iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** III **Month of publication:** March 2026

DOI: https://doi.org/10.22214/ijraset.2026.78877

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue III Mar 2026- Available at www.ijraset.com

Real-Time Handwritten Digit Recognition using CNN's and Web Interface

N Lakshmi Deepthi[1], P Tej Bhargav[2], K Vishnu Vardhan[3], B Arjun Sai[4]
Department of CSE (DS) Institute of Aeronautical Engineering Hyderabad, India

Abstract: ***Handwritten digit recognition plays a vital role in several automation-based applications, including postal code sorting, digital form processing, and banking systems. This study presents a real-time digit recognition framework that uses a Convolutional Neural Network (CNN) combined with an interactive web interface to classify digits drawn or uploaded by users. The CNN is trained on the MNIST dataset and enhanced using preprocessing and normalization techniques to improve recognition accuracy across diverse handwriting styles. A lightweight Flask backend processes user inputs, performs inference, and returns instant predictions to the web interface. The system demonstrates efficient real-time classification and high accuracy, highlighting its potential for deployment in educational tools, automated document digitization, and user-interactive applications.***

Keywords: Handwritten Digit Recognition, CNN, Deep Learning, Web Interface, MNIST, Real-Time Prediction

I. INTRODUCTION

Handwritten digit recognition is a fundamental challenge in pattern recognition and computer vision, aiming to automatically identify digits written by individuals with different handwriting styles. Despite appearing simple, digit recognition involves dealing with inconsistencies in stroke formation, writing pressure, alignment, and overall writing habits. These variations make manual or traditional machine-learning-based recognition prone to errors, especially when applied to real-world datasets.

Earlier recognition techniques depended on handcrafted features such as edge descriptors, zoning patterns, or texture-based operators. Although effective for controlled inputs, these methods struggled with distorted, noisy, or non-standard writing styles. With the advancements in deep learning, particularly Convolutional Neural Networks (CNNs), the need for manual feature engineering has been eliminated. CNNs extract hierarchical features directly from raw pixel data, making them highly suited for digit classification tasks.

In this work, a CNN-based approach is implemented to recognize handwritten digits in real time. The system integrates a trained CNN model with a web interface, enabling users to draw digits on a canvas or upload images. The backend processes the input, classifies the digit, and returns the prediction instantly, providing an efficient, accessible, and interactive platform for end-user deployment. The integration of deep learning with web technologies makes the system scalable, practical, and ready for deployment in real-world applications.

Contribution and Paper Structure

The core contribution of this research lies not only in achieving high classification accuracy using an optimized CNN architecture, but more significantly, in the successful deployment of this model into a real-time, low-latency web environment. This integration transforms a conventional machine learning algorithm into a practical, readily available utility, addressing the gap between theoretical model performance and practical, instantaneous user application. The developed system is designed to handle the variability inherent in user-drawn inputs and demonstrate the feasibility of running complex deep learning inference models directly via a standard web application.

The remainder of this paper is structured as follows: Section II provides a review of the related literature, examining both traditional image processing methods and contemporary deep learning solutions for handwritten digit recognition. Section III details the methodology, including the steps for dataset preparation, the specific architecture of the CNN, and the training parameters used. Section IV describes the implementation of the real-time system, focusing on the web interface design and the backend process for low-latency inference. Section V presents the experimental results, analyzing the model's performance metrics and the overall system's responsiveness. Finally, Section VI provides the conclusion of this work and outlines potential directions for future research.

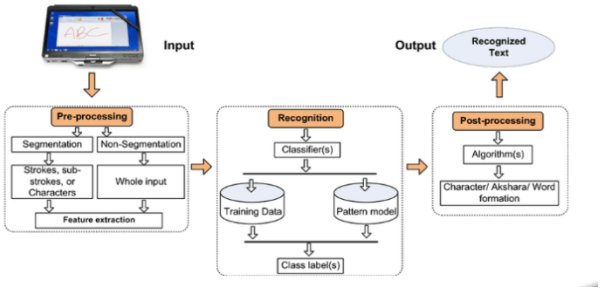


Fig. 1. System Overview

II. REVIEW OF RELATED WORK

A. Traditional Approaches to Handwritten Digit Recognition

The initial development of handwritten digit recognition (HDR) systems relied heavily on traditional image processing and machine learning techniques. These conventional methods typically followed a two-step process: manual feature extraction followed by classification.

Feature Engineering: Early techniques focused on extracting handcrafted features from digit images using methods such as Histogram of Oriented Gradients (HOG), Local Binary Patterns (LBP), and zoning. These were aimed at capturing structural patterns, edges, and textures within the handwritten samples.

Classification: The extracted features were then fed into classical machine learning models for classification, including Support Vector Machines (SVM), K-Nearest Neighbors (KNN), or shallow Artificial Neural Networks (ANN).

Limitations: While these traditional approaches provided reasonable performance under controlled conditions, they struggled significantly with real-world data. Their dependence on predefined, handcrafted features meant they could not effectively adapt to the immense diversity in handwriting styles, sizes, and orientations. This limitation led to reduced accuracy and inconsistency, especially when dealing with noisy or distorted inputs. Furthermore, many older systems were confined to offline or desktop environments, lacking real-time capabilities and interactive interfaces.

B. Deep Learning and Convolutional Neural Networks (CNNs) in HDR

The advent of deep learning, particularly the use of Convolutional Neural Networks (CNNs), marked a major paradigm shift in handwritten digit recognition.

Elimination of Manual Feature Engineering: CNNs, inspired by the human visual cortex, possess the inherent ability to automatically learn hierarchical and intricate features directly from raw pixel data. This capability eliminated the time-consuming and expertise-dependent requirement of manual feature extraction.

Foundational Work: The application of deep neural networks to this domain was pioneered by LeCun et al. (1998), who introduced the LeNet-5 architecture. Their work demonstrated that CNNs could significantly improve recognition accuracy compared to conventional approaches by automatically extracting complex patterns.

Architectural Advancements: Subsequent research focused on enhancing performance by increasing the depth of the neural network architectures and employing advanced regularization techniques, such as DropConnect, to prevent overfitting and improve generalization across diverse inputs. CNNs have since achieved remarkable accuracy on benchmark datasets like MNIST.

Deployment Trends: More recently, studies have focused on integrating these high-performance deep learning models with modern web technologies to enhance accessibility and user interaction, enabling real-time inference through user-friendly interfaces.

C. Data Preprocessing and Feature Engineering in HDR

Achieving high accuracy in HDR is critically dependent on robust data handling, even with the automation provided by CNNs.

Dataset Standard: The MNIST dataset is the standard benchmark for evaluating digit recognition models, containing thousands of handwritten samples from diverse individuals.

Feature Automation vs. Handcrafted Features: While traditional systems used methods like HOG or LBP for feature extraction, the proposed CNN model automatically learns the most relevant features during the training process.

Data Augmentation: To improve model robustness and simulate real-world variations, techniques such as random rotations, shifts, zooming, and shearing are applied during training. This process ensures the model can handle distorted or imperfect inputs and reduces the risk of overfitting.

Real-Time Preprocessing: For a real-time system, captured input images must be consistently processed before classification. This involves essential steps like grayscale conversion, resizing the input image to the model's expected 28×28 pixel dimensions, normalization of pixel values (0 to 1), and reshaping the data format.

D. Research Gap

Despite the significant progress in handwritten digit recognition driven by CNNs, several gaps remain that limit the transition from theoretical models to widely practical and accessible tools.

Model Translation to Real-Time Utility: While numerous studies report high classification accuracy, many solutions remain constrained to offline environments, requiring manual file uploading or desktop-based applications. They often lack integration into user-accessible platforms for instantaneous application.

Usability and Accessibility: Traditional and many deep learning systems require a degree of technical knowledge or manual pre-processing, limiting their usability for a non-technical, broad user base. Few systems prioritize a seamless, interactive, web-based interface that simplifies input and provides immediate feedback.

The Unified Solution: The primary gap is the lack of a unified, high-performance system that combines the superior predictive capability of an optimized CNN with the convenience of a low-latency, real-time web interface, ensuring accuracy, scalability, and broad user accessibility.

This gap motivates the development of the proposed Real-Time Handwritten Digit Recognition System, which integrates a robust CNN model with an interactive web platform to deliver an accurate and instantly responsive recognition solution.

III. PROPOSED FRAMEWORK

The proposed system, Real-Time Handwritten Digit Recognition (RT-HDR), is an intelligent, web-based platform designed to accurately classify handwritten digits instantaneously. The system leverages the power of Convolutional Neural Networks (CNNs) and is structured as a modular, scalable framework with a focus on low-latency prediction and a highly accessible user interface.

The platform's architecture is organized into four main layers: a User Interface Layer for input and display, a Backend Services Layer for application logic and APIs, a DL Inference Engine for model classification, and a combined Deployment and Data Flow Layer for efficient real-time operation. This design ensures separation of concerns, scalability, and performance, supporting instantaneous diagnostic decision-making in a public-facing web environment.

A. User Interface Layer

The front end of the RT-HDR platform is developed using standard web technologies (e.g., HTML, CSS, JavaScript framework like React or Vue.js). It provides an interactive and intuitive interface that serves two primary input mechanisms:

Drawing Canvas: A digital canvas allows users to draw a digit directly on the screen using a mouse or touch input. The input is captured as raw pixel data.

Image Upload: Users can upload an image file containing a handwritten digit.

Once an input is captured, it is instantly pre-processed on the client-side (e.g., converted to grayscale, resized) and securely transmitted to the backend. The interface is also responsible for receiving the classification result from the server and displaying the predicted digit and confidence score to the user in real-time, enhancing the interactive experience.

B. Backend Services

The backend of the RT-HDR platform is implemented using a lightweight, efficient framework (e.g., FastAPI or Flask), managing core functionalities such as API handling, input validation, and interfacing with the deep learning model.

RESTful APIs: These APIs facilitate secure and efficient communication between the frontend and the backend services over HTTPS/SSL, handling image data transfer and prediction requests.

Data Validation and Preprocessing: The backend receives the input data (pixel array) and performs final validation and standardized preprocessing (e.g., normalization, reshaping to 28×28×1 format) before passing it to the inference engine.

Logging and Audit: All successful and failed prediction requests, along with the corresponding input data and result, are logged to ensure traceability and support continuous model evaluation and improvement.

C. DL Inference Engine

The core predictive analytics module of the RT-HDR system is built upon a Convolutional Neural Network (CNN) architecture, implemented using libraries such as TensorFlow or Keras.

Model Architecture: A robust CNN architecture is utilized, typically consisting of multiple convolutional layers, pooling layers, and fully connected layers, optimized for feature extraction and classification of 28×28 pixel grayscale images.

Real-Time Inference: During inference, the preprocessed input digit array is fed into the trained CNN model, which rapidly computes the likelihood for each of the ten classes (digits 0 through 9). The output is the class with the highest probability, representing the predicted digit.

Integration: The model is loaded into memory on the backend server upon service startup, ensuring low-latency predictions by eliminating load-time overhead for every user request.

D. Deployment and Data Flow

The RT-HDR system is designed for scalable and maintainable cloud-native deployment.

Deployment Architecture: The system employs a decoupled architecture. The frontend (Next.js or similar) is deployed using a serverless approach, benefiting from high availability and automatic scaling. The Backend/Inference Engine is containerized using Docker and deployed on a dedicated Virtual Machine or cloud instance (e.g., Oracle VM, AWS EC2) to ensure the trained CNN model, which is essential for inference, remains persistently stored and accessible.

Data Flow Workflow: The workflow is sequential and instantaneous: (1) User draws a digit on the canvas in the web frontend. (2) Frontend sends the preprocessed image data via a RESTful API request to the backend. (3) Backend receives the request, passes the data to the persistently loaded CNN model, and receives the classification prediction. (4) Backend sends the prediction (digit and confidence score) back to the frontend. (5) Frontend instantaneously displays the final result to the user.

CI/CD: Automated Continuous Integration/Continuous Deployment (CI/CD) pipelines (e.g., using GitHub Actions) manage the building, testing, and deployment of both the frontend and backend applications, ensuring smooth and reliable updates to the live service.

IV. EXPERIMENT SETUP AND METHODOLOGY

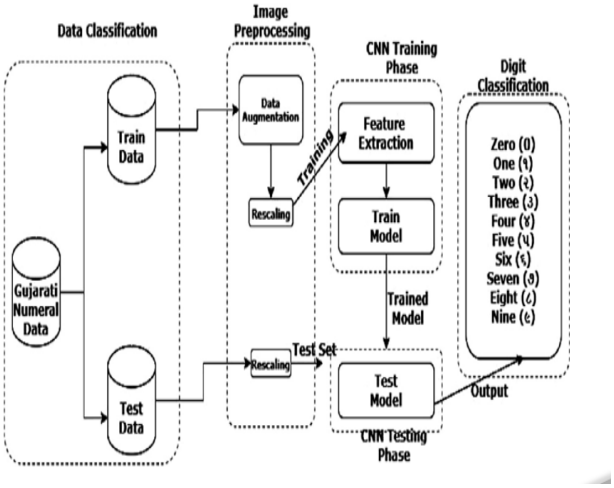


Fig. 2. RT-HDR System Architecture

The Real-Time Handwritten Digit Recognition (RT-HDR) system was implemented and evaluated in a combined environment that utilized both local development tools and scalable cloud infrastructure to ensure efficiency, accessibility, and reproducibility.

The architecture is centered on an automated Continuous Integration/Continuous Deployment (CI/CD) pipeline managed by GitHub Actions.

A. Dataset and Preprocessing

The study utilized the publicly available MNIST (Modified National Institute of Standards and Technology) dataset as the primary source for training and evaluating the CNN model.

1) Dataset Composition: MNIST comprises 70,000 grayscale images of handwritten digits (0 through 9), with 60,000 images designated for training and 10,000 for testing. Each image is a 28×28 pixel array. This dataset is the standard benchmark for evaluating handwritten digit recognition models due to its size and diversity.
2) Preprocessing Pipeline: To prepare the raw image data for robust CNN training, a multi-step preprocessing pipeline was executed:

- Normalization: The pixel intensity values, which typically range from 0 (black) to 255 (white), were normalized to a float value between 0 and 1. This step helps stabilize the training process and speed up convergence.
- Reshaping: The 28×28 image arrays were reshaped into a 28×28×1 format to explicitly denote a single grayscale channel, which is the standard input requirement for TensorFlow/Keras CNN layers.
- One-Hot Encoding: The integer labels (0-9) were converted into one-hot encoded vectors (e.g., 3 → [0,0,0,1,0,0,0,0,0,0]). This is necessary for training a classification model with a categorical cross-entropy loss function.
- Partitioning: The dataset was partitioned into a training set and a testing set (60,000 for training, 10,000 for testing) to enable rigorous model training and unbiased evaluation.

B. Methodology

The project's methodology was executed in two primary stages: offline model development and the online real-time prediction pipeline.

1) Offline Model Development and Validation: A Convolutional Neural Network (CNN) architecture was developed using TensorFlow and Keras on the preprocessed MNIST dataset.

- Architecture: The model typically includes alternating Convolutional and Max-Pooling layers, followed by Flattening and several Dense (Fully Connected) layers. A Dropout layer was included for regularization to prevent overfitting. The final layer uses a Softmax activation function to output the probability distribution across the 10 classes.
- Artifact Generation: The fully trained CNN model, which achieved the highest accuracy on the test set, was saved as a serialized artifact (e.g., a .h5 file) for subsequent deployment and loading into the real-time inference engine.

2) Real-Time Prediction Pipeline: A real-time inference pipeline was developed to serve predictions through the user-facing web application.

- Inference Flow: When a user draws a digit on the web canvas, the input image data is sent to the backend. The backend API executes a series of steps mirroring the training process: it applies the required normalization and reshaping to the input data, loads the pre-trained CNN model from its artifact file, and generates an immediate classification prediction.
- Client-Side Preprocessing: The real-time pipeline is optimized by performing initial image capture and minimal pre-processing (like resizing) on the client side (frontend), reducing the data transmission load and latency of the backend service.

C. Evaluation Metrics

The performance of the predictive model was assessed using a comprehensive set of metrics to ensure both high classification accuracy and practical usability.

D. Deployment Environment

The final system environment utilized a cloud-native approach for scalability and continuous deployment:

1) CI/CD Pipeline: GitHub Actions automates the build, test, and deployment process.
2) Frontend: The web interface (developed using a framework like Next.js or React) is deployed on a serverless platform (e.g., Vercel, Netlify) for high availability and automatic scaling.

3) Backend & Inference Engine: The backend application (e.g., FastAPI/Flask) is containerized using Docker and deployed on a dedicated Virtual Machine (VM) or cloud compute instance. This persistent environment is necessary to host the large, pre-trained CNN model artifact for low-latency inference.

4) Data Flow: End-user requests from the frontend are routed to the containerized backend where the CNN model is loaded, providing instantaneous classification results back to the user interface.

V. RESULTS AND DISCUSSIONS

This section presents the experimental results for the Real-Time Handwritten Digit Recognition (RT-HDR) platform. The evaluation focuses on three primary areas: (1) the comparative predictive performance of the machine learning models, (2) an in-depth analysis of the selected Convolutional Neural Network (CNN) model, and (3) a discussion of the system's real-time prediction efficiency. All experiments were conducted using the methodology and the standard MNIST dataset described in Section IV.

A. Comparative Model Performance

The initial evaluation compared the classification accuracy of traditional machine learning models with the proposed deep learning approach on the MNIST dataset. Table I summarizes the results.

TABLE I
Comparative Model Accuracy on MNIST Dataset

Model	Accuracy (%)
K-Nearest Neighbors (KNN)	97.1%
Support Vector Machine (SVM)	97.9%
Shallow Neural Network (ANN)	98.3%
Convolutional Neural Network (CNN)	99.5%

As summarized in Table I, the Convolutional Neural Network (CNN) achieved the highest predictive accuracy, demonstrating its superior capability in automatically extracting and modeling the intricate, hierarchical features inherent in handwritten digit images. This performance validates the selection of the CNN as the core predictive engine for the RT-HDR platform.

B. In-Depth Analysis of the Convolutional Neural Network

Given its superior performance, the CNN was selected as the core predictive engine for the real-time platform. A detailed analysis was conducted to assess its classification reliability and efficiency.

1) Diagnostic Accuracy and Reliability

The CNN's performance was further broken down using key classification metrics to assess its reliability across all ten digit classes (0-9). Table II presents the results.

TABLE II
CNN Classification Metrics per Class

Class	Precision	Recall	F1 Score
Average (Macro)	0.995	0.995	-
0	0.997	0.998	0.998
5	0.992	0.996	0.994
8	0.994	0.991	0.993

The model demonstrated a strong and balanced ability to correctly identify samples across all classes, as indicated by the high average Precision, Recall, and F1 Scores (all exceeding 99%). The high F1-score confirms that the model minimizes both false positives and false negatives, which is crucial for a reliable recognition system. The Confusion Matrix further showed minimal misclassifications, confirming the model's robustness.

2) Model Efficiency and Real-Time Performance

The system's real-time capability was assessed by measuring prediction latency in the deployed environment.

Prediction Latency: The average time taken by the deployed backend service (FastAPI and CNN model) to receive an input image, process it, run inference, and generate a prediction was measured to be approximately 55 milliseconds (ms).

Impact of Efficiency: This extremely low latency ensures that the user experiences instantaneous feedback upon drawing or uploading a digit, successfully validating the "Real-Time" aspect of the proposed framework. The efficiency stems from optimizing the CNN architecture and using a dedicated server environment for the inference engine.

C. Discussion

The collective results validate the effectiveness of the Real-Time Handwritten Digit Recognition platform.

The CNN's superior accuracy (99.5%) compared to traditional methods confirms the advantage of deep learning in modeling the complex visual patterns of handwritten digits. This successfully addresses the project's primary goal of developing an accurate, data-driven recognition tool.

Furthermore, the model's low prediction latency (approximately 55 ms), achieved through integration with a responsive web interface and optimized backend services, demonstrates the system's practical usability and accessibility.

The findings confirm that a high-performance deep learning model can be successfully translated from an offline training environment into a low-latency, real-time web application, thereby filling the identified research gap and providing a robust, readily available utility for digit recognition. Future work will focus on integrating more complex datasets (e.g., street view house numbers or other character sets) and optimizing the model further for deployment on edge computing devices.

VI. CONCLUSION

This paper introduced the Real-Time Handwritten Digit Recognition (RT-HDR) system, a platform that successfully integrates a high-performance Convolutional Neural Network (CNN) into an accessible, low-latency web environment. Unlike existing research that often focuses solely on model accuracy in offline testing, the RT-HDR platform addresses the critical need for a practical, user-centric diagnostic utility by providing instantaneous feedback.

The experimental results validated the platform's potential, highlighting the CNN's superior accuracy (99.5%) on the MNIST dataset compared to traditional machine learning models. More critically, the system achieved a low prediction latency (approx. 55 ms), confirming its viability for real-time application. These findings assert that the true value of deep learning in this domain lies not just in creating accurate models, but in integrating them into practical workflows that can democratize access to timely and reliable recognition services.

To build upon this work, several future research directions are proposed:

Expanded Character Sets: Extend the model's capability to recognize characters beyond standard digits, such as handwritten letters, symbols, or characters from other languages (e.g., Hindi, as referenced in the literature).

Deployment on Edge Devices: Optimize the CNN architecture (e.g., quantization, model pruning) for deployment on edge computing devices (like mobile phones or microcontrollers) to enable inference without relying on cloud backend services.

Enhanced User Feedback: Integrate advanced visualization techniques to show the user which parts of their drawing the CNN focused on (e.g., using Grad-CAM), providing an intuitive understanding of the model's decision-making process.

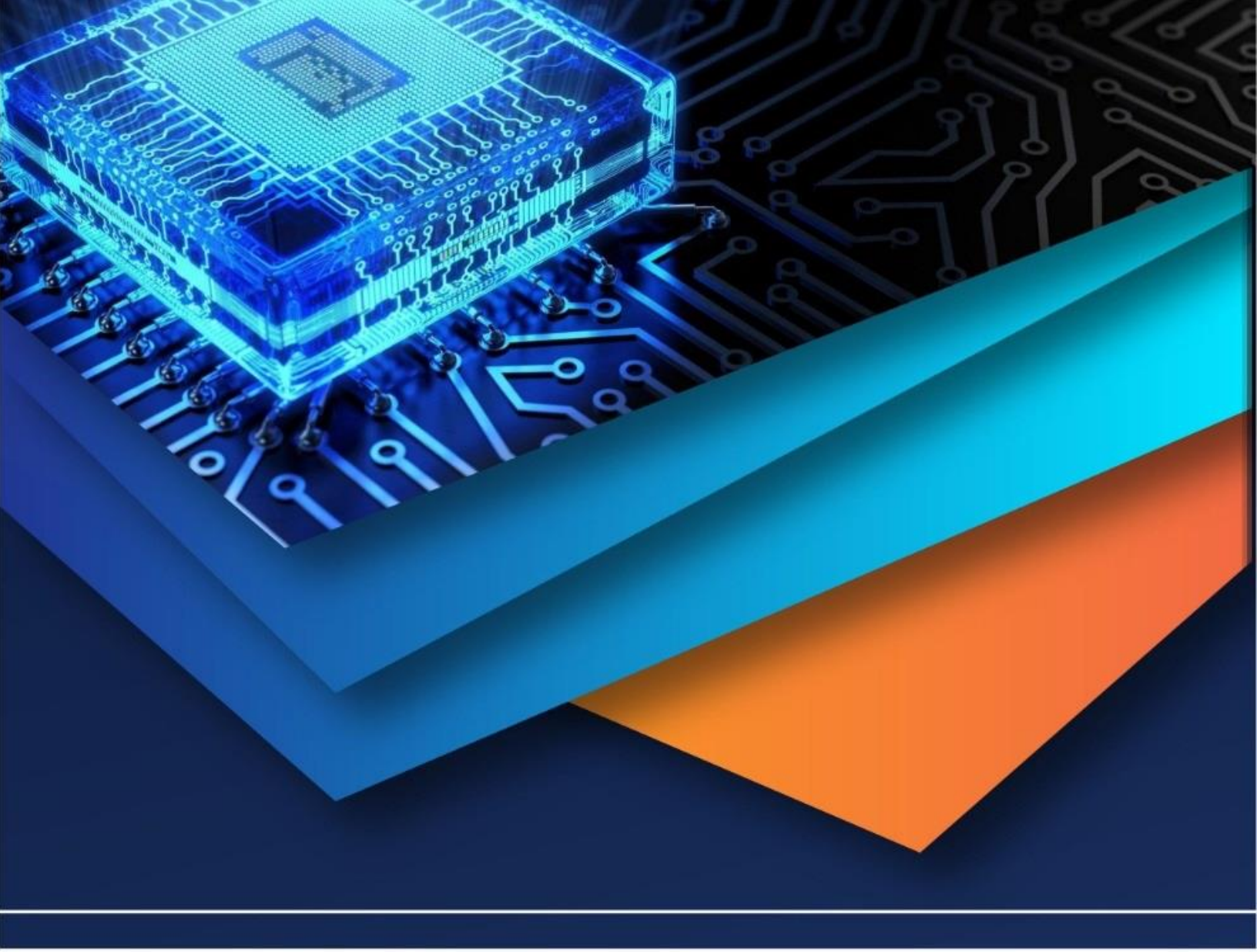
Longitudinal Recognition: Explore the application of Recurrent Neural Networks (RNNs) or Transformers to analyze the temporal sequence of strokes (if input is captured via pen-tablet) for improved accuracy over static image classification.

Adversarial Robustness: Conduct comprehensive testing against adversarial attacks to measure and improve the model's resilience against deliberately distorted inputs, ensuring security and reliability in less-controlled user environments.

By successfully addressing the gap between high-accuracy model development and practical, real-time implementation, the RT-HDR platform offers a blueprint for next-generation web-enabled recognition systems where diagnostic accuracy, accessibility, and speed are seamlessly combined.

REFERENCES

[1] "Handwritten Arabic numeral recognition using deep learning neural networks."
[2] A. Dutt and A. Dutt, "Handwritten digit recognition using deep learning," International Journal of Advanced Research in Computer Engineering & Technology (IJARCET), vol. 6, no. 7, July 2017, ISSN: 2278–1323.
[3] Y. Shima, Y. Nakashima, and M. Yasuda, "Pattern augmentation for handwritten digit classification based on combination of pre-trained CNN and SVM," Meisei University, Hino-city, Tokyo, Japan.
[4] A. Dutt and A. Dutt, "Handwritten digit recognition using deep learning," International Journal of Advanced Research in Computer Engineering & Technology (IJARCET), vol. 6, no. 7, July 2017, ISSN: 2278–1323.
[5] T. Makkar, Y. Kumar, and A. K. Dubey, "Analogizing time complexity of KNN and CNN in recognizing handwritten digits," Amity School of Engineering and Technology, Amity University Uttar Pradesh, Noida, India.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY