BASAVARAJESWARI GROUP OF INSTITUTIONS

BALLARI INSTITUTE OF TECHNOLOGY & MANAGEMENT

NACC Accredited Institution*

(Recognized by Govt. of Karnataka, approved by AICTE, New Delhi & Affiliated to Visvesvaraya Technological University, Belagavi)

"JnanaGangotri" Campus, No.873/2, Ballari-Hospet Road, Allipur, Ballar1-583 104 (Karnataka) (India)

Ph: 08392 – 237100 / 237190, Fax: 08392 – 237197

DEPARTMENT OF CSE-DATA SCIENCE

A Mini-Project Report On

"HANDWRITTEN DIGIT RECOGNITION USING CNN MODEL"

A report submitted in partial fulfillment of the requirements for the

NEURAL NETWORK AND DEEP LEARNING

Submitted By

S VARSHINI BHAT **USN: 3BR22CD055**

Under the Guidance of

Mr. Azhar Biag

Asst. Professor

Dept of CSE (DATA SCIENCE),

BITM, Ballari

Visvesvaraya Technological University

Belagavi, Karnataka 2025-2026

BASAVARAJESWARI GROUP OF INSTITUTIONS

BALLARI INSTITUTE OF TECHNOLOGY & MANAGEMENT

NAAC Accredited Institution*
(Recognized by Govt. of Karnataka, approved by AICTE, New Delhi &
Affiliated to Visvesvaraya Technological University, Belagavi)
"JnanaGangotri" Campus, No.873/2, Ballari-
Hospet Road, Allipur, Ballari-583 104 (Karnataka)
(India)
Ph: 08392 – 237100 / 237190, Fax: 08392 – 237197

DEPARTMENT OF CSE-DATA SCIENCE

CERTIFICATE

This is to certify that the Mini Project of NEURAL NETWORK AND DEEP LEARNING title "HANDWRITTEN DIGIT RECOGNITION USING CNN MODEL" has been successfully presented by S VARSHINI BHAT 3BR22CD055 student of semester B.E for the partial fulfillment of the requirements for the award of Bachelor Degree in CSE(DS) of the BALLARI INSTITUTE OF TECHNOLOGY& MANAGEMENT, BALLARI during the academic year 2025-2026. It is certified that all corrections and suggestions indicated for internal assessment have been incorporated in the report deposited in the library. The Mini Project has been approved as it satisfactorily meets the academic requirements prescribed for the Bachelor of Engineering Degree. The work presented demonstrates the required level of technical understanding, research depth, and documentation standards expected for academic evaluation.

Signature of Coordinators

Mr. Azhar Baig

Ms. Chaithra B M

Signature of HOD

Dr. Aradhana D

ABSTRACT

Handwritten digit recognition is one of the most widely explored applications in the field of image classification and deep learning. It serves as a benchmark problem for evaluating the performance of neural network models due to the availability of large and well-structured datasets like MNIST. The objective of this project is to design and implement a Convolutional Neural Network (CNN) capable of accurately classifying digits (0–9) from 28×28 grayscale images.

The CNN model used in this work consists of convolutional layers, max-pooling layers, flattening, and dense layers with ReLU and softmax activations. The MNIST dataset, containing 60,000 training and 10,000 testing samples, is preprocessed by normalizing pixel intensities and reshaping into appropriate tensor formats.

After training, the model achieves high accuracy and is deployed using a Streamlit-based user interface. Users can draw digits on a canvas, and the system predicts the corresponding digit using the trained CNN. The project demonstrates the power of deep learning in computer vision tasks and showcases its potential for real-world applications such as postal code recognition, bank cheque processing, and automated digitization systems.

ACKNOWLEDGEMENT

The satisfactions that accompany the successful completion of our mini project on HANDWRITTEN DIGIT RECOGNITION USING CNN MODEL would be incomplete without the mention of people who made it possible, whose noble gesture, affection, guidance, encouragement and support crowned my efforts with success. It is our privilege to express our gratitude and respect to all those who inspired us in the completion of our mini-project.

I am extremely grateful to my Guide **Mr. Azhar Baig** for their noble gesture, support co-ordination and valuable suggestions given in completing the mini-project. I also thank **Dr. Aradhana D,** H.O.D. Department of CSE(DS), for his co-ordination and valuable suggestions given in completing the mini-project. We also thank Principal, Management and non-teaching staff for their co-ordination and valuable suggestions given to us in completing the Mini project.

Name	USN
S VARSHINI BHAT	3BR22CD055

TABLE OF CONTENTS

Ch No	Chapter Name	Page
I	Abstract	I
1	Introduction 1.1 Project Statement 1.2 Scope of the project 1.3 Objectives	1-3
2	Literature Survey	4
3	System requirements 3.1 Hardware Requirements 3.2 Software Requirements 3.3 Functional Requirements 3.4 Non-Functional Requirements	5-6
4	Description of Modules	7-8
5	Implementation	9
6	System Architecture	10-13
7	Code Implementation	14-15
8	Result	16-17
9	Conclusion	18
10	References	19

1. INTRODUCTION

Handwritten digit recognition is one of the most influential and widely studied problems in the field of computer vision, pattern recognition, and deep learning. Over the decades, the need for automated systems that can accurately interpret handwritten data has grown dramatically. This demand is driven by sectors such as banking, postal services, finance, government documentation, examination evaluation systems, and modern digital transformation initiatives. With large volumes of handwritten data generated daily, manual data entry becomes inefficient, time-consuming, and prone to errors, ultimately making automation not only desirable but essential.

Handwritten digits vary significantly between individuals due to differences in writing styles, stroke thickness, alignment, pressure, cultural habits, and even emotional state. Factors such as writing tools, paper texture, lighting conditions, and scanning quality further increase variability. Because of these challenges, traditional machine learning algorithms often fail to achieve high accuracy unless extensive feature engineering is performed. Manual feature extraction methods — like edge detection, zoning, histogram of oriented gradients (HOG), and template matching — require domain expertise, are sensitive to noise, and do not generalize well.

This problem paved the way for neural networks and, in particular, **Convolutional Neural Networks (CNNs)**, which have revolutionized image-processing tasks. CNNs automatically learn hierarchical features from images, eliminating the need for handcrafted features. Early layers identify basic patterns like edges and gradients, while deeper layers extract complex shapes like curves, loops, and digit-like structures. This ability makes CNNs extremely powerful for handwritten digit recognition.

One of the major breakthroughs in this field came with the introduction of the **MNIST dataset**, created by Yann LeCun. MNIST is considered the “Hello World” of deep learning because it provides a simple yet challenging environment to test neural network architectures. The dataset consists of 70,000 grayscale images of handwritten digits, each of size 28×28 pixels. Despite its simplicity, the variations in handwriting make it a perfect benchmark to evaluate the robustness of recognition models.

Thus, the handwritten digit recognition system developed in this mini-project represents a complete deep-learning pipeline from dataset preprocessing to training, evaluation, and deployment. The accuracy achieved by the model, combined with the intuitive Streamlit interface, makes this project technically impactful and practically useful. It also lays a strong foundation for more complex tasks such as multi-digit recognition, handwritten character recognition, and document digitization.

1.1 Problem Statement

Handwritten digit recognition is a challenging task due to the wide variations in human handwriting, where digits differ in shape, size, stroke style, and writing patterns from person to person. Manual interpretation of handwritten digits in applications such as bank cheque processing, postal code reading, form digitization, and document automation is slow and error-prone, highlighting the need for a reliable automated system. Traditional machine-learning methods struggle with such variability because they depend on manually engineered features, which fail to generalize across diverse handwriting styles. Therefore, the problem addressed in this project is to develop an accurate, robust, and efficient system that can automatically recognize handwritten digits using a Convolutional Neural Network (CNN). The model should learn directly from pixel data, classify digits from 0 to 9, and perform consistently across different writing styles. Additionally, the system must be user-friendly and capable of real-time prediction, which is achieved by integrating the trained model with a Streamlit-based interface that allows users to draw digits for instant recognition.

1.2 Scope of the project

The scope of this project includes designing, training, and deploying a Convolutional Neural Network (CNN) model capable of accurately recognizing handwritten digits from images. It covers all major stages of a deep-learning pipeline, including dataset preprocessing, model building, training, testing, and performance evaluation using accuracy and loss metrics. The project also extends to developing a Streamlit-based user interface that allows users to draw digits and obtain real-time predictions. While the system is limited to the MNIST dataset and single-digit recognition, the methodology can be expanded to larger datasets, multi-digit recognition, and advanced computer vision applications in real-world document processing.

1.3 Objectives

- To develop a CNN-based model capable of accurately recognizing handwritten digits.
- To preprocess and normalize the MNIST dataset for efficient model training.
- To train and evaluate the model using accuracy, loss, and visualization metrics.
- To deploy the trained model through a Streamlit interface for real-time prediction.
- To create a user-friendly system that can classify digits from 0 to 9 reliably.

2. LITERATURE SURVEY

[1] **Review: *Handwritten recognition techniques — comprehensive survey* (MDPI, 2024)** A thorough review of character-recognition methods covering classical feature-based approaches, CNNs, hybrid methods, preprocessing steps and open datasets; highlights the effectiveness of CNNs and the importance of preprocessing and augmentation. Good for your literature section to justify using CNNs and to cite recommended preprocessing/augmentation practices.

[2] **Lightweight CNNs for image recognition (ScienceDirect review, 2024)** Survey and evaluation of lightweight CNN architectures and strategies (depthwise separable convs, pruning, quantization) for resource-constrained devices. Reports that carefully designed small CNNs can reach near-state-of-the-art accuracy with far fewer parameters. Use this to motivate a small/efficient CNN for fast predictions in a Streamlit app or on weak hardware (no huge models).

[3] **"A Novel Technique for Handwritten Digit Recognition" (Wiley, Jan 2023)** Proposes architectural tweaks and preprocessing that yield very high MNIST accuracy (authors report >99% on MNIST) and discusses design choices that reduce overfitting. Good concrete example to cite when you discuss architecture choices (layers, activations, regularization) and expected accuracy ranges.

[4] **Upgraded LeNet/CNN variants on MNIST (R. Wang et al., 2023)** Demonstrates improved LeNet-style architectures and training recipes for MNIST; includes experiments on layer sizes, learning rates and data normalization showing consistent gains. Useful for your ablation/discussion section — shows even small changes to classic LeNet can improve performance.

[5] **Practical deployment & demo guides (Streamlit / deployment tutorials, 2023–2024)** Hands-on guides and short papers/documentation that explain packaging trained models and exposing them via Streamlit UI (handling input preprocessing in the app, latency considerations, and lightweight model usage). Helps justify your choice of Streamlit for the front end and gives practical tips for productionizing the App.py workflow (canvas → preprocess → model.predict → return label).

3. SYSTEM REQUIREMENTS

The successful development of a handwritten digit recognition system using Convolutional Neural Networks (CNNs) relies heavily on a well-defined set of system requirements that support efficient model training, evaluation, and deployment. Since deep-learning models involve mathematically intensive operations such as convolutions, feature extraction, backpropagation, and image preprocessing, the system must provide the necessary computational power and software support to handle these tasks smoothly. A suitable combination of hardware and software ensures that the model can process large amounts of image data, learn meaningful patterns, and deliver accurate predictions without performance bottlenecks. Furthermore, because this project integrates a real-time user interface through Streamlit—allowing users to draw digits and receive instant predictions—the system must also be capable of rendering interactive components and executing the trained model with minimal latency. Therefore, defining clear system requirements is essential to guarantee that all stages of the project—from dataset handling and CNN model construction to training, testing, and user-side deployment—operate efficiently, reliably, and in alignment with the intended project objectives.

3.1 Software Requirements

- **Operating System:** Windows / Linux / macOS
- **Programming Language:** Python 3.8 or above
- **Libraries & Frameworks:**

  TensorFlow / Keras (for CNN model development)

  NumPy, Pandas (for data handling and preprocessing)

  Matplotlib / Seaborn (for visualization)

  Streamlit & streamlit-drawable-canvas (for GUI deployment)

  PIL (for image processing)

- **Development Environment:**

  VS Code / PyCharm / Jupyter Notebook

3.2 Hardware Requirements

- **Processor:** Minimum Dual-Core CPU (Intel i3 or equivalent)
- **Memory (RAM):** Minimum 4 GB (8 GB recommended for faster training)
- **Storage:** At least 1 GB free space for dataset, model files, and logs
- **GPU:** Optional; NVIDIA GPU recommended for faster CNN training
- **Input Device:** Mouse/Touchpad (for drawing digits in Streamlit UI)

3.3 Functional Requirements

- The system must load and preprocess the MNIST dataset.
- It should train a CNN model capable of classifying digits from 0 to 9.
- The system must evaluate model performance using accuracy and loss metrics.
- It must support a graphical interface for drawing handwritten digits.
- The model must predict the digit drawn by the user in real time.

3.4 Non-Functional Requirements

- **Accuracy:** The model should provide high and reliable prediction accuracy.
- **Usability:** The interface should be simple, intuitive, and easy for users to interact with.
- **Performance:** Predictions should be processed quickly without noticeable delay.
- **Scalability:** The system should allow further extension (multi-digit recognition or larger datasets).
- **Maintainability:** The codebase should be modular and easy to update or retrain

4. DESCRIPTION OF MODULES

The handwritten digit recognition system is divided into several modules, each responsible for a specific stage of the deep-learning workflow. These modules work together to ensure efficient preprocessing, model training, prediction, and deployment. The modular design also improves clarity, maintainability, and scalability of the project.

4.1 Data Preprocessing Module

This module handles the preparation of input data before it is fed into the CNN model. The MNIST dataset is loaded and converted into the appropriate numerical format. Images are normalized by scaling pixel values between 0 and 1 to improve training stability. The data is reshaped into the format required by CNNs: (28, 28, 1) to indicate grayscale images. The dataset is then split into training and testing sets to evaluate the model's ability to generalize. This module ensures clean, standardized, and structured input for effective model learning.

4.2 CNN Model Construction Module

This module defines and builds the architecture of the Convolutional Neural Network used for digit recognition. The design includes convolutional layers for feature extraction, max-pooling layers for dimensionality reduction, and dense layers for classification. Activation functions such as ReLU and Softmax are applied to introduce non-linearity and produce probability outputs. The model is compiled using the Adam optimizer and categorical cross-entropy loss. This module forms the core intelligence of the system by learning meaningful patterns from image data.

4.3 Model Training Module

After the model is constructed, this module trains the CNN using the preprocessed dataset. During training, the model learns weights and filters that help identify features like curves, edges, and shapes associated with different digits. Training is performed for a fixed number of epochs, with accuracy and loss recorded at each step. A portion of the training data is used as validation data to monitor performance

and detect overfitting. This module transforms the raw CNN architecture into a functional classification system.

4.4 Model Evaluation Module

This module evaluates the model's performance on unseen test data. Metrics such as accuracy, loss, confusion matrix, and classification reports are computed to measure how well the model recognizes digits. The confusion matrix helps analyze correctly predicted classes and misclassified samples. Visualizations such as accuracy and loss graphs provide insights into training behavior. This module ensures the model meets expected performance standards before deployment.

4.5 Prediction Module

The prediction module uses the trained model to classify new handwritten digits. When a user draws a digit on the Streamlit canvas, the image is captured, converted to grayscale, resized to 28×28 pixels, normalized, and passed to the CNN model. The system outputs the predicted digit based on the highest probability. This module brings the model into real-world use by enabling instantaneous digit recognition.

4.6 Data Splitting Module

This module provides an interactive interface for users. Using Streamlit and the drawable canvas component, users can draw digits directly on the screen. Buttons such as "Predict Now" initiate the prediction process. This module enhances the usability and accessibility of the system, making the project not only functional but also user-friendly.

5. IMPLEMENTATION

The implementation of the handwritten digit recognition system is carried out through a sequence of well-defined steps that transform raw image data into accurate predictions using a Convolutional Neural Network (CNN). The entire workflow integrates dataset preprocessing, model development, training, evaluation, and deployment through a graphical interface. Each stage is implemented using Python, TensorFlow/Keras, NumPy, and Streamlit to ensure efficiency, clarity, and real-time usability.

The first step involves **loading and preprocessing the MNIST dataset**, which contains 60,000 training images and 10,000 testing images of handwritten digits. Each image is converted from its original 28×28 grayscale format into normalized arrays by dividing pixel values by 255. This normalization ensures faster convergence and stable model training. The dataset is then reshaped into (28,28,1) format to match the input requirements of CNNs, and split into training and testing sets.

Next, the **CNN model is constructed** using Keras' Sequential API. The architecture includes two convolutional layers with ReLU activation to extract important visual features, followed by max-pooling layers that reduce spatial dimensions and computational load. The extracted features are flattened and passed through dense layers, culminating in a Softmax output layer that classifies digits from 0 to 9. The model is compiled using the Adam optimizer and categorical cross-entropy loss, ideal for multi-class classification.

The **model training phase** involves feeding the training data into the CNN for several epochs. During training, the model continuously adjusts its weights to minimize loss and improve accuracy. A validation split is used to monitor overfitting and ensure the model generalizes well to unseen data. Accuracy and loss values are recorded for every epoch and later visualized using Matplotlib to track the learning progress.

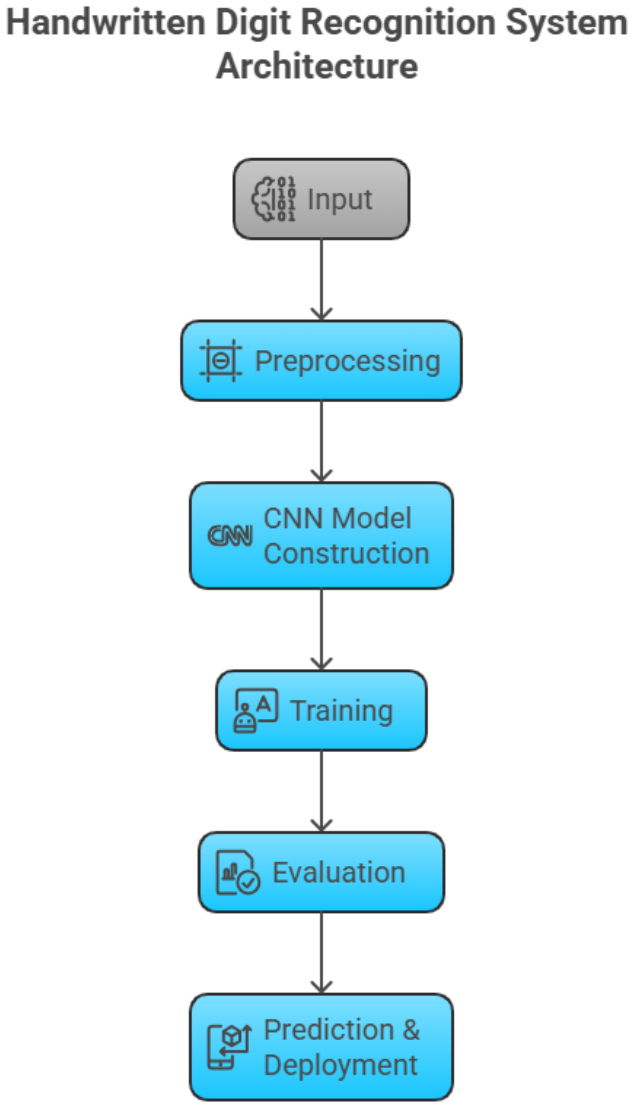
After training, the model is **evaluated** on the test dataset to measure performance. Metrics such as test accuracy, confusion matrix, and per-class predictions are analyzed to verify the model's effectiveness. The trained network typically achieves high

accuracy due to the suitability of CNNs for pattern recognition tasks like handwritten digits.

Once evaluated, the model is saved as a `.h5` file and integrated into a **Streamlit-based deployment interface**. Using the `streamlit-drawable-canvas` library, users can draw digits on a virtual canvas. The drawn image is captured, converted to grayscale using PIL, resized to 28×28 pixels, normalized, and passed to the trained model. The prediction is then displayed instantly on the screen. This interface demonstrates real-time digit recognition and showcases how deep learning models can be deployed in practical applications.

Through these steps, the implementation combines machine-learning techniques, image processing, and front-end design to create a complete handwritten digit recognition system that is accurate, fast, and user-friendly.

6. SYSTEM ARCHITECTURE



Input

The system begins with two possible inputs: the MNIST handwritten digit dataset or a digit drawn interactively by the user through the Streamlit interface. The MNIST dataset provides 70,000 grayscale images of handwritten digits (0–9) and serves as the primary source for training and testing the CNN model. In real-time prediction, users draw digits on a digital canvas, and the drawing is captured as an image. Both types of inputs ensure the system can learn from standardized data while also responding to dynamic, user-generated inputs during deployment.

Preprocessing

Before feeding images into the CNN, the preprocessing module standardizes them to ensure consistent learning. Pixel values are normalized to the range 0–1 to improve model stability and training efficiency. Images are resized to 28×28 pixels and reshaped into a 4D tensor format suitable for convolutional operations. The dataset is also divided into training and testing splits to evaluate model generalization. This preprocessing guarantees that both dataset images and user-drawn digits match the model's expected input structure.

CNN Model Construction – Conv, Pooling, Flatten, Dense

In this stage, the Convolutional Neural Network architecture is built using multiple layers designed to extract and learn meaningful image features. Convolution layers detect low-level patterns such as edges and curves, while max-pooling layers reduce spatial dimensions and computational cost. The Flatten layer converts feature maps into a one-dimensional vector, which is fed into fully connected Dense layers for classification. A final Softmax output layer produces probability scores for each digit from 0 to 9. This module forms the core intelligence of the system.

Training

During training, the preprocessed images are passed through the CNN in multiple epochs, allowing the model to learn patterns by adjusting its internal weights. The Adam optimizer and cross-entropy loss function guide the learning process. Throughout training, accuracy and

loss values are recorded for both training and validation sets to monitor progress and detect overfitting. This iterative process transforms the model from an untrained network into a highly accurate digit classifier.

Evaluation – Accuracy, Confusion Matrix, Metrics

Once training is complete, the model undergoes evaluation using the test dataset. Accuracy is measured to determine how well the model predicts unseen digits, while a confusion matrix helps identify misclassified examples. Additional performance metrics may also be used to assess robustness and reliability. This evaluation module ensures the model is sufficiently accurate and ready for deployment before it is integrated into the prediction interface.

Prediction & Deployment – Streamlit Interface

In the final stage, the trained CNN model is deployed using a Streamlit-based graphical user interface. Users draw digits on a canvas, and the system preprocesses the drawn image to match the training format. The CNN then predicts the digit in real time and displays the result immediately. This module demonstrates practical usability, making the system interactive, intuitive, and suitable for educational or real-world applications.

7. CODE IMPLEMENTATION

Algorithm: Handwritten Digit Recognition using CNN.

Input: User-drawn digit on Streamlit canvas.

Output: Predicted digit (0–9)

1. Start

2. Load Dataset
   2.1 Load the pretrained CNN model from: model/handwritten.h5
   2.2 Ensure the model is ready for inference.
3. Capture User Input
   3.1 User draws a digit on the canvas in the Streamlit app.
   3.2 Extract the canvas image as RGBA array.
   3.3 Convert the array into a PIL image.
4. Preprocess the Image
   4.1 Convert the image to grayscale using `ImageOps.grayscale()`.
   4.2 Resize the image to 28×28 pixels.
   4.3 Convert the image to a NumPy array.
   4.4 Normalize pixel values by dividing by 255.
   4.5 Expand dimensions if required to match model shape.
5. Predict Digit
   5.1 Pass the processed image into the CNN model.
   5.2 Model outputs a probability vector of 10 classes.
   5.3 Use argmax () to get the predicted digit.

6. Display Prediction

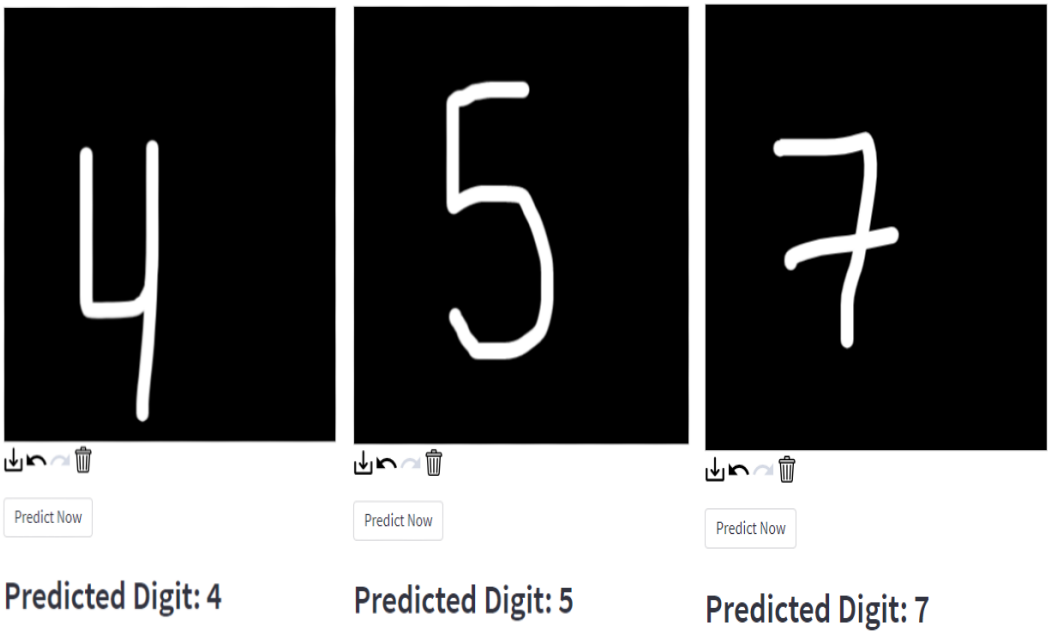
   6.1 Show the predicted digit in Streamlit.

   6.2 Allow user to draw again.

7. End.

8. RESULT

```
Epoch 1/10
469/469 [==============================] - 17s 35ms/step - loss: 0.2129 - accuracy: 0.9382 - val_loss: 0.0598 - val_accurac
y: 0.9816
Epoch 2/10
469/469 [==============================] - 18s 37ms/step - loss: 0.0588 - accuracy: 0.9822 - val_loss: 0.0417 - val_accurac
y: 0.9870
Epoch 3/10
469/469 [==============================] - 18s 38ms/step - loss: 0.0433 - accuracy: 0.9866 - val_loss: 0.0408 - val_accurac
y: 0.9866
Epoch 4/10
469/469 [==============================] - 18s 39ms/step - loss: 0.0315 - accuracy: 0.9901 - val_loss: 0.0360 - val_accurac
y: 0.9873
Epoch 5/10
469/469 [==============================] - 18s 38ms/step - loss: 0.0239 - accuracy: 0.9926 - val_loss: 0.0309 - val_accurac
y: 0.9889
Epoch 6/10
469/469 [==============================] - 18s 37ms/step - loss: 0.0193 - accuracy: 0.9938 - val_loss: 0.0390 - val_accurac
y: 0.9875
Epoch 7/10
469/469 [==============================] - 18s 37ms/step - loss: 0.0150 - accuracy: 0.9951 - val_loss: 0.0281 - val_accurac
y: 0.9909
Epoch 8/10
469/469 [==============================] - 18s 38ms/step - loss: 0.0118 - accuracy: 0.9963 - val_loss: 0.0280 - val_accurac
y: 0.9920
Epoch 9/10
469/469 [==============================] - 18s 38ms/step - loss: 0.0097 - accuracy: 0.9967 - val_loss: 0.0335 - val_accurac
y: 0.9883
Epoch 10/10
469/469 [==============================] - 18s 38ms/step - loss: 0.0090 - accuracy: 0.9973 - val_loss: 0.0393 - val_accurac
y: 0.9877
313/313 [==============================] - 1s 4ms/step - loss: 0.0393 - accuracy: 0.9877
Accuracy: 98.77%
```


Predict Now
Predicted Digit: 4
Predict Now
Predicted Digit: 5
Predict Now
Predicted Digit: 7

9. CONCLUSION

The handwritten digit recognition system developed in this project successfully demonstrates the practical application of deep learning in solving real-world image classification problems. By using a pretrained Convolutional Neural Network model and integrating it with a Streamlit-based interactive interface, the system is able to accurately recognize digits drawn by the user in real time. The project highlights the effectiveness of CNNs in extracting meaningful features from handwritten images and showcases how deep learning models can be deployed in user-friendly applications without requiring direct access to the training dataset. Overall, the system provides fast, reliable digit predictions and serves as a strong foundation for more advanced handwriting recognition tasks, such as multi-digit processing, character recognition, and document automation.

10. REFERENCES

[1] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, *"Gradient-Based Learning Applied to Document Recognition,"* Proceedings of the IEEE, vol. 86, no. 11, pp. 2278–2324, 1998.

[2] L. Deng, *"The MNIST Database of Handwritten Digit Images for Machine Learning Research,"* IEEE Signal Processing Magazine, vol. 29, no. 6, pp. 141–142, 2012.

[3] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, MIT Press, 2016.

[4] F. Chollet, *Deep Learning with Python*, Manning Publications, 2017.

[5] TensorFlow Documentation, *"Convolutional Neural Networks (CNNs) for Image Classification,"* Available: https://www.tensorflow.org. Accessed: 2025.

[6] Keras Documentation, *"Image Classification Using CNN Models,"* Available: https://keras.io. Accessed: 2025.

[7] Streamlit Documentation, *"Building Interactive Web Apps for Machine Learning,"* Available: https://docs.streamlit.io. Accessed: 2025.

[8] Vinay Uniyal, *"Handwritten Digit Recognition – GitHub Repository,"* Available: https://github.com/Vinay2022/Handwritten-Digit-Recognition. Accessed: 2025.

[9] K. O'Shea and R. Nash, *"An Introduction to Convolutional Neural Networks,"* arXiv:1511.08458, 2015.

[10] D. P. Kingma and J. Ba, *"Adam: A Method for Stochastic Optimization,"* arXiv:1412.6980, 2015.