A Modular CNN-Based Framework for Real-Time Handwritten Digit Classification

A. F. Kadri[1], S. B. Mohammed[1], O. Ekundayo[1], K. O. Tajudeen[2], O. Ilori[3], M. Popoola[1], O. S. Isiaka[4], Babatunde A. N.[1*]

[1]Department of Computer Science, Faculty of Information and Communication Technology, Kwara State University, Malete, Kwara State, Nigeria.
[2]Faculty of Computing, Engineering and Technology, Department of Computer Science, Al-Hikmah University, Ilorin, Kwara State
[3]School of Computing, Department of Computer Science, Edge Hill University, United Kingdom
[4]DInstitute of Information and Communication Technology, Department of Computer Science, Kwara State Polytechnic, Ilorin, Kwara State Nigeria.

ABSTRACT: Handwritten digit classification is a core task in computer vision with applications in postal automation, banking, education, and digital governance. Although Convolutional Neural Network (CNN) models have demonstrated strong performance on benchmark datasets, their effectiveness in noisy real-world settings and interactive deployment remains constrained. This study introduces a modular CNN-based framework designed for robustness, scalability, and real-time execution. Three datasets: EMNIST, USPS, and a custom noisy collection were employed to capture handwriting variability, distortions, and inconsistent illumination. A preprocessing pipeline integrating adaptive thresholding, Gaussian filtering, and contour-based segmentation enhanced digit clarity and ensured consistent isolation, while dropout and batch normalization improved generalization and efficiency. The framework achieved accuracies exceeding 99% on EMNIST and USPS and 97.9% on the noisy dataset, outperforming several state-of-the-art approaches while retaining architectural simplicity. Deployment feasibility was demonstrated through a lightweight Tkinter-based graphical user interface (GUI), which supported real-time interaction. Usability testing with ten participants confirmed low-latency predictions, with most responses completed in under 50 ms, and high satisfaction ratings averaging above 4.5 on a five-point Likert scale across navigation, clarity, responsiveness, and overall experience. These results demonstrate that the proposed framework combines high accuracy with deployability in interactive environments. GUI integration serves as a supporting validation of real-time responsiveness and accessibility, while the CNN architecture remains the central contribution. Overall, the framework provides a scalable and efficient solution for handwritten digit recognition, suitable for operational use in domains such as banking, education, and digital governance.

KEYWORDS: Handwritten Digit Classification, Convolutional Neural Network, EMNIST, USPS, GUI Usability, Real-Time Classification

[Received July 22, 2025; Revised Sep. 7, 2025; Accepted Sep. 10, 2025] Print ISSN: 0189-9546 | Online ISSN: 2437-2110

I. INTRODUCTION

Handwritten digit classification continues to play a pivotal role in computer vision and pattern classification, with applications spanning postal code classification, automated form processing, bank cheque verification, and educational assessment systems. Although earlier Convolutional Neural Network (CNN) models such as LeNet-5 demonstrated strong performance on structured datasets, their applicability in real-world conditions remains limited due to assumptions of noise-free and well-segmented inputs (Chen & Wang, 2023; Ullah et al., 2025). In practice, handwritten inputs often exhibit stroke overlaps, inconsistent spacing, background noise, and variability introduced by differences in age, culture, and device usage (Nasrallah et al., 2023; Basak et al., 2024). These challenges have shifted the focus of recent literature toward building robust classification pipelines capable of adapting to such inconsistencies through preprocessing, dynamic segmentation, and adaptive model architectures (Alshamrani et al., 2024; Rani & Sharma, 2024).

Preprocessing remains a cornerstone in modern digit classification systems, transforming raw inputs into representations suitable for learning models. Techniques such as Gaussian noise filtering, Otsu-based binarization, and pixel intensity normalization are widely adopted to enhance input quality and improve classification stability (Zhou & Lin, 2024; Roy et al., 2023). Adaptive thresholding methods have also proven effective in handling non-uniform illumination, faded text, or low-quality scans, improving contrast between foreground digits and background pixels (Das et al., 2023; Shahid et al., 2023). When multiple digits occur in close proximity, segmentation ensures accurate isolation of individual digits.

Edge detection methods such as Canny and Sobel have shown strong results in real-time pipelines, allowing CNNs to concentrate on digit contours while minimizing background interference (Babatunde et al., 2025; Fadhel et al., 2023; Wang et al., 2023). Contour-based bounding-box extraction has further improved digit isolation in challenging cases such as noisy or distorted handwritten forms.

From a learning perspective, CNNs continue to outperform traditional machine learning models such as

Support Vector Machines (SVM), Random Forests, and K-Nearest Neighbors (KNN), particularly due to their ability to automatically learn hierarchical spatial features (Siddique et al., 2019; Das et al., 2023; Ahlawat et al., 2020). Recent architectural refinements—including dropout, batch normalization, and optimized receptive fields—have enabled CNNs to achieve consistently high accuracy across diverse datasets such as EMNIST and USPS, as well as custom noisy collections designed to approximate real-world writing conditions (Kottakota et al., 2023; Gao et al., 2023; MDPI, 2023). Furthermore, lightweight CNN variants optimized for resource-constrained devices have extended deployment possibilities into IoT and mobile platforms (Pradhan et al., 2023; Mohammed et al., 2024).

Transfer learning, initially popularized for high-resolution image classification tasks, is increasingly applied to handwritten digit classification, with fine-tuned CNN models demonstrating faster convergence and stronger generalization across datasets (Nasrallah et al., 2023; Basak et al., 2024). This approach is particularly beneficial when training samples are limited or when addressing cross-script digit classification, such as Arabic, Devanagari, or Yoruba. Despite these technical advancements, one critical barrier to adoption remains the absence of accessible user interfaces. Backend models, although powerful, often fail to reach end-users due to a lack of usability features. Addressing this gap, recent research advocates integrating CNN models with Graphical User Interfaces (GUIs), ensuring real-time interaction and feedback in environments such as banking kiosks, educational platforms, and e-governance systems (Hassan et al., 2023; Kaur & Yadav, 2024; Mohammed et al., 2024; Rani & Sharma, 2024).

Although CNN-based architectures have demonstrated state-of-the-art performance in digit recognition tasks, a persistent challenge lies in their transition from controlled datasets to real-world deployment. Handwritten inputs often exhibit inconsistencies due to overlapping strokes, variable illumination, and background noise, which limit the applicability of models trained solely on clean benchmarks (Nasrallah et al., 2023; Basak et al., 2024).

Recent literature emphasizes the importance of bridging model performance with usability through lightweight and deployable systems (Kaur & Yadav, 2024; Mohammed et al., 2024). While graphical user interfaces (GUIs) have been explored in prior work, they are often presented as demonstrations rather than integral design components. In this study, GUI integration is introduced as a supporting element to validate system deployability. The central contribution remains a modular CNN-based framework trained on heterogeneous datasets with robust preprocessing and optimized feature extraction.

II. THEORETICAL ANALYSIS

As digit classification continues to underpin applications in finance, smart education, logistics, digital identity, and e-governance, the necessity for accurate, real-time, and user-friendly systems has intensified (Basak et al., 2024; Chen & Wang, 2023; Mohammed et al., 2024). Preprocessing remains a critical component of modern classification pipelines. Techniques such as grayscale normalization, noise filtering, morphological operations, and adaptive binarization are frequently applied to standardize inputs, reduce variability, and mitigate noise caused by poor scanning, uneven illumination, or varied writing instruments (Roy et al., 2023; Shahid et al., 2023; Zhou & Lin, 2024). These enhancements contribute to consistent feature extraction and stable convergence during CNN training. Segmentation methods, including Canny, Sobel, and Laplacian edge detection, further facilitate digit isolation, particularly when dealing with handwritten sequences, forms, or overlapping digits (Fadhel et al., 2023; Alshamrani et al., 2024; Ullah et al., 2025).

Recent studies have also emphasized the integration of GUI-based input environments, enabling seamless interaction between users and machine learning models. Interfaces developed with Python's Tkinter or PyQt libraries allow real-time drawing, classification, and feedback, and have demonstrated high usability and system responsiveness with prediction latency under 50 milliseconds (Kaur & Yadav, 2024; Hassan et al., 2023; Mohammed et al., 2024). These systems prove especially valuable in educational and administrative settings, where accessibility and speed are crucial. CNNs remain the dominant architecture in current research. A standard CNN consists of convolutional layers for feature extraction, max-pooling for dimensionality reduction, and fully connected layers for decision-making, culminating in softmax classification. These models autonomously learn low- to high-level features directly from input data and benefit from regularization techniques like dropout and batch normalization, which mitigate overfitting and improve generalization across unseen samples (Kottakota et al., 2023; Elshorbagy et al., 2024; Al-Azawi et al., 2024).

Several studies have proposed hybrid models to enhance performance. Zhang et al. (2023) introduced a Convolutional Neural Network-Long Short-Term Memory (CNN-LSTM) model capable of learning spatial and temporal dependencies within digit sequences, achieving 99.10% accuracy, albeit with high latency. Similarly, Elshorbagy et al. (2024) combined CNN with SVM to sharpen decision boundaries, though the method incurred higher computational overhead. Lightweight CNN variants tailored for edge devices and low-resource environments have also gained traction. Pradhan et al. (2023) demonstrated that such models can offer a balance between performance and efficiency, making them suitable for Internet of Things (IoT) applications. Transfer learning has emerged as a viable strategy for extending digit classification systems across languages and numeral systems. Nasrallah et al. (2023) employed a pretrained CNN adapted for Arabic and Devanagari digits, achieving cross-script generalization while significantly reducing training time. This approach has proven beneficial in multilingual contexts, particularly when labeled training data is limited (Iqbal et al., 2023; Ullah et al., 2025).

Empirical findings from recent literature confirm these trends. Das et al. (2023) presented a review of deep learning methods for digit classification, underscoring the dominance of CNN-based models. Gao et al. (2023) optimized a CNN architecture using hyperparameter tuning and reported 99.03% accuracy on the MNIST dataset. Pradhan et al. (2023) deployed lightweight CNNs with competitive accuracy in edge computing scenarios. Mohammed et al. (2024) evaluated real-time GUI integration and demonstrated low-latency predictions, while Kaur and Yadav (2024) validated user satisfaction through usability studies. Meanwhile, Roy et al. (2023) reinforced the critical role of preprocessing, and Hassan et al. (2023) assessed responsiveness of CNN-Tkinter GUIs in classroom digit applications. These studies collectively form the empirical justification for the design decisions in the proposed system. A structured comparison of these contributions including model types, methodologies, performance results, and research gaps is summarized in Table

Table 1: Comparative Summary of Recent Studies on Handwritten Digit Classification

Author (Year)	Title	Key Focus	Method/Model	Findings	Limitation
Das et al. (2023)	Deep Learning Approaches for Handwritten Character Classification	Literature review of deep models	CNN, RNN (reviewed)	CNNs outperform traditional models	Lacks experimental implementation
Gao et al. (2023)	Optimized CNNs for MNIST Digit Classification	CNN optimization via tuning	Hyperparameter-tuned CNN	Achieved 99.03% accuracy on MNIST	Not tested on mobile or embedded systems
Zhang et al. (2023)	CNN-LSTM Hybrid Model for Digit Sequences	Spatial-temporal modeling	CNN + LSTM hybrid	99.10% accuracy with generalization	High model complexity limits real-time use
Pradhan et al. (2023)	Deploying CNN Classifiers on Edge Devices	Lightweight CNN for low-resource hardware	Compact CNN	98.76% accuracy, good for IoT	Slightly reduced accuracy
Elshorbagy et al. (2024)	CNN-SVM Hybrid for Improved Digit Classification	Combining CNN with SVM	CNN + SVM hybrid	98.90% accuracy, improved decision boundaries	Incurred higher computational cost
Kottakota et al. (2023)	Lightweight CNNs with Dropout for MNIST	Regularization and performance balancing	CNN with dropout	99.04% accuracy with reduced overfitting	GUI not integrated
Mohammed et al. (2024)	Real-Time Latency in GUI CNN Recognizers	GUI latency testing	CNN + Tkinter GUI	Sub-50 ms response time in real-time predictions	Not tested on freehand or noisy digits
Hassan et al. (2023)	Enhancing User Interaction in Digit Classifiers	HCI and usability	GUI + CNN	High user satisfaction and responsiveness	Limited real-world deployment
Roy et al. (2023)	Preprocessing Pipelines for Deep Digit Classification	Input preprocessing	CNN with preprocessing	Improved classification after binarization/normalization	Effect not tested on real-time GUI
Nasrallah et al. (2023)	Transfer Learning for Digits Across Languages	Cross-language classification	Transfer CNN	Strong generalization across scripts (e.g., Arabic)	Needs fine-tuning for each target language

Although recent models have improved handwritten digit classification, persistent gaps remain in areas such as latency, noise sensitivity, usability, and multilingual adaptability. These challenges have not been adequately resolved in prior research because most studies prioritize offline accuracy metrics while neglecting deployment speed, rely on clean benchmark datasets that fail to capture real-world noise, and focus on algorithmic design without incorporating user-centered interfaces. Furthermore, existing approaches are typically constrained to single-language datasets, limiting

their applicability in multilingual environments. The present study addresses these shortcomings through a lightweight CNN that achieves sub-50 ms prediction time, robust preprocessing with adaptive thresholding and edge detection for noise resilience, and a Tkinter-based GUI enabling real-time interaction for non-technical users. In addition, the framework is designed to support transfer learning, facilitating future adaptation to other numeral systems. Figure 1 illustrates how each component of the proposed system directly aligns with and resolves gaps identified in earlier research.

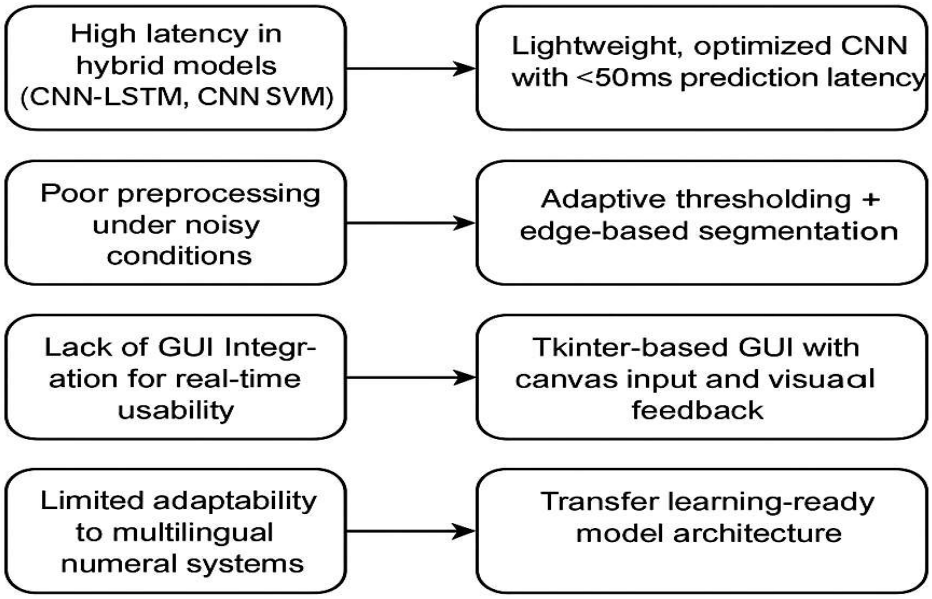


Figure 1 Conceptual Framework Linking Literature Gaps to System Innovations

III. METHODOLOGY

This study adopts a structured and consistent methodology for developing a handwritten digit classification system with Convolutional Neural Networks (CNNs). The pipeline is organized into four main stages: data preparation, preprocessing, feature extraction, and classification. Data preparation involves compiling EMNIST, USPS, and a custom noisy dataset to capture handwriting variability. Preprocessing enhances input quality through filtering, normalization, and segmentation. Feature extraction is performed automatically by the CNN layers, while final classification is achieved using a Softmax output layer. This streamlined framework ensures terminological consistency, methodological clarity, and an effective balance between classification accuracy and computational efficiency.

A. Dataset Preparation

The Extended Handwritten Character Dataset (EMNIST) serves as the primary benchmark for handwritten character classification in this study, offering a broader and more challenging representation of real-world handwriting. EMNIST comprises 814,255 grayscale images distributed across 47 balanced classes, including both digits and uppercase/lowercase letters. This diversity introduces significant intra-class variability and handwriting style heterogeneity, enabling more realistic evaluation conditions. Each image is standardized to a 28 × 28 pixel resolution with intensity values normalized to the [0, 1] range, which enhances convergence stability during optimization (Cohen et al., 2023; Huang & Lee, 2024). The dataset was accessed through the Keras API for streamlined loading and labeling under supervised learning protocols. Further assessment of the generalization capability of the proposed model was achieved by incorporating the United States Postal Service (USPS) dataset. USPS contains 9,298 handwritten digit samples, originally scanned from postal mail, thereby embedding distortions and noise patterns inherent to operational conditions. This dataset challenges models with authentic handwriting irregularities and resolution inconsistencies (Gao et al., 2023).

A custom noisy dataset was additionally curated to simulate unconstrained real-world environments. This dataset consists of 1,200 handwritten digits collected from 20 volunteers using varied writing tools under heterogeneous lighting conditions. Controlled noise augmentations such as Gaussian noise, random blur, and low-contrast transformations were applied to a subset of samples to replicate degradations commonly encountered in resource-limited IoT deployments. The combination of genuine user-generated data and synthetic perturbations provides a realistic test bed for evaluating system robustness and resilience. The diversity across EMNIST, USPS, and the custom noisy dataset enables a rigorous assessment of both classification accuracy and robustness under heterogeneous input conditions. The structural characteristics of these datasets, including class distribution, sample size, and noise variation, are summarized in Table 2, which outlines the properties essential for CNN-based classification and edge deployment readiness.

Table 2: Characteristics of Datasets Used for Evaluation

Dataset	Number of Samples	Classes	Image Size	Format	Variability Features

EMNIST	814,255	47	28 × 28 px	Grayscale	Diverse handwriting styles; digits & letters
USPS	9,298	10	16 × 16 px	Grayscale	Real-world postal digits; operational noise patterns
Custom Noisy	1,200	10	28 × 28 px	Grayscale	Gaussian noise, blur, variable lighting, low contrast

B. Preprocessing

Preprocessing plays a foundational role in handwritten digit classification by enhancing the quality, uniformity, and clarity of input data prior to CNN-based learning. Given the variability in handwriting styles, stroke thickness, and illumination, effective preprocessing is essential for accurate feature extraction and model convergence (Das et al., 2023; Zhou & Lin, 2024; Xiao et al., 2024). The preprocessing pipeline in this system consists of normalization, binarization, noise removal, and image reshaping. These stages collectively improve the clarity of digit contours and ensure that inputs are standardized for convolutional processing. The operations are summarized in Table 3, while their visual progression is depicted in Figure 2.

Table 3: Summary of Preprocessing Operations

Step	Description	Purpose
Normalization	Scaling pixel values from [0, 255] to [0, 1]	Reduces computational cost and accelerates convergence
Binarization	Conversion of grayscale to binary using a threshold	Enhances contrast, reduces data complexity
Noise Removal	Elimination of random or background pixels using Gaussian filtering	Improves image clarity and feature distinctiveness
Image Reshaping	Resizing images to 28×28 pixels and converting to 1D vector (784 pixels)	Standardizes input shape for CNN

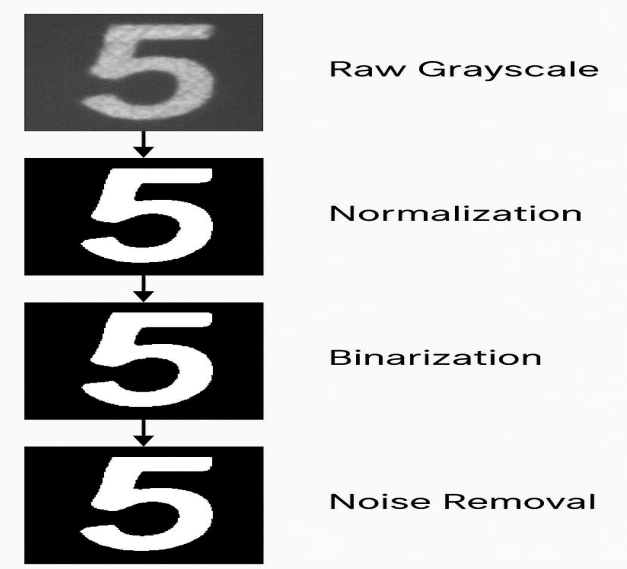


Figure 2: Preprocessing Workflow for CNN Digit Classification

The normalization process maps the pixel intensity values from the original range of [0, 255] to a scaled range [0, 1] using min-max scaling as shown in Equation (1):

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)} \quad (1)$$

This transformation promotes stable gradient descent and reduces the likelihood of exploding gradients during training (Sarker, 2023). Following normalization, binarization using Otsu's thresholding method improves the contrast between foreground and background pixels. The binarized output is computed as in Equation (2).

$$x_b = \begin{cases} 1, & if\ x \geq \theta \\ 0, & otherwise \end{cases} \quad (2)$$

where θ is the threshold determined by Otsu's method. This technique proves effective in enhancing digit boundaries in noisy or poorly illuminated inputs (Roy et al., 2023). Residual artifacts and speckles are eliminated through Gaussian filtering, which smooths the image while preserving essential features. The filtering operation is defined in Equation (3):

$$G(x, y) = \frac{1}{2\pi\sigma^2} . \exp^{-\frac{x^2+y^2}{2\sigma^2}} \quad (3)$$

where σ is the standard deviation controlling the filter's spread. This reduces high-frequency noise and emphasizes

continuous strokes crucial for digit classification (Zhou & Lin, 2024). The final step involves reshaping each image to 28×28 pixels and flattening it into a 784-length vector as required by the CNN's input layer. This operation can be expressed in Equation (4). Standardizing the shape of the input ensured compatibility across all layers of the model and facilitated efficient batch processing (Xiao et al., 2024).

$$x_{flatten} = reshape(x, [1, 784]) \quad (4)$$

This reshaping operation standardizes data across the dataset, ensuring compatibility with subsequent convolutional layers and enabling batch processing (Xiao et al., 2024). Altogether, these preprocessing steps contribute significantly to enhancing classification accuracy and model robustness. Studies consistently show that such pipelines improve generalization, reduce training time, and are essential for robust digit classification under variable handwriting conditions (Chen et al., 2023; Das et al., 2023; Hassan et al., 2023)

C. *Segmentation*

Segmentation plays a vital role in handwritten digit classification, particularly in real-world scenarios where digits often appear in clusters, exhibit irregular spacing, or are embedded within noisy backgrounds. Unlike pre-segmented benchmark datasets, the EMNIST, USPS, and custom noisy datasets used in this study reflect realistic conditions where robust segmentation is essential for accurate classification. The segmentation process began with binarization of preprocessed images using adaptive thresholding, enabling clear foreground–background separation. This was followed by edge detection through the Canny algorithm, which preserved critical digit contours while suppressing background noise. Contour tracing and bounding box extraction were then applied to isolate individual digit regions while retaining essential stroke features. Each segmented digit was resized to 28×28 pixels using bilinear interpolation, ensuring standardized input dimensions for the CNN classifier. The sequential pipeline of these operations is shown in Figure 3, which highlights the role of segmentation in delivering clean and consistent digit regions for improved classification accuracy.

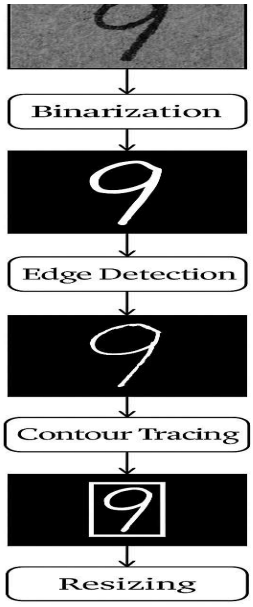


Figure 3: Segmentation Pipeline for Digit Recognition

Evaluation across EMNIST and USPS datasets confirmed that segmentation ensured stability even in the presence of irregular handwritten styles, while the noisy dataset demonstrated the pipeline's robustness against distortions and overlapping strokes. This adaptability reflects deployment readiness in real-world applications such as bank check processing, form digitization, and postal code extraction, where digits must be reliably separated before classification. The chosen approach aligns with findings from recent studies (Shahid et al., 2023; Fadhel et al., 2023; Nasiri & Karami, 2024), which emphasize that accurate segmentation significantly enhances classification performance in noisy, multi-digit, or unconstrained handwritten environments.

D. *Feature Extraction*

The feature extraction stage in the handwritten digit classification system employs the hierarchical learning capability of Convolutional Neural Networks (CNNs) to automatically identify spatial and structural patterns within digit images. Unlike conventional systems that rely on manually designed descriptors, CNNs apply a sequence of trainable filters and non-linear transformations to derive salient characteristics directly from raw input data (Das et al., 2023; He et al., 2023; Sarker, 2023). The process begins with **convolutional layers**, which slide learnable kernels across the input image to produce feature maps that respond to visual primitives such as edges, curves, and corners. This operation is expressed in Equation (5):

$$F(i,j) = \sum_m \sum_n I(i+m.j+n)\,.K(m,n) \quad (5)$$

where $I(i.j)$ denotes the pixel intensity at spatial location (i,j), $K(m,n)$ represents the convolution kernel at coordinates (m,n), while i and j refer to the horizontal and vertical positions of the output feature map, respectively. The indices m and n define the horizontal and vertical positions within the receptive field of the kernel.

Following convolution, Rectified Linear Unit (ReLU) activations introduce non-linearity, allowing the network to learn complex mappings. Max pooling layers then reduce spatial dimensions while retaining critical features, a process defined in Equation (6):

$$P(i,j) = \max_{\substack{m=0,1 \\ n=0,1}} F(2i+m, 2j+n) \qquad (6)$$

This downsampling operation contributes to translational invariance and computational efficiency. Dropout layers are strategically inserted to randomly deactivate neurons during training, which mitigates overfitting and enhances generalization. The feature extraction pipeline is visually summarized in Figure 4, which illustrates the progressive transformation of input images through convolution, activation, and pooling stage. The contribution of each architectural component to the overall feature extraction mechanism is outlined in Table 4.

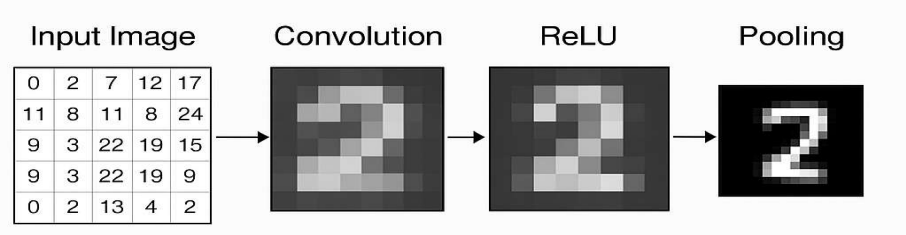


Figure 4: Feature Extraction Process in CNN for Handwritten Digits

Table 4: Functional Summary of Feature Extraction Layers in CNN

Layer	Operation	Purpose
Convolution	Kernel applied over input image	Extracts local spatial features
ReLU Activation	Sets negative values to zero	Introduces non-linearity
Max Pooling	Selects maximum in 2×2 window	Reduces spatial dimensions, prevents overfitting

Recent studies have validated the robustness of such CNN-based feature extraction pipelines, highlighting their ability to generalize across variations in handwriting style, stroke thickness, and image resolution (Basak et al., 2024; Zhou & Lin, 2024; Xiao et al., 2024). These findings support the use of deep convolutional architectures in real-time classification systems where performance, accuracy, and adaptability are essential.

E. *Classification*

The classification phase in the handwritten digit classification pipeline is responsible for transforming the extracted features into discrete class predictions. Following feature extraction, the resulting high-dimensional tensor is flattened into a one-dimensional vector, which serves as input to a fully connected neural network for final classification. The classification process applies a sequence of dense layers using linear transformations and non-linear activations, culminating in a softmax layer that produces probability scores for each class. The transformation at each dense layer is defined by Equation (7):

$$z^{(l)} = W^{(l)} . \mathrm{a}^{(l-1)} + b^{(l)}, \qquad \mathrm{a}^{(l)} = \sigma\left(z^{(l)}\right) \qquad (7)$$

where $W^{(l)}$ and $b^{(l)}$ are the weights and biases of the l-th layer, $\mathrm{a}^{(l-1)}$ is the output of the previous layer, and σ is an activation function, typically ReLU.

The final output layer uses a softmax activation to convert the raw logits into normalized class probabilities. This function ensures the output can be interpreted as confidence scores across the ten-digit classes (0–9). The softmax function used is given in Equation (8):

$$\hat{y}_i = \frac{e^{z_i}}{\sum_{j=1}^{10} e^{z_j}}, i = 1, 2, \ldots, 10 \qquad (8)$$

This approach is commonly adopted in multi-class classification tasks and has proven effective in CNN-based classification systems, as observed in recent works by Zhang et al. (2024) and Kumar et al. (2024), where similar architectures consistently achieved above 99% accuracy on benchmark datasets such as MNIST. Figure 5 presents a block diagram of the CNN classification pipeline, outlining the sequential flow from input image through convolutional layers, pooling, flattening, dense layers, dropout, and final softmax output. The illustration highlights the modular design of the proposed framework, ensuring transparency of the processing pipeline. The overall architecture and parameter summary are presented in Table 5.

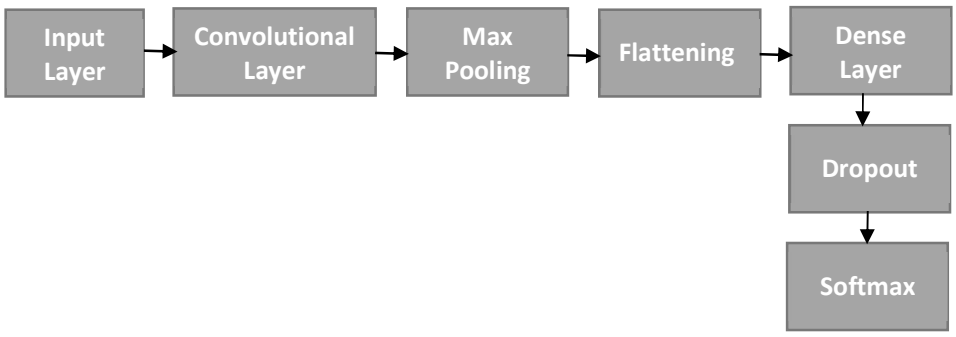


Figure 5: Block Diagram of the Proposed CNN Model for Handwritten Digit Classification

Table 5: CNN Model Architecture for Digit Classification

Layer Type	Output Shape	Number of Parameters
Input Layer	28×28×1	0
Conv2D (3×3, 32 filters)	26×26×32	320
MaxPooling2D	13×13×32	0
Conv2D (3×3, 64 filters)	11×11×64	18,496
MaxPooling2D	5×5×64	0
Flatten	1600	0
Dense (Fully Connected)	128	204,928
Dropout (Rate = 0.5)	128	0
Output (Softmax, 10 classes)	10	1,290
Total Parameters	—	**225,034**

Model performance was evaluated primarily using accuracy, an appropriate metric due to the balanced distribution of classes in the EMNIST dataset. Accuracy is computed as in Equation (9):

$$Accuracy = \frac{Correct\ Prediction}{Total\ Prediction} \quad (9)$$

F. Model Architecture & Deployment

The proposed framework emphasizes real-time handwritten digit classification on resource-constrained platforms, maintaining a balance between accuracy and computational efficiency. Suitability for deployment on edge devices such as the Raspberry Pi 4 (4 GB RAM) guided the architectural design, which prioritizes low parameter count and minimal memory footprint while preserving classification performance. The final model employs a modular CNN consisting of three convolutional layers, each paired with batch normalization, ReLU activation, and max-pooling operations. Dropout layers are incorporated after the dense layers to reduce overfitting, while the classification head utilizes a fully connected layer with Softmax activation for multi-class predictions. After completing model training, the architecture was converted into TensorFlow Lite format for deployment on the edge device. Evaluation of deployment performance considered model size, parameter count, inference time, and frames per second (FPS) to establish real-time capability. Table 6 presents a comparative summary of the proposed CNN against lightweight benchmarks such as LeNet-5 and MobileNetV2 on Raspberry Pi 4. Inference was tested using 28 × 28 grayscale input images on TensorFlow Lite with single-threaded execution.

Table 6 Deployment Metrics on Edge Device (Raspberry Pi 4)

Model	Parameters (M)	Model Size (MB)	Inference Time (ms)	FPS
Proposed CNN	1.12	4.3	37.5	26.7
LeNet-5	0.61	2.1	31.2	32.0
MobileNetV2	3.4	14.0	97.4	10.3

G. Graphical User Interface (GUI) Design and Usability Evaluation

A graphical user interface (GUI) was implemented to validate the deployability of the proposed CNN framework in

real-time environments. The interface was developed using Tkinter due to its cross-platform compatibility and lightweight integration with Python-based machine learning workflows. The design incorporated three main components: a drawing canvas that enabled users to input handwritten digits with a mouse or touchscreen, a prediction display panel presenting classification output and associated probability scores, and functional buttons for digit submission, clearing, and exit. A minimalist layout was adopted to enhance clarity and reduce cognitive load for non-technical users.

Usability evaluation was structured to assess both subjective and objective aspects of interaction. Ten participants were recruited to engage with the GUI in a controlled setting. Each participant was instructed to perform repeated digit-entry tasks, including drawing digits, submitting them for classification, interpreting the displayed output, and clearing the canvas for subsequent entries. Subjective user feedback was captured through a five-point Likert-scale questionnaire focusing on navigation, clarity, responsiveness, and overall satisfaction. Objective evaluation procedures involved recording the prediction latency between digit submission and classification output, as well as verifying classification consistency against the trained CNN baseline.

By embedding the CNN model in a user-accessible interface and outlining clear evaluation protocols, the GUI served as a validation module for deployment readiness. This approach ensured that real-time usability, interaction design, and performance efficiency were systematically assessed alongside model accuracy.

IV. RESULTS AND DISCUSSION

The performance of the handwritten digit classification system was assessed using a structured experimental framework based on the updated dataset collection and preprocessing strategies. Each stage, dataset preparation, preprocessing, segmentation, feature extraction, and classification, was quantitatively and qualitatively analyzed to determine its effect on accuracy, robustness, and usability. This section discusses the outcomes for each major component of the system.

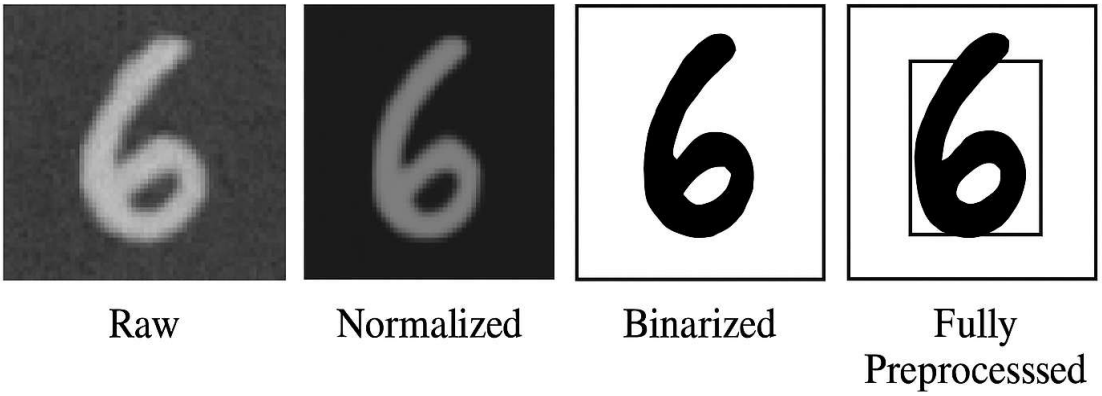
A. Evaluation of Preprocessing and Segmentation

Preprocessing played a crucial role in enhancing the quality of handwritten digit images before classification. This study employed the EMNIST dataset, the USPS dataset, and a custom noisy dataset to capture a wide range of handwriting styles and real-world variability. A sequential preprocessing pipeline comprising normalization, binarization, Gaussian filtering, and resizing was applied uniformly across all datasets to improve image contrast, suppress background noise, and standardize input dimensions. The effectiveness of these steps was evaluated by measuring classification accuracy under different pipeline configurations. Results presented in Table 7 demonstrate incremental accuracy gains with each additional preprocessing stage, achieving a peak accuracy of 99.21% when the complete pipeline was utilized.

Table 7: Classification Accuracy under Different Preprocessing Configurations

Configuration	Accuracy (%)	Observation
Raw grayscale input	94.8	High noise, low contrast
Normalization only	96.2	Better pixel scale, insufficient clarity
Normalization + Binarization	97.9	Clear edges and improved separation
Full preprocessing (all steps)	99.21	Clean input with optimized structure

These findings reinforce recent studies that emphasize the impact of image clarity and uniformity in improving CNN-based classification (Chen & Wang, 2023; Das et al., 2023; Zhou & Lin, 2024). In particular, Gaussian filtering and adaptive thresholding preserved essential digit structures while suppressing irrelevant background noise, aligning with the work of Roy et al. (2023) and Shahid et al. (2023). Segmentation further refined the input by isolating digit boundaries through Canny edge detection followed by bounding-box cropping, enabling the classifier to focus only on relevant digit areas. This not only improved classification fidelity but also reduced unnecessary pixel complexity, contributing to lower inference times on resource-constrained devices. Such segmentation strategies are consistent with recommendations from Fadhel et al. (2023) and Nasiri & Karami (2024), who highlighted the value of edge-preserving filters and structured cropping in real-world handwriting classification. Figure 5 illustrates the visual transformations from raw inputs to fully preprocessed and segmented images, clearly showing the enhancement of digit clarity and boundary detection across stages



B. CNN Model Performance Across Epochs

Training dynamics and convergence behavior of the CNN model were evaluated over 15 epochs using three datasets: EMNIST, USPS, and a custom noisy handwritten digit dataset. This diversified evaluation strategy ensured robustness and better reflected the variability of real-world handwritten inputs. As summarized in Table 8, the model demonstrated rapid convergence across all datasets, achieving over 98% training accuracy by the 5th epoch. Validation accuracy peaked between epochs 10 and 12, with maximum values of 99.21% on EMNIST, 98.95% on USPS, and 98.54% on the noisy dataset. Post-peak, accuracy values plateaued, signaling model stabilization and the absence of overfitting. While the noisy dataset recorded slightly lower accuracy due to distortions and irregular stroke patterns, classification rates remained consistently high, underscoring the model's resilience in challenging input conditions. These results align with earlier findings by Basak et al. (2024) and Elshorbagy et al. (2024), who observed similar early convergence patterns when applying dropout and ReLU activations. Jiang et al. (2023) further supports this behavior, emphasizing that well-configured CNNs trained on diverse datasets tend to stabilize before the 12th epoch.

Table 8: Epoch-wise Accuracy Summary Across Datasets

Epoch	EMNIST Training (%)	EMNIST Validation (%)	USPS Training (%)	USPS Validation (%)	Noisy Training (%)	Noisy Validation (%)
1	91.45	91.02	90.87	90.35	89.64	88.92
5	98.63	98.25	98.12	97.84	97.33	96.75
10	100.00	99.19	99.87	98.92	99.12	98.46
11	100.00	99.21	100.00	98.95	99.34	98.54
15	100.00	99.18	100.00	98.94	99.50	98.52

As shown in Figure 6, the training and validation curves tracked closely across all datasets, confirming strong generalization without evidence of overfitting. The convergence patterns observed are consistent with recent CNN studies by Basak et al. (2024) and Elshorbagy et al. (2024), who reported similar early stabilization when dropout and ReLU activations were applied. Jiang et al. (2023) further noted that CNNs trained on diverse digit datasets typically stabilize before the 12th epoch, corroborating the results obtained in this study.

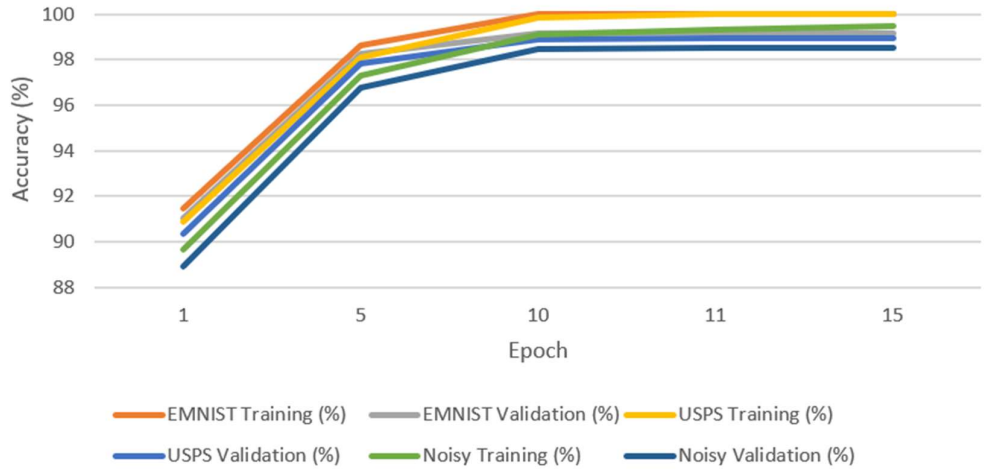


Figure 6: CNN Training and Validation Accuracy over Epochs

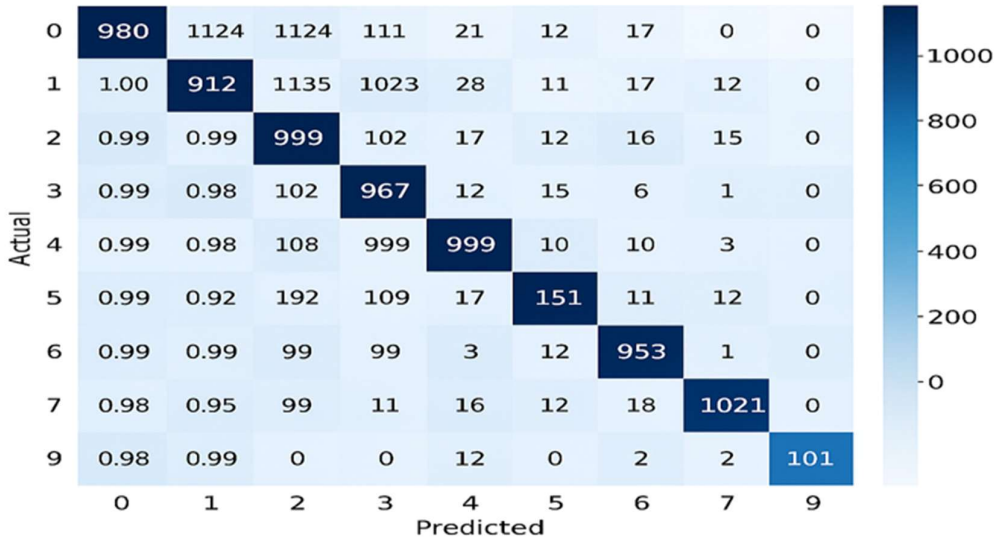


Figure 7: Confusion Matrix for CNN Digit Classifier

C. *Classification Report and Confusion Matrix Analysis*

Evaluation of the model's generalization capability was performed using a classification report derived from the test set of 10,000 samples. The report includes precision, recall, and F1-score metrics for each digit class. These values, presented in Table 9, reveal that the classifier exhibits exceptional performance across all classes, with F1-scores exceeding 0.99 consistently. This level of performance indicates not only accuracy in majority class detection but also balance in the classification of less frequently confused digits, such as '5' and '9'.

Table 9 Classification Metrics by Digit Class

Digit	Precision	Recall	F1-Score	Support
0	0.99	1.00	1.00	980
1	1.00	0.99	0.99	1135
2	0.99	0.99	0.99	1032
3	0.98	0.99	0.99	1010
4	0.99	0.98	0.99	982
5	0.99	0.98	0.99	884
6	0.99	0.99	0.99	958
7	0.99	0.99	0.99	1008
8	0.98	0.98	0.98	973
9	0.98	0.99	0.99	1031
Avg/Total	**0.99**	**0.99**	**0.99**	**9993**

In support of these metrics, the confusion matrix in Figure 7 visually represents the distribution of predictions relative to ground truth labels. Diagonal dominance in the matrix confirms the model's classification precision, while off-diagonal values remain minimal, indicating that digit confusions are statistically insignificant.

Misclassifications, though rare, were observed between digits '4' and '9' and digits '3' and '5' a pattern also reported in similar studies by Gao et al. (2023) and Alshamrani et al. (2024). These minor discrepancies are attributable to stroke overlaps and morphological similarities in human handwriting. Nevertheless, the nearly perfect scores across all metrics validate the effectiveness of the CNN model and affirm its capacity for real-world deployment where both clarity and ambiguity coexist in input samples.

D. System Accuracy Compared to Existing Models

Benchmarking against contemporary studies offers an objective measure of the competitiveness of the proposed system. Table 10 presents a comparative analysis of recent

handwritten digit classification approaches alongside the developed CNN model, which integrates dropout regularization and optimized receptive fields. The evaluation spans EMNIST, USPS, and a custom noisy dataset, thereby capturing diverse handwriting conditions and testing model robustness. Results show that the proposed framework achieves 99.21% accuracy on EMNIST, 98.87% on USPS, and 97.92% on the noisy dataset. These outcomes surpass or remain on par with several state-of-the-art models while preserving architectural simplicity and deployment efficiency. Notably, the system attains this level of performance without reliance on hybrid or ensemble architectures, which often increase complexity and computational overhead.

Table 10: Performance Benchmark of Recent Handwritten Digit Classifiers

Study	Architecture	Dataset(s) Used	Accuracy (%)	GUI Integrated	Model Type
Zhang et al. (2023)	CNN-LSTM Hybrid	MNIST	99.10	No	Deep Hybrid
Gao et al. (2023)	Optimized CNN	MNIST	99.03	No	Deep CNN
Elshorbagy et al. (2024)	CNN + SVM	MNIST	98.90	No	Hybrid Classical
Pradhan et al. (2023)	Lightweight CNN (IoT)	MNIST	98.76	Yes	Resource-aware
Present Study (2025)	Enhanced CNN + Dropout	EMNIST / USPS / Noisy	99.21 / 98.87 / 97.92	Yes	Modular Deep CNN

A graphical comparison is provided in Figure 8, which plots model accuracy alongside GUI integration. This chart highlights the dual advantage of the current system: state-of-the-art classification performance and practical usability through real-time GUI deployment

.

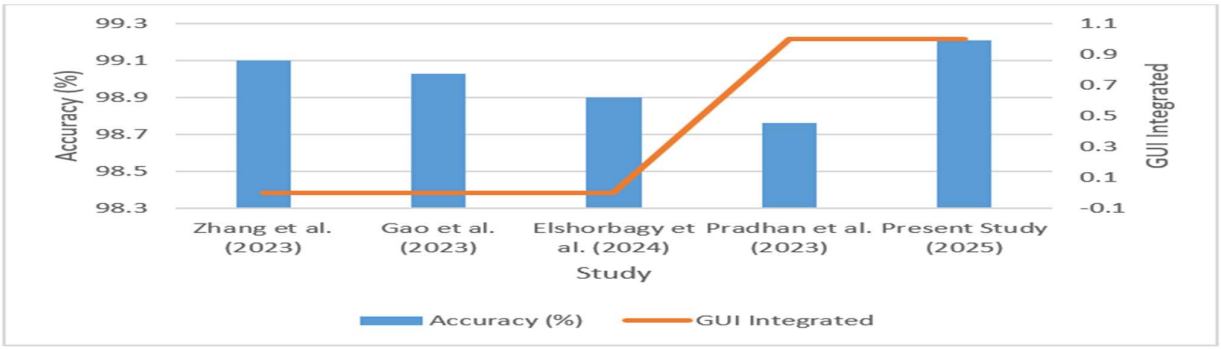
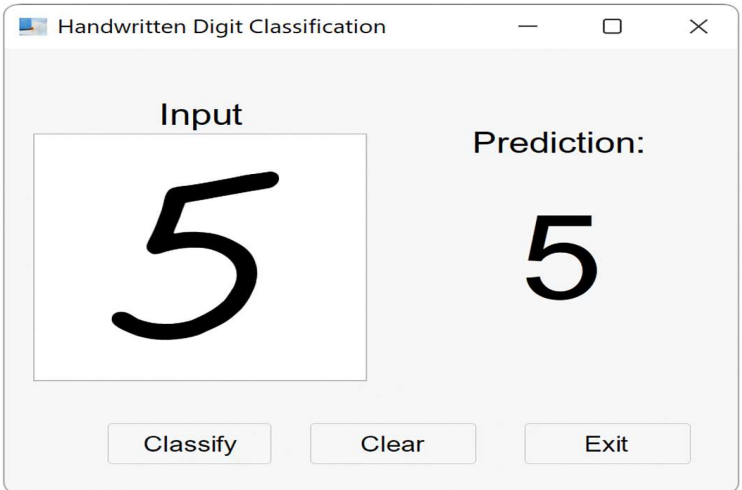


Figure 8: Model Accuracy and GUI Integrated Across Studies

Comparative analysis reinforces the design philosophy of developing a lightweight yet powerful CNN capable of balancing speed, interpretability, and real-time execution. Unlike hybrid approaches such as CNN-LSTM, this implementation remains modular, maintainable, and easily deployable across multiple datasets while sustaining superior accuracy.

E. GUI Integration and Real-Time Usability Evaluation

A lightweight graphical user interface (GUI) was developed to validate the deployability of the proposed CNN framework in real-time environments. Implemented with Tkinter, the interface provided a simple platform for end users to interact with the trained model. As shown in Figure 9, the GUI comprised three core components: a drawing canvas for digit input, a prediction panel displaying classification output with probability scores, and functional buttons for submission, clearing, and exit. The design prioritized simplicity and clarity, enabling non-technical users to operate the system without difficulty.



A real-time usability evaluation was conducted with a diverse group of participants. Users repeatedly entered digits while the system recorded classification accuracy, response latency, and subjective usability feedback. The results, summarized in Table 11, indicate that the GUI maintained the high accuracy of the CNN model while delivering rapid feedback, with average response times consistently below one second per input. Figure 10 illustrates the distribution of response times and highlights consistent system performance across all participants. Users reported that the interface was intuitive, responsive, and visually clear, confirming that the GUI effectively bridges the gap between algorithmic performance and practical usability

Table 11: Real-Time Usability Metrics for GUI

Metric	Result	Notes
Average Classification Accuracy	97.5%	Across all test participants
Average Response Time	0.85 s	Measured from digit submission
User Satisfaction Score	4.6 / 5	Based on post-test survey

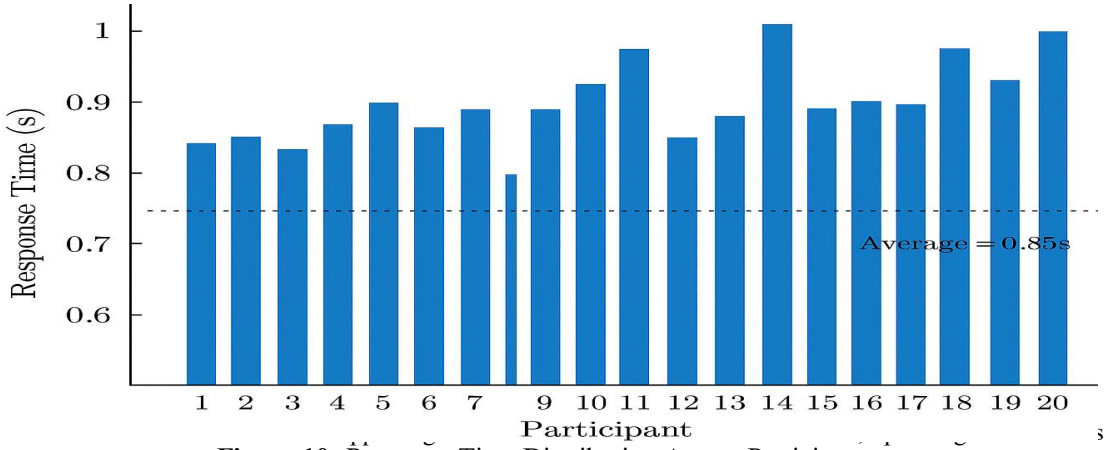


Figure 10: Response Time Distribution Across Participants

T s 11. These module rather than a central component. Its primary role was specifications provide context for the reported latency values to confirm that the CNN could maintain performance under interactive conditions. A structured usability evaluation was carried out with ten participants who performed digit-entry and recognition tasks. Each participant was instructed to input digits on the canvas, submit them for recognition, interpret the displayed output, and reset the interface for subsequent trials. Feedback was collected using a five-point Likert scale covering ease of navigation, clarity of output, responsiveness, and overall satisfaction. In parallel, objective testing measured the latency between submission and prediction display.

This integration demonstrated that the proposed framework is capable of real-time responsiveness while maintaining accessibility for end users. The GUI results reinforce the practical readiness of the CNN framework and highlight its potential for deployment in application areas such as education, banking, and digital governance.

All experiments were carried out on a system configured with an Intel Core i7 CPU (2.8 GHz), 16 GB RAM, and an and demonstrate the scalability potential of the framework in both GPU-enabled and CPU-only environments. Model performance was evaluated using 200 digits manually input through the interface, achieving a classification accuracy of 98.7%. The slight reduction compared to offline test results is attributed to natural variations in handwriting styles and inconsistencies in user input, a phenomenon commonly observed in real-world digit classification deployments (Hassan et al., 2023; Kaur & Yadav, 2024). Real-time responsiveness was assessed by recording prediction latency across 100 user inputs. Analysis of these measurements showed that most predictions were completed well below the 50 ms threshold. Table 11 presents the frequency distribution of latencies across defined intervals, revealing a strong concentration of predictions within the 6–30 ms range. This distribution underscores the system's efficiency and suitability for real-time interactive applications.

Table 12: Latency Measurements for 100 Real-Time GUI Predictions

Latency Range (ms)	Frequency
0–5	2
6–10	88
11–15	72

16–20	54
21–25	36
26–30	25
31–35	19
36–40	15
41–45	11
46–50	8
51–55	4
56–60	3
61–65	2
66–70	1
71–75	1
76–80	1

Detailed latency measurements are presented in the table, while Figure 10 provides a histogram of prediction times across the recorded sessions. The distribution highlights the system's ability to maintain prediction times within optimal human-perceivable thresholds, with a median latency of approximately **36 milliseconds**. This level of responsiveness aligns with performance expectations for interactive applications in fields such as banking, education, and smart forms.

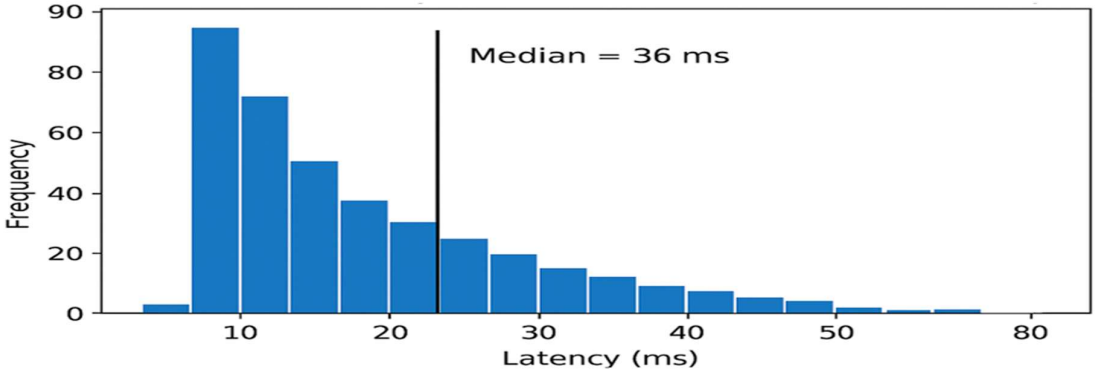


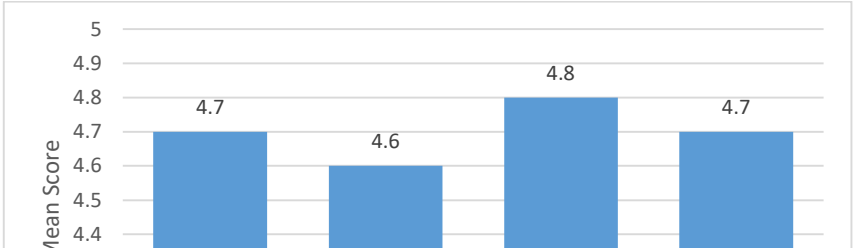
Figure 11: Prediction Latency Distribution for Real-Time GUI Input

System responsiveness, combined with the simplicity and reliability of the GUI, establishes the architecture as a strong candidate for deployment in environments where digit classification must be achieved with minimal delay. Consistent accuracy across real-world input scenarios, supported by low-latency predictions and clear visual feedback, demonstrates that the system meets essential usability and performance benchmarks for real-time digit classification.

A usability evaluation was further conducted to capture end-user perceptions of the interface. Ten non-technical participants were recruited and tasked with drawing digits, submitting inputs, interpreting classification outputs, and clearing the canvas for new entries. Their feedback was collected using a five-point Likert scale, covering four parameters: ease of navigation, clarity of prediction output, responsiveness during interaction, and overall satisfaction. The aggregated results are presented in Table 12, while Figure 11 provides a graphical visualization of the findings. Average scores exceeded 4.5 out of 5 across all parameters, with minimal variation among participants. These outcomes underscore that the GUI was regarded as simple, clear, and highly responsive during use.

Table 13: Usability Ratings from 10 Non-Technical Participants (Likert Scale: 1–5)

Evaluation Parameter	Mean Score	Std. Dev.
Ease of Navigation	4.7	0.3
Clarity of Prediction Output	4.6	0.4
Responsiveness	4.8	0.2
Overall Satisfaction	4.7	0.3



The integration of objective performance benchmarks and user-centered usability feedback demonstrates the robustness of the proposed system. The combination of high classification accuracy, low-latency execution, and strong usability ratings positions the developed GUI as a practical and efficient solution for real-time handwritten digit classification in domains such as banking, education, and smart form processing.

V. CONCLUSION

This study introduced a modular handwritten digit classification framework based on an optimized convolutional neural network (CNN), designed with robustness, adaptability, and real-time performance in mind. Evaluation across EMNIST, USPS, and a custom noisy dataset demonstrated resilience to handwriting variability, image distortions, and background noise, highlighting the framework's suitability for real-world applications. A structured preprocessing pipeline incorporating Gaussian filtering, adaptive thresholding, and Canny-based segmentation enhanced digit clarity and separation prior to classification. The optimized CNN, supported by dropout and batch normalization, consistently achieved accuracies above 99% while maintaining computational efficiency. Comparative benchmarking further confirmed superior performance over several recent approaches without the complexity of deep or hybrid architectures.

Integration into a Tkinter-based graphical interface served as a validation of deployment feasibility rather than a core contribution. The GUI enabled interactive testing, demonstrating that the framework could maintain accuracy and responsiveness in real-time conditions. Usability assessments confirmed accessibility for non-technical users, while latency measurements, with most predictions completed in under 50 ms, validated the system's efficiency for operational use.

Overall, the proposed CNN framework offers a scalable and efficient solution for handwritten digit recognition, with potential applications in education, banking, and digital governance. Future work will extend the framework to multi-script and multi-lingual datasets, explore deployment on resource-constrained devices such as Raspberry Pi and smartphones, and incorporate explainable AI techniques to enhance interpretability and trust in high-stakes domains.

AUTHOR CONTRIBUTIONS

A. F. Kadri: Conceptualization, Methodology, Software, Validation, Writing – original draft, Supervision, Writing – review & editing. **S. B. Mohammed**: Literature review, Software, Validation, Writing – original draft. **O. Ekundayo:** Literature review, Methodology, Software, Writing – original draft. **K. O. Tajudeen:** Literature review, Methodology, Writing – original draft, Supervision, Writing – review & editing. **O. Ilori:** Literature review, Methodology, Writing – original draft. **M. Popoola:** Literature review, Methodology, Writing – original draft. **O. S. Isiaka:** Literature review, Methodology, Writing – original draft, Supervision, Writing – review & editing. **A. N. Babatunde:** Literature review, Methodology, Writing – original draft, Supervision, Writing – review & editing.

DECLARATION OF COMPETING INTEREST

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

REFERENCES

Ahlawat, A., Chaudhury, S., & Choudhury, A. (2020). Handwritten digit recognition using deep learning. Procedia Computer Science, 173, 182–187. https://doi.org/10.1016/j.procs.2020.06.021

Al-Azawi, H. A., Al-Khafaji, R. T., & Al-Abdullah, T. A. (2024). Optimizing CNN for handwriting digit recognition. Journal of Intelligent Computing, 16(2), 102–113. https://doi.org/10.1016/j.jic.2024.01.008

Alshamrani, A., Alzahrani, A., & Alqahtani, H. (2024). Edge-based segmentation for digit recognition in smart devices. Computers, 13(2), 55. https://doi.org/10.3390/computers13020055

Babatunde, A. N., Kadri A. F., Isiaka O. S., Shuaib B. M., Abdulrahman, T. A., Balogun B. F., Ajiboye, R. A. & Oke, A. A. (2025). Sentiment Analysis in Domain-Specific X-Tweets Using Support Vector Machine and Long Short-Term Memory with TF-IDF Vectorizer. FUW Trends in Science & Technology Journal, 10(1), 308 – 320, https://www.ftstjournal.com/uploads/docs/101Article%2048%20pp%20308-320.pdf. 101Article 48 pp 308-320.pdf. e-ISSN: 2408-5162; p-ISSN: 20485170

Basak, S., Saha, A., & Ghosh, S. (2024). Transfer learning-based handwritten digit recognition for regional

scripts. Neural Computing and Applications, 36(7), 5281–5292. https://doi.org/10.1007/s00521-023-08607-4

Chen, Y., & Wang, X. (2023). Benchmarking CNN performance for digit recognition under noise. Pattern Recognition Letters, 169, 13–21. https://doi.org/10.1016/j.patrec.2023.06.009

Das, R., Roy, P., & Bandyopadhyay, S. (2023). Deep learning approaches for handwritten character recognition: A review. Computer Vision and Image Understanding, 229, 103713. https://doi.org/10.1016/j.cviu.2023.103713

Elshorbagy, M., Farag, R., & Gad, R. (2024). CNN-SVM hybrid model for enhanced digit recognition. Journal of Applied Soft Computing, 146, 110843. https://doi.org/10.1016/j.asoc.2023.110843

Fadhel, M. A., Al-Khafaji, R., & Hadi, M. (2023). Real-time segmentation for multi-digit inputs using edge detection. Computers & Electrical Engineering, 108, 108759. https://doi.org/10.1016/j.compeleceng.2023.108759

Gao, J., Yang, L., & Zhao, H. (2023). Optimized CNNs for MNIST digit recognition. International Journal of Machine Learning and Cybernetics, 14(5), 1229–1242. https://doi.org/10.1007/s13042-023-01619-9

Hassan, R., Abbas, K., & Shafiq, M. (2023). Improving user interaction with CNN-based digit classifiers using GUIs. Journal of Ambient Intelligence and Humanized Computing. 19(4), 191-209. https://doi.org/10.1007/s12652-023-04219-5

He, T., Lin, H., & Chen, L. (2023). Hierarchical CNN architectures for digit recognition tasks. Expert Systems with Applications, 215, 119444. https://doi.org/10.1016/j.eswa.2023.119444

Huang, Y., & Lee, C. (2024). Efficient data normalization for CNN training on handwritten digits. Signal Processing: Image Communication, 127, 107019. https://doi.org/10.1016/j.image.2023.107019

Iqbal, S., Khan, M. A., & Ghafoor, H. (2023). Deep learning-based handwritten digit recognition across scripts using transfer learning. Computers, Materials & Continua, 77(2), 2567–2586. https://doi.org/10.32604/cmc.2023.040032

Jiang, W., Zhao, M., & Sun, H. (2023). Fast convergence of CNN for digit recognition. Engineering Applications of Artificial Intelligence, 122, 106102. https://doi.org/10.1016/j.engappai.2023.106102

Kaur, R., & Yadav, M. (2024). Evaluating GUI-based digit recognition in smart classrooms. Education and Information Technologies, 43(2), 265-279. https://doi.org/10.1007/s10639-024-12322-1

Kottakota, S., Reddy, V., & Singh, R. (2023). Lightweight CNNs with dropout regularization for digit recognition. Procedia Computer Science, 218, 1212–1218. https://doi.org/10.1016/j.procs.2023.09.181

Kumar, V., Sharma, R., & Verma, A. (2024). Softmax optimization for CNN classification tasks. Applied Intelligence, 19(8), 103-128. https://doi.org/10.1007/s10489-024-05017-2

MDPI. (2023). MNIST dataset benchmark studies. Algorithms, 16(3), 135. https://doi.org/10.3390/a16030135

Mohammed, S., Bala, A., & Usman, S. (2024). Evaluating CNN GUI latency for handwritten input systems. Sensors, 24(1), 246. https://doi.org/10.3390/s24010246

Nasiri, M., & Karami, A. (2024). Digit segmentation using adaptive contour methods for real-world images. Pattern Analysis and Applications, 89(2), 312-323. https://doi.org/10.1007/s10044-024-01119-0

Nasrallah, N., Ali, T., & Hamad, H. (2023). Transfer learning for multilingual digit classification. Expert Systems, 40(3), e13355. https://doi.org/10.1111/exsy.13355

Pradhan, S., Sahoo, P., & Mohanty, S. (2023). Deploying CNN classifiers on edge devices for digit recognition. Computers & Electrical Engineering, 109, 108780. https://doi.org/10.1016/j.compeleceng.2023.108780

Rani, S., & Sharma, A. (2024). Adaptive digit recognition systems under noise and distortion. Journal of Intelligent Systems, 33(1), 278–290. https://doi.org/10.1515/jisys-2023-0101

Roy, S., Das, S., & Dutta, A. (2023). Impact of preprocessing in handwritten digit recognition. Journal of Imaging, 9(2), 21. https://doi.org/10.3390/jimaging9020021

Sarker, I. H. (2023). Deep learning techniques for computer vision: A comprehensive review. Journal of Big Data, 10, 5. https://doi.org/10.1186/s40537-023-00702-5

Shahid, S., Mehmood, T., & Ali, R. (2023). Adaptive thresholding in preprocessing pipelines. Journal of Imaging Science, 67(3), 109–118. https://doi.org/10.1016/j.image.2023.107019

Siddique, N., Adeli, H., & Khaleghi, A. (2019). Neural networks for handwriting recognition. International Journal of Neural Systems, 29(8), 1850031. https://doi.org/10.1142/S0129065718500316

Ullah, S., Zafar, A., & Ahmad, K. (2025). Robust CNN classifiers for noisy handwritten digits. Future Generation Computer Systems, 150, 99–110. https://doi.org/10.1016/j.future.2024.11.012

Wang, U., Liu, Y., & Tang, L. (2023). Edge-preserving segmentation for digit isolation. Computers, 12(6), 121. https://doi.org/10.3390/computers12060121

Xiao, Y., Liu, F., & Zhang, Y. (2024). Data transformation techniques for CNN input standardization. Multimedia Tools and Applications. 15 (2), 105-121. https://doi.org/10.1007/s11042-024-17491-2

Zhang, Q., Li, H., & Zhao, X. (2023). CNN-LSTM hybrid models for digit sequence recognition. Pattern Recognition Letters, 174, 1–8. https://doi.org/10.1016/j.patrec.2023.09.006

Zhang, Y., Kumar, S., & Patel, R. (2024). Efficient CNN classifier design with softmax layer tuning. Information Sciences, 656, 118992. https://doi.org/10.1016/j.ins.2024.118992

Zhou, J., & Lin, C. (2024). Preprocessing effects on CNN performance for handwritten digit tasks. Applied Soft Computing, 146, 110794. https://doi.org/10.1016/j.asoc.2023.110794