A NOVEL METHOD FOR HANDWRITTEN DIGIT RECOGNITION

1.INTRODUCTION

1.1 PROJECT OVERVIEW

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. Handwritten text recognition is one of the significant areas of research and development with a streaming number of possibilities that could be attained. Handwriting recognition (HWR), also known as Handwritten Text Recognition (HTR), is the ability of a computer to receive and interpret intelligible handwritten input from sources such as paper documents, photographs, touch-screens and other devices.

1.2 PURPOSE

The aim of a handwriting digit recognition system is to convert handwritten digits into machine readable formats. The main objective of this work is to ensure effective and reliable approaches for recognition of handwritten digits and make banking operations easier and error free. Handwritten digit recognition system (HDR) is meant for receiving and interpreting handwritten input in the form of pictures or paper documents. Traditional systems of handwriting recognition have relied on handcrafted features and a large amount of prior knowledge. Training an Optical character recognition (OCR) system based on these prerequisites is a challenging task. Convolutional neural networks (CNNs) are very effective in perceiving the structure of handwritten digits in ways that help in automatic extraction of distinct features and make CNN the most suitable approach for solving handwriting recognition problems. It has many applications like number plate recognition, postal mail sorting, bank check processing, etc.

2. LITERATURE SURVEY

2.1. EXISTING PROBLEM

EXISTING PROBLEMS	DRAWBACKS
Ensemble neural networks that combined with ensemble decision tree	Less accuracy with accuracy rate of 84%
Using classifier methods in handwritten digit recognition • Baseline Linear Classifier • LeNet 1 • Le Net 4 • Large fully connected multi network	Much complex networks with high computation time with accuracy rates • 92.2% • 98.3% • 98.9% • 98.4%
Digit recognition using single layer neural Network with principal component analysis	• Consumes more training time • Accuracy rate - 98.39%
Recognition using Simple Neural network and back propagation	• Higher processor required • High cost • Time consuming • Accuracy rate - 99.1%

In addition to this systems, KNN and SVM predict all the classes of dataset correctly with 99.26% accuracy but the thing process goes little complicated with MLP when it was having trouble classifying number 9.

2.2 REFERENCES

1. L. Bottou, C. Cortes, “Comparison of Classifier methods a case study in handwritten digit recognition”, Pattern Recognition, 1994. Vol. 2 Conference B; Image Processing, Proceedings of the 12th IAPR International. Conference IEEE, 06 August 2002.

2. Hongjian Zhan, ShujingLyu, Yue Lu Shanghai (August 2018), “Handwritten Digit String Recognition using Convolutional Neural Network”, 24th International Conference on Pattern Recognition (ICPR), pp. 3729-3734.

3. V.C.Bharati, K.Veningston, and P.V.Venkateswara Rao, “Query -Based Word Spotting in Handwritten Documents Using HMM", Springer Nature Singapore pte Ltd.2020,Data Engineering and communication Technology, Advances in Intelligent systems and computing 10789.

2.3. PROBLEM STATEMENT DEFINITION

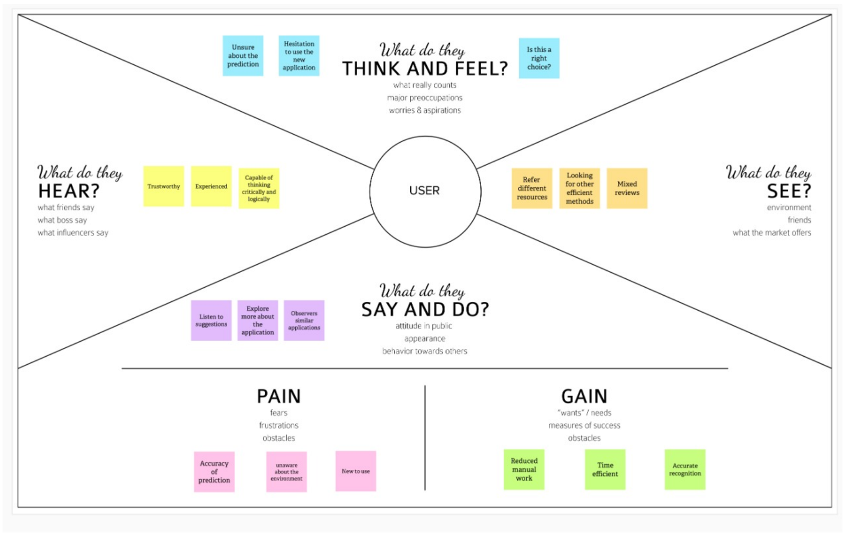
A solution to recognize the handwritten digit present in the paper or any other media to prevent the wastage of money spend on converting the handwritten information present in paper to digital form.To develop a handwritten digit recognition system using CNN which is capable of meeting the following constraints,

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.

3. IDEATION AND PROPOSED SOLUTION

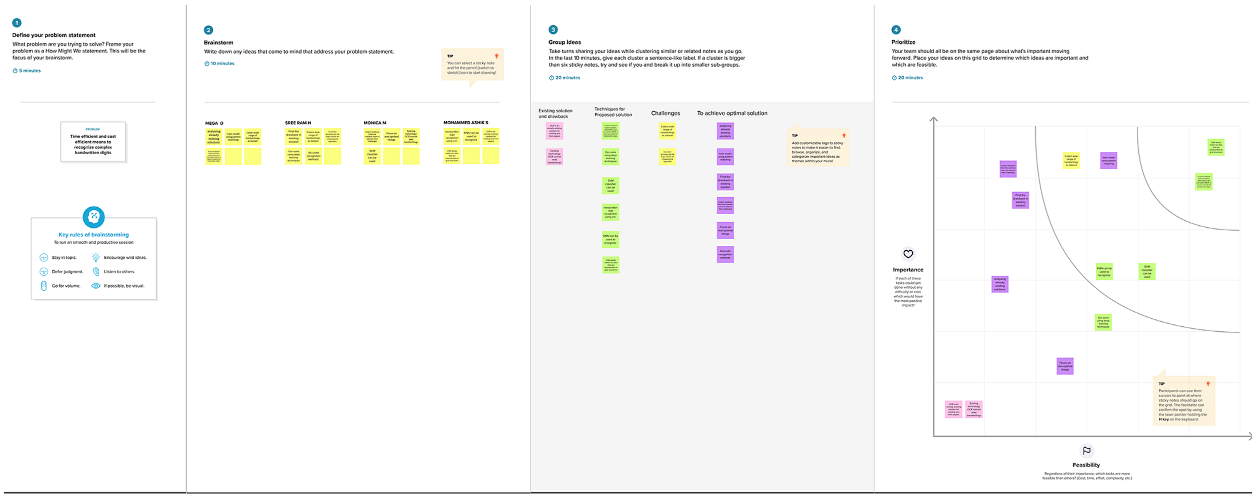
3.1. EMPATHY MAP CANVAS

An empathy map is a simple, easy-to-digest visual that captures knowledge about a user’s behaviors and attitudes. It is a useful tool to helps teams better understand their users.Creating an effective solution requires understanding the true problem and the person who is experiencing it. The exercise of creating the map helps participants consider things from the user’s perspective along with his or her goals and challenges.



3.2. IDEATION AND BRAINSTROMING

Brainstorming provides a free and open environment that encourages everyone within a team to participate in the creative thinking process that leads to problem solving. Prioritizing volume over value, out-of-the-box ideas are welcome and built upon, and all participants are encouraged to collaborate, helping each other develop a rich amount of creative solutions.

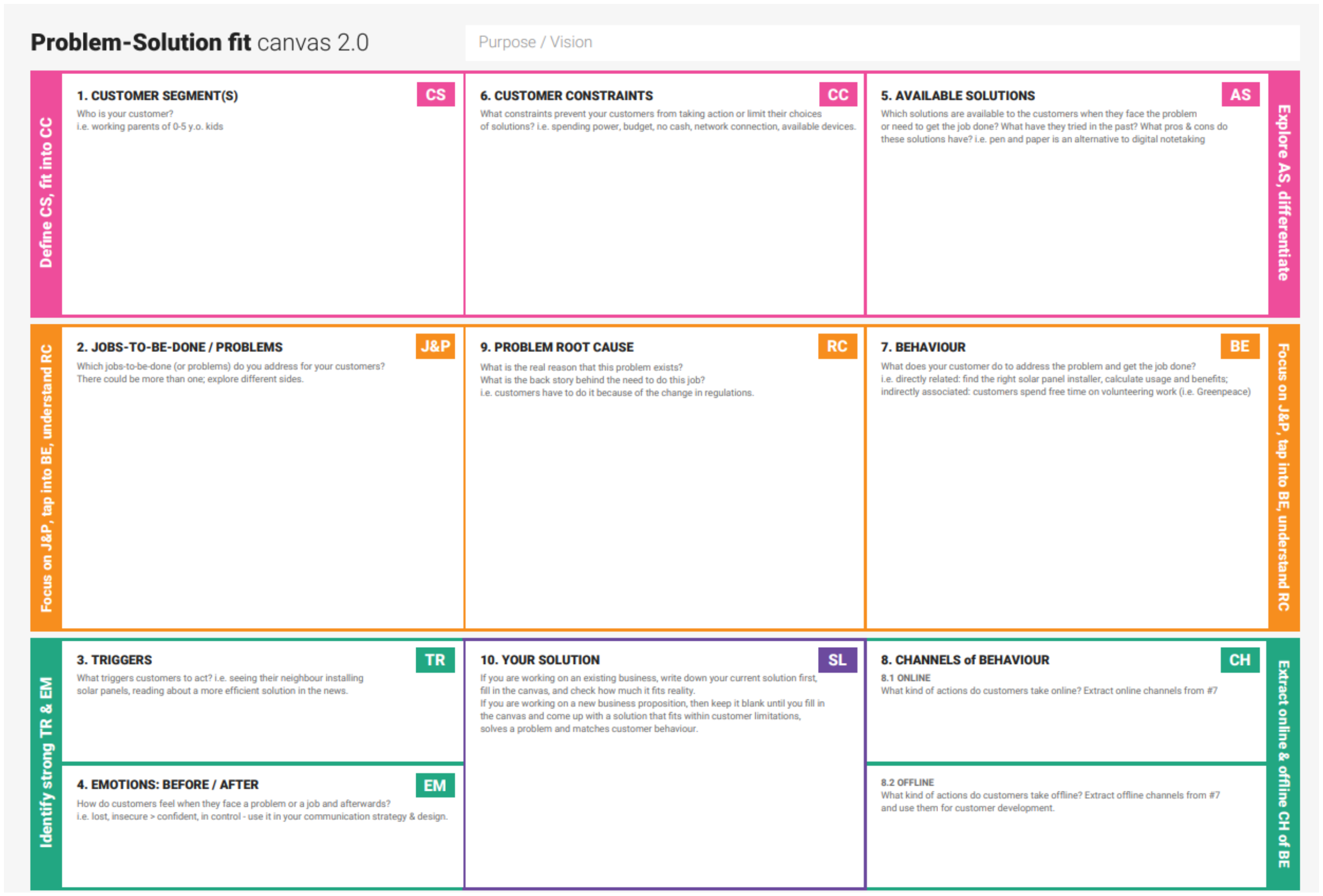


3.3. PROPOSED SOLUTION

- Convolutional Neural Networking (CNN) is being used in many fields like object detection, face recognition, spam detection, image classification.
- Many algorithms have been developed for hand written digit recognition. But due to infinite variation in writing styles they are still not up to mark.
- Poor contrast, image text vagueness, disrupted text stroke, unwanted objects, deformation, disoriented patterns and also inter-class and intraclass similarity also cause mis-classification in handwritten numeral recognition system.
- The drawbacks of the existing systems can be overcome by using CNN algorithm for training on the Modified National Institute of Standards and Technology (MNIST) dataset using OpenCV, a machine learning library written in python can provide an accuracy rate of 99.63%

S.No	Parameter	Description
1.	Problem Statement (Problem to be solved)	I am a person working in a bank. I'm trying to feed the account details into the database but it takes long time because the account details need to be entered manually which makes me feel frustrated.
2.	Idea / Solution description	A Novel method to recognize handwritten digits using Conventional Neural Networks (CNN) using the MNIST dataset.
3.	Novelty / Uniqueness	Accurately recognize the digits rather than recognizing all the characters like OCR.
4.	Social Impact / Customer Satisfaction	● Requires less time rather than entering the details manually ● More accurate as it is dedicatedly designed to recognize the digits ● Reduces the customer's waiting time
5.	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 pin-code details easily.
6.	Scalability of the Solution	● Ability to recognise digits in more noisy environments. ● There is no limit in the number of digits it can be recognized.

3.4. PROBLEM SOLUTION FIT


Problem-Solution fit canvas 2.0

Purpose / Vision

Define CS, fit into CC

1. CUSTOMER SEGMENT(S) CS
Who is your customer?
i.e. working parents of 0-5 y.o. kids

6. CUSTOMER CONSTRAINTS CC
What constraints prevent your customers from taking action or limit their choices of solutions? i.e. spending power, budget, no cash, network connection, available devices.

5. AVAILABLE SOLUTIONS AS
Which solutions are available to the customers when they face the problem or need to get the job done? What have they tried in the past? What pros & cons do these solutions have? i.e. pen and paper is an alternative to digital notetaking

Explore AS, differentiate

Focus on J&P, tap into BE, understand RC

2. JOBS-TO-BE-DONE / PROBLEMS J&P
Which jobs-to-be-done (or problems) do you address for your customers?
There could be more than one; explore different sides.

9. PROBLEM ROOT CAUSE RC
What is the real reason that this problem exists?
What is the back story behind the need to do this job?
i.e. customers have to do it because of the change in regulations.

7. BEHAVIOUR BE
What does your customer do to address the problem and get the job done?
i.e. directly related: find the right solar panel installer, calculate usage and benefits; indirectly associated: customers spend free time on volunteering work (i.e. Greenpeace)

Focus on J&P, tap into BE, understand RC

Identify strong TR & EM

3. TRIGGERS TR
What triggers customers to act? i.e. seeing their neighbour installing solar panels, reading about a more efficient solution in the news.

4. EMOTIONS: BEFORE / AFTER EM
How do customers feel when they face a problem or a job and afterwards?
i.e. lost, insecure > confident, in control - use it in your communication strategy & design.

10. YOUR SOLUTION SL
If you are working on an existing business, write down your current solution first, fill in the canvas, and check how much it fits reality.
If you are working on a new business proposition, then keep it blank until you fill in the canvas and come up with a solution that fits within customer limitations, solves a problem and matches customer behaviour.

8. CHANNELS of BEHAVIOUR CH
8.1 ONLINE
What kind of actions do customers take online? Extract online channels from #7

8.2 OFFLINE
What kind of actions do customers take offline? Extract offline channels from #7 and use them for customer development.

Extract online & offline CH of BE

4. REQUIREMENT ANALYSIS

4.1. FUNCTIONAL REQUIREMENT

FR.NO	FUNCTIONAL REQUIREMENT	SUB-REQUIREMENT
FR 1	User Registration	Registration through form
FR 2	User Login	Login via registered Username and Password
FR 3	Uploading images	Can able to input the handwritten images into the application
FR 4	Recognising digits	Display the recognized digits from the input images to the user

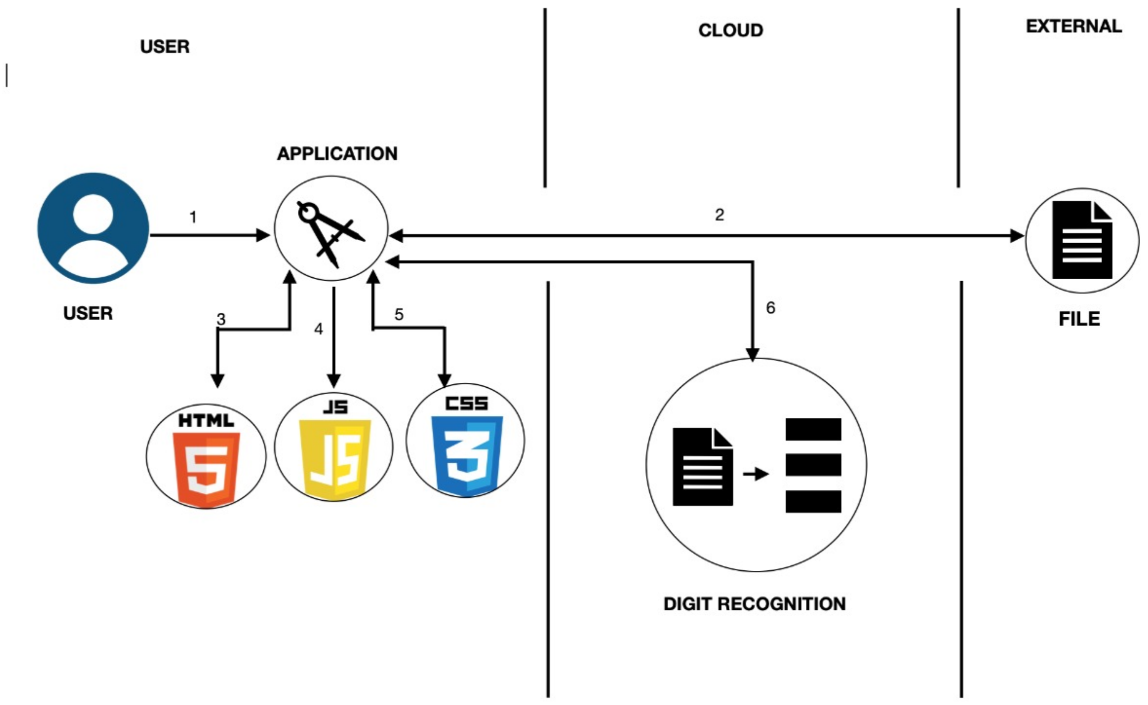
4.2 NON-FUNCTIONAL REQUIREMENT

FR.NO	NON-FUNCTIONAL REQUIREMENT	DESCRIPTION
NFR 1	Usability	The application needs to respond smoothly so that the user can use the application effectively and need to be an user friendly application.
NFR 2	Security	Ensure the security by authenticating the users using their username and password.
NFR 3	Reliability	The application does not show any error during the recognition of the digits from the uploaded images.
NFR 4	Performance	Needs to respond fast and provide the output even for the complex handwritings.
NFR 5	Availability	Need to available for all users at any time and can able to input the handwritten images to the application easily.
NFR 6	Scalability	It can able to handle N numbers of users at the same time with faster response and recognize the digits effectively.

5. PROJECT DESIGN

5.1. DATA FLOW DIAGRAMS

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.



5.2. SOLUTION AND TECHNICAL ARCHITECTURE

Technology Architecture:

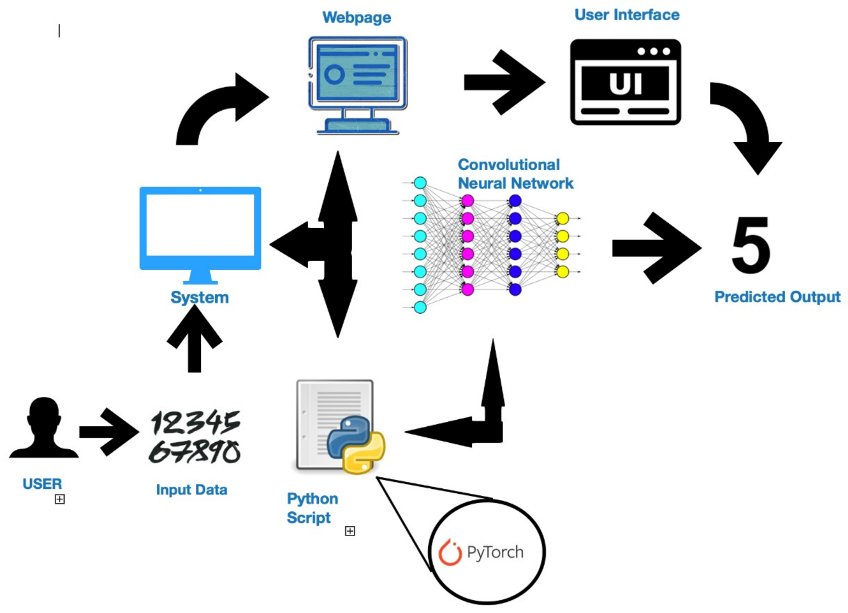


Table-1 : Components & Technologies:

S.No	Component	Description	Technology
1.	User Interface	The user interacts with the application by using the Web UI	HTML, CSS, JavaScript
2.	Application Logic-1	Logic for a process in the application	Java / Python
3.	Application Logic-2	Logic for a process in the application	CNN and Pytorch
4.	Application Logic-3	Logic for a process in the application	IBM Watson Studio
5.	Database	Data Type, Configurations etc.	MySQL
6.	Cloud Database	Database Service on Cloud	IBM Cloudant
7.	File Storage	File storage requirements	IBM storage or Local FileStorage
8.	External API-1	Purpose of External API used in the application	-
9.	External API-2	Purpose of External API used in the application	-
10.	Machine Learning Model	Purpose of Machine Learning Model	Convolutional Neural Network(CNN)
11.	Infrastructure (Server / Cloud)	Application Deployment on Local System / Cloud Local Server Configuration Cloud Server Configuration	IBM CLOUD SERVICE

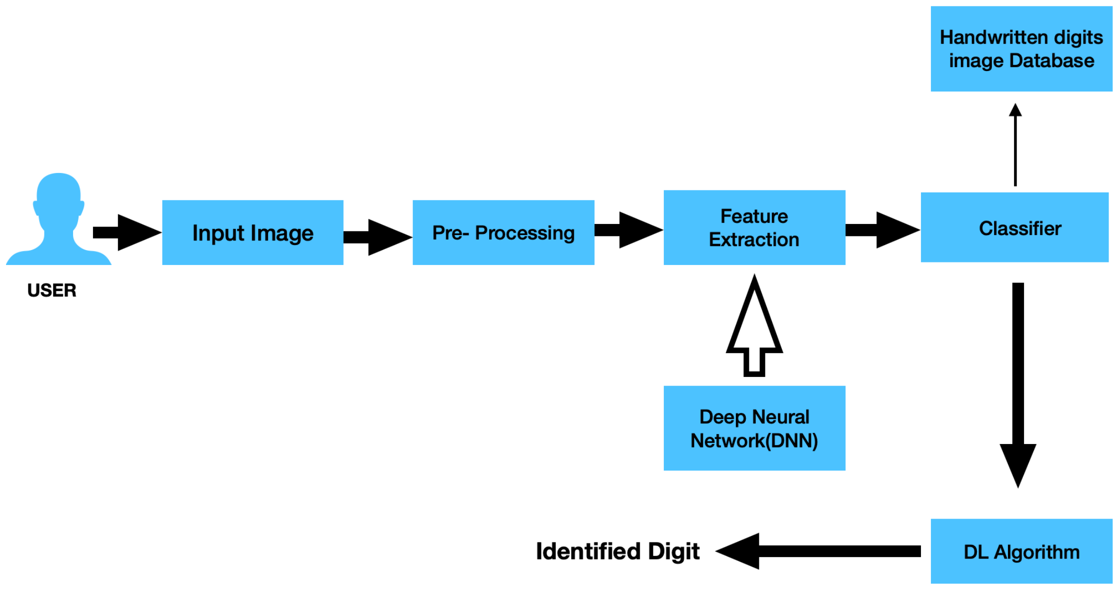
Table-2: Application Characteristics:

S.No	Characteristics	Description	Technology
1.	Open-Source Frameworks	List the open-source frameworks used	CNN, TENSOR FLOW
2.	Security Implementations	List all the security / access controls implemented, use of firewalls etc.	User Authentication by password authentication
3.	Scalable Architecture	Justify the scalability of architecture (3 – tier, Micro-services)	AWS service
4.	Availability	Justify the availability of application (e.g. use of load balancers, distributed servers etc.)	IBM load balancers
5.	Performance	Design consideration for the performance of the application (number of requests per sec, use of Cache, use of CDN's) etc.	IBM load balancers

Solution Architecture:

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

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



5.3. USER STORIES

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Custom er (Web user)	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	I can access my account	High	Sprint-1
	Login	USN-2	As a user, I can log into the application by entering email & password	I can login into my account	High	Sprint-1
	Upload Image	USN-3	As a user, I can able to draw the images of the digits in the given canvas	I can draw the images of digits	Medium	Sprint-2
		USN-4	As a user, I can able to input the images of the handwritten documents or images to the application	I can upload the images of the handwritten documents or images	High	Sprint-3
	Recognise Digit	USN-5	As a user I can able to get the recognised digit as output from the drawn images	I can access the recognized digits from digits from the drawn images	Medium	Sprint-4

		USN-6	As a user I can able to get the recognised digit as output from the images of handwritten documents or images	I can access the recognized digits from handwritten documents or images	High	Sprint-4

6. PROJECT PLANNING AND SCHEDULING

6.1. SPRINT PLANNING AND ESTIMATION

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

SPRINT 1

EFFORT	HOURS	DAY 1	DAY 2	DAY 3	DAY 4	DAY 5	DAY 6
ACTUAL EFFORT	20	4	6	3	5	2	0
ESTIMATED EFFORT	20	5	5	5	5	5	0

SPRINT 2

EFFORT	HOURS	DAY 1	DAY 2	DAY 3	DAY 4	DAY 5	DAY 6
ACTUAL EFFORT	15	4	2	5	2	2	0
ESTIMATED EFFORT	15	3	3	3	3	3	0

SPRINT 3

EFFORT	HOURS	DAY 1	DAY 2	DAY 3	DAY 4	DAY 5	DAY 6
ACTUAL EFFORT	25	4	7	6	6	2	0
ESTIMATED EFFORT	25	5	5	5	5	5	0

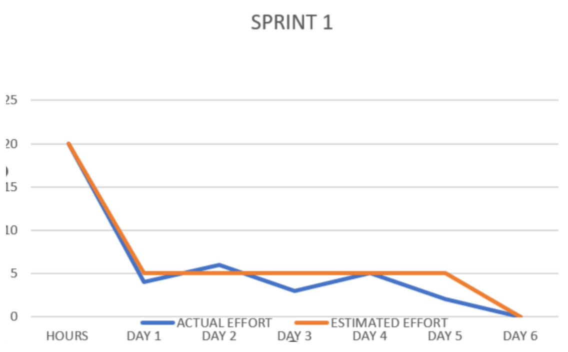
SPRINT 4

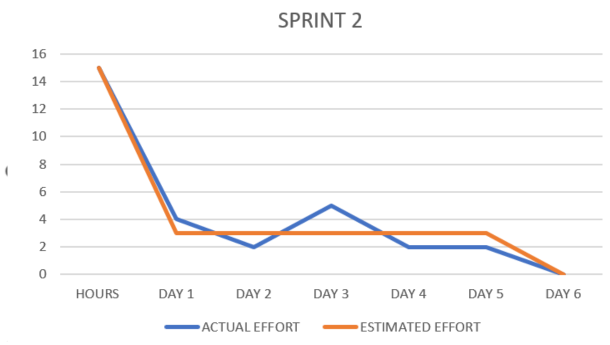
EFFORT	HOURS	DAY 1	DAY 2	DAY 3	DAY 4	DAY 5	DAY 6
ACTUAL EFFORT	40	9	7	10	9	5	0
ESTIMATED EFFORT	40	8	8	8	8	8	0

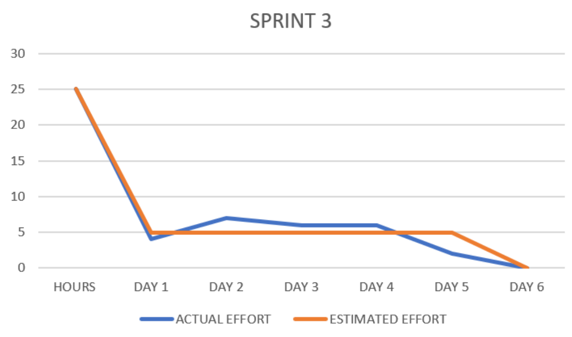
6.2. SPRINT DELIVERY SCHEDULE

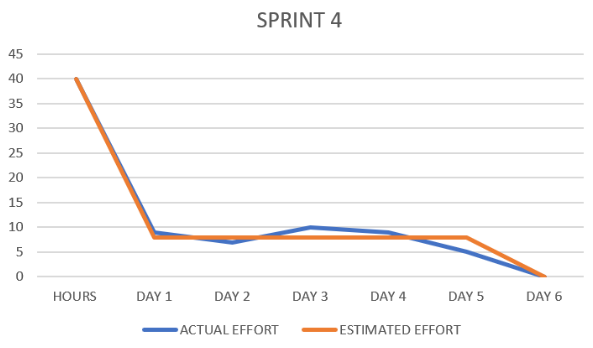
Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)
Sprint-1	20	6 Days	24 Oct 2022	29 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022

6.3. REPORTS FROM JIRA









7. CODING AND SOLUTIONING

7.1. FEATURE 1

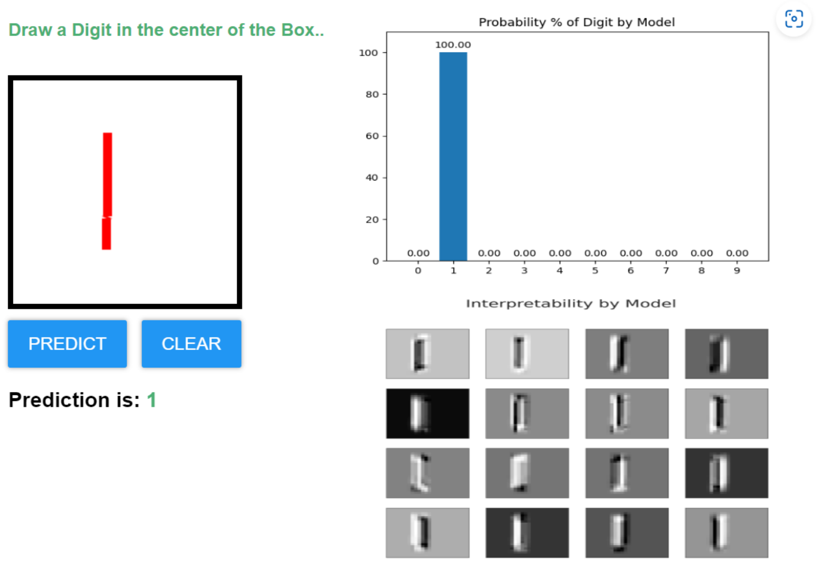
The application facilitates the user to draw the digit on a given canvas and predict the drawn digit.The predicted digit is displayed by means of a prediction graph along with the accuracy rate.

CODE FOR FEATURE 1 :

```
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):
    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 home():
    return render_template("index.html")
@app.route("/recognize_page", methods=["GET", "POST"])
def recog_page():
    return render_template("Recognize.html")
@app.route("/recongize", methods=["GET", "POST"])
def recog():
    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)
```

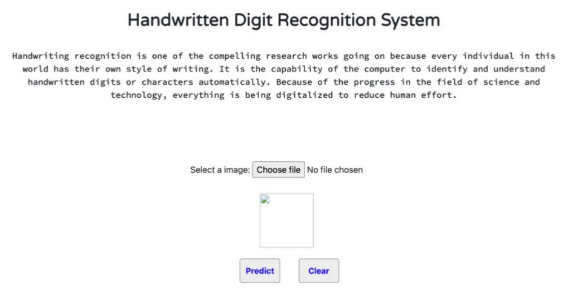

Draw a Digit in the center of the Box..
PREDICT
CLEAR
Prediction is: 1
Probability % of Digit by Model
100.00
0.00
Interpretability by Model

7.2. FEATURE 2

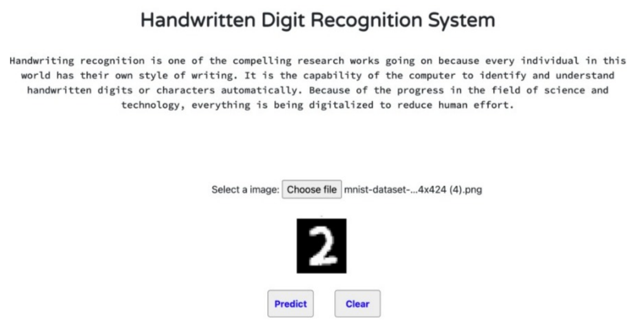
The application facilitates the user to upload the image of the handwritten digit and predict the digit in the uploaded document

CODING FOR FEATURE 2

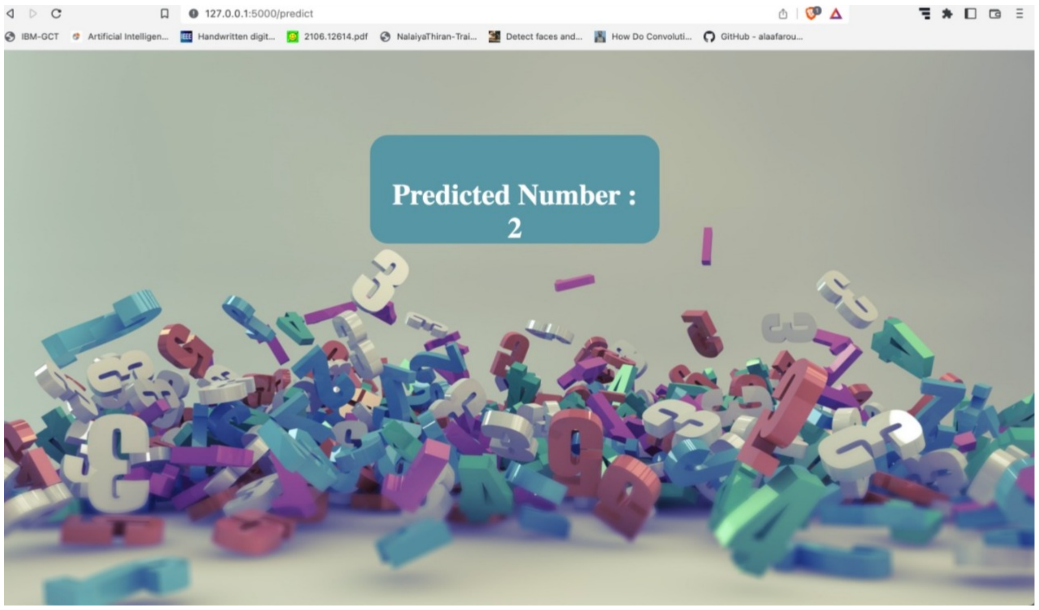
```
import numpy as np
import os
from PIL import Image
from flask import Flask, request, render_template, url_for
from werkzeug.utils import secure_filename, redirect
from gevent.pywsgi import WSGIServer
from keras.models import load_model
from keras.preprocessing import image
from flask import send_from_directory
UPLOAD_FOLDER = 'uploads'
app = Flask(__name__)
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER
model = load_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)
```


Handwritten Digit Recognition System

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.


Handwritten Digit Recognition System

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.



Predicted Number : 2

8. TESTING

8.1. TEST CASES

TEST CASES FOR SPRINT 1 :

	Test Scenarios
1	Verify user is able to see the sigin page or not?
2	Verify user is able to signin into the application or not?
3	Verify user is able to see the sigin page or not?
4	Verify user is able to signin into the application or not?

TEST CASES FOR SPRINT 2 :

	Test Scenarios
1	Verify user is able to view the index page
2	Verify user is able to view the recognize option or not?
3	Verify user is able to click the recognize option or not?
4	Verify user is able to view the page to draw the image on the canvas or not?
5	Verify user is able to draw the image on the canvas or not?
6	Verify user is able to view the clear icon or not?
7	Verify user is able to click the predict option or not?
8	Verify user is able to view the predict page or not?
9	Verify user is able to view the output of the drawn image in the digital format or not?
10	Verify user is able to view the predicted graph of the drawn image in the canvas along with the accuracy rate or not?
11	Verify user is able to clear the already drawn image on the canvas and able to redraw a new image on the canvas or not?

TEST CASES FOR SPRINT 3 :

	Test Scenarios
1	Verify user is able to view the index page
2	Verify user is able to view the recognize option or not ?
3	Verify user is able to click the recognize option or not ?
4	Verify user is able to view the page to upload the image on the canvas or not?
5	Verify user is able to upload image or not?
6	Verify user is able to view the cancel icon or not?
7	Verify user is able to click the predict option or not?
8	Verify user is able to view the predict page or not?
9	Verify user is able to view the output of the uploaded image or not?
10	Verify user is able to view the cancel icon and able to cancel the already uploaded file and can able to reupload a new file or not?

TEST CASES FOR SPRINT 4:

	Test Scenarios
1	Verify user can able to login and signup or not
2	Verify user is able to view the index page
3	Verify user is able to view the recognize option or not?
4	Verify user is able to click the recognize option or not?
5	Verify user is able to view the recognize page to view the recognize options for prediction by uploading the image of the handwritten digit and prediction by drawing the image on the
6	Verify user is able to predict the digit drawn on the canvas by clicking the recognize option for prediction by drawing on the canvas or not?
7	Verify user is able to predict the digit in the uploaded image of the handwritten digit or not?

8.2. USER ACCEPTANCE TESTING

DEFECT ANALYSIS

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
By Design	10	4	2	3	19
Duplicate	1	0	3	0	4
External	2	3	0	1	6
Fixed	11	2	4	20	37
Not Reproduced	0	0	0	0	0
Skipped	0	0	0	0	0
Won't Fix	0	0	0	0	0

TEST CASES ANALYSIS

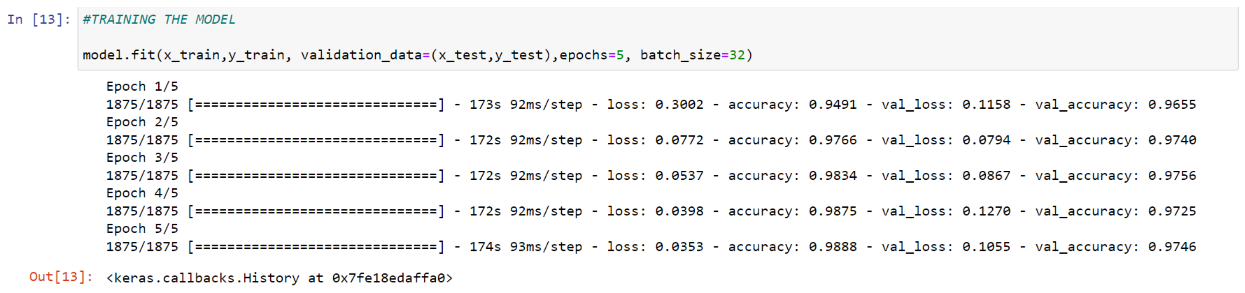
Section	Total Cases	Not Tested	Fail	Pass
Print Engine	8	0	0	8
Client Application	35	0	0	35
Security	4	0	0	4
Outsource Shipping	0	0	0	0
Exception Reporting	0	0	0	0
Final Report Output	3	0	0	3
Version Control	0	0	0	0

9. RESULTS

9.1. PERFORMANCE METRICS

S.No.	Parameter	Values
1.	Model Summary	-
2.	Accuracy	Training Accuracy - 0.97708 Validation Accuracy -0.97244

Model summary:



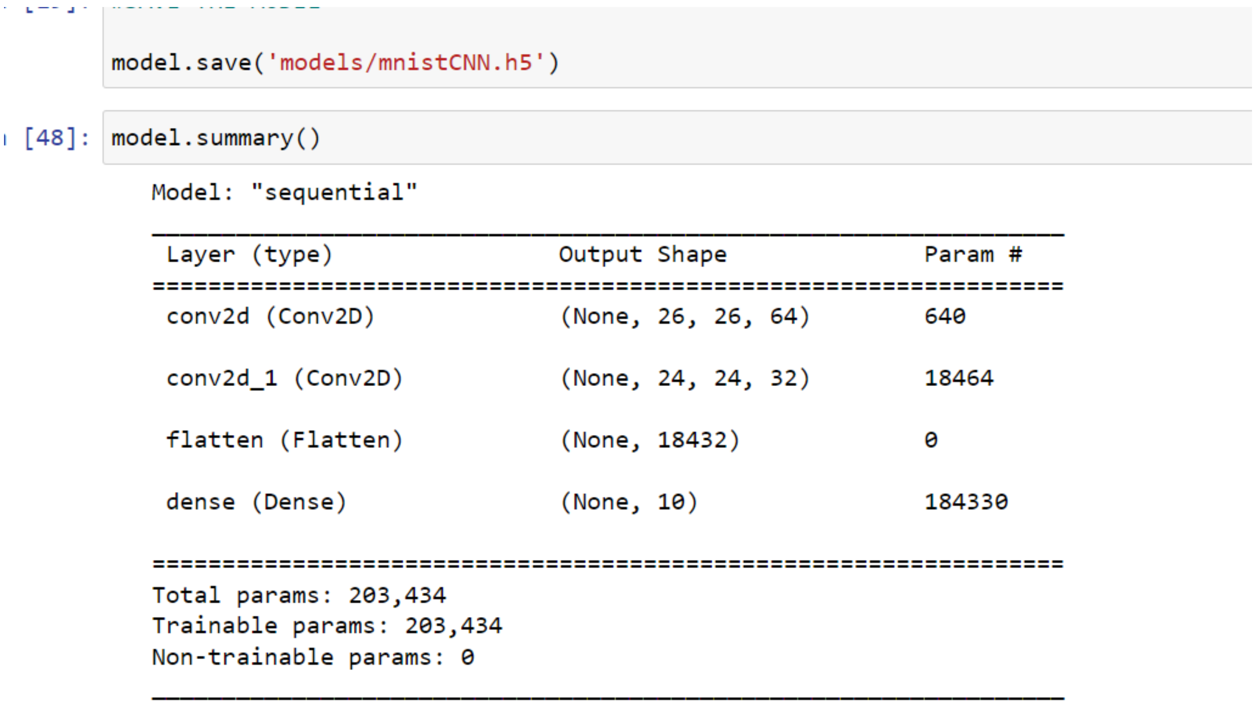
```
In [13]: #TRAINING THE MODEL

         model.fit(x_train,y_train, validation_data=(x_test,y_test),epochs=5, batch_size=32)

           Epoch 1/5
           1875/1875 [==============================] - 173s 92ms/step - loss: 0.3002 - accuracy: 0.9491 - val_loss: 0.1158 - val_accuracy: 0.9655
           Epoch 2/5
           1875/1875 [==============================] - 172s 92ms/step - loss: 0.0772 - accuracy: 0.9766 - val_loss: 0.0794 - val_accuracy: 0.9740
           Epoch 3/5
           1875/1875 [==============================] - 172s 92ms/step - loss: 0.0537 - accuracy: 0.9834 - val_loss: 0.0867 - val_accuracy: 0.9756
           Epoch 4/5
           1875/1875 [==============================] - 172s 92ms/step - loss: 0.0398 - accuracy: 0.9875 - val_loss: 0.1270 - val_accuracy: 0.9725
           Epoch 5/5
           1875/1875 [==============================] - 174s 93ms/step - loss: 0.0353 - accuracy: 0.9888 - val_loss: 0.1055 - val_accuracy: 0.9746

Out[13]: <keras.callbacks.History at 0x7fe18edaffa0>
```

Accuracy:



```
model.save('models/mnistCNN.h5')
```

```
[48]: 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
_________________________________________________________________
```

10. ADVANTAGES AND DISADVANTAGES

10.1. ADVANTAGES

- By using CNN algorithm for training on the Modified National Institute of Standards and Technology (MNIST) dataset using OpenCV, a machine learning library written in python can provide an accuracy rate of 99.63%.
- The application enables the user to predict the digits by uploading images of the handwritten digits or by drawing the digits on the provided canvas.
- Easily navigatable.
- User-friendly
- This application is particularly designed to predict the handwritten digits.

10.2. DISADVANTAGES

- As the application is particularly designed to predict the handwritten digits it cannot be used to predict the handwritten characters.
- The digits drawn with close proximity to the actual digit only recognised with high accuracy rate.

11. CONCLUSION

We have implemented the models for handwritten digit recognition using MNIST datasets, based on CNN.Support vector machines are one of the basic classifiers that’s why it’s faster than most algorithms and in this case, gives the maximum training accuracy rate but due to its simplicity, it’s not possible to classify complex and ambiguous images as accurately as achieved with CNN algorithms. We have found that CNN gave the most accurate results for handwritten digit recognition. So, this makes us conclude that CNN is best suitable for any type of prediction problem including image data as an input.

12. FUTURE SCOPE

The future development of the applications based on CNN is practically boundless. In the future, we can work on a denser or hybrid algorithm than the current set of algorithms with more manifold data to achieve the solutions to many problems.

In future, the application of these algorithms lies from the public to high-level authorities, as from the differentiation of the algorithms above and with future development we can attain high-level functioning applications which can be used in the classified or government agencies as well as for the common people, we can use these algorithms in hospitals application for detailed medical diagnosis, treatment and monitoring the patients, we can use it in surveillance system to keep tracks of the suspicious activity under the system, in fingerprint and retinal scanners, database filtering applications, Equipment checking for national forces and many more problems of both major and minor category.

SOURCE CODE:

1. LOGIN AND SIGNUP PAGE

```
<!DOCTYPE html>
<html>
  <head>
    <title>Login Form &middot; UI Library</title>
    <link type="text/css" media="screen" rel="stylesheet"
href="http://fonts.googleapis.com/css?family=Roboto:100,300,400,500,700">
    <link type="text/css" media="screen" rel="stylesheet" href="./main.css">
  </head>
  <body>
    <h1 class="pro-title">Handwritten Digit Recognition System</h1>
    <div class="form">
      <div class="form-toggle"></div>
```

```html
<div class="form-panel one">
  <div class="form-header">
    <h1>Account Login</h1>
  </div>
  <div class="form-content">
    <form>
      <div class="form-group">
        <label for="username">Username</label>
        <input type="text" id="username" name="username" required>
      </div>
      <div class="form-group">
        <label for="password">Password</label>
        <input type="password" id="password" name="password" required>
      </div>
      <div class="form-group">
        <label class="form-remember">
          <input type="checkbox">Remember Me
        </label><a class="form-recovery" href="#">Forgot Password?</a>
      </div>
      <div class="form-group">
        <button type="submit">Log In</button>
      </div>
    </form>
  </div>
</div>
<div class="form-panel two">
  <div class="form-header">
    <h1>Register Account</h1>
  </div>
  <div class="form-content">
    <form>
      <div class="form-group">
        <label for="username">Username</label>
        <input type="text" id="username" name="username" required>
      </div>
      <div class="form-group">
        <label for="password">Password</label>
        <input type="password" id="password" name="password" required>
      </div>
      <div class="form-group">
        <label for="cpassword">Confirm Password</label>
        <input type="password" id="cpassword" name="cpassword" required>
      </div>
      <div class="form-group">
        <label for="email">Email Address</label>
        <input type="email" id="email" name="email" required>
      </div>
      <div class="form-group">
```

```
          <button type="submit">Sign up</button>
         </div>
        </form>
       </div>
      </div>
    </div>
    <script src="./jquery-2.1.4.min.js"></script>
    <script src="./script.js"></script>
  </body>
</html>
```

CSS FOR LOGIN AND SIGNUP PAGE:

```
$(document).ready(function() {
  var panelOne = $('.form-panel.two').height(),
    panelTwo = $('.form-panel.two')[0].scrollHeight;
  $('.form-panel.two').not('.form-panel.two.active').on('click', function(e) {
    e.preventDefault();
    $('.form-toggle').addClass('visible');
    $('.form-panel.one').addClass('hidden');
    $('.form-panel.two').addClass('active');
    $('.form').animate({
      'height': panelTwo
    }, 200);
  });
  $('.form-toggle').on('click', function(e) {
    e.preventDefault();
    $(this).removeClass('visible');
    $('.form-panel.one').removeClass('hidden');
    $('.form-panel.two').removeClass('active');
    $('.form').animate({
      'height': panelOne
    }, 200);
  });
});
```

2.INDEX PAGE

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <link rel="stylesheet" href="./Style.css">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/6.2.0/css/all.min.css">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <style>
        @import url('https://fonts.googleapis.com/css2?family=Poppins:wght@500;700&display=swap');
*{
    margin: 0;
    padding: 0;
}
body{
    background-image: url(./bg.jpg);
    background-size: cover;
}
.nav-container{
    display: flex;
    justify-content: end;
    height: 100px;
}
.nav-list{
    padding-right: 50px;
    display: flex;
    width: 20%;
    justify-content: end;
    align-items: center;
}
.nav-item{
    width: 70%;
    display: flex;
    justify-content: space-around;
}
.nav-item .nav-links {
    list-style: none;
}
.nav-item .nav-links input{
    text-decoration: none;
    font-family: 'Poppins', sans-serif;
    color: black;
    border: none;
    background-color: white;
    font-size: 16px;
    cursor: pointer;
}
.heading{
    display: flex;
    justify-content: center;
}
.heading .sub-div {
    text-align: center;
    width: 40%;
    font-family: 'Poppins', sans-serif;
    font-weight: 700;
    margin-bottom: 50px;
}
.content{
    display: flex;
    justify-content: center;
}
.des{
    width: 70%;
    text-align: center;
    font-family: 'Poppins', sans-serif;
    justify-content: center;
    display: flex;
    background-color: rgba(0, 0, 0, 0.126);
    border-radius: 30px;
}
.sub-des{
   font-size: 20px;
   padding: 70px 0;
   width: 80%;
}
    </style>
</head>
<body>

    <div class="nav-container">
        <div class="nav-list">
            <ul class="nav-item">
                <form action="/" method="post">
                <li class="nav-links home">
                    <i class="fa-solid fa-house"></i>
                    <input type="submit" value="Home"></a>
                     </li>
                    </form>
            <form action="/recognize_page" method="post">
                <li class="nav-links"><input type="submit" value="Recognize"></input></li>
            </form>
            </ul>
        </div>
    </div>
    <div class="heading">
        <div class="sub-div">
            <h1>Handwritten Digit Recognition System</h1>
        </div>
    </div>
    <div class="content">
        <div class="des">
            <div class="sub-des">
                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) using Conventional Neural Network(CNN). Digit recognition has many applications like
number plate recognition, postal mail sorting, bank cheque processing etc.,
            </div>
        </div>
    </div>
</body>
</html>
```

3.RECOGNIZE PAGE

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <link rel="stylesheet" href="./recog.css">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/6.2.0/css/all.min.css">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <style>
@import url('https://fonts.googleapis.com/css2?family=Poppins:wght@500;700&display=swap');
*{
    margin: 0;
    padding: 0;
}
.nav-container{
    display: flex;
    justify-content: end;
    height: 100px;
}
.nav-list{
    padding-right: 50px;
    display: flex;
    width: 20%;
    justify-content: end;
    align-items: center;
}
.nav-item{
    width: 70%;
    display: flex;
    justify-content: space-around;
}
.nav-item .nav-links {
    list-style: none;
}
.nav-item .nav-links input{
    text-decoration: none;
    font-family: 'Poppins', sans-serif;
    color: black;
    border: none;
    background-color: white;
    font-size: 16px;
    cursor: pointer;
}
.heading{
    display: flex;
    justify-content: center;
}
.heading .sub-div {
    text-align: center;
    width: 40%;
    font-family: 'Poppins', sans-serif;
    font-weight: 700;
    margin-bottom: 50px;
}
.content-container{
    display: flex;
    justify-content: center;
    height: 400px;
}
.content-container .boxs{
    position: relative;
    display:flex;
    justify-content: space-around;
    width: 100%;
}
.content-container .boxs .box1{
    align-items: center;
    flex-direction: column;
    display:flex;
   width: 40%;
   background-color: rgba(117, 117, 117, 0.219);
   border-radius: 20px;
   position: relative;
}
.content-container .boxs .box1 .box1-title{
    font-family: 'Poppins', sans-serif;
    margin: 20px;
}
.content-container .boxs .box2 .box2-title{
    font-family: 'Poppins', sans-serif;
    margin: 17px;
}
.content-container .boxs .box2{
    align-items: center;
    flex-direction: column;
    display:flex;
   width: 40%;
   background-color: rgba(117, 117, 117, 0.219);
   border-radius: 20px;
```

```
}
.content-container .boxs .box1 .btn-1{
    display: flex;
    justify-content: center;
}
.input-box1{
    width: 400px;
    height:100px;
    position: relative;
    padding-top: 50px;
    text-align: center;
    margin: 30px 0;
    font-family: 'Poppins', sans-serif;
}
.input-box2{
    padding-top: 30px;
    text-align: center;
    width:400px;
    height:100px;
    position: relative;
    margin: 30px 0;
    font-family: 'Poppins', sans-serif;
}
.btn-1 input{
    border: none;
    width: 130px;
    height: 40px;
    border-radius: 30px;
    background-color: rgb(16, 197, 61);
    color: white;
    font-family: 'Poppins', sans-serif;
}
.btn-2 input{
    border: none;
    margin-top: 10px;
    width: 130px;
    height: 40px;
    border-radius: 30px;
    background-color: rgb(197, 58, 16);
    color: white;
    font-family: 'Poppins', sans-serif;
}
    </style>
</head>
<body>
        <div class="nav-container">
            <div class="nav-list">
                <ul class="nav-item">
                <li class="nav-links home">
                    <form action="/" method ="post">
                        <i class="fa-solid fa-house"></i>
                    <input type="submit" value="Home">
                     </li>
                    </form>
                <form action="/recognize_page" method ="post">
                <li class="nav-links"><input type="submit" value="Recognize"></li></form>
            </ul>
        </div>
    </div>
  <div class="heading">
      <div class="sub-div">
          <h1>Handwritten Digit Recognition System</h1>
      </div>
  </div>
  <div class="content-container">
      <div class="boxs">
          <div class="box1">
          <form action="/recognize" method="post">
             <div class="box1-title"><h3>Recognizing Digits from Drawing Images</h3></div>
             <div class="input-box1">Draw the digit on the given canva and click predict to recognize the
drawn digit </div>
             <div class="btn-1"><input type="submit" value="Recognize"></div>
          </form>
          </div>
          <div class="box2">
              <div class="box2-title"><h3>Recognizing Handwritten Digits from Uploaded
Document</h3></div>
              <div class="input-box2">Upload the image containing the handwritten digit and click predict to
recognize the digit in the image </div>
              <div class="btn-2"><input type="submit" value="Recognize"></div>
          </div>
      </form>
      </div>
  </div>
</body>
</html>
```

4.RECOGNIZING DIGITS BY DRAWING DIGITS IN THE CANVAS:

```
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):
    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 home():
    return render_template("index.html")
@app.route("/recognize_page", methods=["GET", "POST"])
def recog_page():
    return render_template("Recognize.html")
@app.route("/recongize", methods=["GET", "POST"])
def recog():
    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)
```

5.RECOGNIZING DIGITS FROM THE UPLOADED IMAGE OF THE HANDWRITTEN DIGIT:

```
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 = 'uploads'
app = Flask(__name__)
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER
model = load_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)
```

CODE FOR TRAINING THE MODEL :

```
#IMPORTING THE REQUIRED LIBRARIES
import numpy
import tensorflow
from tensorflow.keras.datasets import mnist
from tensorflow.keras.models import Sequential
from tensorflow.keras import layers
from tensorflow.keras.layers import Dense, Flatten
from tensorflow.keras.layers import Conv2D
from tensorflow.keras.optimizers import Adam
from keras.utils import np_utils
#LOADING THE DATA
(x_train,y_train),(x_test,y_test)=mnist.load_data()
print(x_train.shape)
print(x_test.shape)
#ANALYSING THE DATA
x_train[0]
y_train[0]
import matplotlib.pyplot as plt
plt.imshow(x_train[0])
#RESHAPING THE DATA

x_train=x_train.reshape(60000,28,28,1).astype('float32')
x_test=x_test.reshape(10000,28,28,1).astype('float32')
#ONE HOT ENCODING
number_of_classes=10
y_train=np_utils.to_categorical(y_train,number_of_classes)
y_test=np_utils.to_categorical(y_test,number_of_classes)
y_train[0]
#ADDING CNN LAYERS
model=Sequential()
model.add(Conv2D(64,(3,3), input_shape=(28, 28, 1),activation='relu'))
model.add(Conv2D(32,(3,3), activation='relu'))
model.add(Flatten())
model.add(Dense(number_of_classes, activation='softmax'))
#COMPILING THE MODEL
model.compile(loss='categorical_crossentropy', optimizer="Adam", metrics=['accuracy'])
#TRAINING THE MODEL
model.fit(x_train,y_train, validation_data=(x_test,y_test),epochs=5, batch_size=32)
#OBSERVING THE METRICS
metrics=model.evaluate(x_test,y_test,verbose=0)
print("Metrics(Test loss & Test Accuracy):")
print(metrics)
#TESTING THE MODEL
prediction=model.predict(x_test[:4])
print(prediction)
import numpy as np
print(np.argmax(prediction,axis=1))
print(y_test[:4])
#OBSERVING THE METRICS
metrics=model.evaluate(x_test, y_test, verbose=0)
print("Metrics(Test loss & Test Accuracy): ")
print(metrics)
#TEST THE MODEL
```

```
prediction=model.predict(x_test[:4])
print(prediction)
#SAVE THE MODEL
model.save('models/mnistCNN.h5')
```

GITHUB LINK :Handwritten digit recognition

PROJECT DEMO LINK :
https://drive.google.com/file/d/1rJnqzSe9utSTclJ0jU46yYAUSNItllFF/view?usp=sharing