BASAVARAJESWARI GROUP OF INSTITUTIONS

**BALLARI INSTITUTE OF TECHNOLOGY & MANAGEMENT**

NBA and NACC Accredited Institution*

**(Recognized by Govt. of Karnataka, approved by AICTE, New Delhi & Affiliated to Visvesvaraya Technological University, Belagavi) "Jnana Gangotri" Campus, No.873/2, Ballari-Hospet Road, Allipur, Ballari – 583104 (Karnataka) (India) Ph:08392–237100/237190, Fax:08392–237197**

**DEPARTMENT OF**

**Computer Science and Engineering (Artificial Intelligence)**

**Neural Network and Deep learning Project Report**

**On**

# "Handwritten Digit Recognition"

***Submitted By***

**Abhilash K R 3BR22CA001**

**Under the Guidance of**

**Prof. Pavan Kumar**

**and Mr. Vijay**

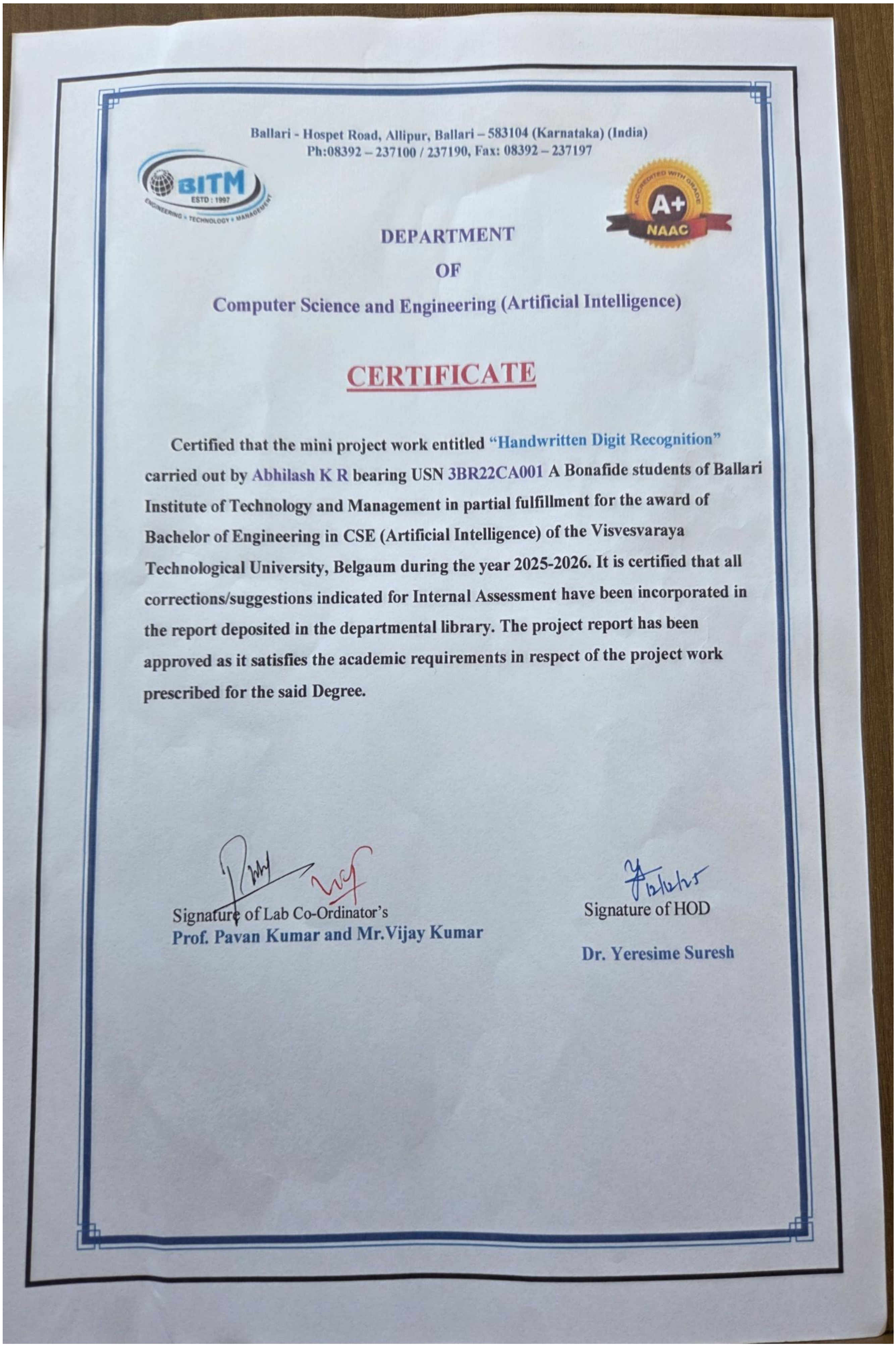
**Kumar**

**Dept of CSE(AI), BITM, Ballari**

**Visvesvaraya Technological University**

**Belagavi, Karnataka**

2024-2025



Ballari - Hospet Road, Allipur, Ballari – 583104 (Karnataka) (India)
Ph:08392 – 237100 / 237190, Fax: 08392 – 237197

**DEPARTMENT**

**OF**

**Computer Science and Engineering (Artificial Intelligence)**

# CERTIFICATE

Certified that the mini project work entitled **"Handwritten Digit Recognition"** carried out by **Abhilash K R** bearing USN **3BR22CA001** A Bonafide students of Ballari Institute of Technology and Management in partial fulfillment for the award of Bachelor of Engineering in CSE (Artificial Intelligence) of the Visvesvaraya Technological University, Belgaum during the year 2025-2026. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the report deposited in the departmental library. The project report has been approved as it satisfies the academic requirements in respect of the project work prescribed for the said Degree.

[illegible] [illegible]

Signature of Lab Co-Ordinator's
**Prof. Pavan Kumar and Mr.Vijay Kumar**

[illegible] 12/12/25

Signature of HOD

**Dr. Yeresime Suresh**

# ABSTRACT

Handwritten Digit Recognition is a classical pattern recognition problem that remains significant in modern automation and digitization systems. This project presents a complete end-to-end handwritten digit recognition model built using Convolutional Neural Networks (CNNs) trained on the MNIST PNG dataset. The system can accurately classify digits (0–9) from both user-drawn inputs and uploaded handwritten images.

The methodology includes dataset preprocessing, augmentation, model training, evaluation, performance visualization, and UI-based deployment using Pygame. Additionally, the system incorporates digit extraction from real images through OpenCV-based thresholding and contour detection. The trained CNN achieves an accuracy of **98–99%**, demonstrating strong generalization and robustness to handwriting variations.

This report covers the entire lifecycle of the project, including objective definition, requirement analysis, model architecture, training strategies, implementation workflow, results, and discussion. Diagrams such as flowchart, use case, sequence diagram, and block diagram illustrate the architecture and workflow clearly.

# ACKNOWLEDGEMENT

The successful completion of this project titled "Handwritten Digit Recognition" would not have been possible without the guidance, support, and encouragement of many individuals. We express our sincere gratitude to our Lab Coordinators for their valuable suggestions, continuous motivation, and knowledgeable guidance throughout the completion of this project.

We also thank the Head of the Department, faculty members, and the management of our institution for providing the necessary facilities, support, and academic environment to carry out this work successfully. Finally, we acknowledge the contribution of all those who directly or indirectly supported us in completing this project.

| Name | USN |
|---|---|
| Abhilash K R | 3BR22CA001 |

# TABLE OF CONTENTS

# LIST OF FIGURES

# CHAPTER 1

# INTRODUCTION

Digit recognition plays a fundamental role in enabling human-machine interaction in various computational systems. With the rapid expansion of automation technologies, machines must interpret handwritten input with the same reliability as typed or printed text. Whether it is reading postal codes, processing bank cheques, validating handwritten applications, or supporting digital pen-based tools, accurate digit classification is essential.

Traditional machine learning techniques rely on handcrafted features, which often fail due to variations in writing patterns, stroke thickness, rotation, background noise, and inconsistent digit shapes. To overcome these limitations, deep learning—specifically Convolutional Neural Networks—provides a more robust solution by learning hierarchical representations directly from raw image data.

This project implements a CNN-based solution capable of recognizing handwritten digits in real-time. Unlike typical MNIST models that only classify centered digits, this system also integrates contour detection to extract digits from full-page images, making it suitable for practical deployments. The final model is embedded into a graphical user interface created using Pygame, enabling users to draw digits, upload images, and receive instant predictions along with confidence scores.

# CHAPTER 2

# OBJECTIVES

**Primary Objectives**

1. To develop a deep-learning-based model capable of recognizing handwritten digits with high accuracy.
2. To design and train a CNN on the MNIST PNG dataset and analyze its performance.
3. To implement a complete user application supporting freehand drawing and image upload for digit recognition.
4. To integrate computer vision techniques for automatic digit extraction from multi-digit images.

**Secondary Objectives**

1. To visualize training performance through accuracy and loss graphs.
2. To evaluate the system using comprehensive metrics including precision, recall, F1-score, and confusion matrices.
3. To ensure the model is scalable, modular, and can be integrated into future real-world applications.
4. To create a user-friendly graphical interface with simple navigation and clear visual feedback.

# CHAPTER 3

## PROBLEM STATEMENT

To build an intelligent system capable of accurately interpreting handwritten digits from diverse sources—including freehand drawing and uploaded photographs—using modern deep learning and computer vision techniques.

# CHAPTER 4

# METHODOLOGY

## 4.1 Dataset Preparation

The MNIST PNG dataset includes:

- Training Set → 60,000 PNG images
- Testing Set → 10,000 PNG images

Each digit is saved as a separate 28×28 grayscale PNG file. Images are loaded using a Keras Directory Generator, which simplifies class labeling.

## 4.2 Image Preprocessing

Preprocessing includes:

- Normalization (pixel values scaled to 0–1)
- Grayscale loading
- Resizing to 28×28
- Augmentation for robustness:
  - rotation (±10°)
  - width/height shift
  - zoom
  - shear

## 4.3 CNN Model Architecture

The CNN includes:

- Two convolutional blocks with Batch Normalization
- MaxPooling after each block
- Dropout layers (0.25 and 0.30) to prevent overfitting
- Flatten layer
- Dense(128) + Batch Normalization
- Softmax classifier for 10 output classes

## 4.4 Training Strategy

- Optimizer: Adam (learning rate=0.001)
- Loss: Categorical Crossentropy
- Training Epochs: 10 (full execution, no early-stop)
- Callbacks:
    - ModelCheckpoint (saves best model)
    - ReduceLROnPlateau (helps model converge)

## 4.5 Prediction on Uploaded Images

Uploaded images undergo:

- Grayscale conversion
- Inversion if background is white
- Gaussian blur + OTSU thresholding
- Contour detection
- Bounding box extraction
- Digit extraction & centering
- CNN prediction

## 4.6 UI Development (Pygame)

The application provides:

- Drawing board screen
- Upload mode screen
- Result display with confidence scores
- Keyboard shortcuts (Clear, Save, Quit)

# CHAPTER 5

# REQUIREMENT ANALYSIS

## 5.1 Functional Requirements

1. The system must train a CNN model using the MNIST PNG dataset.
2. It should recognize digits drawn on the screen as well as digits extracted from uploaded images.
3. The system must detect multiple digits using contour detection and display predictions with confidence scores.
4. The application should allow users to draw, upload images, and navigate between screens.
5. It should generate evaluation metrics such as accuracy, confusion matrix, precision–recall curves, and classification reports.

## 5.2 Non-Functional Requirements

1. Accuracy: The model must achieve at least 98% test accuracy.
2. Performance: Predictions should occur in real-time.
3. Usability: Interface must be simple and intuitive with minimal user steps.
4. Reliability: The system must handle various handwriting styles and image backgrounds.
5. Maintainability: Code should be modular and easy to update or extend.

## 5.3 Hardware Requirements

1. A computer with at least Intel/AMD dual-core processor
2. 8 GB RAM

3. 1–2 GB free storage for dataset and libraries

## 5.4 Software Requirements

1. Python 3.10
2. TensorFlow, Keras, NumPy, Matplotlib
3. OpenCV for image preprocessing
4. Pygame for graphical interface
5. PlantUML for diagrams (optional)

# CHAPTER 6

# DESIGN

## FLOWCHART

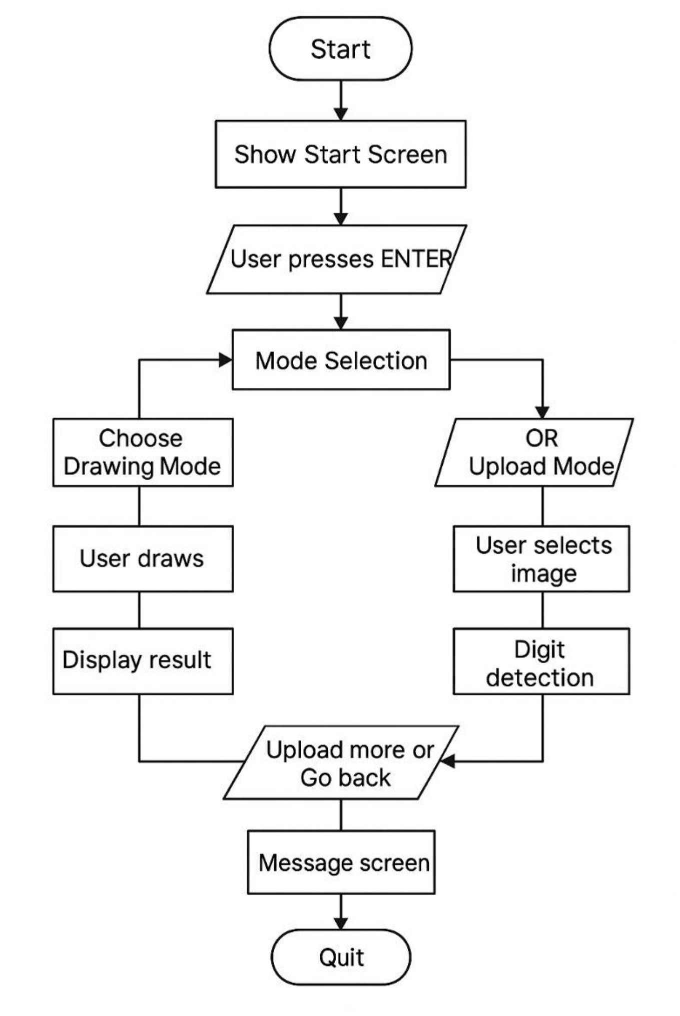


**Fig 6.1 Flow Chart**

## USE CASE DIAGRAM

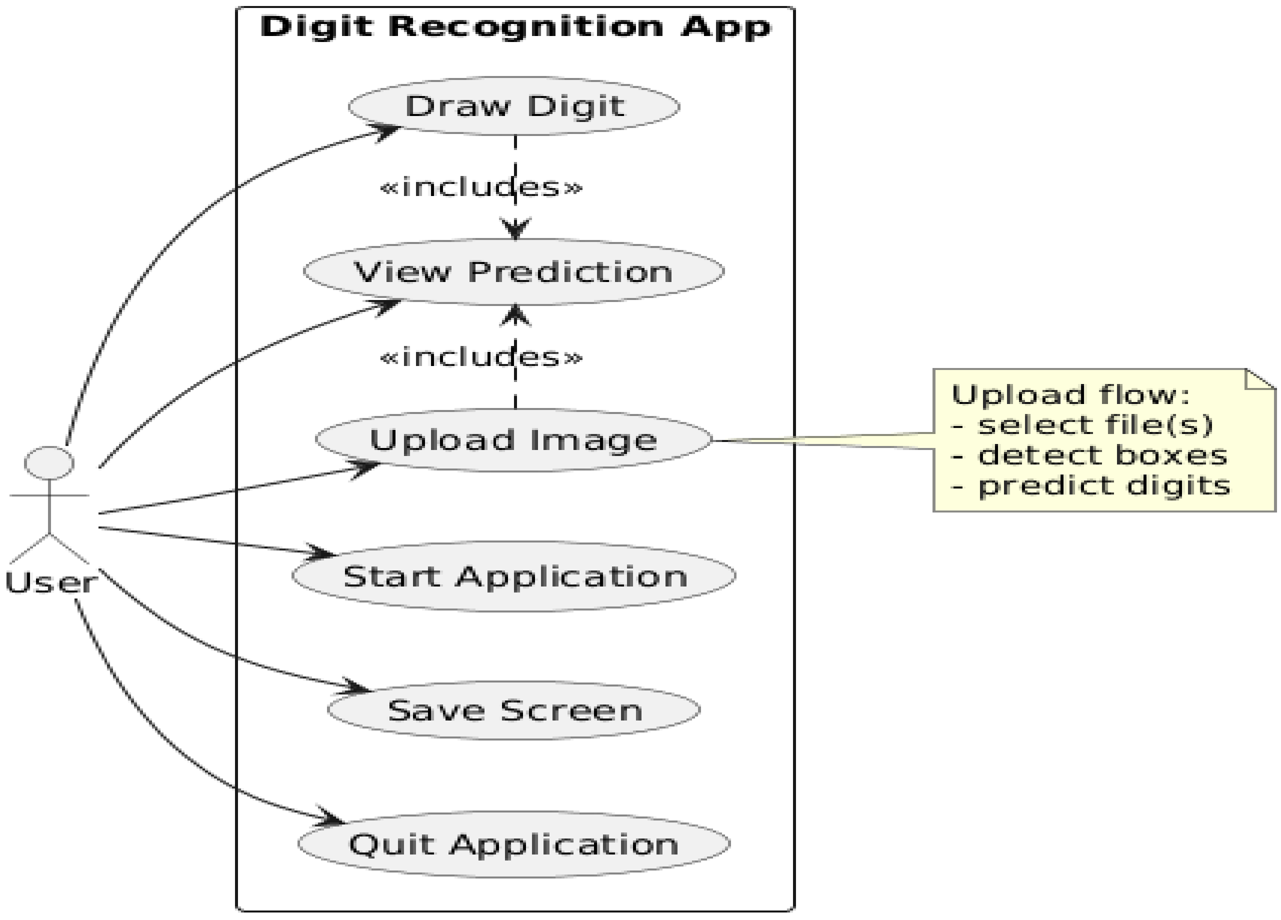


**Fig 6.2 Use Case Diagram**

## SEQUENCE DIAGRAM

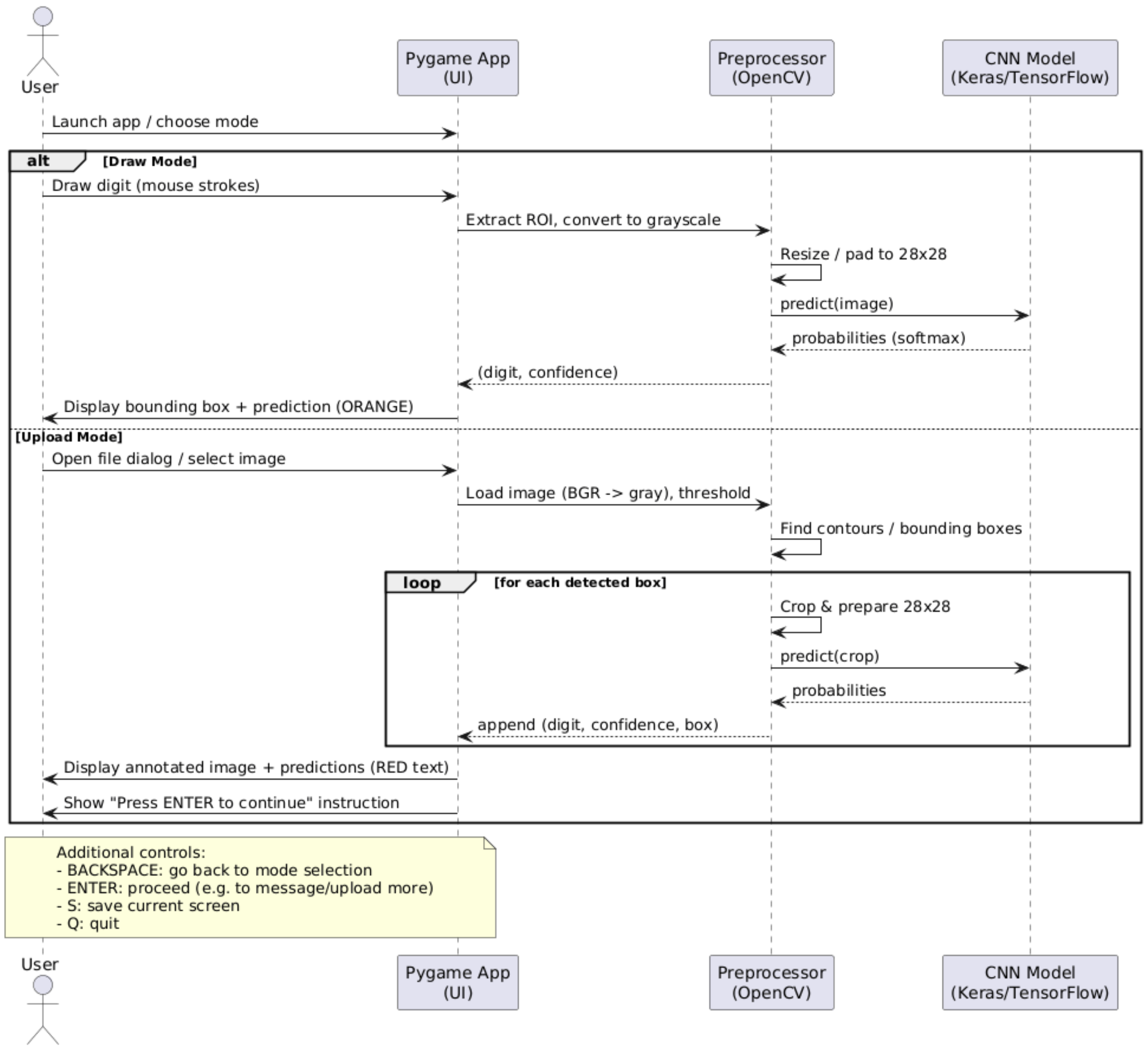

User
Pygame App (UI)
Preprocessor (OpenCV)
CNN Model (Keras/TensorFlow)
Launch app / choose mode
alt [Draw Mode]
Draw digit (mouse strokes)
Extract ROI, convert to grayscale
Resize / pad to 28x28
predict(image)
probabilities (softmax)
(digit, confidence)
Display bounding box + prediction (ORANGE)
[Upload Mode]
Open file dialog / select image
Load image (BGR -> gray), threshold
Find contours / bounding boxes
loop [for each detected box]
Crop & prepare 28x28
predict(crop)
probabilities
append (digit, confidence, box)
Display annotated image + predictions (RED text)
Show "Press ENTER to continue" instruction
Additional controls:
- BACKSPACE: go back to mode selection
- ENTER: proceed (e.g. to message/upload more)
- S: save current screen
- Q: quit
User
Pygame App (UI)
Preprocessor (OpenCV)
CNN Model (Keras/TensorFlow)

**Fig 6.3 Sequence Diagram**

.

# CHAPTER 7

# IMPLEMENTATION

## 10.1 Data Loading Module

- Uses Image Data Generator to load and augment images
- Automatically assigns labels based on folder names

## 10.2 Model Training Module

- Builds CNN architecture
- Trains using training dataset
- Validates using test dataset
- Saves the best-performing model

## 10.3 Evaluation Module

- Computes accuracy, precision, recall, F1-score
- Generates confusion matrix for both train & test
- Produces precision–recall curves

## 10.4 Image Preprocessing Module

Uses OpenCV to:

- Convert input images to grayscale
- Perform thresholding
- Detect contours
- Extract digit ROIs
- Center digits in a 28×28 canvas

### 10.5 Pygame Application Module

Provides:

- Start screen
- Mode selection
- Drawing interface
- Upload interface
- Screen saving capabilities

### 10.6 Prediction Module

Applies CNN model to:

- Freehand drawings
- Extracted digit ROIs
- Displays class + confidence %

# CHAPTER 8

# RESULTS AND DISCUSSION

## 11.1 Model Performance

1. Test Accuracy: **98–99%**
2. Low training loss, stable validation loss
3. Balanced precision and recall across all digits
4. Confusion matrix shows minimal misclassification

## 11.2 Observations

1. Digits with similar shapes (e.g., 9 vs 4, 3 vs 8) sometimes produce lower confidence.
2. Handwritten input drawn using a mouse introduces noise but still yields reliable predictions.
3. Uploaded images with uneven lighting still perform well due to robust preprocessing.

## 11.3 Strengths

1. High accuracy
2. Real-time inference
3. Effective image preprocessing
4. User-friendly interface

## 11.4 Limitations

1. Sensitive to extremely noisy backgrounds
2. Performance depends on contour detection quality
3. Predictions may drop for highly stylized handwriting

## 11.5 Future Improvements

1. Expand dataset with more diverse handwriting samples
2. Implement digit sequence reading (e.g., full numbers)
3. Deploy model on mobile or web platforms
4. Add support for alphabets (A–Z)

# CHAPTER 9

# CONCLUSION

This project demonstrates a complete, working handwritten digit recognition system combining deep learning, computer vision, and interactive UI development. The designed CNN model performs exceptionally well, achieving nearly 99% accuracy and strong performance on handwritten inputs.

The integration of preprocessing, contour detection, and real-time prediction forms a comprehensive system suitable for real-world digit recognition tasks. Through structured methodology and detailed analysis, the project fulfills its objectives and provides a scalable foundation for future extensions.

# CHAPTER 10

## REFERENCES

1. LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). *Gradient-Based Learning Applied to Document Recognition*. Proceedings of the IEEE.
2. MNIST Dataset: http://yann.lecun.com/exdb/mnist/
3. TensorFlow Documentation: https://www.tensorflow.org/
4. Keras API Reference: https://keras.io/api/
5. OpenCV Documentation: https://docs.opencv.org/
6. Pygame Documentation: https://www.pygame.org/wiki/GettingStarted
7. Bishop, C. (2006). *Pattern Recognition and Machine Learning*.
8. Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press.