Int. J. Advanced Networking and Applications
Volume: 17 Issue: 04 Pages: 7036-7044 (2026) ISSN: 0975-0290
https://doi.org/10.35444/IJANA.2025.17406

TinyML-Powered Handwritten Digit Recognition Device for the Visually Impaired

Abdullateef Ogundipe
Department of Electrical and Electronics Engineering, University of Ibadan, Ibadan, Nigeria
Email: a.ogundipe@ui.edu.ng
Temiloluwa Ifeoluwapo Oloye
Department of Electrical and Electronic Engineering, University of Ibadan, Ibadan, Nigeria
Email: temiloluwaoloye@gmail.com
Abdul Rasak Zubair
Department of Electrical and Electronic Engineering, University of Ibadan, Ibadan, Nigeria
Email: ar.zubair@ui.edu.ng

----------**ABSTRACT**----------
Accessing information in an easily understandable format remains a significant challenge for visually impaired individuals. Conventional handwritten digit recognition systems often rely on computationally intensive models, limiting their deployment on portable, low-cost devices. This paper presents a TinyML-based system for real-time handwritten digit recognition designed specifically to assist visually impaired users. Leveraging a Convolutional Neural Network (CNN) deployed on a Raspberry Pi, the system delivers accurate digit recognition with both visual and audio feedback, enabling independent identification of hand-written digits encountered in everyday activities. The integrated solution combines a Pi camera module, a 3.5-inch display, and an audio speaker, resulting in a compact, portable, and user-friendly device. The methodology involves training the CNN on a curated dataset and optimizing it for edge deployment using TinyML techniques, ensuring low-latency and energy-efficient operation. Experimental results demonstrate the system's capability for efficient and reliable digit recognition, highlighting its potential to enhance accessibility and empower visually impaired individuals through affordable, real-time assistive technology.
Keywords - **Handwritten digits recognition, Convolutional Neural Network (CNN), Visual impairment, Raspberry Pi, TinyML, Assistive technology development**

Date of Submission: October 26, 2025 Date of Acceptance: December 12, 2025

1. INTRODUCTION

Tiny Machine Learning (TinyML) refers to the deployment of machine learning models on resource- constrained devices, typically embedded systems with low- power processors and limited memory [1]. By performing inference directly on these devices, TinyML enables real- time decision-making without constant connectivity to cloud servers, improving privacy, reducing latency, and conserving bandwidth [2].

Handwritten digit recognition has been a central research topic in machine learning and computer vision for decades, particularly in applications aimed at assisting visually impaired individuals. Traditional approaches, such as back-propagation networks, have been widely applied for this task [3]. Support Vector Machines (SVMs) have also been explored, though they often underperform compared to Convolutional Neural Networks (CNN) due to dataset size limitations [4]. Comparatively, Multilayer Perceptron (MLP) achieve higher accuracy in digit classification [5].

CNN are highly effective for image recognition due to their ability to automatically extract hierarchical features from raw pixel data. They have been successfully applied in object detection, facial recognition, and medical imaging tasks [6], [7]. CNN process images through multiple layers, including convolutional, pooling, and fully connected layers, where lower-level features often provide high discriminative power [8]. Multi-Attention CNN (MA-CNN) further improve fine-grained recognition by incorporating attention mechanisms, channel grouping, and part classification sub- networks, achieving state-of-the-art performance without requiring manual annotations [9], [10].

Development of assistive technology for the old and persons with desirability is an active area of research [11]. Recent advancements in TinyML have enabled the development of assistive technologies for the visually impaired. Navigation aids using touchscreens typically provide vibration, audio feedback, or both, with audio feedback offering faster and more efficient responses for many users [12], [13]. Previous works, while effective, either rely on computationally intensive models or lack real-time portability and multimodal feedback suitable for daily use, indicating a clear gap in

Volume: 17 Issue: 04 Pages: 7036-7044 (2026) ISSN: 0975-0290
https://doi.org/10.35444/IJANA.2025.17406

accessible, portable, and efficient handwritten digit recognition systems.

This paper presents a TinyML-based system for handwritten digit recognition specifically designed to aid visually impaired individuals. The proposed system leverages a CNN model deployed on a Raspberry Pi 4 Model B, integrating a Pi camera module, a 3.5-inch display, and an audio speaker to deliver both visual and audio feedback in real-time. The development of a lightweight, portable, and low-cost TinyML solution for handwritten digit recognition, integration of multimodal feedback mechanisms for enhanced accessibility, and demonstration of real-time performance and accuracy suitable for practical use by visually impaired users are the key contributions of this paper.

2. RELATED WORK

Alajlan and Ibrahim provided a comprehensive review of the TinyML revolution, highlighting its capability to enable deep learning inference on ultra-low-power devices [14]. Various datasets and microcontroller units were analysed. Shifting from cloud-based processing to local device processing significantly reduces latency, conserves bandwidth, and enhances data privacy. The theoretical foundation and benefits of TinyML such as energy efficiency and low cost were established.

Lamaakal et al. introduced a Tiny Deep Learning (TinyDL) model specifically designed for recognizing Arabic numerals and letters through air handwriting [15]. Utilizing an Arduino Nano 33 BLE Sense equipped with inertial measurement unit (IMU) sensors, a high recognition accuracy of 97.5% was achieved by capturing motion data rather than visual data. The complexity of Arabic script was effectively addressed without requiring a camera; the model relies on the user performing specific gestures in the air, which differs from the need to recognize static written text on paper that visually impaired individuals often encounter.

Saha et al. developed a low-cost system for controlling appliances using static hand gestures, employing an ESP32-CAM module [16]. The two-stage Convolutional Neural Network (CNN) approach successfully detects palms and classifies gestures with over 92% accuracy from a distance, proving the viability of vision-based TinyML on resource-constrained microcontrollers. However, the solution focuses on home automation and controlling electrical loads rather than providing descriptive feedback for visually impaired users. Additionally, while the system uses a camera, it is optimized for specific hand shapes rather than the intricate feature extraction required for character or digit recognition.

Chittepu et al. explored the integration of TinyML with voice assistants to enhance user interfaces in healthcare and smart home domains [17]. The advantages of processing voice commands locally to mitigate privacy concerns and latency associated with cloud-based assistants were highlighted. While audio feedback is crucial, current TinyML voice models face challenges with noise interference and computational constraints on microcontrollers. The solution in this work underscores the importance of audio feedback mechanisms but focuses on speech processing rather than computer vision tasks for reading assistance.

Kunhoth et al. proposed "VisualAid+," a wearable assistive system that combines a Raspberry Pi for object detection with a server-based module for image captioning [18]. The system employs a hierarchical approach where simple person detection runs on a microcontroller with recognition accuracy of 81.15%, but complex visual scene narration relies on internet connectivity to a remote server. While effective for general environmental awareness, the dependency on a server for complex tasks introduces potential latency and connectivity requirements that may limit usability in offline scenarios. Furthermore, the system is designed for general object detection rather than the specific, high-precision task of recognizing handwritten digits.

In this paper, a TinyML-based handwritten digit recognition system specifically designed to aid visually impaired individuals. Unlike systems that rely on cloud connectivity or server-side processing, the proposed solution utilizes a CNN model deployed directly on a Raspberry Pi 4 Model B to perform inference entirely on the edge. This ensures complete data privacy and functionality even without an internet connection. By integrating a Pi camera for capturing input and providing both visual verification via a 3.5-inch display and distinct audio feedback through a speaker, the proposed system offers a robust, multimodal, and fully portable assistive tool.

3. GENERAL SYSTEM DESIGN

3.1 Hardware components

The proposed system assists visually impaired individuals in recognizing handwritten digits using TinyML technology. It features a CNN model deployed on a Raspberry Pi 4 Model B, interfaced with peripherals for image capture, visual output, and audio feedback as illustrated in Fig. 1.

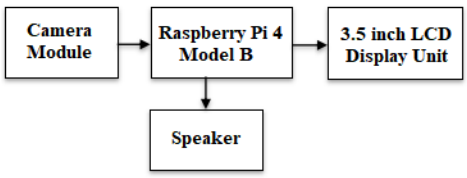

Fig. 1: System overview of the TinyML-based handwritten digit recognition system.

The Pi camera module captures images of handwritten digits placed in front of the camera lens. These images are processed as input for digit recognition. The captured images are fed into the deployed CNN model for inference. The model predicts the digits present in the images with high accuracy, leveraging its learned features from the training dataset.

Volume: 17 Issue: 04 Pages: 7036-7044 (2026) ISSN: 0975-0290
https://doi.org/10.35444/IJANA.2025.17406

The recognized digits are displayed in real-time on the 3.5-inch display connected to the Raspberry Pi 4 Model B. This visual feedback aids users in verifying the accuracy of the digit recognition process. Simultaneously, the predicted digits are communicated audibly through the integrated audio speaker. This auditory feedback further assists visually impaired users in understanding and verifying the recognized digits.

3.1.1 Raspberry Pi 4 Model B

The Raspberry Pi 4 Model B represents a significant advancement in embedded computing, offering enhanced performance and versatility for various projects [12]. It features a Broadcom BCM2711 quad-core Cortex-A72 64-bit SoC at 1.5 GHz with 2 GB, 4 GB, or 8 GB LPDDR4-3200 SDRAM options, making it suitable for resource-intensive tasks. Key features include dual micro-HDMI ports supporting 4K at 60 fps, Gigabit Ethernet, dual-band Wi-Fi, Bluetooth 5.0, multiple USB 3.0 and USB 2.0 ports, and extensive GPIO pins enabling diverse multimedia and networking applications.

The Raspberry Pi 4 Model B has 40 GPIO pins for connecting external components:

• Power Pins provide power (3.3 V, 5 V, and GND) for devices such as sensors and LEDs.
• I/O Pins support input and output functions for interfacing with sensors and switches, including I2C and UART communication.
• Special Function Pins include PWM for motor control, SPI for high-speed data transfer, and other specific functions.
• Ground Pins ensure proper signal integrity by providing a reference voltage for circuits.

3.1.2 Interfacing Peripherals

The Pi camera module is interfaced with the Raspberry Pi by connecting its ribbon cable to the Camera Serial Interface (CSI) port as shown in Fig. 2. The Raspberry Pi must be powered off to prevent damage during setup. The CSI interface enables high-speed data transfer between the camera module and the Raspberry Pi's processing unit.

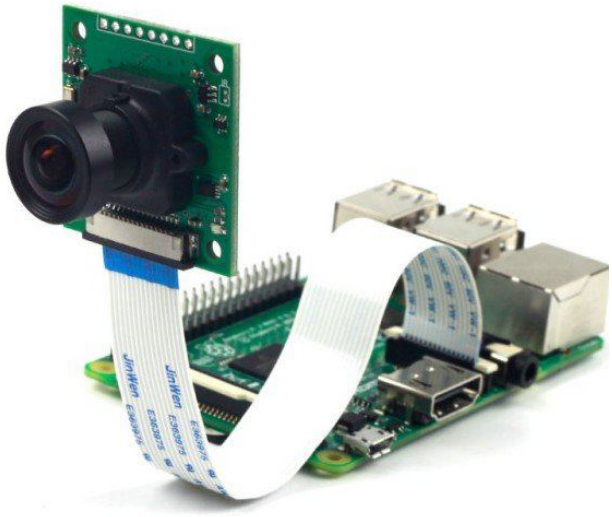

Fig. 2. The Pi Camera Module Interfaced with the Raspberry Pi 4 Model B

After connecting the ribbon cable, the camera interface is enabled through the Raspberry Pi Configuration menu. The functionality of the camera module is then tested using command-line tools such as raspistill, raspivid, or libcamera to capture images or record videos Camera settings, including resolution, frame rate, and exposure, are adjusted as needed to optimize image quality and performance.

Integrating the 3.5-inch display with the Raspberry Pi 4 Model B involves connecting the display's pins to the appropriate GPIO pins as illustrated in Fig. 3 and Fig. 4. Given the compact nature of the display, precision in pin selection is crucial to avoid potential conflicts or operational disruptions.

To provide auditory feedback, an audio speaker is interfaced with the Raspberry Pi 4 Model B as shown in Fig. 3. The audio speaker conveys the predicted digit through sound signals, enhancing system accessibility. The speaker connects directly to the standard audio jack of the Raspberry Pi 4 Model B, allowing seamless integration with external audio devices.

3.2 Software components

3.2.1 Convolutional Neural Network (CNN) Model

The core of the system is the CNN model which is trained using a dataset of handwritten digits. The architecture of the CNN is optimized for efficient inference on resource-constrained devices like the Raspberry Pi.

Volume: 17 Issue: 04 Pages: 7036-7044 (2026) ISSN: 0975-0290
https://doi.org/10.35444/IJANA.2025.17406

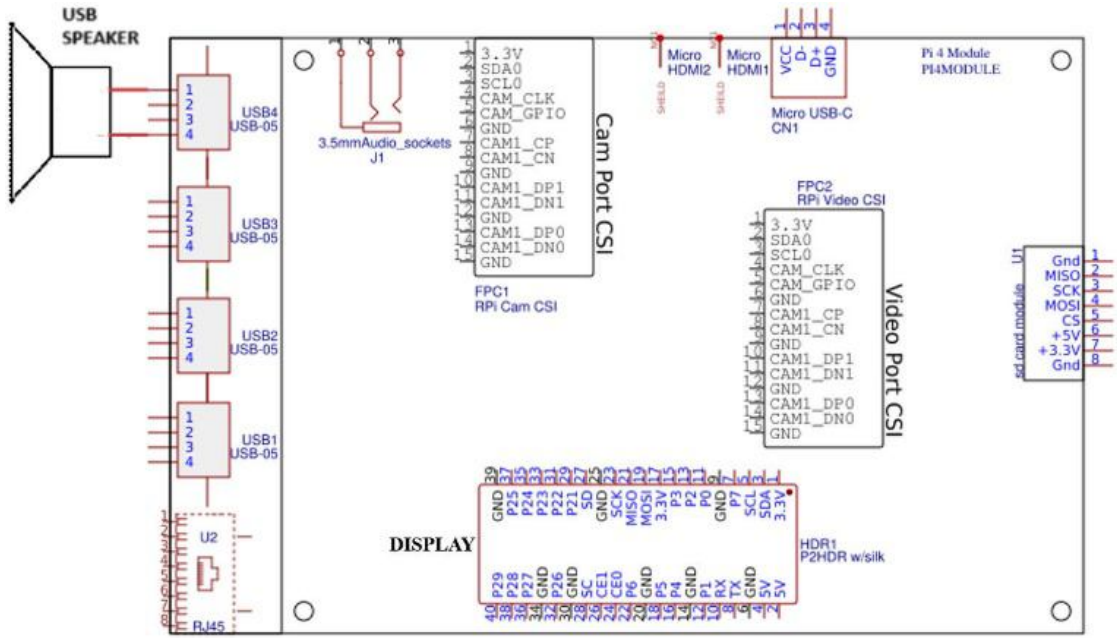

Fig. 3: Schematic diagram of the proposed TinyML-based handwritten digit recognition system.

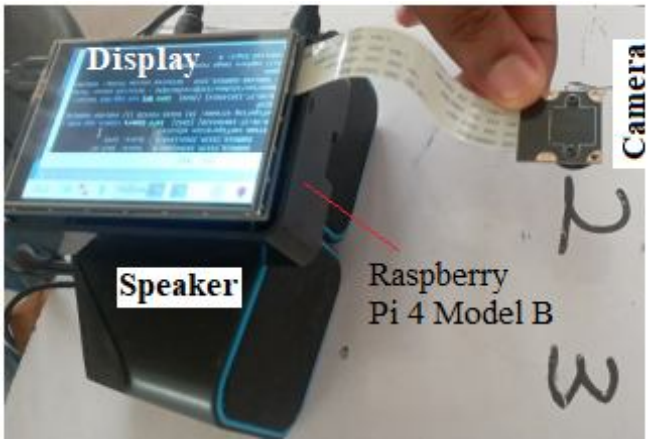

Fig. 4: Pictorial view the TinyML-based handwritten digit recognition system.

3.2.2 TensorFlow Lite

TensorFlow Lite is a lightweight version of the TensorFlow framework optimized for mobile and embedded devices. It is utilized for deploying the trained CNN model on the Raspberry Pi 4 Model B.

3.2.3 Python Programming Language

The system is primarily developed using Python programming language due to its versatility and extensive support for machine learning frameworks and Raspberry Pi interfacing libraries.

3.2.4 Libraries

For smooth runing of the system, a number of libraries are utilized. The essential libraries are TensorFlow, Keras, NumPy, Pandas, Matplotlib, Subprocess, Time module, PIL (Python Imaging Library), Pygame, and OpenCV.

For deploying machine learning models on the Raspberry Pi 4 Model B, TensorFlow Lite is chosen due to its lightweight footprint and efficient performance on edge devices [13]. TensorFlow Lite offers an optimized runtime for low-latency inference and compatibility with hardware accelerators such as Edge TPU and NVIDIA GPUs, enabling real-time performance while conserving resources [19]. Setting up the Raspberry Pi for handwritten digit recognition involves:

• System Setup - Installing and updating the Raspbian OS.
• Library Installation - Installing Python, TensorFlow Lite, OpenCV, and GPIO libraries.
•.Configuration-Optimizing TensorFlow Lite for Raspberry Pi and configuring GPIO pins for peripheral interaction
• Testing - Ensuring the functionality of installed libraries and peripherals through thorough testing and troubleshooting.

3.3 The Dataset

The Modified National Institute of Standards and Technology (MNIST) dataset is a benchmark dataset in handwritten digit recognition. It originated from NIST Special Database 3 and Special Database 1 to promote advancements in pattern recognition and machine learning [20]. The dataset contains 28×28-pixel grayscale images of handwritten digits (0–9) and has been widely adopted for evaluating classification algorithms and neural networks [21]. Samples of the MNIST dataset are shown in Fig. 5.

Developed by Yann LeCun, Corinna Cortes, and Christopher J.C. Burges, the MNIST dataset includes 70,000 grayscale images of handwritten digits [3]. These are split into a training set of 60,000 images and a test set of 10,000 images. Each image has a resolution of 28 × 28 pixels, simplifying preprocessing and ensuring uniformity. The dataset is

Volume: 17 Issue: 04 Pages: 7036-7044 (2026) ISSN: 0975-0290
https://doi.org/10.35444/IJANA.2025.17406

balanced, with each digit class having about 6,000 samples in the training set and 1,000 in the test set. This prevents bias and ensures reliable classification performance.

Fig. 5: Sample images from the MNIST dataset.

The MNIST dataset requires minimal preprocessing due to its clean and standardized nature, free from noise or missing values. Pixel values are normalized to the range [0,1] by dividing by 255. This normalization enhances model convergence and stability during training, making the dataset well-suited for CNN training [3].

3.4 The Recognition Process

The camera module is initialized and shows an image preview for a few seconds to ensure correct framing. Once captured, the image is processed by the CNN model for inference. Finally, the predicted digit is displayed on the screen and simultaneously conveyed as audio feedback for the visually impaired user. A flowchart illustrating the working process is presented in Fig. 6.

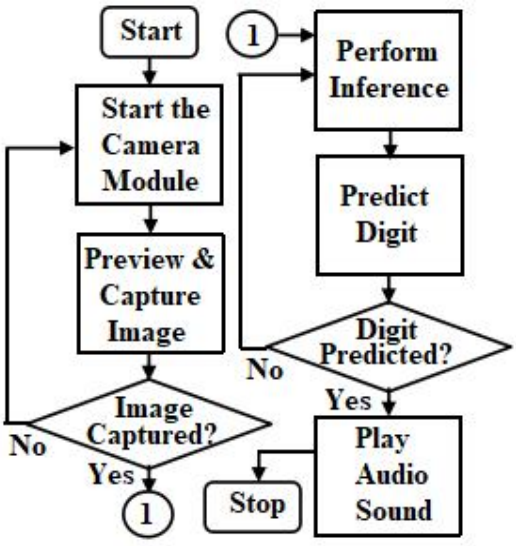

Fig. 6: Flowchart of the working process of the proposed handwritten digit recognition system.

3.5 Selection of Sounds Corresponding to Predicted Digits

Distinct sound tones are selected to represent each digit (0–9) to ensure clarity and ease of interpretation. Considerations include clarity, distinguishability, and user experience. Higher pitches correspond to higher digits, while lower pitches represent lower digits. Each tone's duration is optimized to balance recognition time and responsiveness, and its timbre is chosen to be pleasant and non-intrusive. This consistent set of tones enables visually impaired users to accurately interpret the recognized digits, even in noisy environments.

3.6 CNN Model Architecture and Training Process

A Convolutional Neural Network (CNN) is designed for image recognition and processing, particularly effective for handling pixel-level image data [22]. The network consists of an input layer, multiple hidden layers (convolutional and pooling), and an output layer. The hidden layers apply activation functions and convolutional operations to learn hierarchical feature representations [23].

3.6.1 Input Layer

The input layer represents the raw input image dimensions: height, width, and channels (grayscale). Each neuron corresponds to a pixel value, forming a three- dimensional grid [24].

3.6.2 Convolutional Layers

Convolutional layers perform feature extraction using multiple filters (kernels) that slide across the input image to generate feature maps [25]. These filters detect patterns such as edges, curves, or textures. Non- linear activation functions such as ReLU are applied to introduce non-linearity, allowing the network to represent complex data distributions.

3.6.3 Max Pooling Layers

Pooling layers down sample feature maps, reducing their spatial dimensions by retaining only the maximum value within non-overlapping regions [1], [25]. This process improves computational efficiency, reduces overfitting, and preserves salient features.

3.6.4 Flatten Layer

The flattening operation converts multi- dimensional feature maps into a one-dimensional vector, preparing data for the dense layers [1], [26].

3.6.5 Dense (Fully Connected) Layers

Dense layers classify the extracted features into output categories. Each neuron in the dense layer is fully connected to the preceding layer, enabling complex mappings between features and labels. Outputs are generated as weighted sums with activation functions [5].

3.6.6 Softmax Activation Layer

The softmax layer is used for multi-class classification, transforming raw scores into probability distributions [20], [27]. It normalizes outputs such that probabilities across the

Volume: 17 Issue: 04 Pages: 7036-7044 (2026) ISSN: 0975-0290
https://doi.org/10.35444/IJANA.2025.17406

10 digit classes sum to one, assigning higher probabilities to the most likely class.

The final CNN model architecture employed in this work is shown in Fig. 7. Each component plays a vital role in feature extraction, dimensionality reduction, and classification, contributing to the overall performance and accuracy of the recognition system [28].

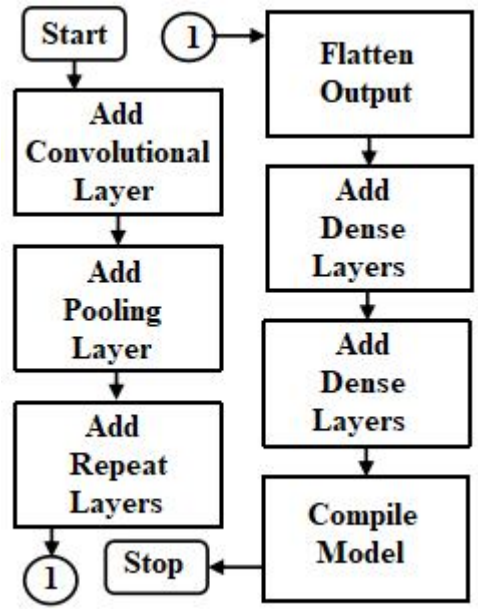

Fig. 7: CNN model architecture used for handwritten digit recognition.

4. RESULTS AND DISCUSSIONS

4.1 Evaluation Metrics

To assess the performance of our handwritten digits' recognition system for the visually impaired, several key metrics were used: accuracy, precision, recall, and F1-score.

Accuracy is given by (1) and its the proportion of correctly classified digits out of the total tested. It indicates the system's overall reliability. Precision is described by (2) and it measures the ratio of true positives to the total predicted positives; showing the accuracy of positive predictions. Recall (sensitivity) quantifies the system's ability to identify all relevant instances, minimizing missed digits and is presented in (3). The F1-Score described in (4) balances precision and recall, providing a single performance metric, especially useful with uneven class distribution.

$$Accuracy = \frac{TP+TN}{TP+FP+TN+FN} \quad (1)$$

$$Precision = \frac{TP}{TP+FP} \quad (2)$$

$$Recall = \frac{TP}{TP+FN} \quad (3)$$

$$F1-Score = \frac{2}{\frac{1}{Precision}+\frac{1}{Recall}} \quad (4)$$

where TP (True Positives) are the correctly identified digits, FP (False Positives) are incorrect positive predictions, FN (False Negatives) are missed digits, and TN (True Negatives) are instances correctly identified as not belonging to a class.

4.2 Model Loss and Accuracy Evaluation

The accuracy of the system was assessed using a test dataset consisting of handwritten digits. The CNN model achieved an accuracy of 98.83%, indicating its effectiveness in digit recognition tasks. The output in Fig. 8 shows the training progress of the neural network model trained on the MNIST handwritten digit dataset over 20 epochs. The loss represents the model's training error, which decreases steadily from about 0.0675 to 0.0366, indicating improved fitting to the training data. The accuracy measures how often the model correctly predicts the digits during training, increasing from around 97.9% to 98.8%. The val_loss (validation loss) shows how well the model generalizes to unseen data, remaining low and stable (around 0.06), while val_accuracy (validation accuracy) stays consistently high (~98%), demonstrating that the model performs well on both training and validation sets without overfitting. Fig. 8 illustrates the accuracy achieved across different epochs during the training process, demonstrating consistent improvement over iterations.

4.3 Precision, Recall, and F1-Score Evaluation

These metrics showcase the high accuracy and reliability of the system across all classes of handwritten digits. The precision, recall, and F1-score consistently demonstrate the system's ability to effectively identify and classify digits. The results are presented in Table 1.

4.4 Experimental Results of the Recognition System

The handwritten digits' recognition system was used to identify real-life handwritten digits, as shown in Fig. 9. We performed five tests for each of the classes 0–9 and recorded the number of correct and incorrect predictions. The results are presented in Table 2.

Volume: 17 Issue: 04 Pages: 7036-7044 (2026) ISSN: 0975-0290
https://doi.org/10.35444/IJANA.2025.17406

```
Epoch 10/20
51/51 [==============================] - 14s 276ms/step - loss: 0.0675 - accuracy: 0.9790 - val_loss: 0.0697 - val_accuracy: 0.9772
Epoch 11/20
51/51 [==============================] - 14s 272ms/step - loss: 0.0640 - accuracy: 0.9801 - val_loss: 0.0860 - val_accuracy: 0.9729
Epoch 12/20
51/51 [==============================] - 14s 277ms/step - loss: 0.0592 - accuracy: 0.9818 - val_loss: 0.0638 - val_accuracy: 0.9809
Epoch 13/20
51/51 [==============================] - 13s 263ms/step - loss: 0.0526 - accuracy: 0.9834 - val_loss: 0.0679 - val_accuracy: 0.9792
Epoch 14/20
51/51 [==============================] - 14s 272ms/step - loss: 0.0503 - accuracy: 0.9845 - val_loss: 0.0641 - val_accuracy: 0.9792
Epoch 15/20
51/51 [==============================] - 14s 277ms/step - loss: 0.0503 - accuracy: 0.9835 - val_loss: 0.0657 - val_accuracy: 0.9790
Epoch 16/20
51/51 [==============================] - 14s 272ms/step - loss: 0.0473 - accuracy: 0.9848 - val_loss: 0.0572 - val_accuracy: 0.9820
Epoch 17/20
51/51 [==============================] - 14s 269ms/step - loss: 0.0427 - accuracy: 0.9865 - val_loss: 0.0630 - val_accuracy: 0.9803
Epoch 18/20
51/51 [==============================] - 14s 269ms/step - loss: 0.0410 - accuracy: 0.9873 - val_loss: 0.0568 - val_accuracy: 0.9803
Epoch 19/20
51/51 [==============================] - 14s 273ms/step - loss: 0.0368 - accuracy: 0.9885 - val_loss: 0.0554 - val_accuracy: 0.9823
Epoch 20/20
51/51 [==============================] - 14s 275ms/step - loss: 0.0366 - accuracy: 0.9883 - val_loss: 0.0619 - val_accuracy: 0,9799
```

Fig. 8: Accuracy achieved across different epochs during training.

(a) Sample Prediction 1 (b) Sample Prediction 2 (c) Sample Prediction 3

Fig. 9: Making Predictions with the Handwritten Digits Recognition System.

Table 1: Precision, Recall, and F1-Score Evaluation of Digits 0–9

Class	Precision	Recall	F1-Score
0	97%	97%	97%
1	97%	98%	98%
2	96%	97%	96%
3	95%	97%	97%
4	98%	97%	96%
5	98%	98%	97%
6	97%	96%	97%
7	97%	96%	96%
8	98%	98%	98%
9	98%	98%	97%

4.5 Discussion of Results

The exemplary performance across all classes underscores the robustness of the system in digit recognition tasks. High precision values in Table 1 indicate minimal false positives, which is critical for aiding visually impaired individuals. Similarly, the high recall values demonstrate the system's ability to capture instances of each digit class, minimizing false negatives.

Table 2: Experimental Results of the Recognition System

Class	Test Count	Correct	Incorrect	Accuracy (100)
0	5	4	1	80
1	5	5	0	100
2	5	4	1	80
3	5	3	2	60
4	5	4	1	80
5	5	5	0	100
6	5	4	1	80
7	5	4	1	80
8	5	5	0	100
9	5	5	0	100

Volume: 17 Issue: 04 Pages: 7036-7044 (2026) ISSN: 0975-0290
https://doi.org/10.35444/IJANA.2025.17406

The F1-score, as the harmonic mean of precision and recall, reflects the balance between these two metrics. The particularly impressive performance observed in classes 1, 5, 8, and 9 suggests the system excels at distinguishing digits with intricate or similar features. Experiments were performed under varying lighting conditions. The results show good performance as seen in Fig 10. Incorrect predictions were observed mostly under dim lighting, which represents the major limitation of the recognition system.

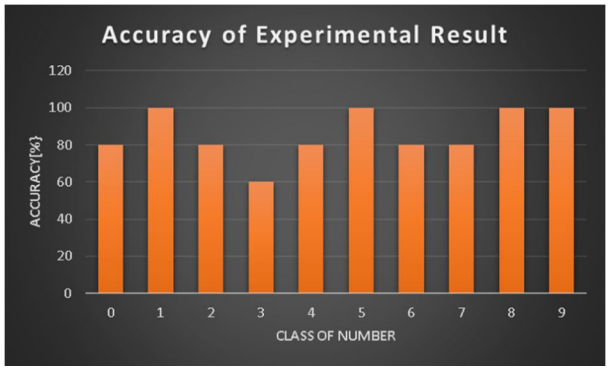

Fig. 10: Accuracy of Experimental result

The consistent performance under various environmental circumstances suggests that the preprocessing and extraction of features phases play a major role in the stability of the system. This implies that the model is not unduly sensitive to small changes in input quality, which is crucial for reliable operation. The majority of misclassifications happen between visually similar digits, according to the analysis of the confusion matrix, suggesting that errors are more often caused by innate form resemblance than by inefficient models.

From a usability perspective, the observed results affirm that the system can deliver timely and accurate digit recognition, which is vital for supporting independent navigation and information access for visually impaired users. Finally, the experimental outcomes validate the feasibility of deploying the proposed system on resource-constrained platforms, as high recognition performance is achieved without excessive computational overhead, making it suitable for portable assistive devices.

5. CONCLUSION

This project successfully achieved its objectives of creating an innovative solution to assist individuals with visual impairments in recognizing handwritten digits. Through the integration of a Convolutional Neural Network (CNN) model with a Raspberry Pi 4 Model B, coupled with peripherals including a Pi camera module, a 3.5-inch display, and an audio speaker, a comprehensive system was developed to provide both visual and auditory feedback for digit recognition.

The evaluation revealed promising results, with high accuracy rates achieved in digit recognition tasks. Low latency during real-time operation indicates the system's suitability for practical use by visually impaired individuals. Comparative analysis with existing solutions highlighted the system's strengths, particularly its portability, affordability, and real-time processing capabilities. Feedback obtained from user testing provided valuable insights into usability, with users appreciating the intuitive interface and multi-modal feedback mechanism.

REFERENCES

[1]. J. Lin, L. Zhu, W.-M. Chen, W.-C. Wang, and S. Han, "Tiny machine learning: Progress and futures [feature]," IEEE Circuits and Systems Magazine, vol. 23, no. 3, pp. 8–34, 2023.

[2]. Y. Abadade, A. Temouden, H. Bamoumen, N. Benamar, Y. Chtouki, andA. S. Hafid, "A comprehensive survey on Tinyml," IEEE Access, vol. 11, pp. 96 892–96 922, 2023.

[3]. Y. LeCun, B. Boser, J. Denker, D. Henderson, R. Howard,W. Hubbard, and L. Jackel, "Handwritten digit recognition with a back-propagation network," in Advances in Neural Information Processing Systems, D. Touretzky, Ed., vol. 2. Morgan-Kaufmann, 1989. [Online]. Available: https://proceedings.neurips.cc/paper files/paper/1989/file/53c3bce66e43be4f209556518c2fcb54-Paper.pdf

[4]. SVM Based Real Time Hand-Written Digit Recognition System, 01, 2019

[5]. M. Shuvo, M. Akhand, and N. Siddique, "Handwritten numeral recognition through superimposition onto printed form," Journal of King Saud University - Computer and Information Sciences, vol. 34, no. 9, pp. 7751–7764, 2022. [Online].Available:https://www.sciencedirect.com/science/article/pii/S1319157822002270

[6]. M. A. Hossain and M. S. A. Sajib, "Classification of image using convolutional neural network (CNN)," Global Journal of Computer Science and Technology, vol. 19, no. 2, pp. 13–14, 2019.

[7]. A. H. Abed, "Smart Aerial Image Classification using Deep Learning for Drone-Based Emergency Monitoring," Int. J. Advanced Networking and Applications, vol. 17. no. 2: 6873-6884, 2025.

[8]. H. Zheng, J. Fu, T. Mei, and J. Luo, "Learning multi-attention convolutional neural network for fine-grained image recognition," in 2017 IEEE International Conference on Computer Vision (ICCV), 2017, pp. 5219–5227.

[9]. S. Varghese Jacob and I. S. Mackenzie, "Comparison of feedback modes for the visually impaired: Vibration vs. audio," in Universal Access in Human-Computer Interaction. Methods, Technologies, and Users, M. Antona and C. Stephanidis, Eds. Cham: Springer International Publishing, 2018, pp. 420–432.

[10]. A. Aswin and Dr. G. Raemsh, "Deep Learning-Based Acoustic Analysis for Early Detection of

Volume: 17 Issue: 04 Pages: 7036-7044 (2026) ISSN: 0975-0290
https://doi.org/10.35444/IJANA.2025.17406

Respiratory Diseases: A Hybrid CNN-LSTM Approach," Int. J. Advanced Networking and Applications, vol. 16. no. 6: 6873-6884, 2025.

[11]. A. R. Zubair, E. S. Olu-Flourish and M. O. Nnaukwu, "Smart Home, Support at Old Age and Support for Persons with Disabilities: Speech Processing for Control of Energy," Int. J. Advanced Networking and Applications, vol. 13. no. 5: 5134-5142, 2022.

[12]. M. Maksimovic, V. Vujovic, N. Davidovic´, V. Milosevic, and B. Perisic, "Raspberry pi as internet of things hardware: Performances and constraints," 06 2014.

[13]. A. Fanariotis, T. Orphanoudakis, K. Kotrotsios, V. Fotopoulos, G. Keramidas, and P. Karkazis, "Power efficient machine learning models deployment on edge IoT devices," Sensors, vol. 23, no. 3, 2023. [Online]. Available: https://www.mdpi.com/1424-8220/23/3/1595

[14]. N. N. Alajlan and D. M. Ibrahim, "TinyML: Enabling of Inference Deep Learning Models on Ultra-Low-Power IoT Edge Devices for AI Applications," Micromachines, vol. 13, no. 6, 851: 1-22, 2022.

[15]. I. Lamaakal, I. Ouahbi, K. El Makkaoui, Y. Maleh, P. Pławiak, and F. Alblehai, "A TinyDL Model for Gesture-Based Air Handwriting Arabic Numbers and Simple Arabic Letters Recognition," IEEE Access, vol. 12, pp. 76589–76605, 2024.

[16]. B. Saha and S. K. Ghosh, "TinyML-Powered Gesture Wizardry: Low-Cost, Low-Power Two-Stage CNN for Static Hand Gesture Classification on MCU in Appliance Control," in Proceedings of the Fourth International Conference on AI-ML Systems (AIMLSystems '24), Baton Rouge, LA, USA, 2024.

[17]. S. Chittepu, S. Martha, and D. Banik, "Empowering voice assistants with TinyML for user-centric innovations and real-world applications," Scientific Reports, vol. 15, no. 1, 2025, Art. no. 15411. [Online]. Available: https://doi.org/10.1038/s41598-025-96588-1

[18]. J. Kunhoth, M. Alkaeed, A. Ehsan, and J. Qadir, "VisualAid+: Assistive System for Visually Impaired with TinyML Enhanced Object Detection and Scene Narration," in 2023 International Symposium on Networks, Computers and Communications (ISNCC), Doha, Qatar, 2023. [Online]. Available: https://www.google.com/search?q=https://doi.org/10.1109/ISNCC58260.2023.10323988

[19]. W. J. Song, "Chapter two - hardware accelerator systems for embedded systems," in Hardware Accelerator Systems for Artificial Intelligence and Machine Learning, ser. Advances in Computers, S. Kim and G. C. Deka, Eds. Elsevier, 2021, vol. 122, pp. 23–49. [Online]. Available:https://www.sciencedirect.com/science/article/pii/S0065245820300917

[20]. A. Baldominos, Y. Saez, and P. Isasi, "A survey of handwritten character recognition with mnist and emnist," Applied Sciences, vol. 9, no. 15, 2019. [Online]. Available: https://www.mdpi.com/2076-3417/9/15/3169

[21]. D. Beohar and A. Rasool, "Handwritten digit recognition of mnist dataset using deep learning state-of-the-art artificial neural network (ANN) and convolutional neural network (CNN)," in 2021 International Conference on Emerging Smart Computing and Informatics (ESCI), 2021, pp. 542–548.

[22]. B. B. Traore, B. Kamsu-Foguem, and F. Tangara, "Deep convolution neural network for image recognition," Ecological Informatics, vol. 48, pp. 257–268,2018. [Online]. Available:https://www.sciencedirect.com/science/article/pii/S1574954118302140

[23]. M. M. Taye, "Theoretical understanding of convolutional neural network: Concepts, architectures, applications, future directions," Computation, vol. 11, no. 3,2023.[Online]. Available: https://www.mdpi.com/2079-3197/11/3/52

[24]. O. Montesinos-Lo´pez, A. Montesinos, and J. Crossa, Multivariate Statistical Machine Learning Methods for Genomic Prediction, 01 2022.

[25]. M. Krichen, "Convolutional neural networks: A survey," Computers, vol. 12, no. 8, 2023. [Online]. Available: https://www.mdpi.com/2073-431X/12/8/151

[26]. A. E. Hassani, B. A. Skourt, And A. Majda, "Efficient lung nodule classification method using convolutional neural network and discrete cosine transform," International Journal of Advanced Computer Science and Applications, vol. 12, no. 2, 2021. [Online]. Available:http://dx.doi.org/10.14569/IJACSA.2021.0120296

[27]. S. Hayat, S. Kun, Z. Tengtao, Y. Yu, T. Tu, and Y. Du, "A deep learning framework using convolutional neural network for multi-class object recognition," in 2018 IEEE 3rd International Conference on Image, Vision and Computing (ICIVC), 2018, pp. 194–198.

[28]. E. A. Alzubaidi L, Zhang J, "Review of deep learning: concepts, CNN architectures, challenges, applications, future directions," Journal of Big Data, vol. 8, no. 1, 2021. [Online]. Available: https://doi.org/10.1186/s40537-021-00444-8