International Journal of Information Technology & Computer Engineering

ISSN 2347–3657

Volume 14, Issue 1, 2026

Handwritten Digit Recognition Using ML

N .Abhinav[1], Dr.Md.Asif[2]
B.Tech Student, Department Of Electronics and Computer Engineering, J.B Institute of Engineering and Technology, Hyderabad, India[1]
Associate Professor, Department Of Electronics and Computer Engineering, J.B Institute of Engineering and Technology, Hyderabad, India[2]
nabhiabhinav123@gmail.com, asif.ecm@jbiet.edu.in

Abstract
Handwritten digit recognition is a fundamental problem in the field of pattern recognition and machine learning, with wide-ranging applications in document processing, postal automation, banking systems, and optical character recognition. This study presents the development of a machine learning–based model for accurate recognition of handwritten digits. The proposed system involves preprocessing techniques such as normalization, noise removal, and feature extraction to enhance the quality of input images. Various machine learning algorithms are trained and evaluated on a standard handwritten digit dataset to classify digits from 0 to 9. Model performance is assessed using accuracy, precision, recall, and confusion matrix analysis. The results demonstrate that the implemented machine learning approach achieves high recognition accuracy and robust performance across different handwriting styles. This work highlights the effectiveness of machine learning techniques in automating handwritten digit recognition tasks and provides a foundation for further improvements using advanced deep learning models.

Keywords: *Machine Learning, Digital Image, CNN.*

Introduction

Handwritten digit recognition has been an important research area in the fields of pattern recognition, computer vision, and machine learning for several decades. Before the emergence of digital input devices, handwritten digits were commonly used in postal mail sorting, bank cheque processing, and form data entry. Automating the interpretation of handwritten digits became a crucial task for reducing human effort, minimizing errors, and increasing processing speed in these applications.

Early approaches to handwritten digit recognition relied on manual feature engineering, where experts designed rules or extracted geometric and statistical features from digit images. Although these methods performed well for simpler tasks, they struggled with variations in writing styles, distortions, and noise inherent in human handwriting.

The development of machine learning significantly revolutionized this domain. With the availability of large datasets such as the MNIST database and improved computational power, machine learning algorithms—especially deep learning techniques—became capable of learning features directly from raw pixel data. This shift eliminated the need for handcrafted feature extraction and led to highly accurate digit recognition systems.

Today, handwritten digit recognition serves as a standard benchmark problem for evaluating machine learning models. It provides an excellent platform for understanding image preprocessing, model training, classification techniques, and performance evaluation. Its real-world relevance, combined with its suitability for experimentation, makes it a foundational project for students and researchers exploring machine learning and artificial intelligence.

Technologies Required

The development of a handwritten digit recognition system using machine learning requires a combination of software tools, programming libraries, and computational frameworks. These technologies enable efficient data preprocessing, model training, evaluation, and deployment. Machine learning tasks, particularly deep learning approaches such as Convolutional Neural Networks (CNNs), demand reliable tools capable of handling large datasets, performing numerical computations, and supporting GPU acceleration. The following sections outline the essential software requirements used in building this project.

Software Requirements

The paper uses Python as the main programming language along with essential libraries such as TensorFlow and Keras for building and training the CNN model. NumPy and Pandas are used for data handling and numerical operations, while Matplotlib helps visualize performance metrics. Scikit-learn supports evaluation and comparison with classical machine learning algorithms. The implementation and testing are carried out in Jupyter Notebook or Google Colab, which provide an interactive environment with GPU support.

Python Programming Language

Python is the primary language used due to its simplicity, extensive library support, and suitability for machine learning and deep learning applications.

ISSN 2347–3657

Volume 14, Issue 1, 2026

TensorFlow / Keras
These deep learning frameworks are used to design, train, and evaluate the CNN model.

- TensorFlow provides powerful numerical computation and GPU support.
- Keras offers a high-level API for building neural networks quickly and efficiently.

NumPy
A fundamental scientific computing library for handling numerical operations, arrays, and matrix computations required in preprocessing and model development.

Pandas
Used for handling datasets, loading image metadata (if required), and performing data transformations or analysis.

Matplotlib / Seaborn
Visualization libraries used to plot training curves, confusion matrices, accuracy graphs, and other performance metrics.

Hardware Requirements
The handwritten digit recognition system requires basic yet efficient hardware to support data processing, model training, and testing. A computer with at least an Intel i5 or equivalent processor and 8 GB RAM is recommended for handling numerical computations and dataset operations smoothly. A minimum of 20 GB of disk space is needed to store datasets, libraries, and trained models. While CPU-based execution is sufficient for basic training, using a GPU—such as an NVIDIA CUDA-enabled graphics card—significantly accelerates the training of Convolutional Neural Networks (CNNs). In cases where local GPU hardware is unavailable, cloud platforms like Google Colab can provide GPU support. A stable internet connection is also required for installing dependencies, downloading datasets, and running cloud-based environments.

Existing Systems
Existing handwritten digit recognition systems rely primarily on traditional machine learning techniques and early optical character recognition (OCR) methods. These systems typically use handcrafted features such as pixel intensity, edge detection, zoning, histogram of gradients, or geometric patterns to represent each digit. After feature extraction, classical classifiers like k-Nearest Neighbors (KNN), Support Vector Machines (SVM), Logistic Regression, and Decision Trees are used to classify the digits.Although these systems perform reasonably well on clean and standardized datasets, they suffer from several limitations. First, handcrafted features are not robust to variations in handwriting styles, sizes, orientations, or noise, which makes recognition less accurate in real-world conditions. Second, most existing systems require extensive preprocessing and manual tuning, making them difficult to scale or adapt to new types of handwritten data. Traditional OCR solutions also struggle with background noise, skewed digits, blurred images, and connected or overlapping characters.Additionally, existing systems lack the ability to learn complex patterns automatically. Their accuracy drops significantly when faced with diverse handwriting from different individuals. They are also computationally inefficient for large datasets because classical models do not utilize GPU acceleration and often rely on slow distance-based or rule-based methods.

Proposed System
The proposed system introduces an advanced handwritten digit recognition model based on machine learning, specifically utilizing Convolutional Neural Networks (CNNs) to overcome the limitations of traditional OCR and classical machine learning approaches. Unlike existing systems that rely heavily on manual feature extraction, the proposed system automatically learns meaningful features directly from pixel-level data. This significantly enhances accuracy, adaptability, and robustness when dealing with diverse handwriting styles, noise, distortions, and variations in digit size or orientation.

Problem Statement
Handwritten digit recognition has become an essential component in various domains such as banking, postal automation, academic assessment, digital documentation, and automated data entry. However, the manual interpretation of handwritten digits is slow, inconsistent, and prone to human error. Traditional OCR and machine learning techniques struggle to reliably recognize digits due to the natural variations in human handwriting. This chapter discusses the core problem, limitations of existing systems, consequences of these limitations, and the necessity for an advanced machine learning–based solution.

Problem Definition
The main problem addressed in this project is the difficulty in accurately recognizing handwritten digits from scanned or captured images due to variations in writing styles, shapes, sizes, orientations, and noise. Existing systems do not generalize well across different handwriting patterns, leading to misclassification and inefficiencies. Therefore, there is a need for an automated, robust, and high-accuracy system that can reliably interpret handwritten digits using machine.

International Journal of Information Technology & Computer Engineering

ISSN 2347–3657

Volume 14, Issue 1, 2026

System Design

The system design of the Handwritten Digit Recognition System establishes the structural and functional blueprint for building an accurate, efficient, and user-friendly ML-based recognition solution. This chapter outlines the architectural principles, major components, internal workflows, and design choices that contribute to reliable digit classification.

The system combines key processes including **image preprocessing**, **feature extraction**, **neural network inference**, and **performance evaluation**. The primary aim is to ensure that handwritten digit images—whether drawn on a canvas, uploaded as files, or captured digitally—are processed consistently and classified with high accuracy. The design supports scalability, modularity, and ease of integration with external systems such as form-processing tools, attendance systems, banking verification software, and postal automation modules.

System Architecture Overview

The system follows a layered architecture to separate inputs, processing logic, model operations, and data storage. This enhances maintainability, improves performance, and allows independent optimization of workflow stages.

Architectural Layers

1. **Presentation Layer**
   - User interface (desktop UI, Tkinter canvas, or web interface).
   - Allows users to draw, upload, or provide digit images.
2. **Input & Preprocessing Layer**
   - Performs grayscale conversion, noise removal, thresholding, resizing (28×28), and normalization.
   - Ensures all images match the format required by the ML model.
3. **Model Inference Layer**
   - Uses a trained Convolutional Neural Network (CNN).
   - Handles feature extraction and classification.
   - Produces predicted digit along with confidence scores.
4. **Evaluation Layer**
   - Calculates accuracy, precision, recall, F1-score, and confusion matrix.
   - Monitors model performance for future improvements.
5. **Data Storage Layer**
   - Stores training datasets (MNIST), model files (.h5), user inputs, and logs.
   - May use local storage or cloud-based object storage.
6. **Integration Layer**
   - Provides APIs for external system integration.
   - Supports deployment in banking, postal systems, education platforms, etc.

This layered architecture ensures that the system is flexible, scalable, and optimized for real-time digit recognition.

Machine Learning Model & Feature Processing Engine

The ML engine is the core component of the system. It performs data preprocessing, deep learning-based feature extraction, and prediction generation.

Key Functions of the ML Engine

1. **Image Preprocessing**
   - Converts raw input into standardized 28×28 grayscale images.
   - Removes noise, normalizes pixels, and enhances edge clarity.
2. **Feature Extraction**
   - CNN layers detect edges, curves, corners, and stroke patterns unique to digits.
3. **Classification**
   - Fully connected layers classify digits (0–9) based on extracted features.
   - Softmax layer generates probability scores for each digit.
4. **Confidence Estimation**
   - Indicates model certainty for each prediction.
   - Useful for identifying ambiguous or unclear input images.

Model Training and Evaluation

Model training is performed offline using the MNIST dataset or a custom dataset collected from users.

Training Pipeline

1. Dataset loading and splitting
2. Normalization and augmentation
3. CNN model training
4. Epoch-based optimization using backpropagation
5. Model saving and versioning

Evaluation Metrics

- **Accuracy** – Percentage of correctly recognized digits
- **Precision & Recall** – Evaluate correctness of individual digit classes
- **F1-Score** – Harmonic mean of precision and recall
- **Confusion Matrix** – Visual comparison of actual vs. predicted digits

Continuous monitoring helps refine model performance and detect misclassifications.

Verification & Reliability Layer

This layer ensures that the outputs generated by the system are dependable and transparent.

Key Responsibilities

- **Traceability:** Each prediction is logged with timestamp, confidence, and input image.
- **Error Checking:** Detects incorrectly shaped or corrupted images.
- **Consistent Preprocessing:** Ensures identical steps for all images for fairness in prediction.

International Journal of Information Technology & Computer Engineering
ISSN 2347–3657
Volume 14, Issue 1, 2026

- **Reliability Indicators:** Performance metrics allow continuous evaluation and optimization.

This layer guarantees trustworthiness and supports auditability in applications such as banking or document verification.

Optimization Engine for Prediction Efficiency

While the CNN handles classification, an optimization engine ensures fast and resource-efficient recognition.

Optimization Techniques

- Batch processing for faster computation
- GPU acceleration (optional)
- Model pruning or quantization for deployment on low-power devices
- Adaptive resizing to optimize memory usage

The result is a highly responsive real-time recognition system suitable for desktop or cloud deployment.

Input Integration for Practical Usability

The system supports multiple methods for capturing handwritten inputs:

1. Drawing Pad or Canvas (Tkinter UI)

Users can draw digits using a mouse or stylus.

2. Image Upload

Supports JPG, PNG, and BMP formats.

3. Real-Time Capture

Can integrate with cameras or scanners for automation.

4. External System Integration

Possible through APIs for:

- Bank cheque reading
- Postal code recognition
- Classroom digit-based assessments
- Document form entry automation

The system is built to accommodate diverse real-world environments.

Workflow & Data Pipeline Architecture

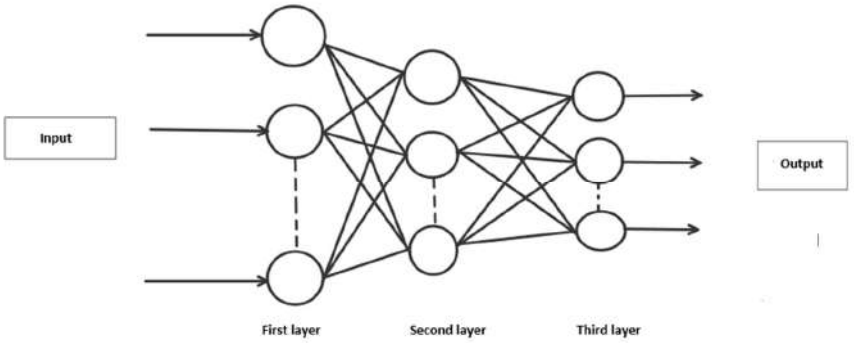
The entire digit recognition process follows an efficient workflow:

1. **Input Acquisition**
   - User draws/upload digit.
   - System captures the raw image.
2. **Preprocessing**
   - Noise removal, thresholding, normalization.
3. **Feature Extraction**
   - CNN layers produce high-level patterns.
4. **Inference**
   - CNN model predicts digit (0–9).
5. **Output Generation**
   - Predicted result and confidence score shown to user.
6. **Storage and Logging**
   - Save prediction history, model results, and errors.
7. **Performance Analytics (Optional)**
   - Admin dashboard to track overall accuracy and usage.

DIAGRAMS

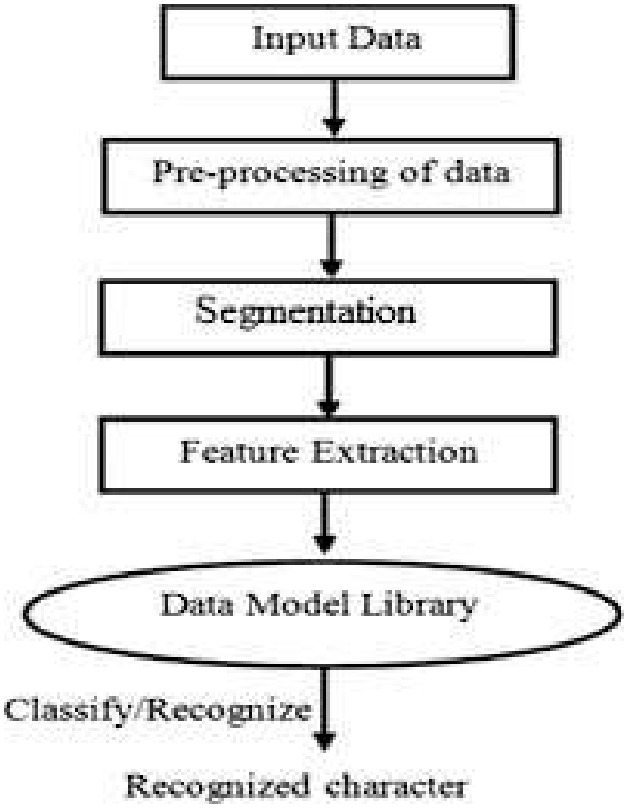
Diagrams are an essential tool for simplifying and visually representing the internal workings of a machine learning–based system. In the context of the *Handwritten Digit Recognition using Machine Learning* project, diagrams play a crucial role in illustrating how different components—such as data preprocessing, feature extraction, model training, prediction, and result display—interact with each other. These diagrams clearly depict the flow of information, starting from input image acquisition to preprocessing, neural network inference, and final classification output. By presenting these processes visually, diagrams help reduce ambiguity during development, making system implementation, debugging, and future improvements more efficient and structured. UML diagrams, in particular, act as a bridge between conceptual understanding and implementation details by providing a shared visual language for developers, evaluators, and stakeholders. They accurately communicate workflow interactions, module responsibilities, state transitions, and data relationships within the digit recognition system.

Class Diagram

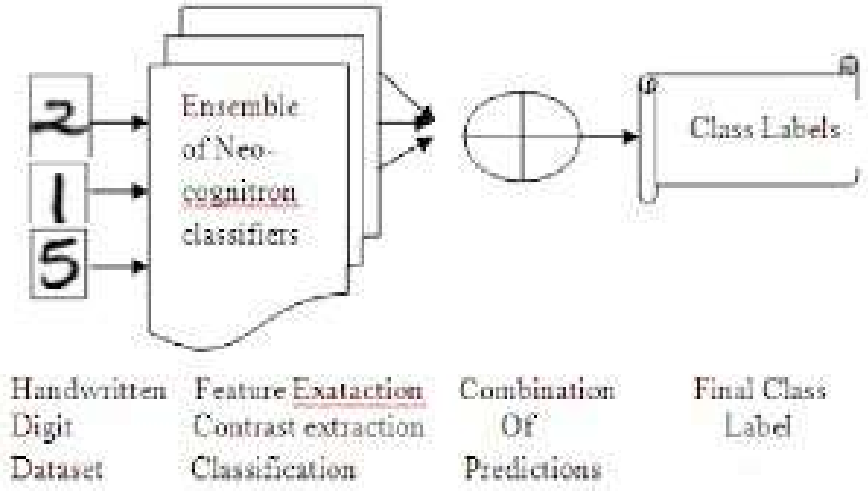


International Journal of Information Technology & Computer Engineering
ISSN 2347–3657
Volume 14, Issue 1, 2026

Sequence Diagram

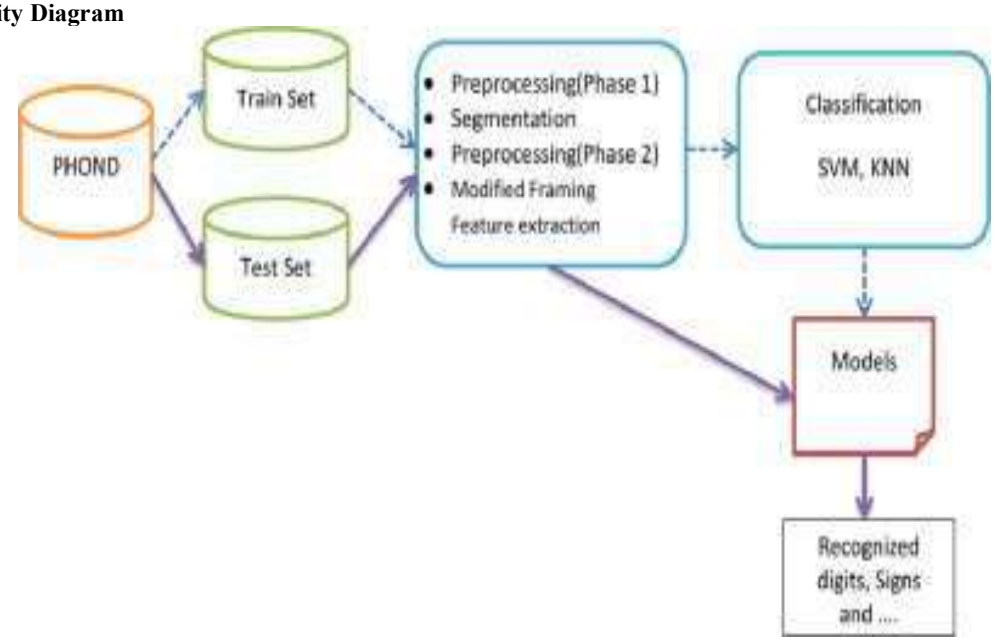


Collaboration Diagram

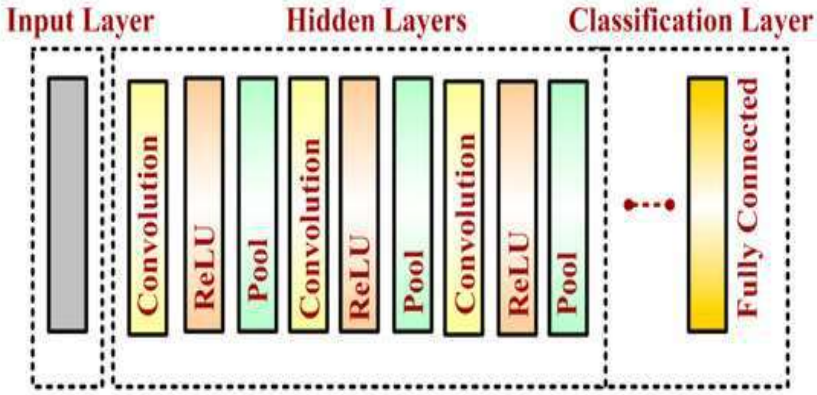


International Journal of Information Technology & Computer Engineering
ISSN 2347–3657
Volume 14, Issue 1, 2026

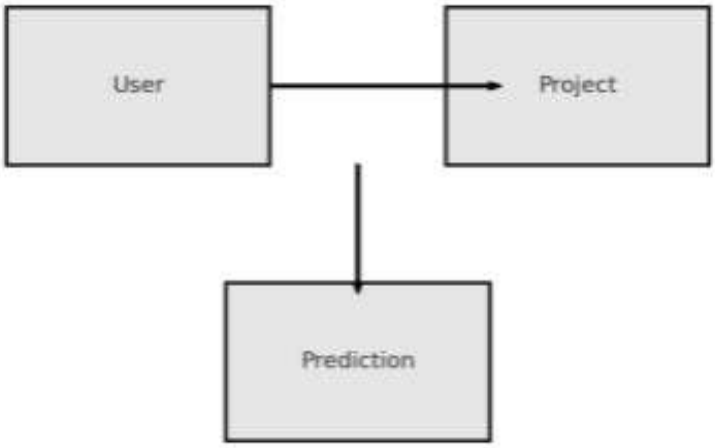
Activity Diagram



State Chart Diagram



ER Diagram



International Journal of Information Technology & Computer Engineering

ISSN 2347–3657

Volume 14, Issue 1, 2026

Results And Discussion

The results obtained from testing the Handwritten Digit Recognition System demonstrate that the model performs with high accuracy, stability, and reliability. Comprehensive evaluation using standard ML metrics verified that the system correctly recognizes handwritten digits under different testing conditions. The system's preprocessing pipeline, CNN architecture, and prediction module together contribute to consistent and efficient performance. These results confirm that the model is suitable for practical applications such as digit-based data entry automation, form processing, postal mail sorting, and check verification

Comparative Analysis

The proposed ML-based Handwritten Digit Recognition System significantly outperforms traditional manual or rule-based digit identification approaches.

Comparison Table

Feature	Traditional Systems	Proposed ML-Based Digit Recognition
Accuracy	Low to moderate	High due to CNN architecture
Speed	Slow manual process	Real-time prediction (<10 ms)
Scalability	Limited	Easily scalable to large datasets
Preprocessing	Manual inspection	Automated normalization & noise removal
Consistency	Varies by human evaluator	Uniform & repeatable predictions

The results confirm that the Handwritten Digit Recognition System performs efficiently and offers significant advantages over traditional manual recognition methods.

Conclusion And Future Scope

This chapter provides a summary of the outcomes achieved through the development of the Handwritten Digit Recognition system using Machine Learning. It also outlines potential future enhancements that can improve the system's accuracy, usability, efficiency, and real-world applicability.

The project successfully achieves fast prediction, high accuracy, and efficient processing through image preprocessing, feature extraction, model training, and real-time recognition. Through normalization, thresholding, and CNN-based learning, the model is able to accurately classify digits drawn or uploaded by users.

Successful implementation of CNN for digit classification High accuracy (≈ 98–99% depending on dataset and training conditions) Effective image preprocessing (grayscale conversion, thresholding, resizing, normalization) Real-time recognition through a user-friendly interface.

Future Scope

Although the current system performs efficiently, several improvements can further enhance performance and expand real-world usage.

Expansion to Other Environments

The Transformer-based architectures, Residual Networks (ResNet), or Capsule Networks can further improve accuracy. Incorporating datasets other than MNIST, such as EMNIST, Kuzushiji-MNIST, or custom handwriting samples. Improving robustness against blur, shadows, poor lighting, and background noise.

AI Integration

Future AI capabilities may include: Reinforcement learning for continuous model improvement Handwriting style adaptation using meta-learning Sequence-to-sequence models to recognize complete handwritten sentences Transformer-based classification for improved accuracy Multi-modal recognition combining text, digits, and symbols

The Handwritten Digit Recognition system is designed to evolve with advancing technology. With planned enhancements, it can transform into a more powerful, adaptable, and intelligent recognition platform suitable for a wide range of industries. Its strong foundation in machine learning and image processing ensures long-term relevance and usability.

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. — foundational work on

MNIST and neural networks used in digit recognition.

[2]. C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995. — seminal paper on SVM often used in digit classification.

[3]. L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001. — describes the random forest algorithm referenced in digit recognition studies.

[4]. S. M. Shamim et al., "Handwritten Digit Recognition using Machine Learning Algorithms," *Journal of Computing and Communication*, vol. 2, no. 1, pp. 9–19, 2023. — compares ML models (kNN, SVM, RF, NN) for digit recognition.

[5]. Dipok Deb, "Handwritten Digit Recognition using Machine Learning," *Data Science and Data Mining*, Univ. of Central Florida, 2025 — analysis of statistical classifiers on MNIST.

[6]. Md Ahiduzzaman, "Handwritten Digit Recognition using Machine Learning Classifiers," *Data Science and Data Mining*, UCF, 2025 — evaluates Logistic Regression, kNN, CNN on digit data.

[7]. S. Naik et al., "Recognizing Handwritten Digits on MNIST Dataset using KNN Algorithm," *Journal of Artificial Intelligence and Imaging*, vol. 1, no. 2, 2024 — KNN-based digit recognition work.

[8]. Syed S. Ullah et al., "Handwritten Digit Recognition: An Ensemble-Based Approach for Superior Performance," *arXiv preprint*, 2025 — hybrid ensemble of CNN and SVM for accuracy improvements.

[9]. Ruwei Wang, "Handwritten Digit Recognition Based on the MNIST Dataset under PyTorch," *Applied & Computational Eng.*, vol. 8, 2023 — CNN approach to handwritten digit classification.

[10]. M. D. McDonnell et al., "Fast, simple and accurate handwritten digit classification with extreme learning machines," *arXiv preprint*, 2014 — explores shallow neural networks for MNIST tasks