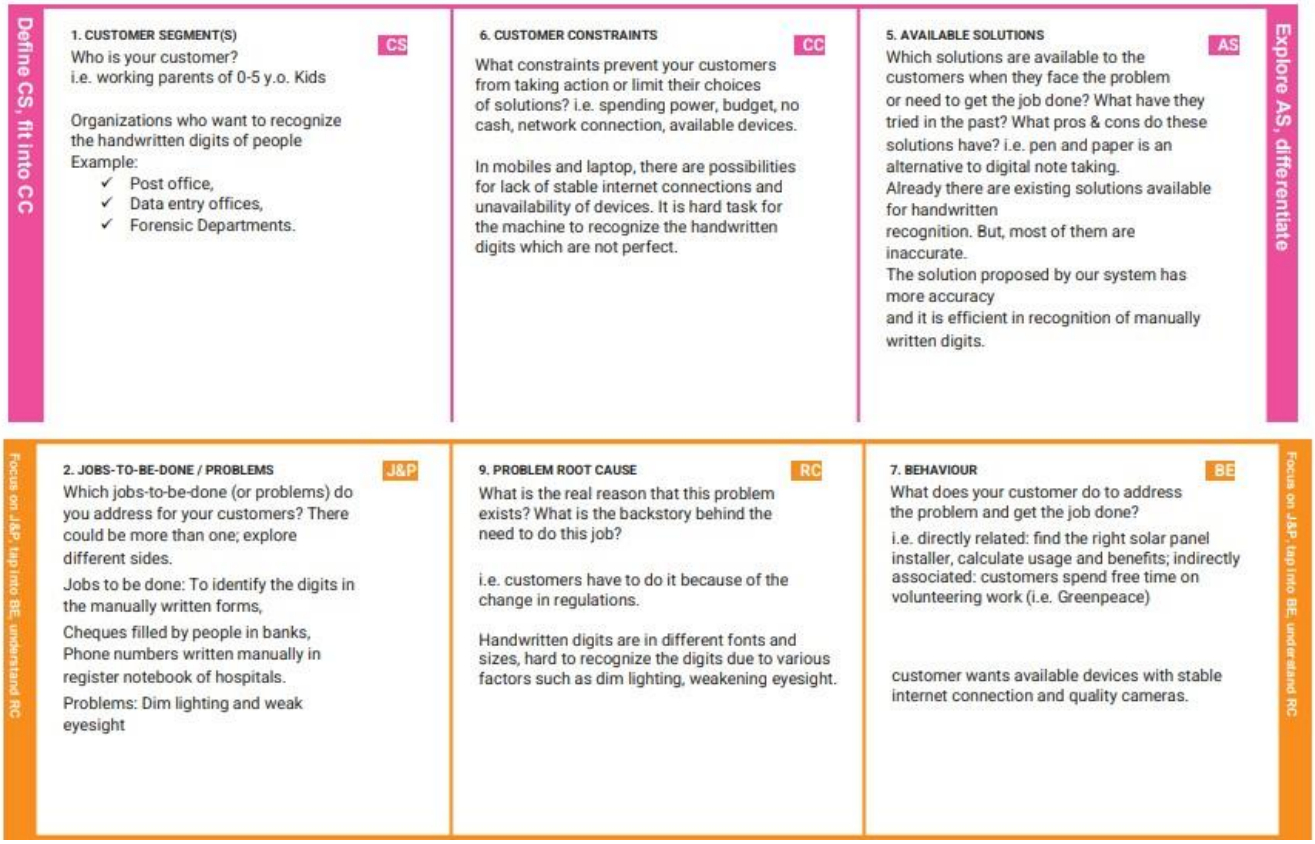
3.3 Proposed Solution

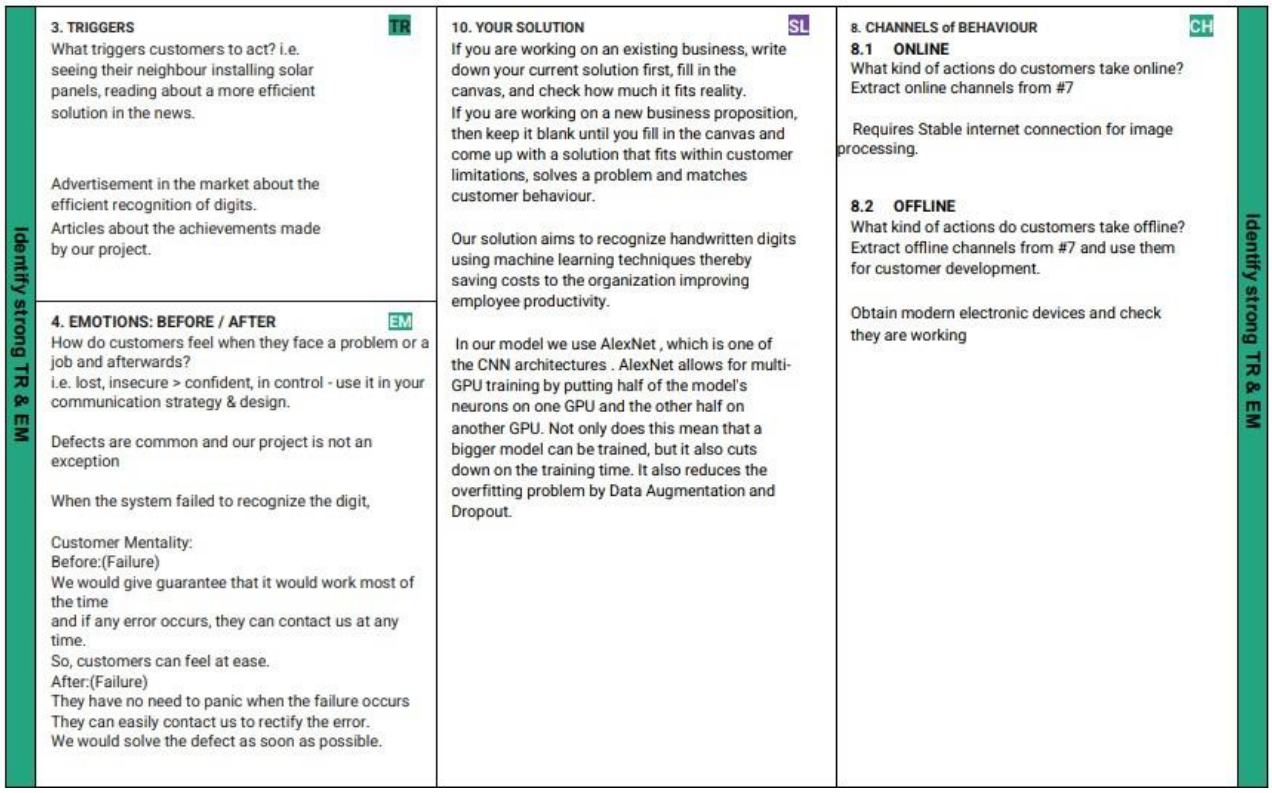
S.No.	Parameter	Description
1	Problem Statement (Problem to be solved)	Handwriting recognition is one of the compelling research works going on because every individual in this world has their own style of writing. It is the capability of the computer to identify and understand handwritten digits or characters automatically. Because of the progress in the field of science and technology, everything is being digitized to reduce human effort. Hence, there comes a need for handwritten digit recognition in many real-time applications. The user interacts with the UI (User Interface) to upload the image as input. The uploaded image is analyzed by the model which is integrated. Once the model analyses the uploaded image, the prediction is showcased on the UI.
2	Idea / Solution description	Convolutional Neural Networks (CNN) has become one of the most appealing approaches and has been an ultimate factor in a variety of recent success and challenging machine learning applications. In our model we use AlexNet , which is one of the CNN architectures . AlexNet allows for multi-GPU training by putting half of the model's neurons on one GPU and the other half on another

		GPU. Not only does this mean that a bigger model can be trained, but it also cuts down on the training time. It also reduces the overfitting problem by Data Augmentation and Dropout.
3	Novelty / Uniqueness	Handwritten Digit Recognition is the capability of a computer to fete the mortal handwritten integers from different sources like images, papers,touch defenses, etc. And classify them into 10 predefined classes(0-9).This is the existing method along with this we add some features to make our project unique among them.
4	Social Impact / Customer Satisfaction	Even the unclear or blurred digits can be recognized after the removal of noise and data preprocessing .One such application is a handwritten digit recognition system that can be used in postal mail sorting, bank check processing, form data entry, etc.,
5	Business Model (Revenue Model)	Handwritten digit recognition is necessary because everything is digitalized. The benefits of handwritten digit recognizer is high. In the banking sector, it is very efficient. It is used to recognize the figures written on cheques.So, Varied handwriting of each and every person in the cheque can be identified. Handwritten addresses are difficult to sort by machine, not

		necessarily because of sloppy handwriting, but because people write all over the envelope.We have hard time segmenting handwritten addresses into their components, such as ZIP code or street address, because very few people print addresses neatly in a prescribed format. So, this problem can be solved using Handwritten digit recognition system.
6	Scalability of the Solution	In our model, AlexNet significantly outperformed as it is trained on a GTX 580 GPU with only 3 GB of memory which couldn't fit the entire network. So the network was split across 2 GPUs, with half of the neurons(feature maps) on each GPU. So, a greater accuracy can be attained by allowing multi-GPU training by putting half of the model's neurons on one GPU and the other half on another GPU.

3.4 Problem Solution fit





4. REQUIREMENT ANALYSIS

4.3 Functional requirement

Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	Input Correlation	Digital image correlation is a technique that combines image registration and tracking methods for accurate 2D measurements of changes in images and recognizes the characters from the images.
FR-2	Data Preparation	Data preparation is the process of preparing raw data so that it is suitable for further processing and analysis.
FR-3	Feature Extraction	Feature extraction refers to the process of transforming raw data into numerical features that can be processed while preserving the information in the original data set.
FR-4	Character Classification	In character classification phase, the attributes of the data in the picture are compared to the classes in the database to determine in which class the picture belongs to.

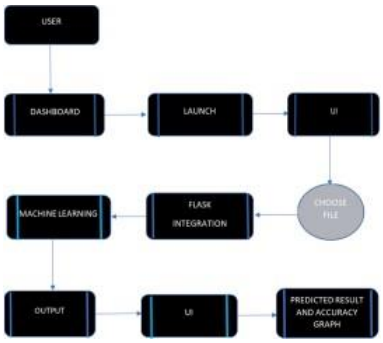
4.4 Non-Functional requirements

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

FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	Handwritten digit recognition is one of the major important issues in pattern recognition applications. Some of the applications for digit recognition include data entry forms, Bank check processing etc,.
NFR-2	**Security**	The applications of handwritten digit recognition can be used in the banking sector where it can be used to maintain the security pin numbers safely. It can be also used for blind-people by using sound output.
NFR-3	**Reliability**	Reliability indicates the probability that the system will perform its intended function for a larger period of sufficient time and also it will operate in a secured environment without any failures.
NFR-4	**Performance**	The standard implementations of neural networks achieve an accuracy of approximately (98–99)
		percent in correctly classifying the handwritten digits.
NFR-5	**Availability**	The features for handwritten digit recognition have been Acquainted. These features are based on shape analysis of the digit image and extract slant or slope information. They are effective in obtaining good recognition of accuracy.
NFR-6	**Scalability**	The scalability in the task of handwritten digit recognition, using a classifier, has great importance and it makes use of online handwriting recognition on computer tablets, recognizing zip codes on mail for postal mail sorting, processing bank check amounts, numeric entries in forms filled up manually(for example - tax forms) and so on.

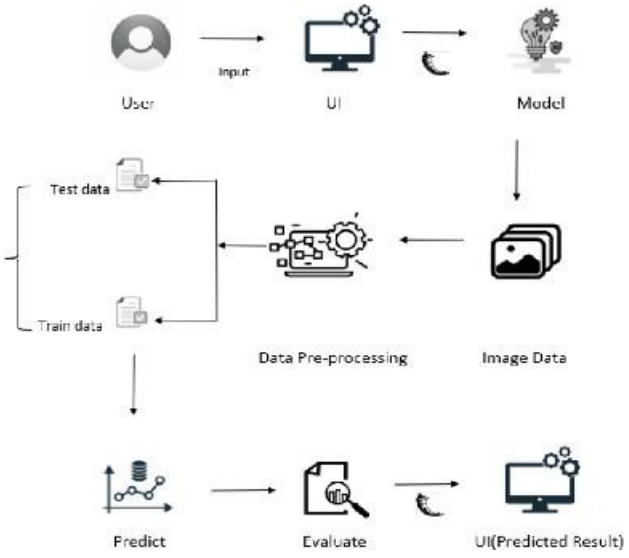
5. PROJECT DESIGN

5.1Data Flow Diagrams



5.2 Solution & Technical Architecture

Solution Architecture



Technology Architecture

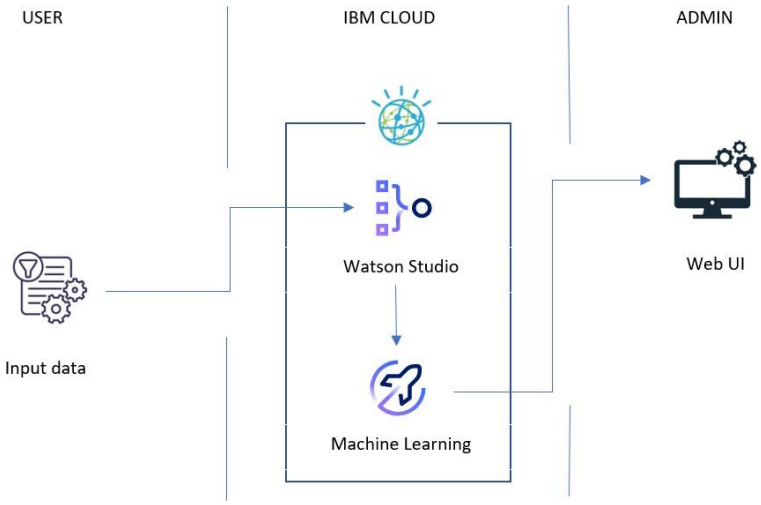


Table-1 : Components & Technologies:

S.No	Component	Description	Technology
1.	User Interface	How user interacts with application e.g. Web UI	HTML, CSS, JavaScript
2.	Application Logic-1	Model is built	Python
3.	Application Logic-2	Python model is deployed	IBM Watson Studio
4.	File Storage	Predicted outputs of the image are stored in a local folder.	Local Filesystem
5.	Machine Learning Model	To predict the image uploaded by the user.	Image Recognition Model
6.	Infrastructure (Server / Cloud)	Application Deployment on Local System / Cloud Local Server Configuration: Flask Cloud Server Configuration : IBM Watson Studio	Local, Cloud Foundry.

Table-2: Application Characteristics:

S.No	Characteristics	Description	Technology
1.	Open-Source Frameworks	List the open-source frameworks used	Flask
2.	Security Implementations	List all the security / access controls implemented, use of firewalls etc.	e.g. SHA-256, Encryptions, IAM Controls, OWASP etc.
3.	Scalable Architecture	High workload can be supported without undergoing any major changes.	Technology used in the architecture is that with Python and the IBM cloud.
4.	Availability	Readily available enables the IT Infrastructure to function when some of the components fail.	Technology used is IBM cloud.
5.	Performance	Performance technology is a field which uses various tools,processes and procedures in a systematic and efficient manner to improve the desired outcomes of individuals and organizations.	Technology used is python.

5.3User Stories

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	Dashboard	USN-1	As a user, they can see the information regarding the prediction of handwritten digit recognition.	I can see the information regarding digit recognition.	High	Sprint 1
	Launch	USN-2	On clicking the launch button, it will redirect the user to a page where the images to be predicted can be uploaded.	I can see the launch button.	High	Sprint 1
	Upload	USN-3	Users can select the image from the local storage.	I can upload the image.	High	Sprint 2
	Predict	USN-4	Once the image is uploaded, it will predict the respective image.		High	Sprint 3
	Display	USN-5	The predicted image will be displayed with the accuracy chart.	I can see the result with accuracy.	High	Sprint 4

6. PROJECT PLANNING & SCHEDULING

6.1Sprint Planning & Estimation

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Data Collection	USN-1	As a user, I can collect the dataset from various resources with different handwritings.	10	Low	Poomani A, Ragavi T, Sowmiya P V, Mahakrishna Moorthy M
Sprint-1	Data Preprocessing	USN-2	As a user, I can load the dataset, handle the missing data, scale, and split data into train and test.	10	Medium	Poomani A, Ragavi T, Sowmiya P V, Mahakrishna Moorthy M
Sprint-2	Model Building	USN-3	As a user, I will get an application with an MLmodel which provides high accuracy of recognized handwritten digits.	5	High	Poomani A, Ragavi T, Sowmiya P V, Mahakrishna Moorthy M

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	Poomani A, Ragavi T, Sowmiya P V, Mahakrishna Moorthy M
Sprint-2	Train & test the model	USN-6	As a user, let us train our model with our image dataset.	6	Medium	Poomani A, Ragavi T, Sowmiya P V, Mahakrishna Moorthy M
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	Poomani A, Ragavi T, Sowmiya P V, Mahakrishna Moorthy M
Sprint-3	Building UI Application	USN-8	As a user, I will upload the handwritten digit image to the application by clicking an upload button.	5	High	Poomani A, Ragavi T, Sowmiya P V, Mahakrishna Moorthy M
Sprint-3		USN-9	As a user, I can know the details of the fundamental usage of the application.	5	Low	Poomani A, Ragavi T, Sowmiya P V, Mahakrishna Moorthy M
Sprint-3		USN-10	As a user, I can see the predicted/recognized digits in the application.	5	Medium	Poomani A, Ragavi T, Sowmiya P V, Mahakrishna Moorthy M
Sprint-4	Train the model on IBM	USN-11	As a user, I train the model on IBM and integrate flask/Django with scoring end points.	10	High	Poomani A, Ragavi T, Sowmiya P V, Mahakrishna Moorthy M
Sprint-4	Cloud Deployment	USN-12	As a user, I can access the web application and make use of the product from anywhere.	10	High	Poomani A, Ragavi T, Sowmiya P V, Mahakrishna Moorthy M

6.2 Sprint Delivery Schedule

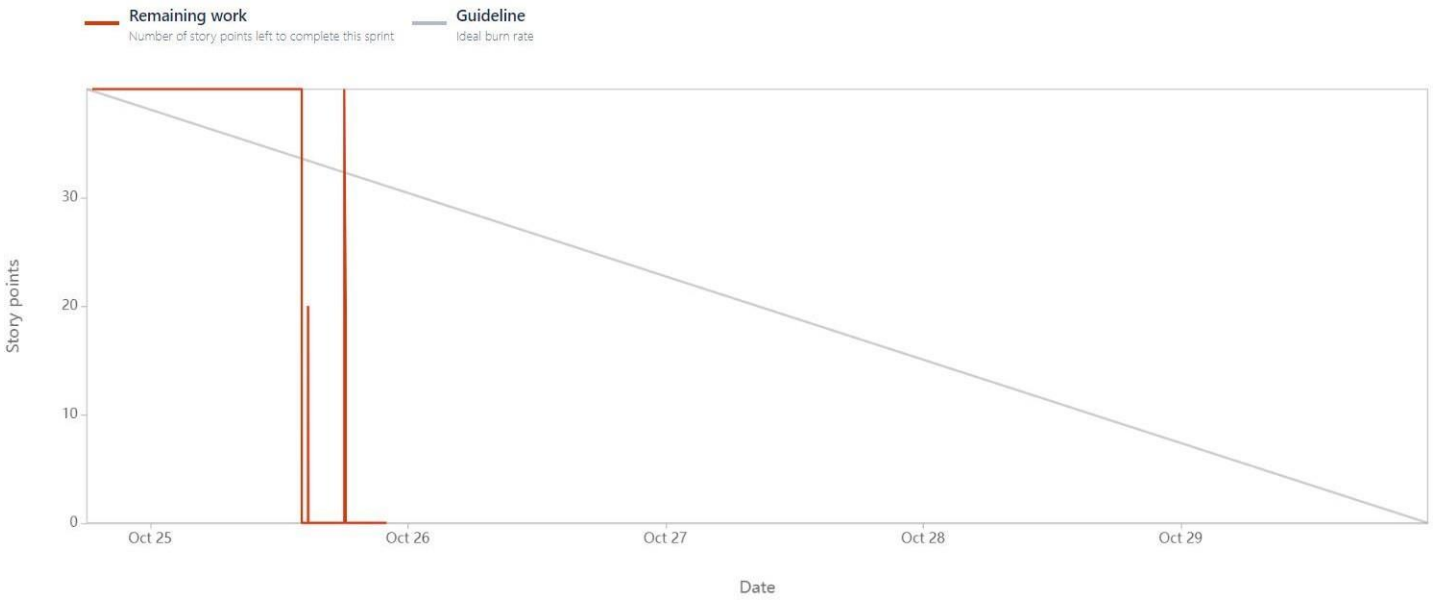
Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	20	6 Days	24 Oct 2022	29 Oct 2022	20	29 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022	20	05 Nov 2022
Sprint-3	20	6 Days	07 Nov 2022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

6.3 Reports from JIRA

Velocity Report

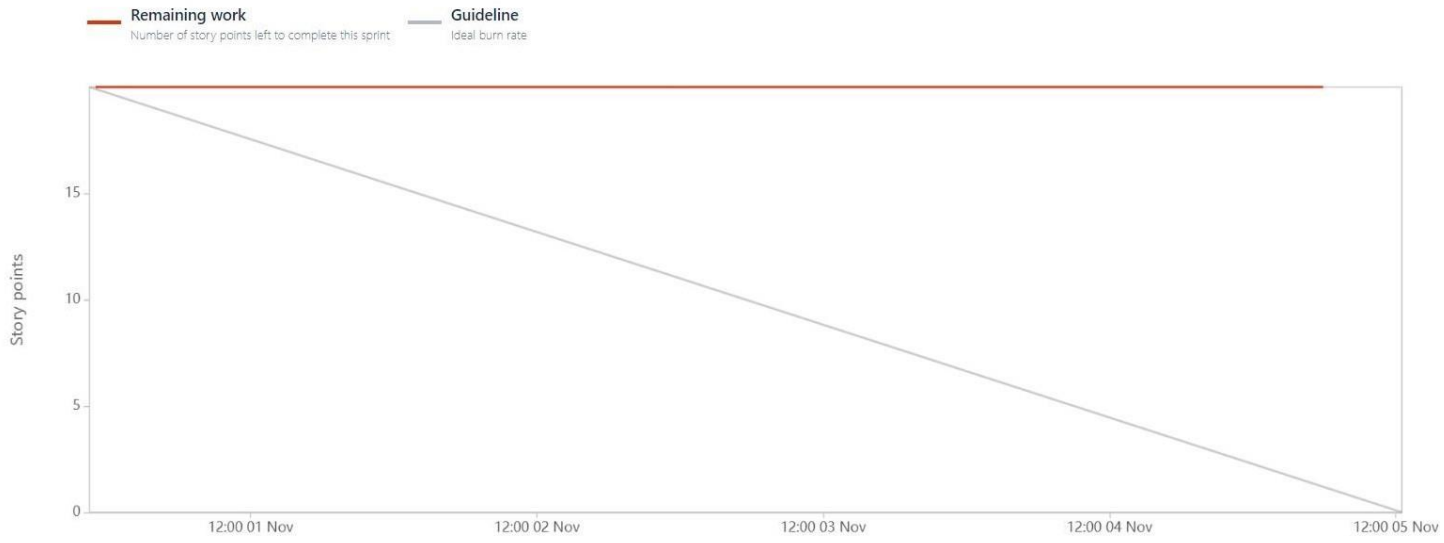
` **Sprint 1**

Date - October 24th, 2022 - October 29th, 2022



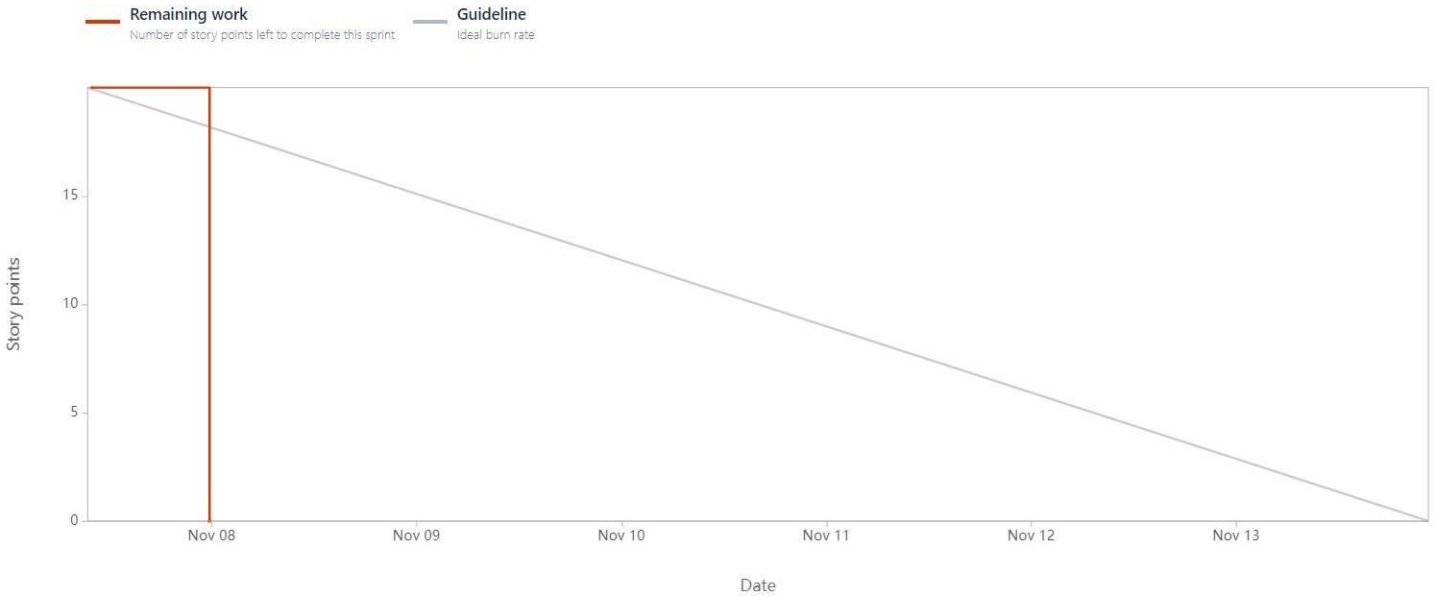
Sprint 2

Date - October 31st, 2022 - November 5th, 2022



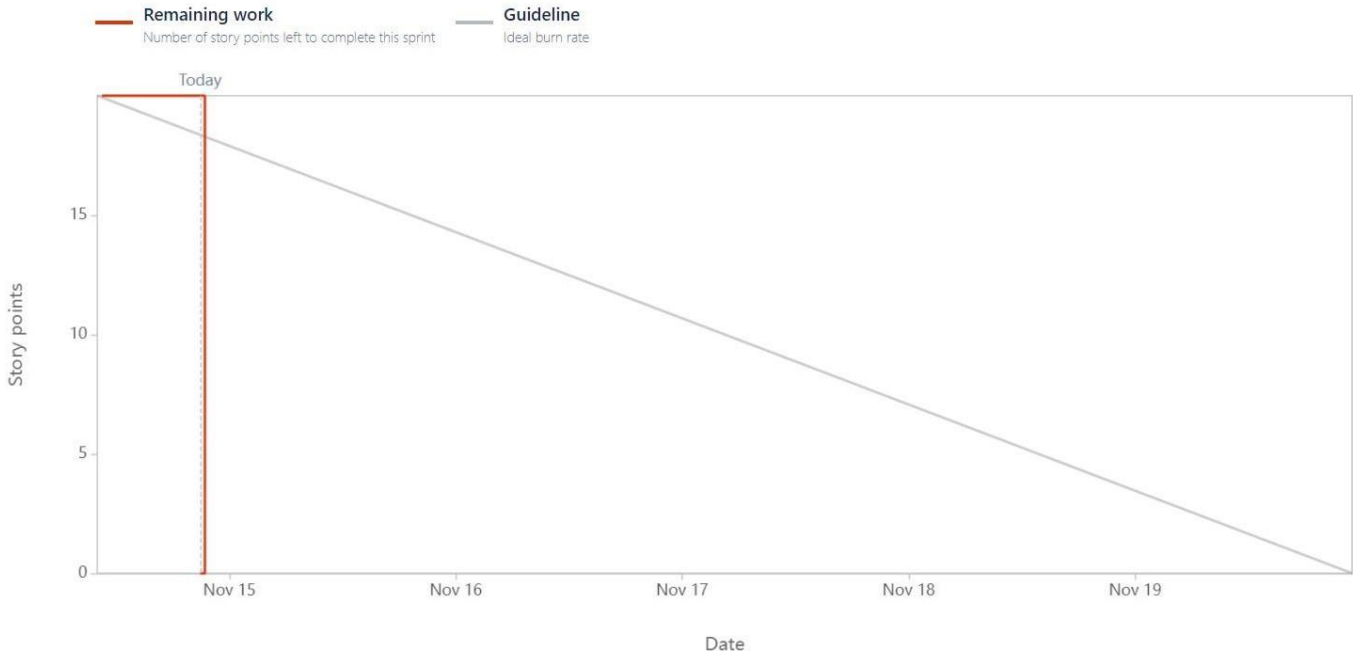
Sprint 3

Date - November 7th, 2022 - November 13th, 2022



Sprint 4

Date - November 14th, 2022 - November 19th, 2022



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

```
import numpy as np
import os
from PIL import Image
from flask import Flask, request, render_template, url_for
from werkzeug.utils import secure_filename, redirect
#from gevent.pywsgi import WSGIServer
from keras.models import load_model
from keras.preprocessing import image
from flask import send_from_directory

UPLOAD_FOLDER = 'D:/ibm/data'


app = Flask(_name_)
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER

model = load_model("./DigitRecog_IBM_model/mnistCNN.h5")


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


@app.route('/predict', methods=['GET', 'POST'])
def upload():
    if request.method == "POST":
        f = request.files["image"]
        filepath = secure_filename(f.filename)
        f.save(os.path.join(app.config['UPLOAD_FOLDER'], filepath))

        upload_img = os.path.join(UPLOAD_FOLDER, filepath)
        img = Image.open(upload_img).convert("L")  # convert image to monochrome
        img = img.resize((28, 28))  # resizing of input image

        im2arr = np.array(img)  # converting to image
        im2arr = im2arr.reshape(1, 28, 28, 1) # reshaping according to our requirement
        pred = model.predict(im2arr)

        num = np.argmax(pred, axis=1) # printing our Labels
        return render_template('predict.html', num=str(num[0]))
if_name_== '_main__':
    app.run(debug=True, threaded=False)
```

8. TESTING

8.1 Test Cases

Test case ID	Feature Type	Component	Test Scenario	Expected Result	Actual Result	Status
Homepage_TC_OO1	Functional	Home Page	Verify user is able to see the Homepage when clicked on the link	Home Page should be displayed.	Working as expected	Pass
Homepage_TC_OO2	UI	Home Page	Verify the UI elements in Homepage	Application should show below UI elements: a.choose file button b.predict button c.clear button	Working as expected	Pass
Homepage_TC_OO3	Functional	Home Page	Verify user is able to choose file from the local system and click on predict	Choose file popup screen must be displayed and user should be able to click on predict button	Working as expected	Pass
Homepage_TC_OO4	Functional	Home page	Verify user able to select invalid file format	Application won't allow to attach formats other than ".png, .jiff, .pjp, .jpeg, .jpg, .pjpeg"	Working as expected	Pass
Predict_TC_OO5	Functional	Predict page	Verify user is able to navigate to the predict to and view the predicted result	User must be navigated to the predict page and must view the predicted result	Working as expected	Pass

8.2 User Acceptance Testing

Defect Analysis

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
By Design	0	0	0	0	0
Duplicate	0	0	0	0	0
External	0	0	0	0	0
Fixed	0	0	0	0	0
Not Reproduced	0	0	0	0	0
Skipped	0	0	0	0	0
Won't Fix	0	0	0	0	0
Totals	0	0	0	0	0

Test Case Analysis

Section	Total Cases	Not Tested	Fail	Pass
Client Application	5	0	0	5
Security	5	0	0	5
Final Report Output	5	0	0	5
Performance	5	0	0	5

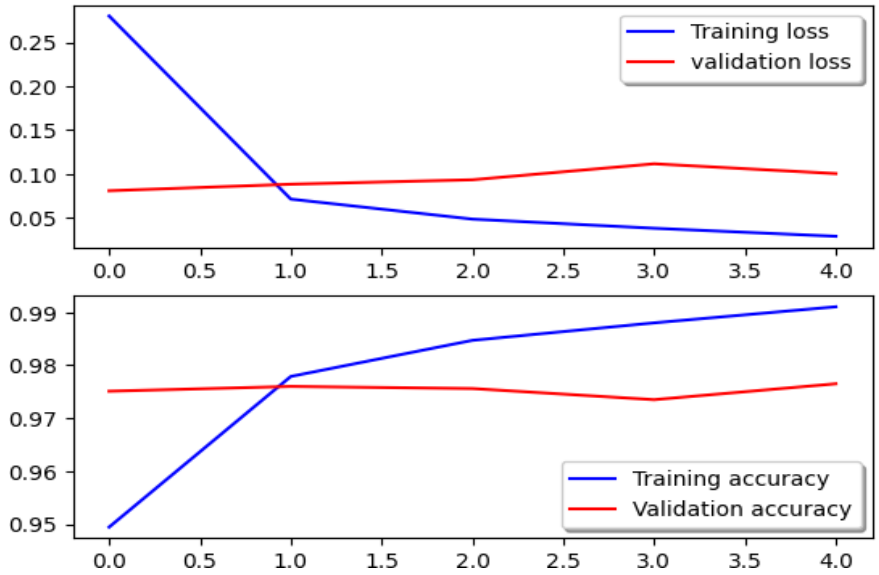
9. RESULTS

9.1 Performance Metrics

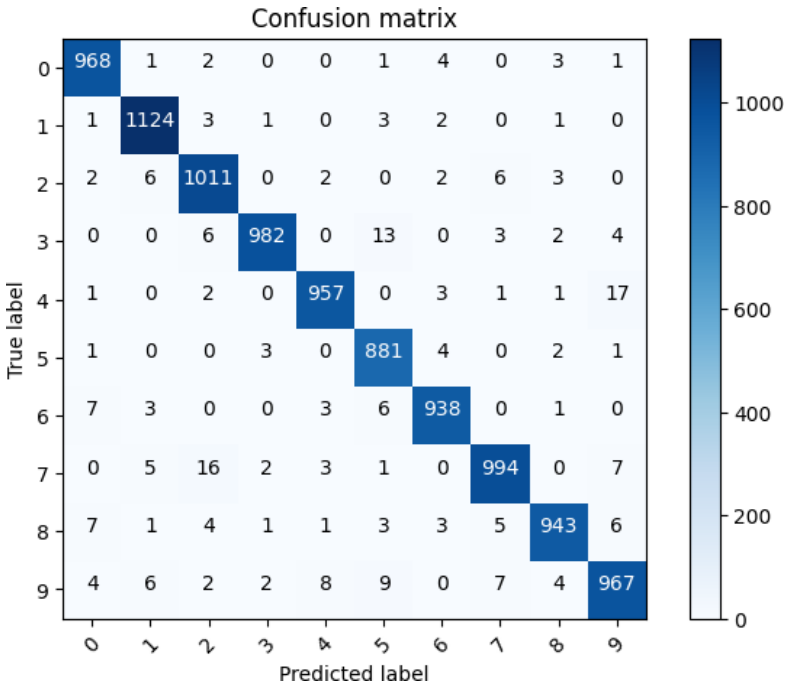
Model Summary:

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 26, 26, 64)        640

 conv2d_1 (Conv2D)           (None, 24, 24, 32)        18464

 flatten (Flatten)           (None, 18432)             0

 dense (Dense)               (None, 10)                184330

=================================================================
Total params: 203,434
Trainable params: 203,434
Non-trainable params: 0
_________________________________________________________________
None
```

Accuracy:



Confusion Matrix:



Classification Report:

	precision	recall	f1-score	support
0	0.98	0.99	0.98	980
1	0.98	0.99	0.99	1135
2	0.97	0.98	0.97	1032
3	0.99	0.97	0.98	1010
4	0.98	0.97	0.98	982
5	0.96	0.99	0.97	892
6	0.98	0.98	0.98	958
7	0.98	0.97	0.97	1028
8	0.98	0.97	0.98	974
9	0.96	0.96	0.96	1009
accuracy			0.98	10000
macro avg	0.98	0.98	0.98	10000
weighted avg	0.98	0.98	0.98	10000

Performance Metrics Result:

Locust Test Report

During: 11/15/2022, 10:52:19 PM - 11/15/2022, 10:56:36 PM

Target Host: http://127.0.0.1:5000/

Script: locustfile.py

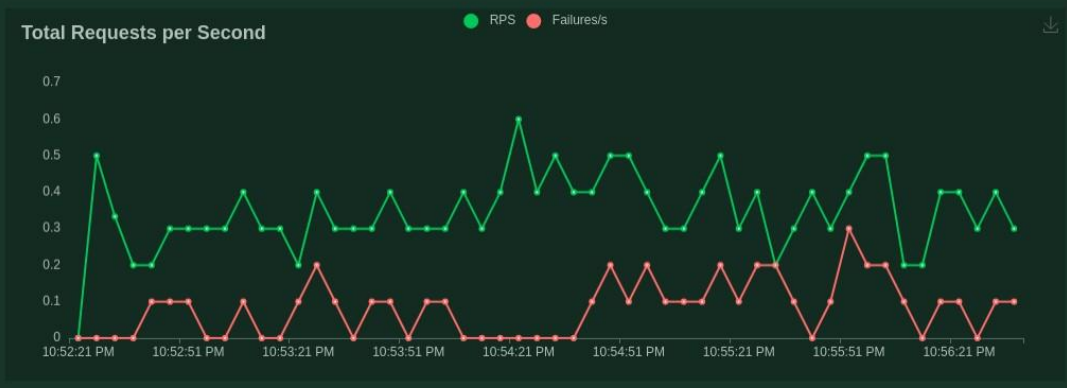
Request Statistics

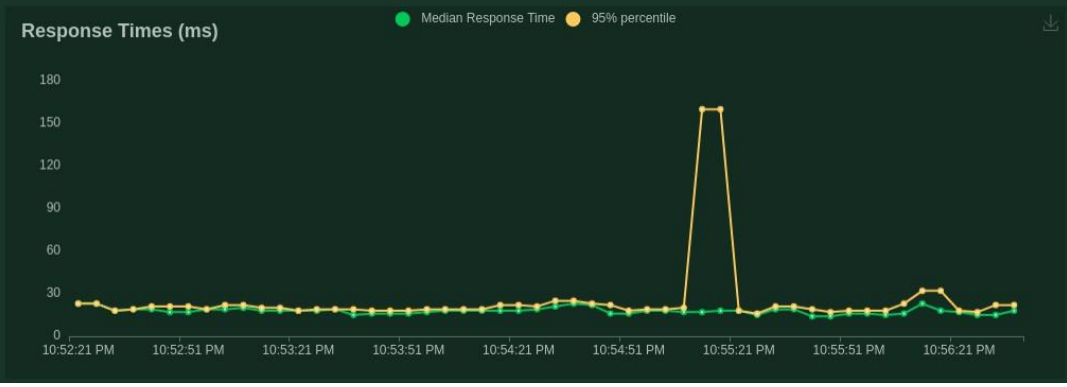
Method	Name	# Requests	# Fails	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	RPS	Failures/s
GET	//	67	0	17	12	24	5875	0.3	0.0
GET	//predict	23	23	21	11	163	265	0.1	0.1
	Aggregated	**90**	**23**	**18**	**11**	**163**	**4441**	**0.4**	**0.1**

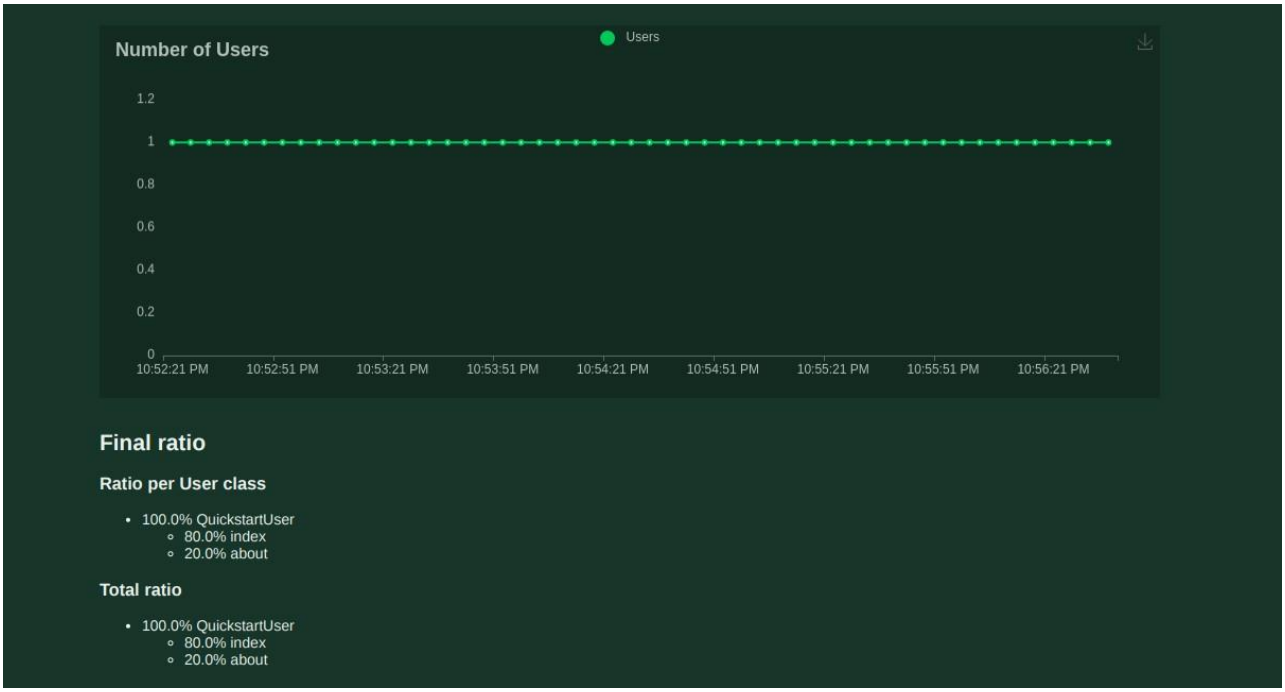
Response Time Statistics

Method	Name	50%ile (ms)	60%ile (ms)	70%ile (ms)	80%ile (ms)	90%ile (ms)	95%ile (ms)	99%ile (ms)	100%ile (ms)
GET	//	18	18	19	19	22	23	25	25
GET	//predict	15	15	16	16	17	32	160	160
	Aggregated	**17**	**18**	**18**	**19**	**22**	**23**	**160**	**160**

Charts







10. ADVANTAGES & DISADVANTAGES

Advantages

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

Disadvantages

- ✓ Cannot handle complex data.
- ✓ All the data must be in digital format.
- ✓ Requires high performance server for faster predictions**.**
- ✓ Prone to occasional errors.

11. CONCLUSION

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

12. FUTURE SCOPE

This project is far from complete and there is a lot of room for improvement. Some of the improvements that can be made to this project are as follows:

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

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

13. APPENDIX

Source Code

HTML AND CSS:

index.html:

```
<html>

<head>
  <title>HDR</title>

  <meta name="viewport" content="width=device-width">

  <link href="https://fonts.googleapis.com/css2?family=Prompt:wght@600&display=swap"
rel="stylesheet">
  <link href="https://fonts.googleapis.com/css2?family=Varela+Round&display=swap"
rel="stylesheet">
  <link href="https://fonts.googleapis.com/css2?family=Source+Code+Pro:wght@500&display=swap"
rel="stylesheet">
  <link
href="https://fonts.googleapis.com/css?family=Calistoga|Josefin+Sans:400,700|Pacifico&display
=swap" rel="stylesheet">

  <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css"
integrity="sha384-ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T"
crossorigin="anonymous">
  <link rel="stylesheet" type= "text/css" href= "{{
url_for('static',filename='css/style.css') }}">

  <script src="https://kit.fontawesome.com/b3aed9cb07.js" crossorigin="anonymous"></script>

  <script src="https://code.jquery.com/jquery-3.3.1.slim.min.js" integrity="sha384-
q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo"
crossorigin="anonymous"></script>
  <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.7/umd/popper.min.js"
integrity="sha384-UO2eT0CpHqdSJQ6hJty5KVphtPhzWj9WO1clHTMGa3JDZwrnQq4sF86dIHNDz0W1"
crossorigin="anonymous"></script>
  <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.min.js"
integrity="sha384-JjSmVgyd0p3pXB1rRibZUAYoIIy6OrQ6VrjIEaFf/nJGzIxFDsf4x0xIM+B07jRM"
crossorigin="anonymous"></script>
  <script src="https://cdn.jsdelivr.net/npm/@tensorflow/tfjs@latest"></script>
  <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@4.6.1/dist/css/bootstrap.min.css">
  <script src="https://cdn.jsdelivr.net/npm/jquery@3.6.0/dist/jquery.slim.min.js"></script>
```

```
  <script
src="https://cdn.jsdelivr.net/npm/popper.js@1.16.1/dist/umd/popper.min.js"></script>
  <script
src="https://cdn.jsdelivr.net/npm/bootstrap@4.6.1/dist/js/bootstrap.bundle.min.js"></script>

</head>
<style>
    body{
     background-image: url('static/images/bc1.jpg');
     background-repeat: no-repeat;
     background-size: cover;
    }
</style>

<script>
  function preview() {
    frame.src=URL.createObjectURL(event.target.files[0]);
}

    $(document).ready(function() {
          $('#clear_button').on('click', function() {
              $('#image').val('');
              $('#frame').attr('src',"");
            });
        });

</script>

<body>
    <h1>HandWritten Digit Recognition System</h1>
        <div class="container p-3 my-3 bg-dark text-white">
            <p>Handwritten Digit Recognition is a technology that is much needed in this
world as of Today.This Digit Recognition System is used to recognize the digits from
different sources like email, posts, cheque etc. Before proper implementation of this
technology we have relied on writing text with our own hands which can result in error.It's
difficult to store and access physical data with efficiency.The project presents in
representing the recognization of handwritten digits (0 - 9) from the famous MNIST dataset.
Here we will be using Convolutional Neural Network for the prediction.</p>
        </div>
        <section id="content">

            <div class="leftside">
            <form action="/predict" method="POST" enctype="multipart/form-data">
            <label>Select a image:</label>
            <input id="image" type="file" name="image" accept="image/png, image/jpeg"
onchange="preview()"><br><br>
              <img id="frame" width="100px" height="100px"/>
              <div class="buttons_div">
```

```
                <button type="submit" class="btn btn-light">Predict</button>
                <button type="button" class="btn btn-light">  Clear  </button>
              </div>
            </form>
            </div>
      </section>


</body>

</html>
```

Predict.html:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Prediction</title>
</head>

<style>
    body{
     background-image: url('static/images/bc1.jpg');
     background-repeat: no-repeat;
     background-size: cover;
    }

    #rectangle{
     width:600px;
     height:150px;
     background-color: #000000;
     border-radius: 25px;
     position:absolute;
     box-shadow: 0px 0px 10px 5px white;
     top:25%;
     left:50%;
     transform:translate(-50%,-50%);
    }

    #num{
  text-align: center;
  font-size: 30px;
  margin: 0 auto;
  padding: 3% 5%;
  padding-top: 8%;
```

```
  color: white;
    }

</style>

<body>

    <div id="rectangle">
        <h1 id="num">Predicted Number is {{num}}</h1>
    </div>

</body>
</html>
```

Style.css

```
  #clear_button{
    margin-left: 15px;
    font-weight: bold;
    color: rgb(0, 174, 255);
  }

  #confidence{
    font-family: 'Josefin Sans', sans-serif;
    margin-top: 7.5%;
  }

  #content{
    margin: 0 auto;
    padding: 2% 15%;
    padding-bottom: 0;
  }

  .welcome{
     text-align: center;
     position:  relative;
     color: rgb(0, 32, 112);
     background-color: skyblue;
     padding-top: 1%;
     padding-bottom: 1%;
     font-weight: bold;
     font-family: 'Bookman', 'URW Bookman L', serif;
  }

  #team_id{
     text-align: right;
```

```
    font-size: 25px;
    padding-right: 3%;
}

#predict_button{
  margin-right: 15px;
  color: rgb(0, 255, 72);
  font-weight: bold;
}

#prediction_heading{
  font-family: 'Josefin Sans', sans-serif;
  margin-top: 7.5%;
}

#result{
  font-size: 5rem;
}

#title{
  padding: 1.5% 15%;
  margin: 0 auto;
  text-align: center;
}

.btn {
    font-size: 15px;
    padding: 10px;
    /* -webkit-appearance: none; */
    background: #eee;
    border: 1px solid #888;
    margin-top: 20px;
    margin-bottom: 20px;
}

.buttons_div{
  margin-bottom: 30px;
  margin-right: 80px;
}

.heading{
  font-family:"American Typewriter", serif;
  font-weight: 700;
  font-size: 2rem;
  display: inline;
}

.leftside{
```

```
    text-align: center;
    margin: 0 auto;
    margin-top: 2%;
    /* padding-left: 10%; */
  }

  #frame{
    margin-right: 10%;
  }

  .predicted_answer{
    text-align: center;
    margin: 0 auto;
    padding: 3% 5%;
    padding-top: 0;
    /* padding-left: 10%; */
  }

  h1{
    text-align: center;
    color: aliceblue;
    padding: 100px 50px 65px 100px;
}

  @media (min-width: 720px) {
    .leftside{
      padding-left: 10%;
    }
  }
```

FLASK:

app.py:

```
import numpy as np
import os
from PIL import Image
from flask import Flask, request, render_template, url_for
from werkzeug.utils import secure_filename, redirect
#from gevent.pywsgi import WSGIServer
from keras.models import load_model
from keras.preprocessing import image
from flask import send_from_directory

UPLOAD_FOLDER = 'D:/ibm/data'


app = Flask(__name__)
```

```
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER

model = load_model("./DigitRecog_IBM_model/mnistCNN.h5")


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


@app.route('/predict', methods=['GET', 'POST'])
def upload():
    if request.method == "POST":
        f = request.files["image"]
        filepath = secure_filename(f.filename)
        f.save(os.path.join(app.config['UPLOAD_FOLDER'], filepath))

        upload_img = os.path.join(UPLOAD_FOLDER, filepath)
        img = Image.open(upload_img).convert("L")  # convert image to monochrome
        img = img.resize((28, 28)) # resizing of input image

        im2arr = np.array(img)  # converting to image
        im2arr = im2arr.reshape(1, 28, 28, 1) # reshaping according to our requirement

        pred = model.predict(im2arr)

        num = np.argmax(pred, axis=1) # printing our Labels

        return render_template('predict.html', num=str(num[0]))


if_name_== '_main__':
    app.run(debug=True, threaded=False)
```

MODEL CREATION:

```
import numpy as np
import tensorflow #open source used for both ML and DL for computation
from tensorflow.keras.datasets import mnist #mnist dataset
from tensorflow.keras.models import Sequential #it is a plain stack of layers
from tensorflow.keras import layers #A Layer consists of a tensor- in tensor-out computat ion
funct ion
from tensorflow.keras.layers import Dense, Flatten #Dense-Dense Layer is the regular deeply
connected r
#faltten -used fot flattening the input or change the dimension
from tensorflow.keras.layers import Conv2D #onvoLutiona l Layer
from keras.optimizers import Adam #opt imizer
from keras. utils import np_utils #used for one-hot encoding
```

```
import matplotlib.pyplot as plt   #used for data visualization
    (x_train, y_train), (x_test, y_test)=mnist.load_data ()
x_train=x_train.reshape (60000, 28, 28, 1).astype('float32')
x_test=x_test.reshape (10000, 28, 28, 1).astype ('float32')
number_of_classes = 10  #storing the no of classes in a variable
y_train = np_utils.to_categorical (y_train, number_of_classes) #converts the output in binary
format
y_test = np_utils.to_categorical (y_test, number_of_classes)
model.add(Conv2D(64, (3, 3), input_shape=(28, 28, 1), activation='relu'))
model.add(Conv2D(32, (3, 3), activation = 'relu'))
model.add(Flatten())
model.add(Dense(number_of_classes,activation = 'softmax'))
model.compile(loss= 'categorical_crossentropy', optimizer="Adam", metrics=['accuracy'])
x_train = np.asarray(x_train)
y_train = np.asarray(y_train)
history = model.fit(x_train, y_train, validation_data=(x_test, y_test), epochs=5,
batch_size=32)

from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
import itertools

fig, ax = plt.subplots(2,1)
ax[0].plot(history.history['loss'], color='b', label="Training loss")
ax[0].plot(history.history['val_loss'], color='r', label="validation loss",axes =ax[0])
legend = ax[0].legend(loc='best', shadow=True)

ax[1].plot(history.history['accuracy'], color='b', label="Training accuracy")
ax[1].plot(history.history['val_accuracy'], color='r',label="Validation accuracy")
legend = ax[1].legend(loc='best', shadow=True)

def plot_confusion_matrix(cm, classes,
                          normalize=False,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues):
    """
    This function prints and plots the confusion matrix.
    Normalization can be applied by setting `normalize=True`.
    """
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)

    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
```

```
    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, cm[i, j],
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")

    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')

# Predict the values from the validation dataset
Y_pred = model.predict(x_test)
# Convert predictions classes to one hot vectors
Y_pred_classes = np.argmax(Y_pred,axis = 1)
# Convert validation observations to one hot vectors
Y_true = np.argmax(y_test,axis = 1)
# compute the confusion matrix
confusion_mtx = confusion_matrix(Y_true, Y_pred_classes)
# plot the confusion matrix
plot_confusion_matrix(confusion_mtx, classes = range(10))

import sklearn
print(sklearn.metrics.classification_report(Y_true, Y_pred_classes))


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

plt.imshow(x_test[5100])

import numpy as np
print(np.argmax(prediction, axis=1))

np.argmax(y_test[5100:5101]) #printing the actual labels
# Save the model
model.save('models/mnistCNN.h5')
```

GitHub

https://github.com/IBM-EPBL/IBM-Project-8071-1658908751