PROJECT REPORT

A NOVEL METHOD FOR HANDWRITTEN DIGIT RECOGNITION SYSTEM

SUBMITTED BY

ASWIN.S	111919104013
SOWMIYA.S	111919104136
VIGNESH.R	111919104160
VIJAYAKUMAR.B	111919104163

TABLE OF CONTENTS

1.INTRODUCTION

1.1 Project Overview

Full Text Available Handwritten digit recognition plays a significant role in many user authentication applications in the modern world. As the handwritten digits are not of the same size, thickness, style and orientation. Therefore these challenges are to be faced to resolve this problem in my project. The objective of this project is to build a Graphical User Interface (GUI) in which we can draw the digit and recognize it straight away. I will be using a special type of deep neural network that is Convolutional Neural Network which is applied in analyzing visual imagery where large set of pixel data in images are converted to conserve useful data of images which can be fed as input layer data to Artificial Neural Network for training purpose. After that system will use hidden layers of CNN to develop a model for handwritten digit recognition. Here we will apply a LeNet-5 Convolution Neural Network algorithm on Modified National Institute of Standards and Technology (MNIST) dataset which includes handwritten digits total of 70,000 images. Keras, a Neural Network library written in python will be used. Stochastic gradient and backpropagation algorithm are used for training the network and the forward algorithm is used for testing. Once the model is ready, user can input their image which consist of digit on our GUI and they will get correct prediction of their input.

1.2 Purpose

This project aims to meet the following objectives:

i. To develop handwritten digit recognizing system that enables users to automate the process of digit recognition using this deep learning model.

ii. To test the accuracy of the model 10 .

iii. Efficient model which is less computation intensive.

2.LITERATURE SURVEY

2.1 Existing problem

Handwritten digit recognition finds its application in various fields such as post mail sorting system where scanned images of mail envelopes are made into queue and extract the section describing postcode to be delivered. With the help of digit recognizer, sorting of mails can be done based on these postcodes according to their region. Another application that utilizes this technique is form processing, digits are extracted from certain columns of a form and users put certain filters to get the desired results they want. But there is no interface for a user to get their images scanned and recognized which makes the task complicated to use for a normal user.

2.2 References

1. https://www.ijnrd.org/papers/IJNRD1704024.pdf -PRIYA, RAJENDRA SINGH.

2. https://www.irjet.net/archives/V9/i6/IRJET-V9I6208.pdf -Dhruv Sharma, Ishaan Singh.

3.https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.228.158&rep=rep1&type=pdf -MALOTHU NAGU, N VIJAY SHANKAR, K. ANNAPURNA .

4. http://ijcsit.com/docs/Volume%207/vol7issue1/ijcsit2016070101.pdf -Ayush Purohit, Shardul Singh Chauhan.

5. http://troindia.in/journal/ijcesr/vol6iss6part2/32-36.pdf -Rohini.M , Dr.D.Surendran

2.3 Problem Statement Definition

1. The Handwritten digits are not always of the same size, width, orientation and justified to margins as they differ from writing of person to person.

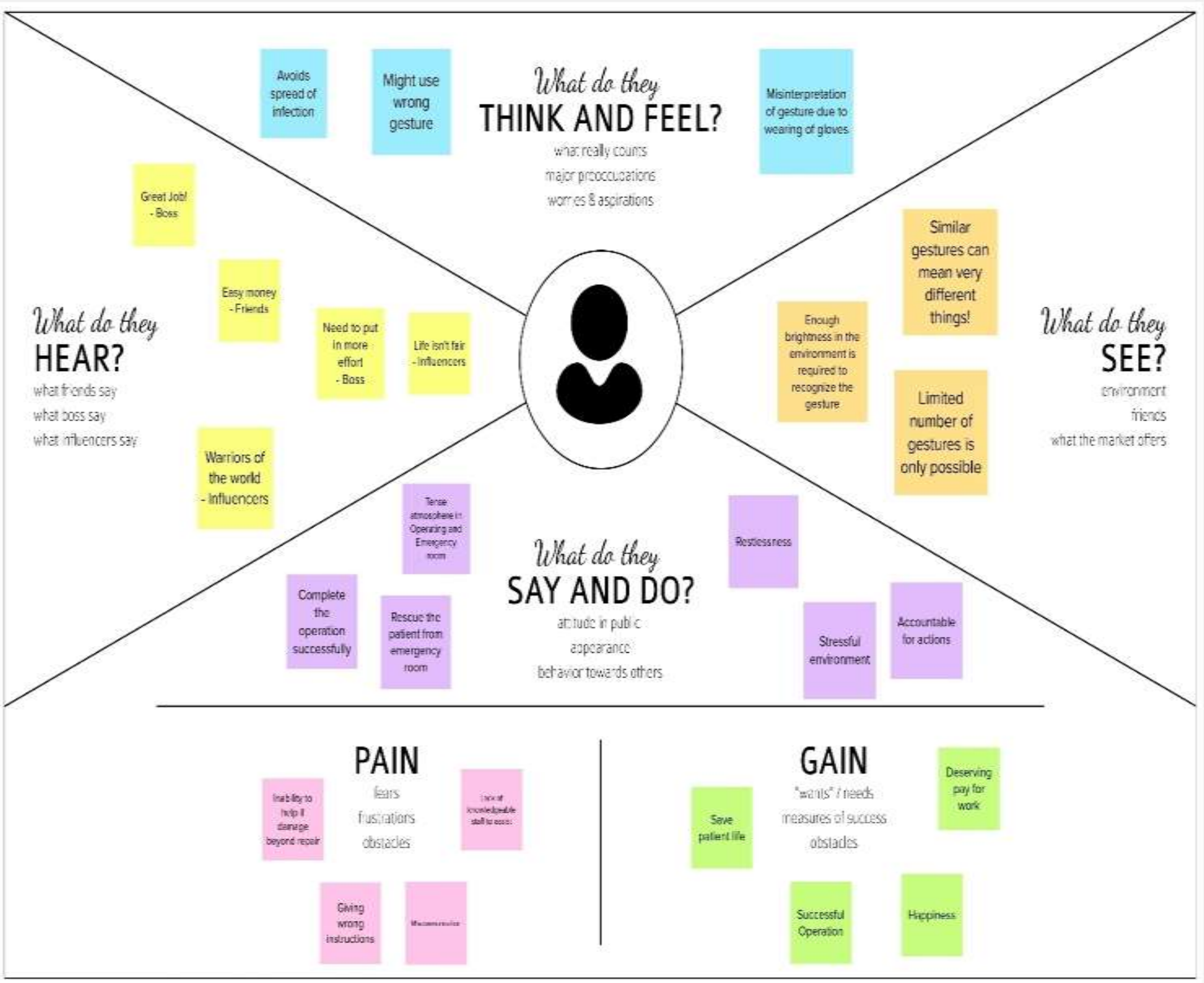
2. The similarity between digits such as 1 and 7, 5 and 6, 3 and 8, 2 and 7 etc. So, classifying between these numbers is also a major problem for computers.

3. The uniqueness and variety in the handwriting of different individuals also influence the formation and appearance of the digits, As the handwritten digits are not of the same size, thickness, style and orientation. Therefore these challenges are to be faced to resolve this problem.

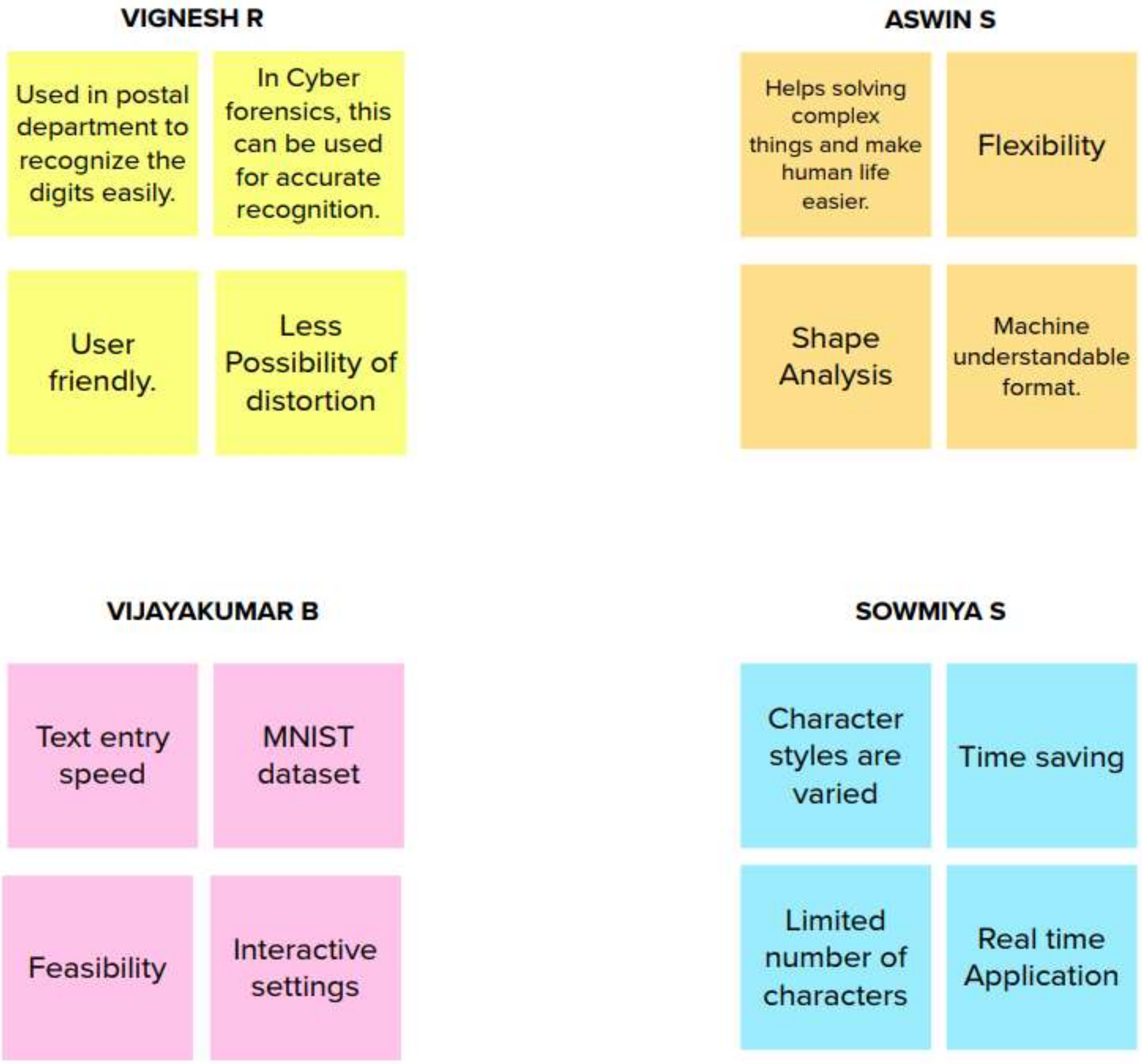
4. There are many types of handwriting ,it is hard to identify , in several application it is more time consuming because The shape of the digits are little bit different, The digits are not written properly.

3. IDEATION & PROPOSED SOLUTION

3.1 EMPATHY MAP CANVAS


What do they
THINK AND FEEL?
what really counts
major preoccupations
worries & aspirations
Avoids spread of infection
Might use wrong gesture
Misinterpretation of gesture due to wearing of gloves
What do they
HEAR?
what friends say
what boss say
what influencers say
Great Job! - Boss
Easy money - Friends
Need to put in more effort - Boss
Life isn't fair - Influencers
Warriors of the world - Influencers
What do they
SEE?
environment
friends
what the market offers
Similar gestures can mean very different things!
Enough brightness in the environment is required to recognize the gesture
Limited number of gestures is only possible
What do they
SAY AND DO?
attitude in public
appearance
behavior towards others
Tense atmosphere in Operating and Emergency room
Complete the operation successfully
Rescue the patient from emergency room
Restlessness
Stressful environment
Accountable for actions
PAIN
fears
frustrations
obstacles
Inability to help & damage beyond repair
Lack of knowledgeable staff to assist
Giving wrong instructions
GAIN
"wants" / needs
measures of success
obstacles
Save patient life
Deserving pay for work
Successful Operation
Happiness

3.2 IDEATION AND BRAINSTROMING



VIGNESH R

Used in postal department to recognize the digits easily.

In Cyber forensics, this can be used for accurate recognition.

User friendly.

Less Possibility of distortion

ASWIN S

Helps solving complex things and make human life easier.

Flexibility

Shape Analysis

Machine understandable format.

VIJAYAKUMAR B

Text entry speed

MNIST dataset

Feasibility

Interactive settings

SOWMIYA S

Character styles are varied

Time saving

Limited number of characters

Real time Application

3.3 PROPOSED SOLUTION

1. Problem Statement (Problem to be solved) - Computer programmes' ability to detect human-written numbers is known as handwritten digit recognition. Because handwritten figures are not always accurate and can take many various forms and sizes, it is a difficult work for the machine.

2. Novelty / Uniqueness - Recognize the digits precisely rather than all the characters like OCR.

3. Idea / Solution description - Using data from various sources, including images, documents, and touch defenses, a computer is able to celebrate the mortal handwritten numbers. It permits users to convert all of their handwritten notes and signatures into text documents in electronic form, using much less physical space than would be needed to store the physical copies of those documents.

4. Social Impact / Customer Satisfaction - The Handwritten Digit Recognizer software was made using artificial intelligence. It approximates the printed word digitally by identifying letters using sophisticated algorithms before producing a digital approximation.

5. Business Model (Revenue Model) - For efficient traffic control, this technology can be connected with traffic surveillance cameras to read license plates. Pin-code details can be easily identified and recognized by integrating with the postal system.

6. Scalability of the Solution - The capacity to recognize numbers in more distracting circumstances. The maximum number of digits that can be recognized is unlimited.

3.4 PROBLEM SOLUTION FIT

1.CUSTOMER SEGMENTS

- Fintech Industries.
- Supply Chain Management.
- Medical data Transcriptions.
- Scientific and Space Research.

2. CUSTOMER CONSTRAINTS

- Speed and Accuracy of the system.
- Size of the vocabulary.
- Spatial layout.
- Lack of feedback-based system.

3. AVAILABLE SOLUTIONS

- Free OCR API.
- Human centric data feed.

4. JOBS-TO-BE-DONE / PROBLEMS

- To design a system that recognizes a wide range of handwriting scripts.
- ML based approach to identify the character quickly and accurately.

- Adaptive learning module to learn from its own instances and get updated.

5. PROBLEM ROOT CAUSE

- In cases where distinct characters look very similar making it hard for a computer to recognize it accurately.
- Different styles of cursive handwriting is another challenge that requires a support system on vocabulary.

4. REQUIREMENT ANALYSIS

4.1 Functional Requirements

Functional Requirement and description:

FR-1	**Image Data**: Handwritten digit recognition is the ability of a computer to recognize the human handwritten digits from different sources like images, papers, touch screens, etc., and classify them into 10 predefined classes (0-9). this has been a topic of boundless-research in the field of deep learning.
FR-2	**Website:** Web hosting makes the files that comprise a website (code, images, etc.) available for viewing online. Every website you've ever visited is hosted on a server. The amount of space allocated on a server to a website depends on the type of hosting. the main types of hosting are shared, dedicated, VPS..
FR-3	**Digit_Classifieí_Model:** Use the MNISľ database of handwritten digits to train a convolutional network to predict the digit given an image. First obtain the training and validation data.

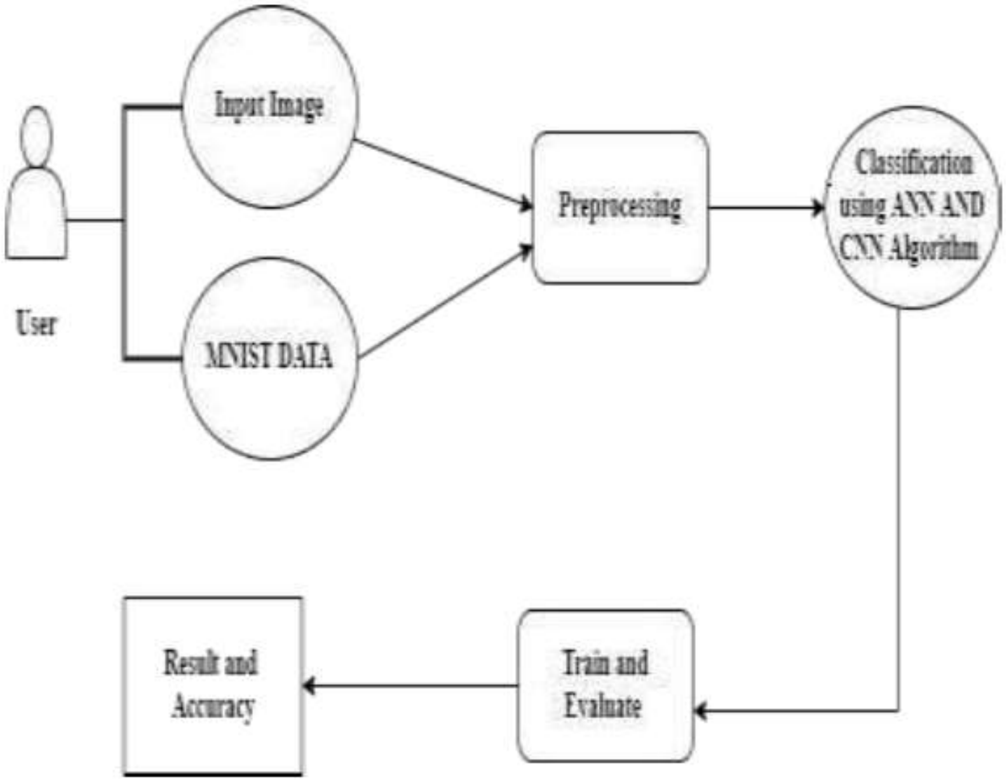
FR-4	**MNISl’ dataset:** the MNIST dataset is an acronym that stands for the Modified National Institute of Standards and Technology dataset. It is a dataset of 60,000 small square 28×28 pixel grayscale images of handwritten single digits between 0 and 9.
FR-5	**Cloud:** The cloud provides a number of IT services such as servers, databases, software, virtual storage, and networking, among others. In layman's terms, Cloud Computing is defined as a virtual platform that allows you to store and access your data over the internet without any limitations.

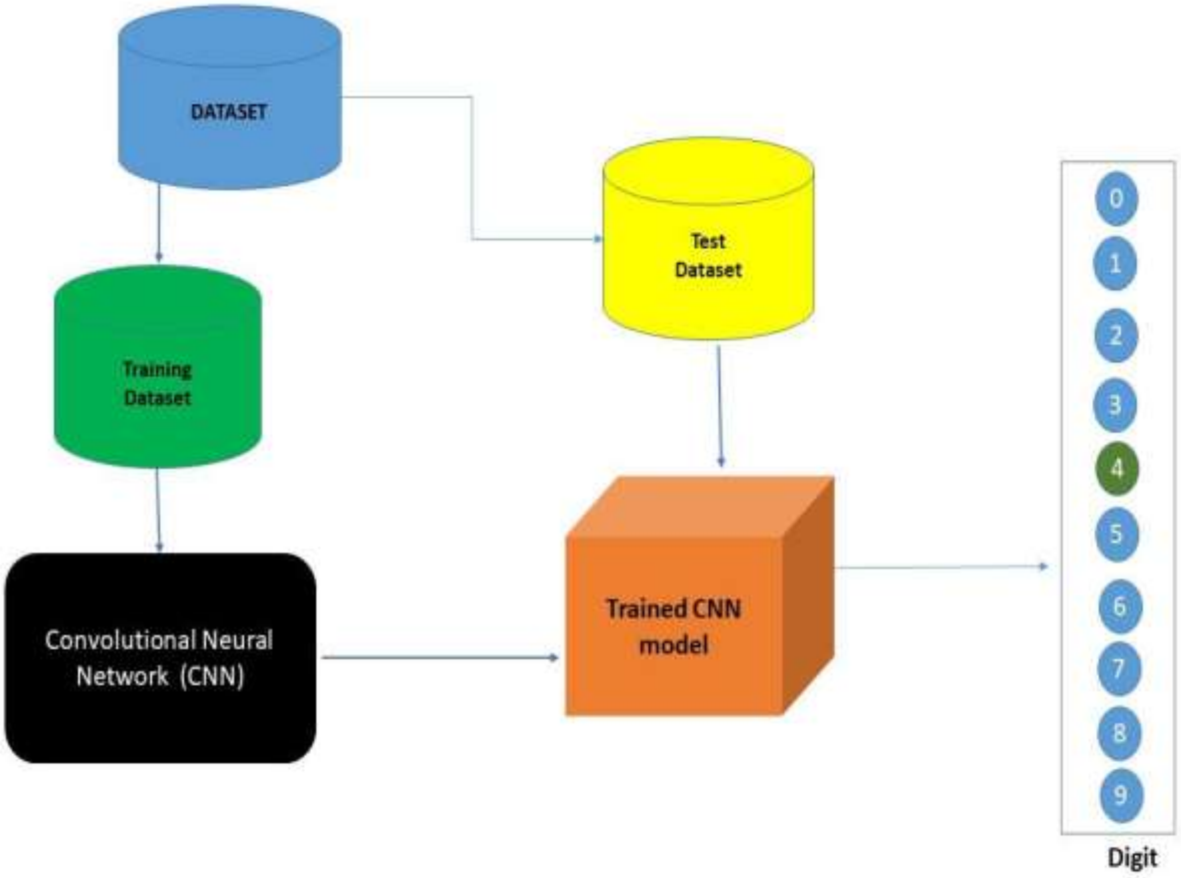
4.2 NON-FUNCTIONAL REQUIREMENTS

NÏR No.	Non-Functional Requirement
NFR-1	**Usability:** Handwritten character recognition is one of the practically important issues in pattern recognition applications. the applications of digit recognition include in postal mail sorting, bank check processing, form data entry, etc.
NFR-2	**Reliability:** 1) the system not only produces a classification of the digit but also a rich description of the instantiation parameters which can yield information such as the writing style. 2) the generative models can perform recognition driven segmentation. 3) the method involves a relatively.
NFR-3	**Performance:** the neural network uses the examples to automatically infer rules for recognizing handwritten digits. Furthermore, by increasing the number of training examples, the network can learn more about handwriting, and so improve its accuracy. there are a number of ways and algorithms to recognize handwritten digits, including Deep Learning/CNN, SVM, Gaussian Naive Bayes, KNN, Decision lees, Random Joists, etc.
NFR-4	**Accuracy:** Optical Character Recognition (OCR) technology provides higher than 99% accuracy with typed characters in high- quality images. However, the diversity in human writing types, spacing differences, and inequalities of handwriting causes less accurate character recognition.

5. PROJECT DESIGN

5.1




DATASET
Test Dataset
Training Dataset
Convolutional Neural Network (CNN)
Trained CNN model
0
1
2
3
4
5
6
7
8
9
Digit

S.No	Components	Description	Technology
1.	User Interface	How user interacts with application e.g., Mobile Application	HTML, CSS, JavaScript / Angular JS / Node Red.
2.	Application Logic-1	Logic for a process in the application	Java / Python
3.	Application Logic-2	Logic for a process in the application	IBM Watson STT service
4.	Application Logic-3	Logic for a process in the application	IBM Watson Assistant
5.	Database	Data Type, Configurations etc.	MySQL, NoSQL, etc.
6.	Cloud Database	Database Service on AI	IBM DB2.
7.	File Storage	File storage requirements	IBM Block Storage or Other Storage Service or Local Filesystem

8.	External API-1	Purpose of External API used in the application	IBM Weather API, etc.

5.3 USER STORIES

USER 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.

USER 2 - 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.

USER 3 - 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.

USER 4 - 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.

USER 5 - As it is a web application, it is installation free

I can use it without the installation of the application or any software.

USER 6 - 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.

6 . PROJECT PLANNING & SCHEDULING

6.1 SPRINT PLANNING & ESTIMATION

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority
Sprint-1	Data Collection	USN-1	As a user, I can collect the dataset from various resources with different handwritings.	10	Low
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
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
Sprint-2	Add CNN layers	USN-4	Creating the model and adding the input, hidden, and output layers to it.	5	High
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
Sprint-2	Train & test the model	USN-6	As a user, let us train our model with our image dataset.	6	Medium
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
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
Sprint-3		USN-9	As a user, I can know the details of the fundamental usage of the application.	5	Low
Sprint-3		USN-10	As a user, I can see the predicted / recognized digits in the application.	5	Medium
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
Sprint-4	Cloud Deployment	USN-12	As a user, I can access the web application and make the use of the product from	10	High

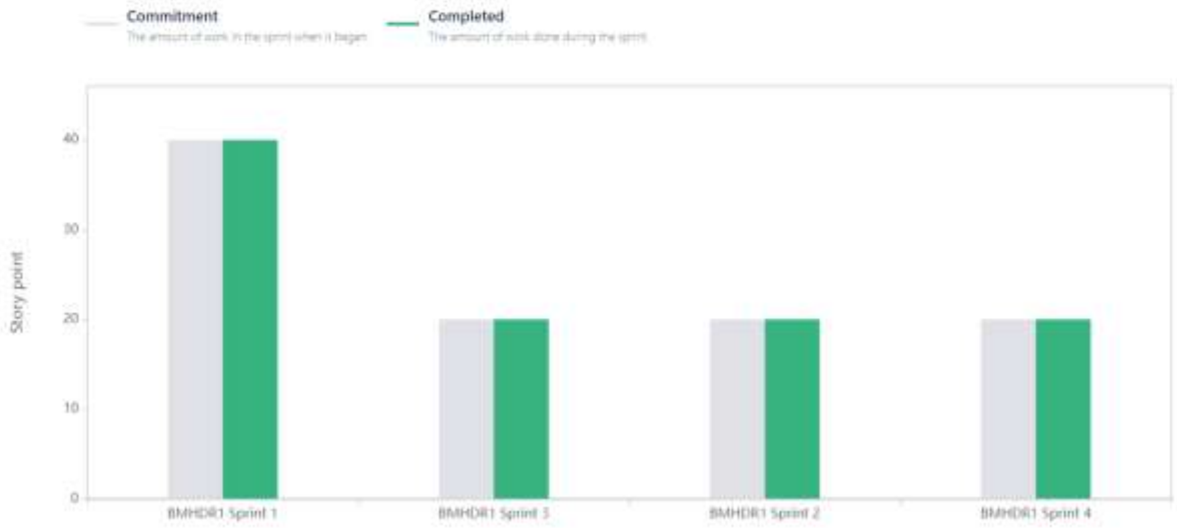
			anywhere.		

6.2 SPRINT DELIVERY SCHEDULE

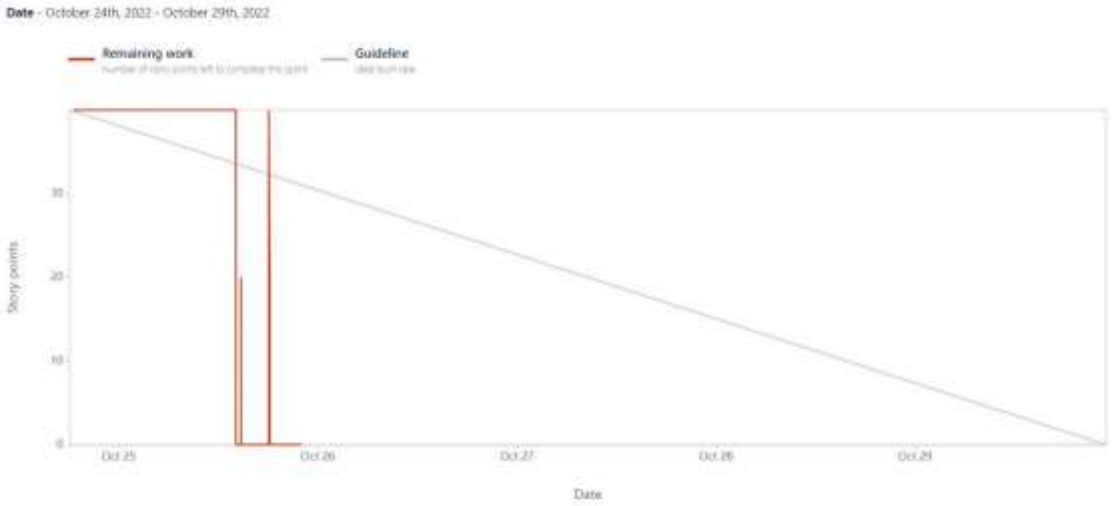
Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	20	6 Days	24 Oct 2022	29 Oct 2022	20	29 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022	20	05 Nov 2022
Sprint-3	20	6 Days	07 Nov 2022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

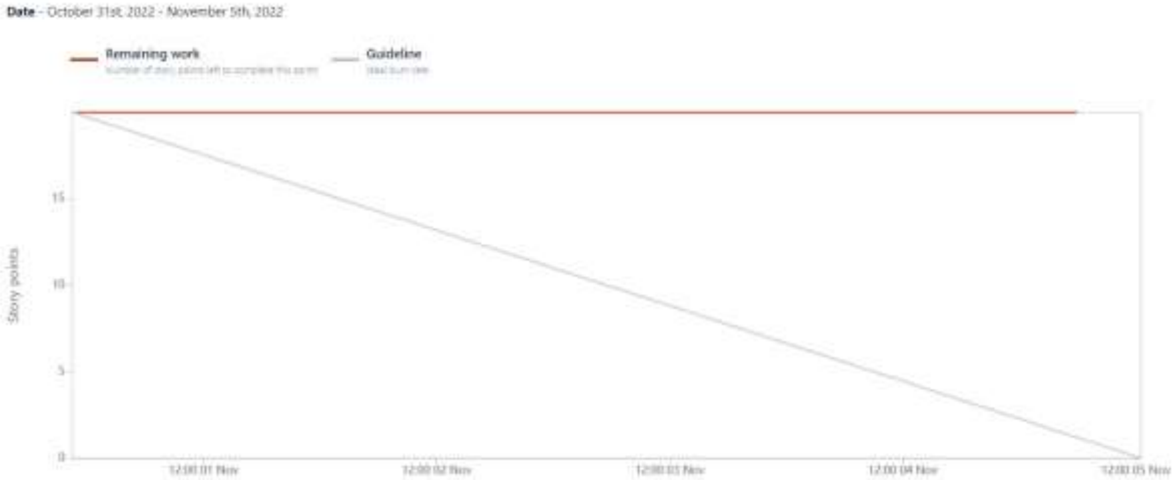
Velocity Report



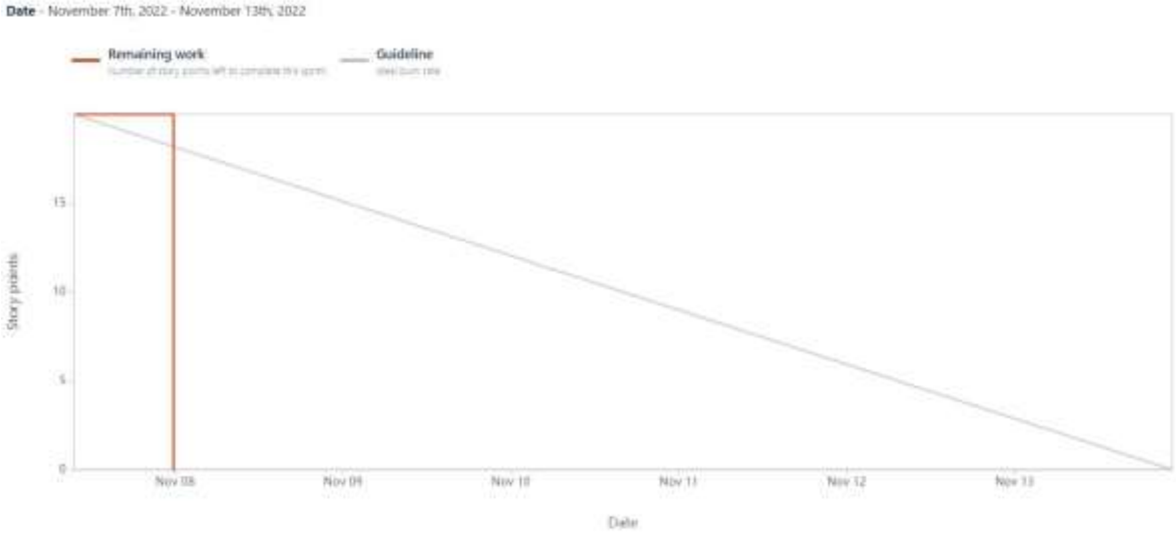
SPRINT 1



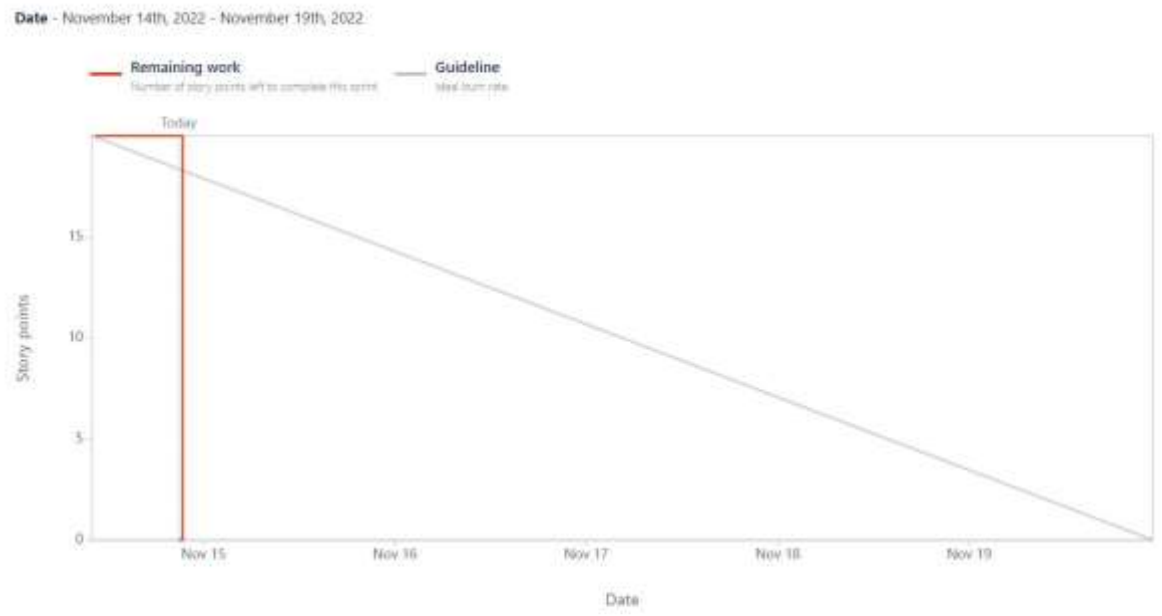
SPRINT 2



SPRINT 3



SPRINT 4


Date - November 14th, 2022 - November 19th, 2022
Remaining work
Number of story points left to complete this sprint
Guideline
Ideal burn rate
Today
Story points
15
10
5
0
Nov 15
Nov 16
Nov 17
Nov 18
Nov 19
Date

7. CODING & SOLUTIONING

```
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)
```

8 TESTING

8. TESTING 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,	Working as expected	Pass

				.pjpeg"		
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

8.1 Test Cases

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

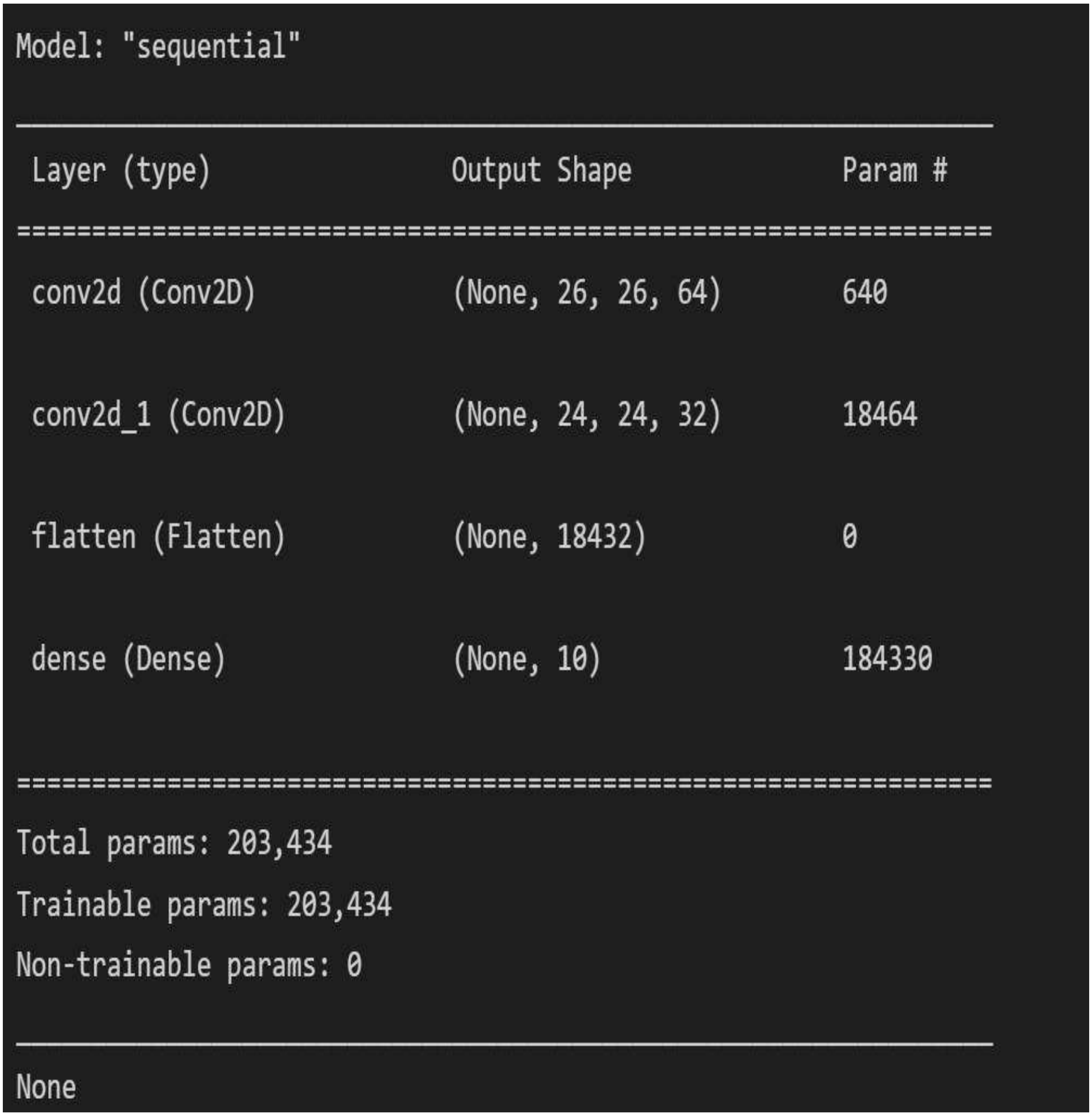
TEST CASES ANALYSIS

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

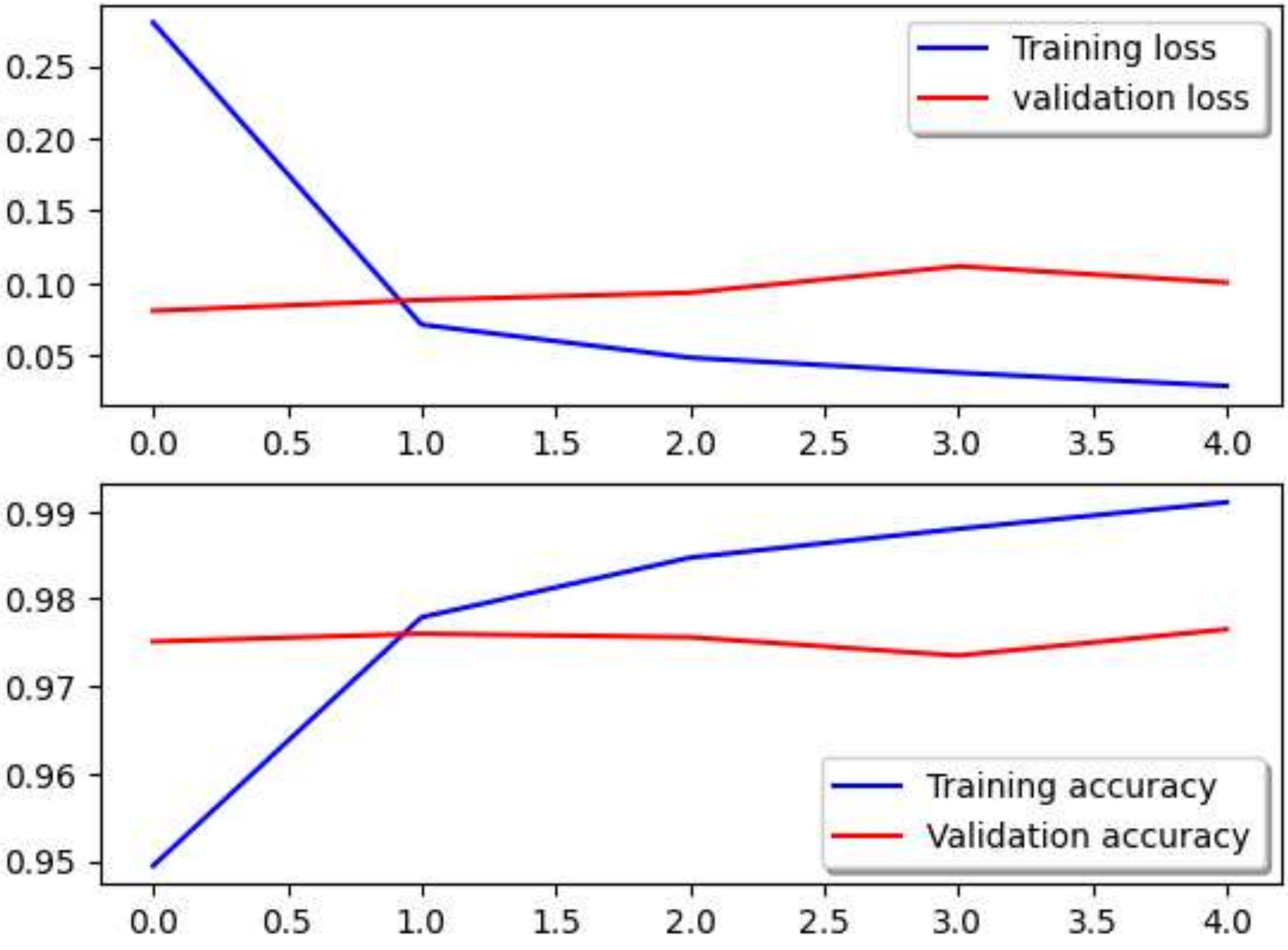
9.1 Performance Metrics

Model Summary:

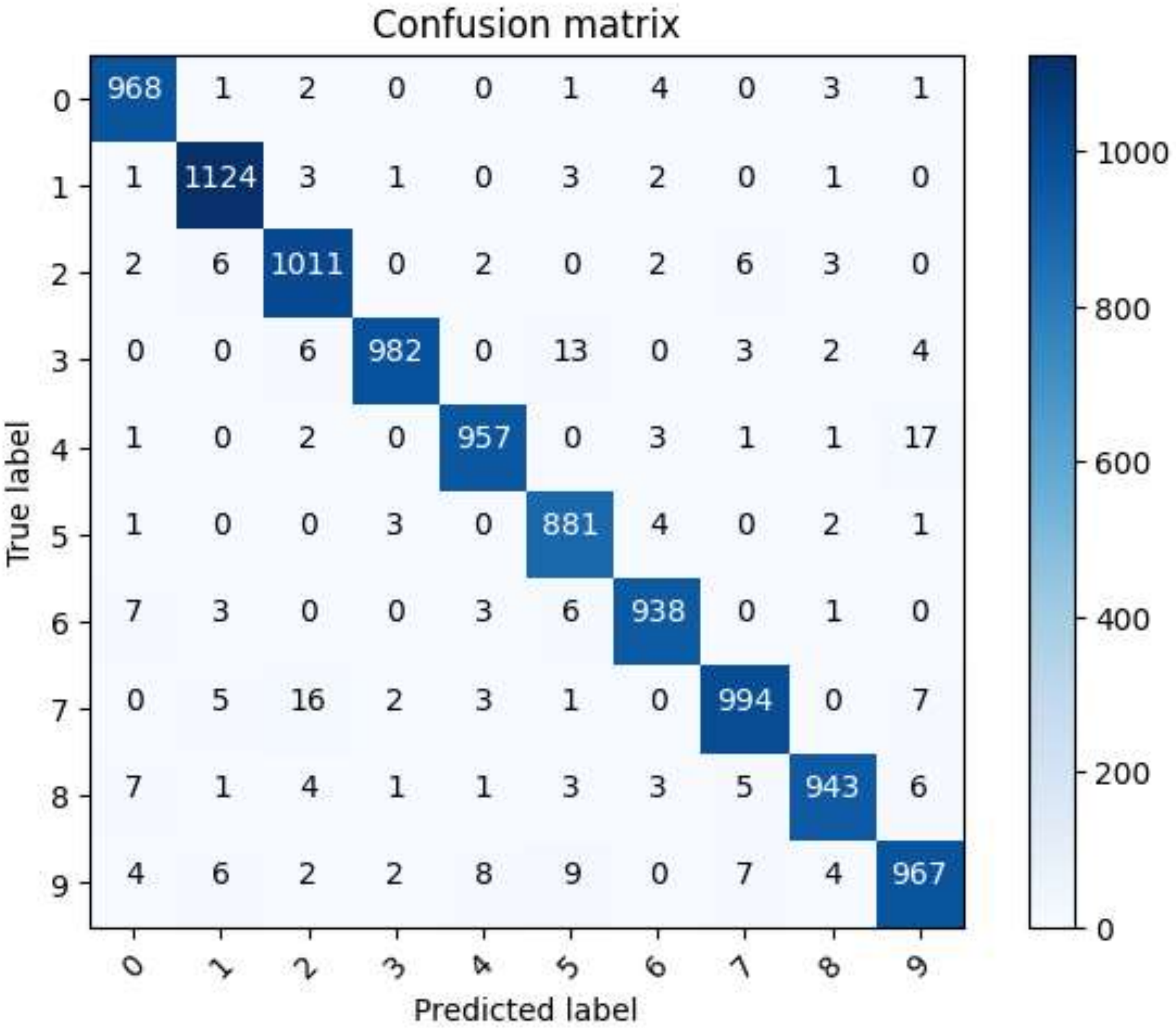


```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 26, 26, 64)        640

 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:

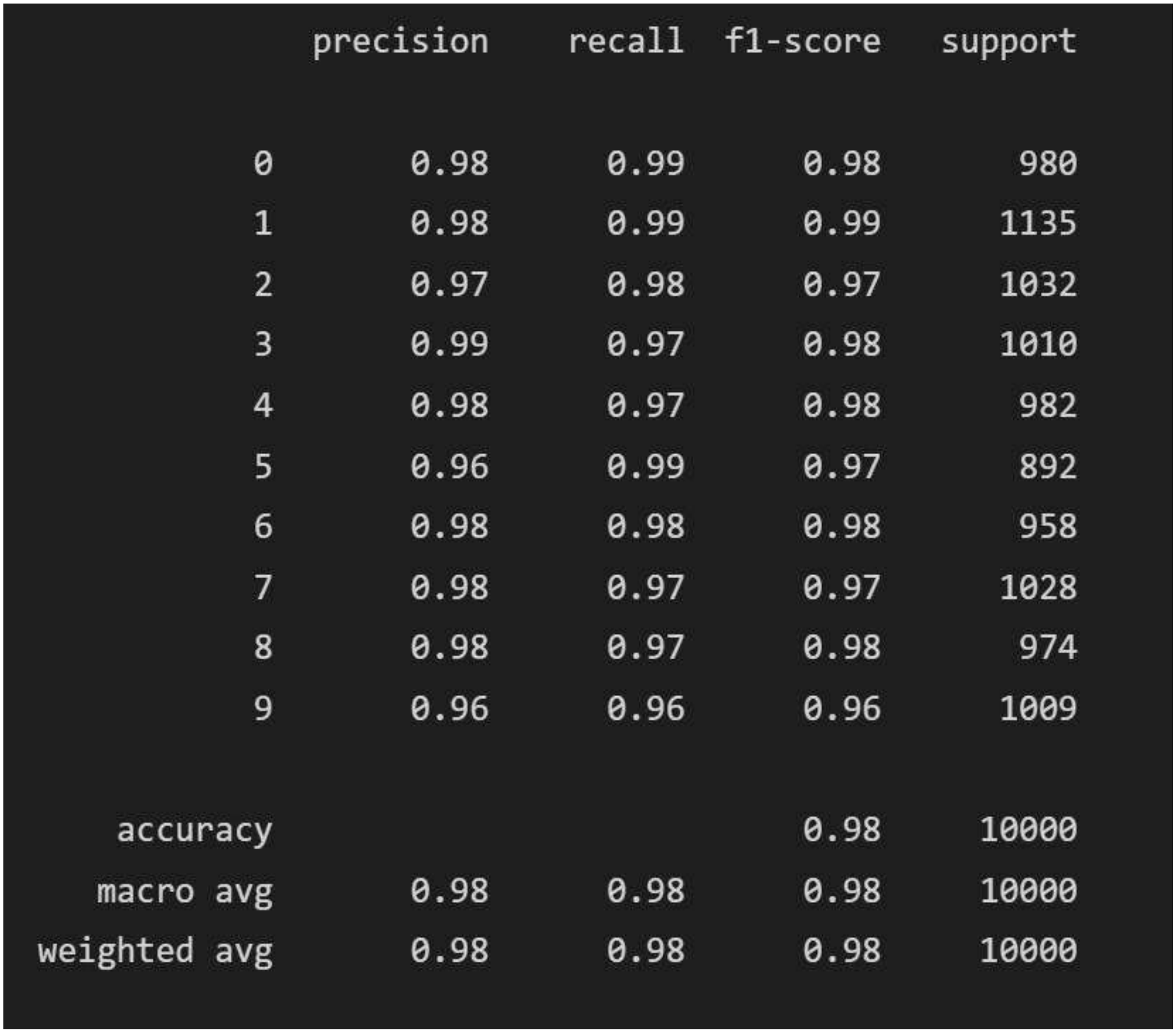

Training loss
validation loss
0.25
0.20
0.15
0.10
0.05
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
Training accuracy
Validation accuracy
0.99
0.98
0.97
0.96
0.95
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0

Confusion Matrix:



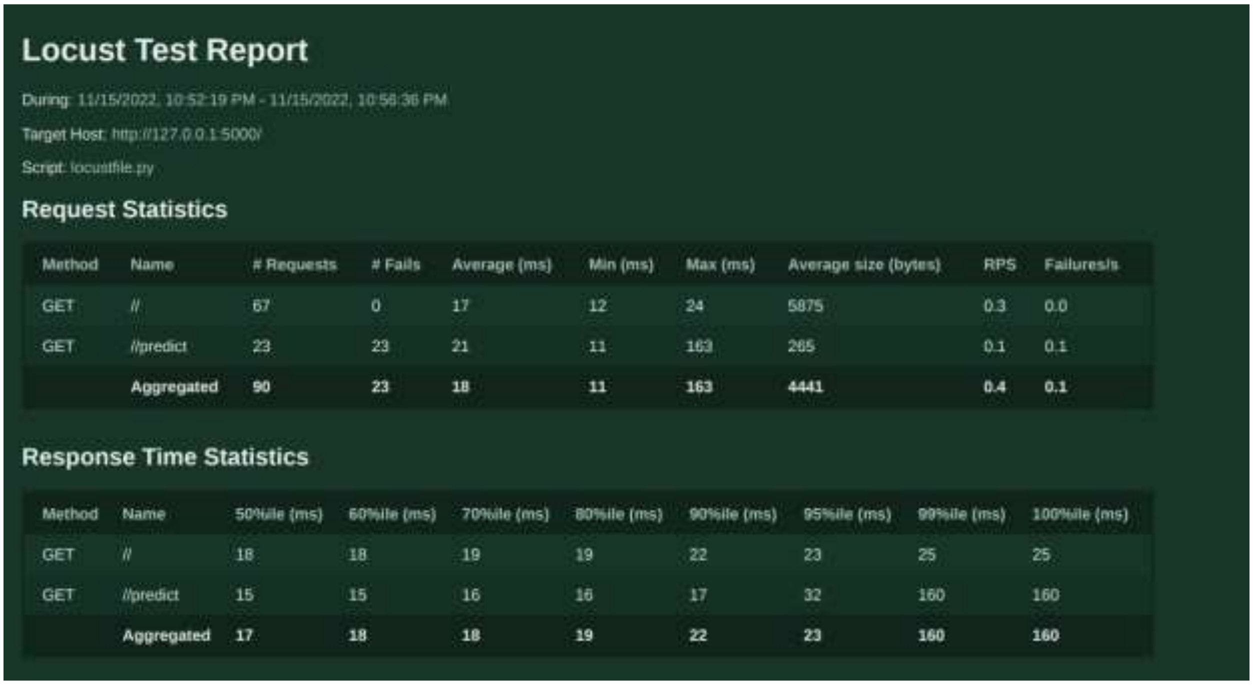
	0	1	2	3	4	5	6	7	8	9
0	968	1	2	0	0	1	4	0	3	1
1	1	1124	3	1	0	3	2	0	1	0
2	2	6	1011	0	2	0	2	6	3	0
3	0	0	6	982	0	13	0	3	2	4
4	1	0	2	0	957	0	3	1	1	17
5	1	0	0	3	0	881	4	0	2	1
6	7	3	0	0	3	6	938	0	1	0
7	0	5	16	2	3	1	0	994	0	7
8	7	1	4	1	1	3	3	5	943	6
9	4	6	2	2	8	9	0	7	4	967

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/

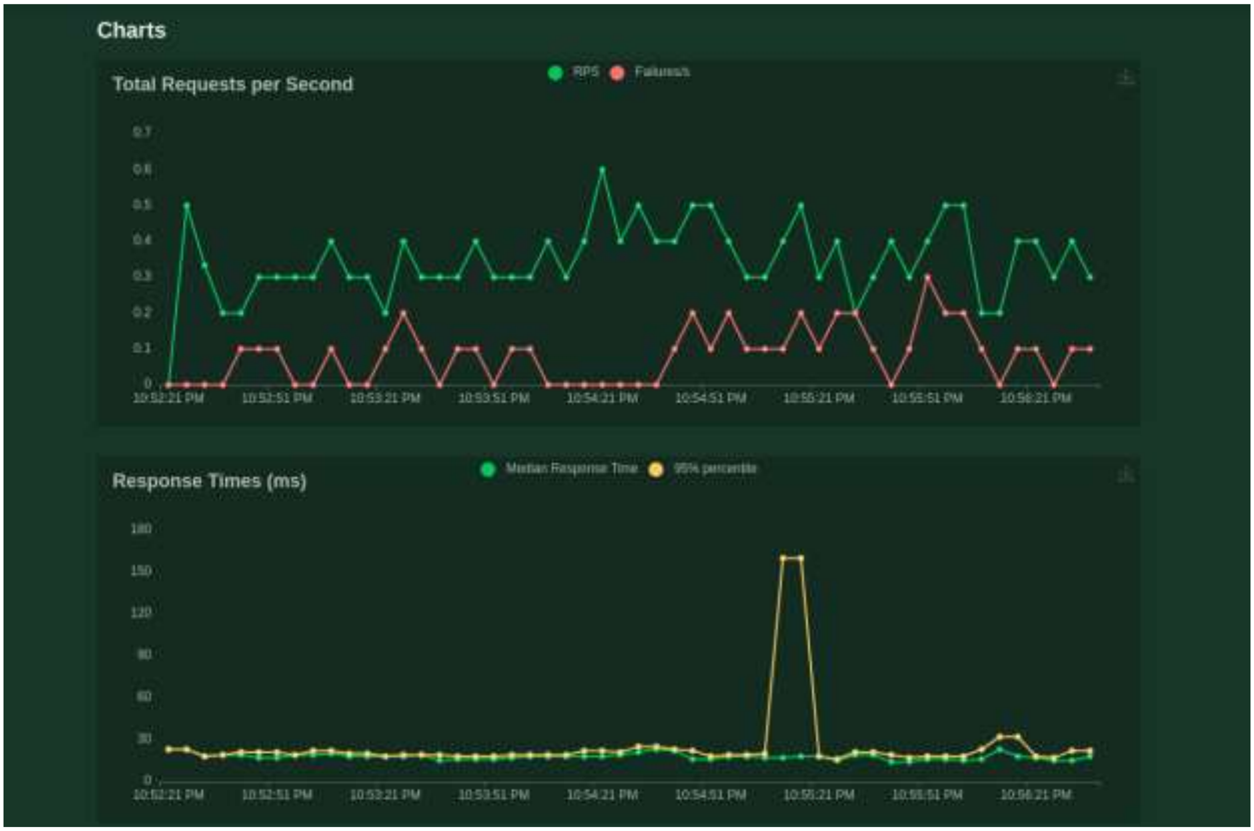
Script: locustfile.py

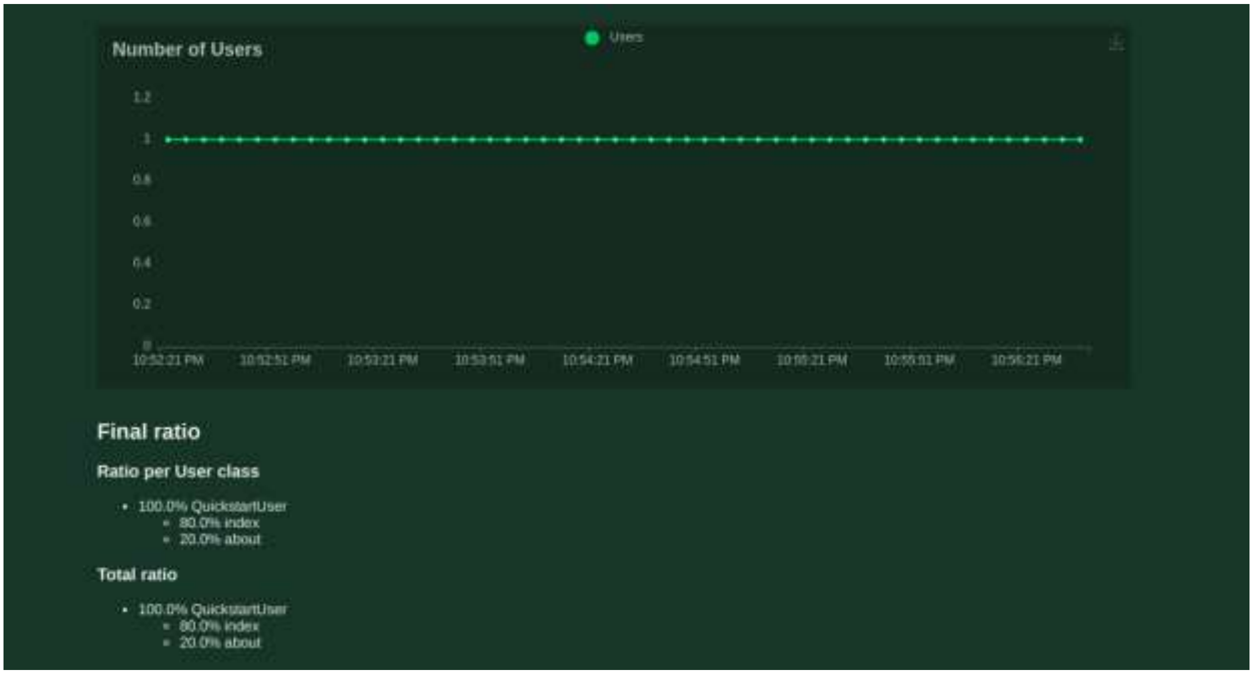
Request Statistics

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**





Final ratio

Ratio per User class

- 100.0% QuickstartUser
    - 80.0% index
    - 20.0% about

Total ratio

- 100.0% QuickstartUser
    - 80.0% index
    - 20.0% about

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 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. 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.

13. APPENDIX

Source Code

HTML AND CSS:

index.html:

```
<html>
         <script type="text/javascript" src="{{url_for('static', filename='jquery.min.js')
         }}"></script>
         <link rel="stylesheet" type="text/css" href="{{url_for('static',
         filename='style.css') }}">
         <script type="text/javascript">
             var canvas, ctx, flag = false,
                 prevX = 0,
                 currX = 0,
                 prevY = 0,
                 currY = 0,
                 dot_flag = false;

             var x = "red",
                 y = 8;

             function init() {
                 canvas = document.getElementById('can');
                 document.getElementById("probs").style.display = "none";
                 document.getElementById("interpret").style.display = "none";
                 ctx = canvas.getContext("2d");
                 w = canvas.width;
                 h = canvas.height;

                 canvas.addEventListener("mousemove", function (e) {
                     findxy('move', e)
                 }, false);
                 canvas.addEventListener("mousedown", function (e) {
                     findxy('down', e)
                 }, false);
                 canvas.addEventListener("mouseup", function (e) {
                     findxy('up', e)
                 }, false);
```

```
    canvas.addEventListener("mouseout", function (e) {
        findxy('out', e)
    }, false);
}


function draw() {
    ctx.beginPath();
    ctx.moveTo(prevX, prevY);
    ctx.lineTo(currX, currY);
    ctx.strokeStyle = x;
    ctx.lineWidth = y;
    ctx.stroke();
    ctx.closePath();
}

function erase() {
    ctx.clearRect(0, 0, w, h);
    document.getElementById("canvasimg").style.display = "none";
    document.getElementById("prediction").style.display = "none";
    document.getElementById("probs").style.display = "none";
    document.getElementById("interpret").style.display = "none";
    b = document.getElementsByTagName("body")[0];
    b.querySelectorAll('a').forEach(n => n.remove());
}

function save() {
    document.getElementById("prediction").style.display = "block";
    document.getElementById("probs").style.display = "block";
    document.getElementById("interpret").style.display = "block";
    var final_image = canvas.toDataURL();
    var a = document.createElement('a');
    a.href = final_image;
    a.download = 'process.png';
    document.body.appendChild(a);
    // a.click();
    $.ajax({
        url: "{{ url_for('process') }}",
        type: 'POST',
        data: final_image,
        success: function (response) {
            endresult = JSON.parse(JSON.stringify(response))
            console.log(endresult)
```

```
                $('#prediction').html('Prediction is: <span id="text">' +
endresult.data + '</span>')
                $('#probs').prop('src', 'data:image/png;base64,' +
endresult.probencoded)
                $('#interpret').prop('src', 'data:image/png;base64,' +
endresult.interpretencoded)
            }
        });
    }

    function findxy(res, e) {
        if (res == 'down') {
            prevX = currX;
            prevY = currY;
            currX = e.clientX - canvas.offsetLeft;
            currY = e.clientY - canvas.offsetTop;

            flag = true;
            dot_flag = true;
            if (dot_flag) {
                ctx.beginPath();
                ctx.fillStyle = x;
                ctx.fillRect(currX, currY, 2, 2);
                ctx.closePath();
                dot_flag = false;
            }
        }
        if (res == 'up' || res == "out") {
            flag = false;
        }
        if (res == 'move') {
            if (flag) {
                prevX = currX;
                prevY = currY;
                currX = e.clientX - canvas.offsetLeft;
                currY = e.clientY - canvas.offsetTop;
                draw();
            }
        }
    }
</script>

<body bgcolor="sky blue" onload="init()">
```

```
    <center>
        <h1> Handwritten Digit Recognition using <span id="text">PyTorch
CNN</span></h1>
<h3>TEAM ID:PNT2022-TMID16448</h3>
    </center>
    <div id="side">
        <h4 id='text'> Draw a Digit in the center of the Box.. </h4>
        <canvas id="can" width="200px" height="200px"></canvas>
        <img id="canvasimg">
        <div style="margin-top: 10;">
            <button class="ripple" id="btn" onclick="save()"> predict </button>
            &nbsp;
            <button id="clr" onclick="erase()"> clear </button>
            <h3 id="prediction"></h3>
        </div>
    </div>
    <div>
        <img id="probs" src="" alt="" height="45%" width="35%">
        <img id="interpret" src="" alt="" height="45%" width="35%">
    </div>
</body>

</html>
```

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)
```

This file is created to help the user include any application configuration for the app. Using this, you can configure some of the attributes of the application. From Application Configuration documentation: Application configuration objects store metadata for an application.

Load.py:

```
import
time
         import torch
         import random
         import functools
         import numpy as np
         from typing import Any, Callable, TypeVar, cast


         def random_seed(seed_value: int) -> None:
             """
             Random Seeds Numpy, Random and Torch libraries
             Args:
                 seed_value (int): Number for seeding
             """
             np.random.seed(seed_value)  # cpu vars
             torch.manual_seed(seed_value)  # cpu vars
             random.seed(seed_value)  # Python
             if torch.cuda.is_available():
                 torch.cuda.manual_seed(seed_value)
                 torch.cuda.manual_seed_all(seed_value)  # gpu vars
                 torch.backends.cudnn.deterministic = True  # needed
                 torch.backends.cudnn.benchmark = False


         F = TypeVar('F', bound=Callable[..., Any])


         def timer(func: F) -> F:
             """ Print the runtime of the decorated function """
             @functools.wraps(func)
             def wrapper_timer(*args, **kwargs):
                 start_time = time.perf_counter()
```

```
        value = func(*args, **kwargs)
        _ = time.perf_counter() - start_time
        hours, _ = divmod(_, 3600)
        minutes, seconds = divmod(_, 60)

        print(f'Execution time of function {func.__name__!r}: {hours:.0f} hrs
{minutes:.0f} mins {seconds:.3f} secs')
        return value
    return cast(F, wrapper_timer)
```

Train.py:

```
import
torch
         from torch import nn, optim
         from torch.utils import data

         from utils import *
         import pandas as pd
         import numpy as np
         from os import makedirs
         from typing import Union
         import matplotlib.pyplot as plt
         from dataclasses import dataclass

         import warnings
         warnings.filterwarnings('ignore')


         class MnistModel(nn.Module):
             """
             Custom CNN Model for Mnist
             """

             def __init__(self, classes: int) -> None:
                 super(MnistModel, self).__init__()

                 self.classes = classes

                 # initialize the layers in the first (CONV => RELU) * 2 => POOL + DROP
                 # (N,1,28,28) -> (N,16,24,24)
                 self.conv1A = nn.Conv2d(
                     in_channels=1, out_channels=16, kernel_size=5, stride=1, padding=0)
                 # (N,16,24,24) -> (N,32,20,20)
                 self.conv1B = nn.Conv2d(
                     in_channels=16, out_channels=32, kernel_size=5, stride=1, padding=0)
                 # (N,32,20,20) -> (N,32,10,10)
                 self.pool1 = nn.MaxPool2d(kernel_size=2)
                 self.act = nn.ReLU()
                 self.do = nn.Dropout(0.25)

                 # initialize the layers in the second (CONV => RELU) * 2 => POOL + DROP
                 # (N,32,10,10) -> (N,64,8,8)
                 self.conv2A = nn.Conv2d(
                     in_channels=32, out_channels=64, kernel_size=3, stride=1, padding=0)
                 # (N,64,8,8) -> (N,128,6,6)
                 self.conv2B = nn.Conv2d(
                     in_channels=64, out_channels=128, kernel_size=3, stride=1, padding=0)
                 # (N,128,6,6) -> (N,128,3,3)
                 self.pool2 = nn.MaxPool2d(kernel_size=2)

                 # initialize the layers in our fully-connected layer set
                 # (N,128,3,3) -> (N,32)
                 self.dense3 = nn.Linear(128*3*3, 32)

                 # initialize the layers in the softmax classifier layer set
                 # (N, classes)
                 self.dense4 = nn.Linear(32, self.classes)

             def forward(self, x: torch.Tensor) -> torch.Tensor:

                 # build the first (CONV => RELU) * 2 => POOL layer set
                 x = self.conv1A(x)
                 x = self.act(x)
                 x = self.conv1B(x)
                 x = self.act(x)
                 x = self.pool1(x)
                 x = self.do(x)

                 # build the second (CONV => RELU) * 2 => POOL layer set
                 x = self.conv2A(x)
                 x = self.act(x)
                 x = self.conv2B(x)
                 x = self.act(x)
                 x = self.pool2(x)
                 x = self.do(x)

                 # build our FC layer set
                 x = x.view(x.size(0), -1)
                 x = self.dense3(x)
                 x = self.act(x)
                 x = self.do(x)

                 # build the softmax classifier
                 x = nn.functional.log_softmax(self.dense4(x), dim=1)

                 return x


         class MnistDataset(data.Dataset):
             """
             Custom Dataset for Mnist
             """

             def __init__(self, df: pd.DataFrame, target: np.array, test: bool = False) ->
         None:
                 self.df = df
                 self.test = test

                 # if test=True skip this step
                 if not self.test:
                     self.df_targets = target

             def __len__(self) -> int:
                 # return length of the dataset
                 return len(self.df)

             def __getitem__(self, idx: int) -> Union[tuple, torch.Tensor]:
                 # if indexes are in tensor, convert to list
                 if torch.is_tensor(idx):
                     idx = idx.tolist()

                 # if test=False return bunch of images, targets
                 if not self.test:
                     return torch.as_tensor(self.df[idx].astype(float)), self.df_targets[idx]
                 # if test=True return only images
                 else:
                     return torch.as_tensor(self.df[idx].astype(float))
```

```
def loss_fn(outputs: torch.Tensor, targets: torch.Tensor) -> torch.Tensor:
    """
    Loss Function
    Args:
        outputs (torch.Tensor): Predicted Labels
        targets (torch.Tensor): True Labels
    Returns:
        torch.Tensor: NLLLoss value
    """
    return nn.NLLLoss()(outputs, targets)


def train_loop_fn(data_loader, model, optimizer, device, scheduler=None):
    """
    Training Loop
    Args:
        data_loader: Train Data Loader
        model: NN Model
        optimizer: Optimizer
        device: Device (CPU/CUDA)
        scheduler: Scheduler. Defaults to None.
    """
    # set model to train
    model.train()
    # iterate over data loader
    train_loss = []
    for ids, targets in data_loader:
        # sending to device (cpu/gpu)
        ids = ids.to(device, dtype=torch.float)
        targets = targets.to(device, dtype=torch.long)

        # Clear gradients w.r.t. parameters
        optimizer.zero_grad()
        # Forward pass to get output/logits
        outputs = model(x=ids)
        # Calculate Loss: softmax --> negative log likelihood loss
        loss = loss_fn(outputs, targets)
        train_loss.append(loss)
        # Getting gradients w.r.t. parameters
        loss.backward()
        optimizer.step()
        if scheduler is not None:
            # Updating scheduler
            if type(scheduler).__name__ == 'ReduceLROnPlateau':
                scheduler.step(loss)
            else:
                scheduler.step()
    print(f"Loss on Train Data : {sum(train_loss)/len(train_loss)}")


def eval_loop_fn(data_loader, model, device):
    """
    Evaluation Loop
    Args:
        data_loader: Evaluation Data Loader
        model: NN Model
        device: Device (CPU/CUDA)
    Returns:
        List of Target Labels and True Labels
    """
    # full list of targets, outputs
    fin_targets = []
    fin_outputs = []
    # set model to eveluate
    model.eval()  # as model is set to eval, there will be no optimizer and scheduler
update

    # iterate over data loader
    for _, (ids, targets) in enumerate(data_loader):
        ids = ids.to(device, dtype=torch.float)
        targets = targets.to(device, dtype=torch.long)

        outputs = model(x=ids)
        loss = loss_fn(outputs, targets)
        loss.backward()

        # Get predictions from the maximum value
        _, outputs = torch.max(outputs.data, 1)

        # appending the values to final lists
        fin_targets.append(targets.cpu().detach().numpy())
        fin_outputs.append(outputs.cpu().detach().numpy())
    return np.vstack(fin_outputs), np.vstack(fin_targets)
```

```
def test_loop_fn(test, model, device):
    """
    Testing Loop
    Args:
        test: Test DataFrame
        model: NN Model
        device: Device (CPU/CUDA)
    Returns:
        List of Predicted Labels
    """
    model.eval()
    # convert test data to FloatTensor
    test = torch.as_tensor(test)
    test = test.to(device, dtype=torch.float)

    # Get predictions
    pred = model(test)
    # Get predictions from the maximum value
    _, predlabel = torch.max(pred.data, 1)
    # converting to list
    predlabel = predlabel.tolist()

    # Plotting the predicted results
    L = 5
    W = 5
    _, axes = plt.subplots(L, W, figsize=(12, 12))
    axes = axes.ravel()

    for i in np.arange(0, L * W):
        axes[i].imshow(test[i].cpu().detach().numpy().reshape(28, 28))
        axes[i].set_title("Prediction Class = {:0.1f}".format(predlabel[i]))
        axes[i].axis('off')

    plt.suptitle('Predictions on Test Data')
    plt.subplots_adjust(wspace=0.5)
    plt.show()

    return predlabel


@timer
```

```
def run(args):
    """
    Function where all the magic happens
    Args:
        args: Arguments for Training
    Returns:
        List of Predicted Labels
    """
    # reading train and test data
    print('Reading Data..')
    dfx = pd.read_csv(args.data_path+'train.csv')
    df_test = pd.read_csv(args.data_path+'test.csv')

    classes = dfx[args.target].nunique()

    print('Data Wrangling..')
    # spliting train data to train, validate
    split_idx = int(len(dfx) * 0.8)
    df_train = dfx[:split_idx].reset_index(drop=True)
    df_valid = dfx[split_idx:].reset_index(drop=True)

    # target labels
    train_targets = df_train[args.target].values
    valid_targets = df_valid[args.target].values

    # reshaping data to 28 x 28 images and normalize
    df_train = df_train.drop(args.target, axis=1).values.reshape(
        len(df_train), 1, 28, 28)/255
    df_valid = df_valid.drop(args.target, axis=1).values.reshape(
        len(df_valid), 1, 28, 28)/255
    df_test = df_test.values.reshape(len(df_test), 1, 28, 28)/255

    print('DataSet and DataLoader..')
    # Creating PyTorch Custom Datasets
    train_dataset = MnistDataset(df=df_train, target=train_targets)
    valid_dataset = MnistDataset(df=df_valid, target=valid_targets)

    # Creating PyTorch DataLoaders
    train_data_loader = data.DataLoader(
        dataset=train_dataset, batch_size=args.BATCH_SIZE, shuffle=True)
    valid_data_loader = data.DataLoader(
        dataset=valid_dataset, batch_size=args.BATCH_SIZE, shuffle=False)
    # device (cpu/gpu)
    device = torch.device('cuda:0' if torch.cuda.is_available() else 'cpu')

    # instatiate model and sending it to device
    model = MnistModel(classes=classes).to(device)
    # instantiate optimizer
    optimizer = optim.SGD(model.parameters(), lr=args.lr)
    # instantiate scheduler
    scheduler = optim.lr_scheduler.CyclicLR(
        optimizer, base_lr=args.lr, max_lr=0.1)

    print('Training..')
    best_accuracy = 0
    # loop through epochs
    for epoch in range(args.NUM_EPOCHS):
        print(f'Epoch [{epoch+1}/{args.NUM_EPOCHS}]')
        # train on train data
        train_loop_fn(train_data_loader, model, optimizer, device, scheduler)
        # evaluate on validation data
        o, t = eval_loop_fn(valid_data_loader, model, device)
        accuracy = (o == t).mean() * 100
        print(f'Accuracy on Valid Data : {accuracy} %')
        if accuracy > best_accuracy:
            torch.save(model.state_dict(), args.model_path)
            best_accuracy = accuracy

    # Predict on test data
    return test_loop_fn(df_test, model, device)


if __name__ == "__main__":
    # variables for training model
    @dataclass
    class Args:
        lr: float = 3e-5
        RANDOM_STATE: int = 42
        NUM_EPOCHS: int = 5
        BATCH_SIZE: int = 100
        target: str = 'label'
        data_path: str = 'data/'
        model_path: str = 'checkpoint/mnist.pt'

        def __post_init__(self):
```

```
        makedirs('checkpoint', exist_ok=True)

arg = Args()
random_seed(arg.RANDOM_STATE)
test_preds = run(args=arg)
```

GitHub & Project Demo Link

GitHub Link

https://github.com/IBM-EPBL/IBM-Project-27926-1660100873