Handwritten Numerical Character Recognition Using Deep Learning

Application Development-I Report Submitted
In partial fulfilment of the requirement for the award of the degree of

Bachelor of Technology
In
Computer Science and Engineering -Artificial Intelligence and Machine Learning

by

V. SAI VENKAT - 22N31A66F9

SK. IMRAN - 22N31A66G2

T. SAI YATISH - 23N35A6618

Under the Guidance of
Mr. T. Hari Babu
Assistant Professor

Computational Intelligence Department
MRCET

DEPARTMENT OF COMPUTATIONAL INTELLIGENCE
MALLAREDDY COLLEGE OF ENGINEERING AND TECHNOLOGY

(Affiliated to JNTU, Hyderabad)

ACCREDITED by AICTE-NBA
Maisammaguda, Dhulapally post, Secunderabad-500014.

2024-2025

DECLARATION

I hereby declare that the project entitled **"Handwritten Numerical Character Recognition Using Deep Learning"** submitted to **Malla Reddy College of Engineering and Technology,** affiliated to Jawaharlal Nehru Technological University Hyderabad (JNTUH) for the award of the degree of **Bachelor of Technology** in **Computer Science and Engineering- Artificial Intelligence and Machine Learning** is a result of original research work done by me.

It is further declared that the project report or any part there of has not been previously submitted to any University or Institute for the award of degree or diploma.

V. Sai Venkat (22N31A66F9)

Sk. Imran (22N31A66G2)

T. Sai Yathish (23N35A6618)

MALLA REDDY COLLEGE
OF ENGINEERING & TECHNOLOGY
(AUTONOMOUS INSTITUTION - UGC, GOVT. OF INDIA)
Affiliated to JNTUH; Approved by AICTE, NBA-Tier 1 & NAAC with A-GRADE | ISO 9001:2015

CERTIFICATE

This is to certify that this is the Bonafide record of the project titled **"Handwritten Numerical Character Recognition Using Deep Learning"** submitted by **V. Sai Venkat (22N31A66F9), Sk. Imran (22N31A66G2), T. Sai Yathish (23N35A6618)** of B. Tech in the partial fulfillment of the requirements for the degree of **Bachelor of Technology** in **Computer Science and Engineering- Artificial Intelligence and Machine Learning**, Dept. of CI during the year 2024-2025. The results embodied in this project report have not been submitted to any other university or institute for the award of any degree or diploma.

Mr. T. Hari Babu
Assistant Professor
INTERNAL GUIDE

Dr. D. Sujatha M. Tech, PhD
Professor
HEAD OF THE DEPARTMENT

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

We feel honored and privileged to place our warm salutation to our college Malla Reddy College of Engineering and technology (UGC-Autonomous), our Director ***Dr. VSK Reddy*** who gave us the opportunity to have experience in engineering and profound technical knowledge.

We are indebted to our Principal ***Dr. S. Srinivasa Rao*** for providing us with facilities to do our project and his constant encouragement and moral support which motivated us to move forward with the project.

We would like to express our gratitude to our Head of the Department ***Dr. D. Sujatha,*** Professor, for encouraging us in every aspect of our system development and helping us realize our full potential.

We would like to thank our Application Development-I coordinator **Dr. P. Hari Krishna**, Associate Professor, as well as our internal guide **Mr. T. Hari Babu,** Assistant Professor, for their structured guidance and never-ending encouragement. We are extremely grateful for valuable suggestions and unflinching co-operation throughout Application Development-I work.

We would also like to thank all supporting staff of department of CI and all other departments who have been helpful directly or indirectly in making our Application Development-I a success.

We would like to thank our parents and friends who have helped us with their valuable suggestions and support has been very helpful in various phases of the completion of the Application Development-I.

By

V. Sai Venkat (22N31A66F9)
Sk. Imran (22N31A66G2)
T. Sai Yathish (23N35A6618)

ABSTRACT

Handwritten digit recognition is a technique or technology for automatically recognizing and detecting handwritten digital data through different Deep Learning models. Deep Learning is used in Recognition technique. The working logic of the handwriting digit recognition process is examined, and the efficiency of different algorithms on the same database is measured. One of the most practical difficulties in pattern recognition applications is handwritten digit recognition. Digit recognition is used in postal mail sorting, bank check processing, form data entry, and other applications. The capacity to design an efficient algorithm that can recognize handwritten digits and which is submitted by users via scanner, tablet, and other digital devices is at the heart of the problem. Handwritten digit recognition is already widely employed in the automated processing of bank checks, postal addresses, and other types of information. Handwriting recognition system plays a very important role in today's world. At present time it is very difficult to find the correct meaning of handwritten documents as different people have different ways of writing digits or texts. There are many areas where we need to recognize the words, alphabets and digits like postal addresses, bank cheques etc. The main aim of this project is to make a GUI based application that allows users to predict the digits drawn on the interface using deep learning and shows the results in percentage how likely the prediction is correct.

TABLE OF CONTENTS

CHAPTER 1
INTRODUCTION

1.1 Purpose:

The purpose of the document for the Handwritten Numerical Character recognition Using Deep Learning is to allow us to explain the implementation of the deep learning models and how the models use the dataset to analyze and predict the outcome. This project allows us to study the neural network of the Handwritten Numerical Recognizer and the working of the model to see how the Deep Learning process works in the background. It shows the working of how the number fed into the model is recognized and displays the number that is recognized after analysis and comparing it with the dataset fed to the model. With the help of the MNIST dataset and the CNN neural network, the project can explain the purpose, working and the benefits of deep learning in our daily lives.

1.2 Motivation of project:

The Handwritten Numerical Character Recognition Using Deep Learning project is the applications that Machine Learning can be used range from self-driving vehicles to predicting deadly diseases and analyzing the pattern that can detect the type of disease and other various things. With the availability of so much data, it is possible now to build any predictive models that can have the ability to study and analyze the various patterns or complex data to find useful information and deliver the outcome needed. This high demand for Machine Learning skills is the motivation behind this project. The motivation of this project is to illustrate Deep Learning with Handwritten Numerical Recognizer and explain the process of the Neural Network used in this project.

1.3 Scope of project:

The project focuses on creating a system that can accurately recognize handwritten numerical characters (0-9). It will use a collection of handwritten digit samples for training and testing, with preprocessing steps like resizing and cleaning the images to improve accuracy. The system will be evaluated for its ability to handle various handwriting styles and noisy inputs. Applications include digitizing handwritten records, automating form processing, and improving accessibility. The project aims to ensure accuracy, efficiency, and robustness, with potential for future expansion to recognize more complex handwriting patterns, with applications in forms processing, accessibility tools, and digitizing records. Future extensions could include recognizing alphabets, multi-language support, and deployment on mobile or web platforms. The project will also focus on performance metrics like accuracy and computational efficiency to ensure practical utility.

1.4 Project Features:

1. **Dataset Handling-**

Support for Standard Datasets: The system can utilize datasets like MNIST or custom datasets with handwritten numeric characters. Data Preprocessing: Image resizing to a fixed size (e.g., 28x28 pixels). Grayscale conversion to reduce computational complexity. Noise removal to improve recognition accuracy.

2. **Model Features-**

Deep Learning Architecture: A Convolutional Neural Network (CNN) designed to extract features and classify numeric characters. Multiple convolutional layers, pooling layers, and fully connected layers for robust learning.

Training: Ability to train the model on large datasets with optimization techniques like Adam or SGD. Use of data augmentation to handle variations in handwriting styles.

Evaluation: Validation of model performance using metrics such as accuracy, precision, recall, and F1-score. Confusion matrix generation to analyse misclassified digits.

3. **Recognition System-**

Single Character Recognition: Recognizes individual handwritten numeric characters in an image.

CHAPTER 2

SYSTEM REQUIREMENTS

2.1 Hardware Requirements:

- RAM 4 GB
- Processor Intel(R)core (TM)i3 or more.
- Hard Disk – 512 GB
- Keyboard
- Mouse

2.2 Software requirements:

Backend:

- Python
- MySQL 8.0

Operating System:

- Windows 11

2.3 Existing System:

- CNN (Convolutional Neural Network)
- MNIST Data Sets.

Various algorithms used for implementing handwritten digit recognition systems consist of:

- Proximal SVM
- Multilayer Perceptron
- Support Vector Machine (SVM)
- Random Forest
- Bayes Net
- Naive Bayes

2.3.1 Drawbacks of existing system:

1. Limited Robustness

Handwriting Variability: Struggle to handle diverse handwriting styles, sizes, and orientations effectively. Poor performance on characters written with slanted, uneven, or exaggerated strokes.
Noise Sensitivity: Inability to handle noisy or distorted input images, such as those with smudges or uneven backgrounds.
Overfitting to Specific Datasets: Models often perform well on datasets like MNIST but fail on unseen real-world data.

2. Dataset Limitations-

Dataset Dependency: Reliance on standardized datasets (e.g., MNIST) that may not fully represent real-world handwriting diversity. Lack of Customization:
Difficulty in adapting models to custom or domain-specific datasets without significant retraining.

3. Computational Challenges-

High Resource Requirements: Deep learning models require significant computational resources (GPU/TPU) for training and inference.
Latency in Real-Time Applications: Some systems exhibit slow processing speeds, making real-time recognition challenging.

2.4 Proposed System:

The proposed Handwritten Numerical recognition Using Deep Learning is the process to provide the ability to machines to recognize human handwritten digits. We can easily import the dataset and start working on that because the keras library already contains many datasets and MNIST is one of them. Handwriting numerical recognition has some more general steps. These are pre-processing segmentation, feature extraction, classification and recognition, post – processing.

CHAPTER 3

SYSTEM DESIGN

3.1 System Architecture

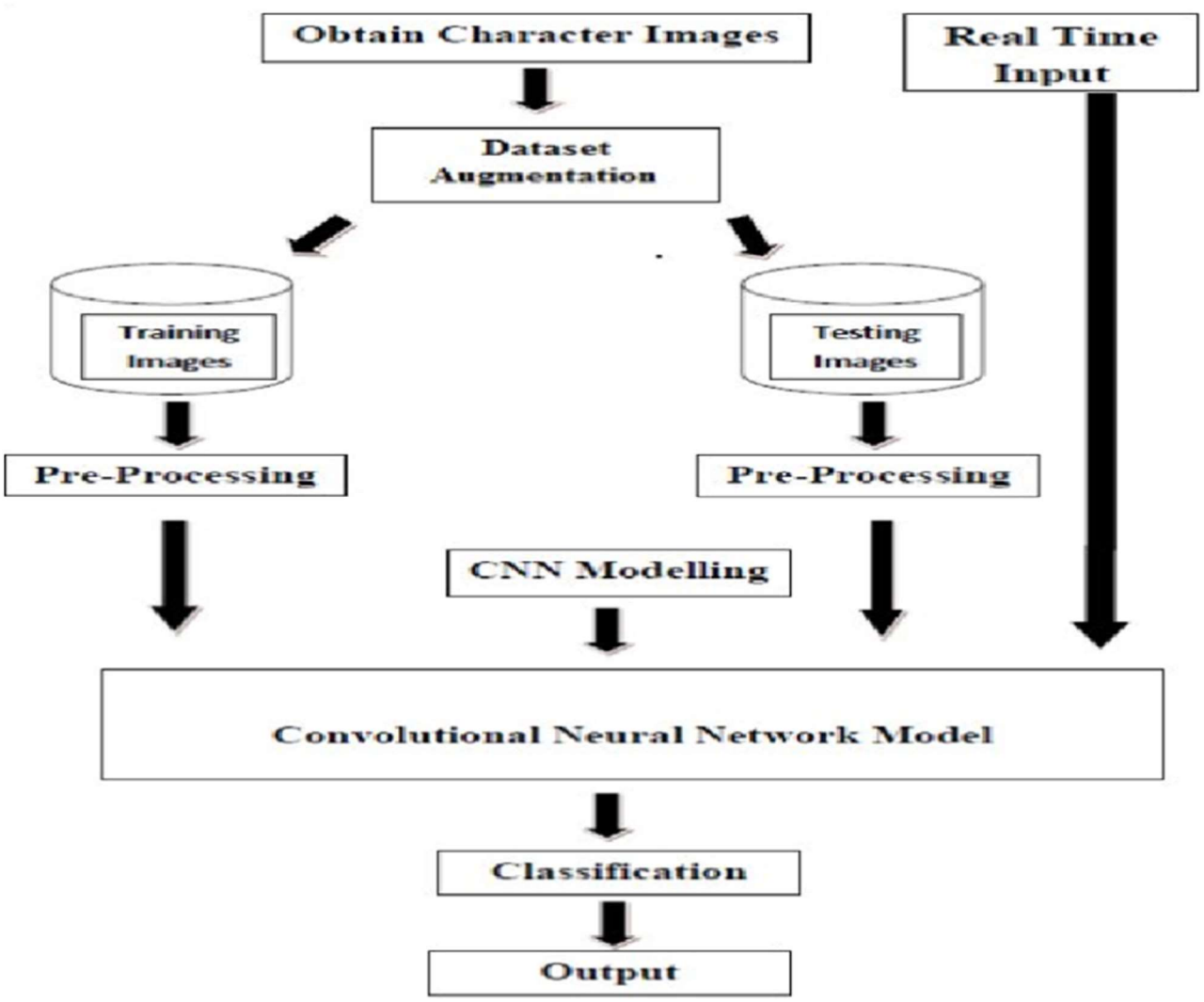

Fig 3.1 System Architecture

3.2 UML Diagrams

3.2.1 Use case diagram

Use Case during requirement elicitation and analysis to represent the functionality of the system. Use case describes a function by the system that yields a visible result for an actor. The identification of actors and use cases result in the definitions of the boundary of the system i.e., differentiating the tasks accomplished by the system and the tasks accomplished by its environment.

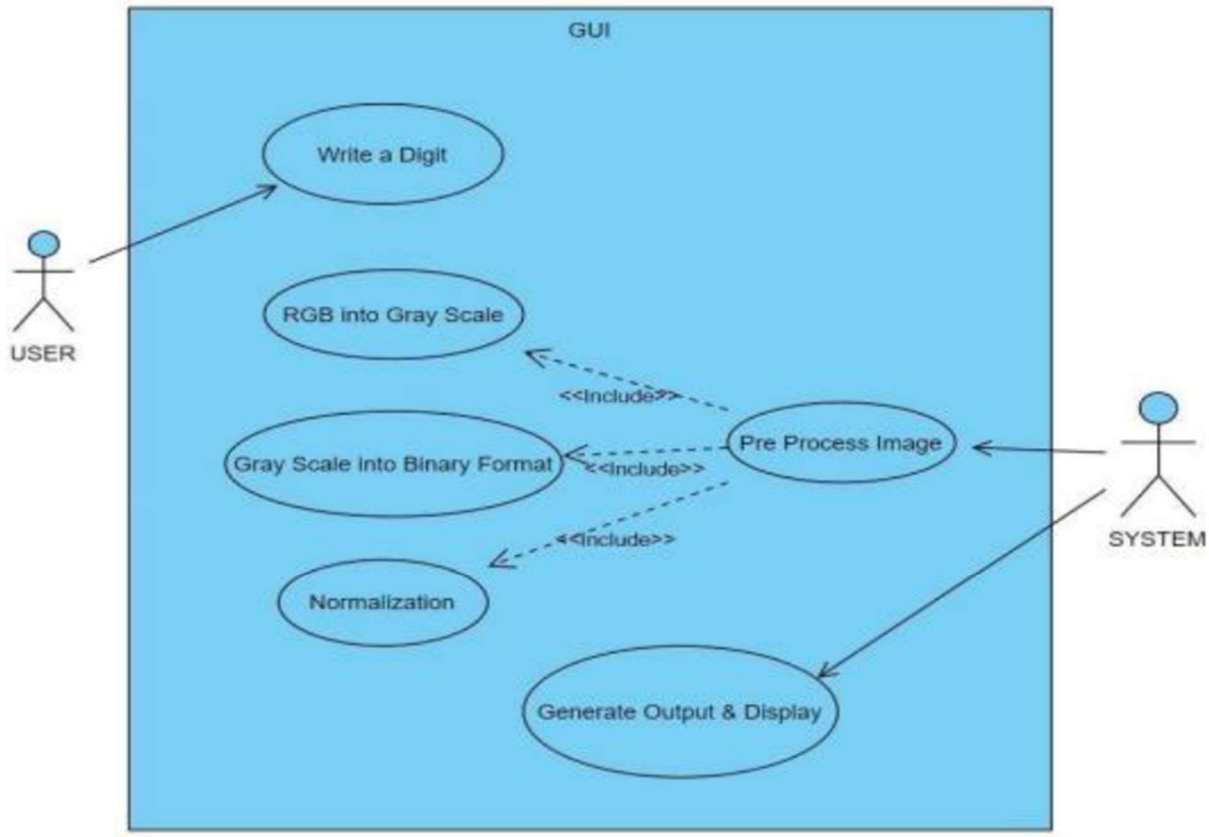

Fig 3.2 Use Case Diagram

3.2.2 Class Diagram

Class diagrams model class structure and contents using design elements such as classes, packages and objects. The class diagram describes the different perspective when designing a system-conceptual, specification and implementation. Classes are composed of three things: name, attributes, and operations. Class diagram also display relationships such as containment, inheritance, association etc. Association relationships are the most common relationship in a class diagram. The association shows the relationship between instances of classes.

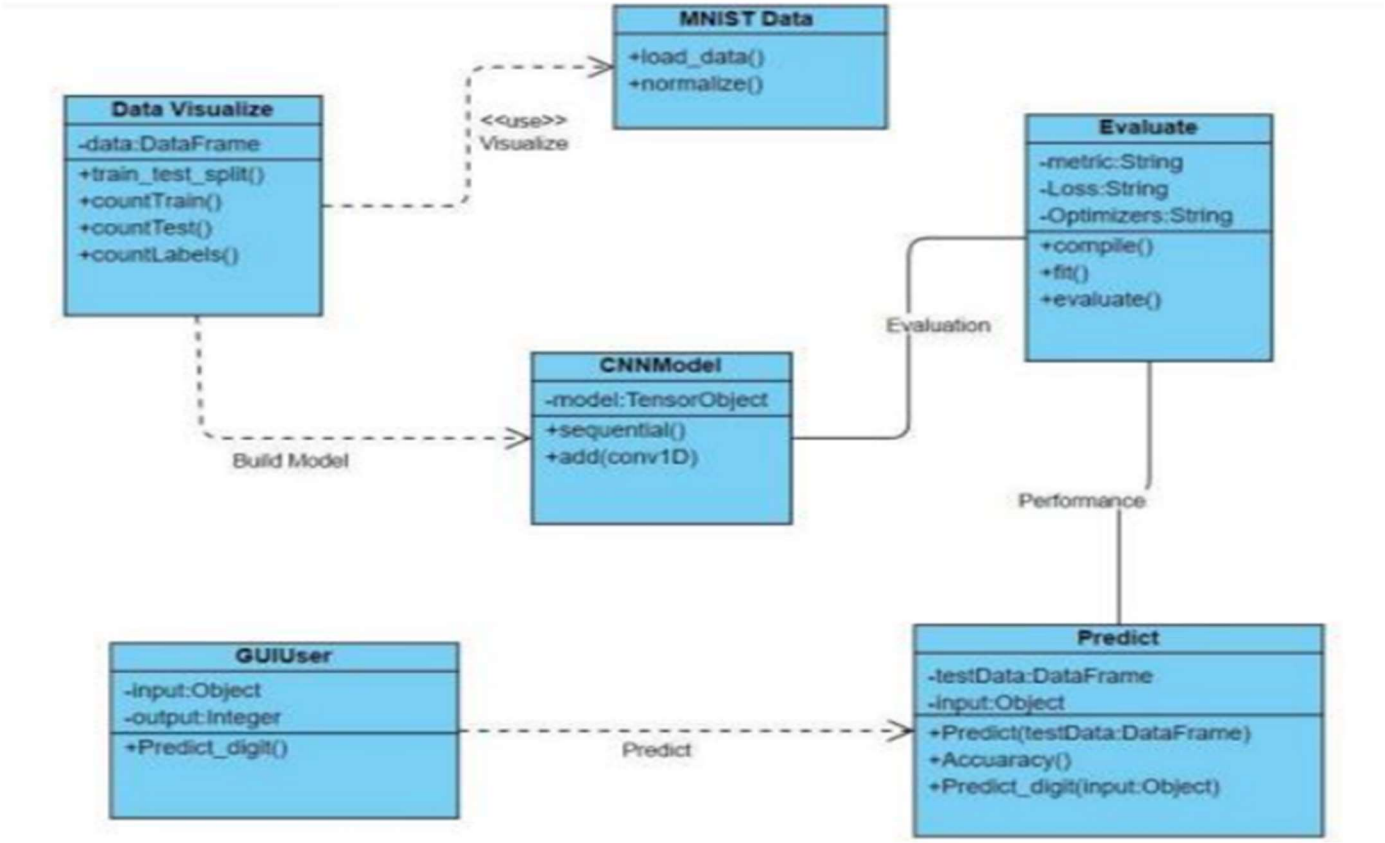

Fig 3.3 Class Diagram

3.2.3 Sequence Diagram

Sequence diagram displays the time sequence of the objects participating in the interaction. This consists of the vertical dimension (time) and horizontal dimension (different objects).

Objects: An object can be thought of as an entity that exists at a specified time and has a definite value, as well as a holder of identity. A sequence diagram depicts item interactions in chronological order. It illustrates the scenario's objects and classes, as well as the sequence of messages sent between them to carry out the scenario's functionality. In the Logical View of the system under development, sequence diagrams are often related with use case realizations. Event diagrams and event scenarios are other names for sequence diagrams. A sequence diagram depicts multiple processes or things that exist simultaneously as parallel vertical lines (lifelines), and the messages passed between them as horizontal arrows, in the order in which they occur. This enables graphical specifications of simple runtime scenarios.

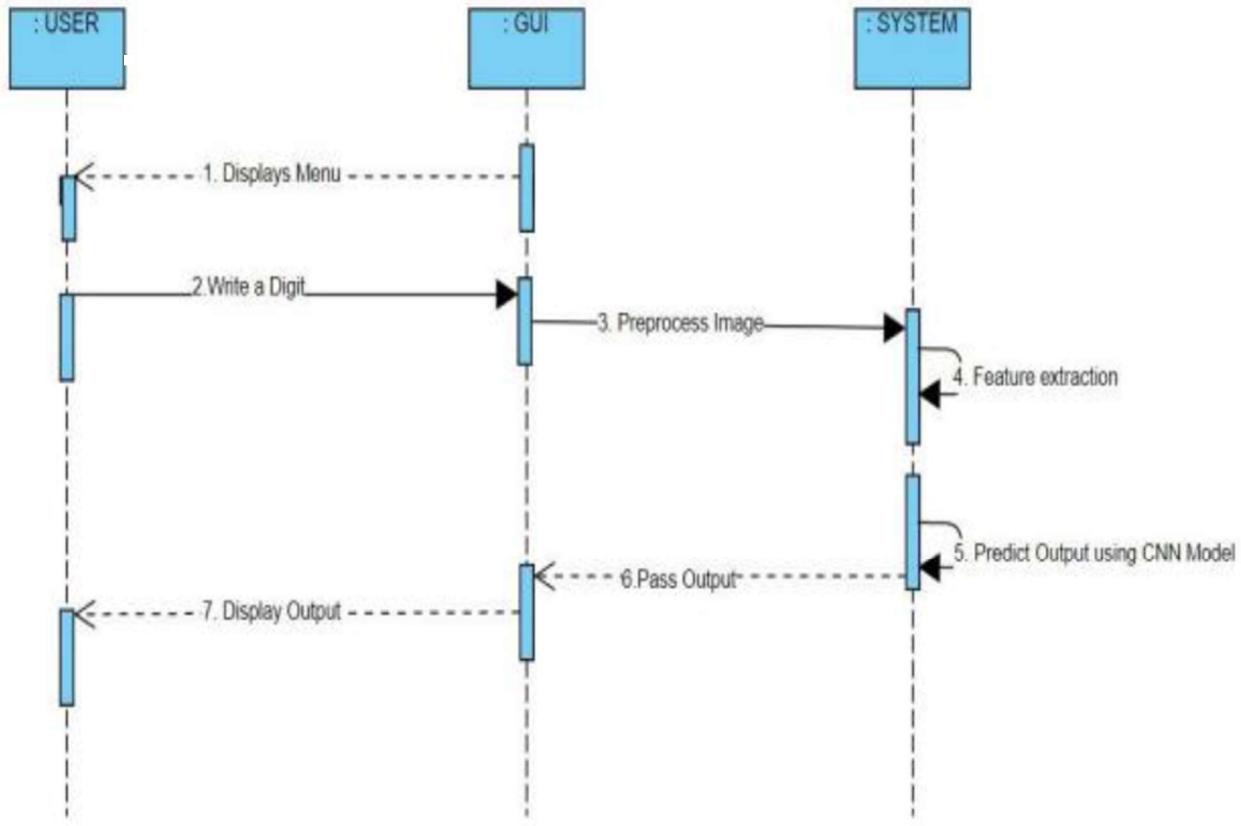

Fig 3.4 Sequence Diagram

3.2.4 Activity Diagram

The process flows in the system are captured in the activity diagram. Similar to a state diagram, an activity diagram also consists of activities, actions, transitions, initial and final states, and guard conditions

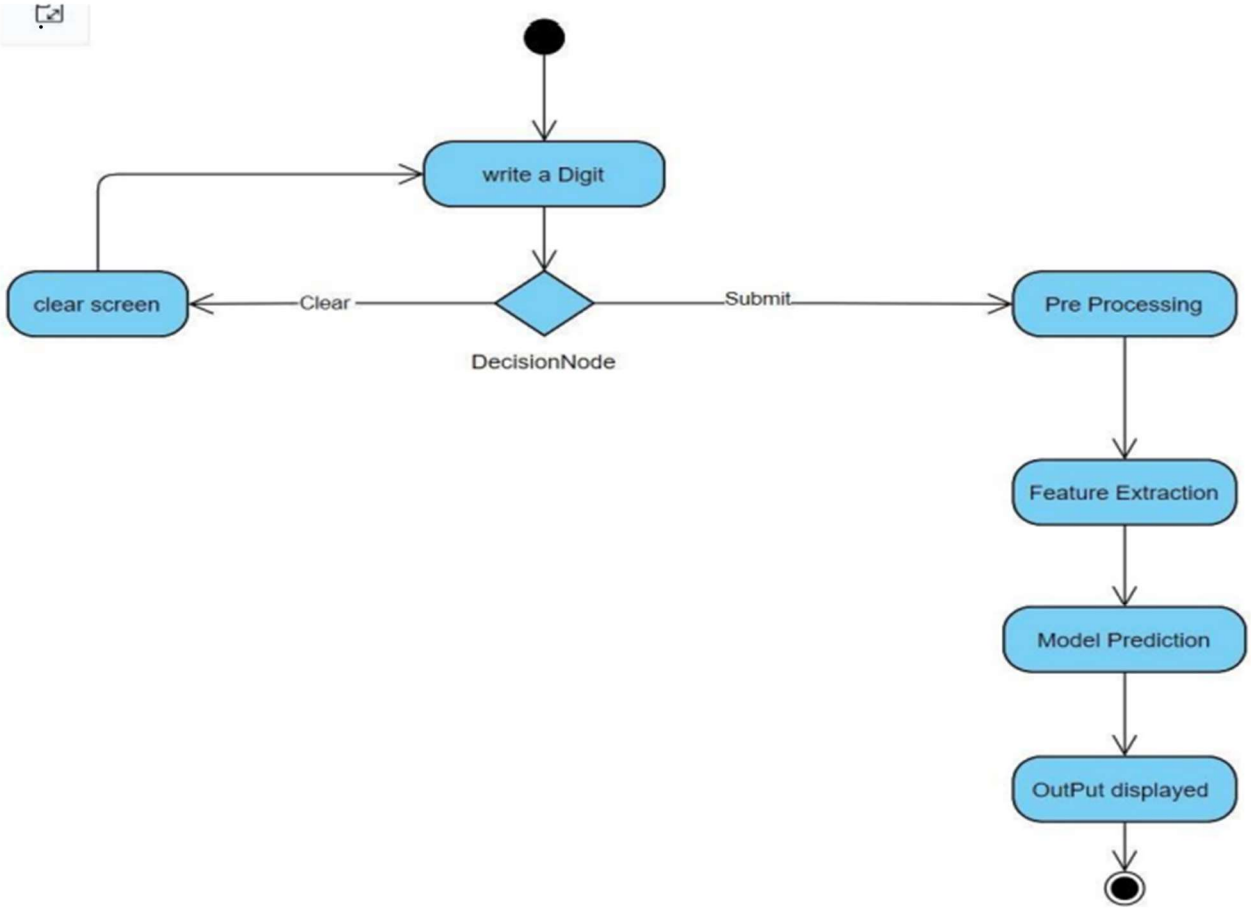

Fig 3.5 Activity Diagram

CHAPTER 4

IMPLEMENTATION

4.1: Code

Main.py

```
from scipy.io import loadmat
import numpy as np
from Model import neural_network
from RandInitialise import initialise
from Prediction import predict

from scipy.optimize import minimize

# Loading mat file
data = loadmat('mnist-original.mat')

# Extracting features from mat file
X = data['data']
X = X.transpose()

# Normalizing the data
X = X / 255

# Extracting labels from mat file
y = data['label']
y = y.flatten()

# Splitting data into training set with 60,000 examples
X_train = X[:60000, :]
y_train = y[:60000]

# Splitting data into testing set with 10,000 examples
X_test = X[60000:, :]
y_test = y[60000:]
```

```
m = X.shape[0]
input_layer_size = 784
hidden_layer_size = 100
num_labels = 10

# Randomly initialise Thetas
initial_Theta1 = initialise(hidden_layer_size, input_layer_size)
initial_Theta2 = initialise(num_labels, hidden_layer_size)

# Unrolling parameters into a single column vector
initial_nn_params = np.concatenate((initial_Theta1.flatten(), initial_Theta2.flatten()))
maxiter = 100
lambda_reg = 0.1  # To avoid overfitting
myargs = (input_layer_size, hidden_layer_size, num_labels, X_train, y_train,
lambda_reg)

# Calling minimize function to minimize cost function and to train weights
results = minimize(neural_network, x0=initial_nn_params, args=myargs,
        options={'disp': True, 'maxiter': maxiter}, method="L-BFGS-B", jac=True)

nn_params = results["x"]  # Trained Theta is extracted

# Weights are split back to Theta1, Theta2
Theta1 = np.reshape(nn_params[:hidden_layer_size * (input_layer_size + 1)], (
                    hidden_layer_size, input_layer_size + 1))  # shape = (100, 785)
Theta2 = np.reshape(nn_params[hidden_layer_size * (input_layer_size + 1):],
                (num_labels, hidden_layer_size + 1))  # shape = (10, 101)

# Checking test set accuracy of our model
pred = predict(Theta1, Theta2, X_test)
print('Test Set Accuracy: {:f}'.format((np.mean(pred == y_test) * 100)))

# Checking train set accuracy of our model
pred = predict(Theta1, Theta2, X_train)
print('Training Set Accuracy: {:f}'.format((np.mean(pred == y_train) * 100)))
```

```
# Evaluating precision of our model
true_positive = 0
for i in range(len(pred)):
    if pred[i] == y_train[i]:
        true_positive += 1
false_positive = len(y_train) - true_positive
print('Precision =', true_positive/(true_positive + false_positive))

# Saving Thetas in .txt file
np.savetxt('Theta1.txt', Theta1, delimiter=' ')
np.savetxt('Theta2.txt', Theta2, delimiter=' ')
```

GUI.py

```
from tkinter import *
import numpy as np
from PIL import ImageGrab
from Prediction import predict

window = Tk()
window.title("Handwritten digit recognition")
l1 = Label()

def MyProject():
    global l1

    widget = cv
    # Setting co-ordinates of canvas
    x = window.winfo_rootx() + widget.winfo_x()
    y = window.winfo_rooty() + widget.winfo_y()
    x1 = x + widget.winfo_width()
    y1 = y + widget.winfo_height()

    img = ImageGrab.grab().crop((x, y, x1, y1)).resize((28, 28))

    # Converting rgb to grayscale image
    img = img.convert('L')
```

```
    x = np.asarray(img)
    vec = np.zeros((1, 784))
    k = 0
    for i in range(28):
        for j in range(28):
            vec[0][k] = x[i][j]
            k += 1

    # Loading Thetas
    Theta1 = np.loadtxt('Theta1.txt')
    Theta2 = np.loadtxt('Theta2.txt')

    # Calling function for prediction
    pred = predict(Theta1, Theta2, vec / 255)

    # Displaying the result
    l1 = Label(window, text="Digit = " + str(pred[0]), font=('Algerian', 20))
    l1.place(x=230, y=420)

lastx, lasty = None, None

# Clears the canvas
def clear_widget():
    global cv, l1
    cv.delete("all")
    l1.destroy()

# Activate canvas
def event_activation(event):
    global lastx, lasty
    cv.bind('<B1-Motion>', draw_lines)
    lastx, lasty = event.x, event.y

# To draw on canvas
def draw_lines(event):
    global lastx, lasty
```

```
    x, y = event.x, event.y
    cv.create_line((lastx, lasty, x, y), width=30, fill='white', capstyle=ROUND,
smooth=TRUE, splinesteps=12)
    lastx, lasty = x, y

# Label
L1 = Label(window, text="Handwritten Digit Recoginition", font=('Algerian', 25),
fg="blue")
L1.place(x=35, y=10)

# Button to clear canvas
b1 = Button(window, text="1. Clear Canvas", font=('Algerian', 15), bg="orange",
fg="black", command=clear_widget)
b1.place(x=120, y=370)

# Button to predict digit drawn on canvas
b2 = Button(window, text="2. Prediction", font=('Algerian', 15), bg="white", fg="red",
command=MyProject)
b2.place(x=320, y=370)

# Setting properties of canvas
cv = Canvas(window, width=350, height=290, bg='black')
cv.place(x=120, y=70)

cv.bind('<Button-1>', event_activation)
window.geometry("600x500")
window.mainloop()
```

Model.py

```
import numpy as np

def neural_network(nn_params, input_layer_size, hidden_layer_size, num_labels, X, y,
lamb):
    # Weights are split back to Theta1, Theta2
    Theta1 = np.reshape(nn_params[:hidden_layer_size * (input_layer_size + 1)],
                        (hidden_layer_size, input_layer_size + 1))
    Theta2 = np.reshape(nn_params[hidden_layer_size * (input_layer_size + 1):],
```

```
                (num_labels, hidden_layer_size + 1))

    # Forward propagation
    m = X.shape[0]
    one_matrix = np.ones((m, 1))
    X = np.append(one_matrix, X, axis=1)  # Adding bias unit to first layer
    a1 = X
    z2 = np.dot(X, Theta1.transpose())
    a2 = 1 / (1 + np.exp(-z2))  # Activation for second layer
    one_matrix = np.ones((m, 1))
    a2 = np.append(one_matrix, a2, axis=1)  # Adding bias unit to hidden layer
    z3 = np.dot(a2, Theta2.transpose())
    a3 = 1 / (1 + np.exp(-z3))  # Activation for third layer

    y_vect = np.zeros((m, 10))
    for i in range(m):
        y_vect[i, int(y[i])] = 1

    # Calculating cost function
    J = (1 / m) * (np.sum(np.sum(-y_vect * np.log(a3) - (1 - y_vect) * np.log(1 - a3)))) +
(lamb / (2 * m)) * (
            sum(sum(pow(Theta1[:, 1:], 2))) + sum(sum(pow(Theta2[:, 1:], 2))))

    # backprop
    Delta3 = a3 - y_vect
    Delta2 = np.dot(Delta3, Theta2) * a2 * (1 - a2)
    Delta2 = Delta2[:, 1:]

    # gradient
    Theta1[:, 0] = 0
    Theta1_grad = (1 / m) * np.dot(Delta2.transpose(), a1) + (lamb / m) * Theta1
    Theta2[:, 0] = 0
    Theta2_grad = (1 / m) * np.dot(Delta3.transpose(), a2) + (lamb / m) * Theta2
    grad = np.concatenate((Theta1_grad.flatten(), Theta2_grad.flatten()))

    return J, grad
```

RandInitialise.py

```
import numpy as np

def initialise(a, b):
    epsilon = 0.15
    c = np.random.rand(a, b + 1) * (
      #Randomly initialise values of thetas between [-epsilon, +epsilon]
      2 * epsilon) - epsilon
    return c
```

Prediction.py

```
import numpy as np

def predict(Theta1, Theta2, X):
    m = X.shape[0]
    one_matrix = np.ones((m, 1))
    X = np.append(one_matrix, X, axis=1)  # Adding bias unit to first layer
    z2 = np.dot(X, Theta1.transpose())
    a2 = 1 / (1 + np.exp(-z2))  # Activation for second layer
    one_matrix = np.ones((m, 1))
    a2 = np.append(one_matrix, a2, axis=1)  # Adding bias unit to hidden layer
    z3 = np.dot(a2, Theta2.transpose())
    a3 = 1 / (1 + np.exp(-z3))  # Activation for third layer
    p = (np.argmax(a3, axis=1))
    return p
```

4.2: Output Screens:

Fig: The model successfully recognizes the handwritten digit '3' and displays it as the prediction result.

Fig: The model successfully recognizes the handwritten digit '5' and displays it as the prediction result.

4.3: Testing

Software testing is a critical element of software quality assurance and represents the ultimate review of specification, design and code generation.

CHAPTER 5
CONCLUSION & FUTURE SCOPE

5.1: Conclusion

The Handwritten Numerical Character Recognition classification accuracy will be tested on the MNIST (Modified National Institute of Standards and Technology) dataset while using training and testing sets. We have built and trained the Convolutional neural network which is very effective for image classification purposes.

5.2: Future Scope

In the future, plan to observe the variation in overall classification accuracy by varying the number of hidden layers and toggling with other parameters of the model.

The model's performance and accuracy were tested after training the model for 50 epochs. With the results given from this work we are more confident in finding other ways to make this better and to make it easier for complex data like converting handwritten paragraphs into text. Plan to make our model more optimized in this direction also make our model recognize the rotated 6 and 9 image(s).

CHAPTER 6

BIBLIOGRAPHY

[1]Arif R. B, Siddique A.B, M. M. R. Khan and Z. Ashrafi, "Study and Observation of the Variations of Accuracies for Handwritten Digits Recognition with Various Hidden Layers and Epochs using Neural Network Algorithm," in 2018 4th International Conference on Electrical Engineering and Information & Communication Technology (iCEEiCT), 2018, pp. 118-123: IEEE.

[2]Ashutosh Aggarwal, Karamjeet Singh, Kamal Preet Singh, ‖Use of Gradient Technique for extracting features from Handwritten Gurmukhi Characters and Numerals‖, 2014, International Conference on Information and Communication Technologies (ICICT 2014).

[3]Akanksha Gaur, Sunita Yadav, ―Handwritten Hindi character Recognition using K-Means Clustering and SVM‖, 978-1-4799- 5532-9/15/$31.00 ©2015 IEEE.

[4]Denker J. S Gardner W. R., Graf, H. P., Henderson, D., Howard, R. E., Hubbard, W., Jackal, L. D., Baird, H. S., and Guyon, I. (1989). Neural Network Recognizer for Hand-Written Digits. In Touretzky, D., editor, Neural Information Processing Systems, volume 1, pages 323-331, Denver, 1988.