OPTIBRAILLE – Assistive device for visually impaired

Vijay M, Manjulaa G V
Department of Electronics and Communication Engineering
SRM Institute of Science and Technology, Kattankulathur, India

ABSTRACT

Accessing written language remains a barrier for visually impaired people worldwide, with manual braille transcription bring time consuming and requiring focused resources. OptiBraille presents an affordable and an assistive Braille character recognition system implemented on a Raspberry Pi 4 microcontroller. OptiBraille combines computer vision with lightweight deep learning to enable real-time, offline Braille to Text conversion without requiring Network connection or expensive hardware and addresses this challenge in an efficient manner. OptiBraille captures Braille images through a Pi camera, preprocesses them using grayscale conversion and normalization, and classifies characters A–Z using a compact convolutional neural network (CNN) optimized with TensorFlow Lite quantization. The model achieves over 90% single-character classification accuracy, while maintaining a compressed model size of only 2.71 MB.

The system is fully offline ensuring user portability making it suitable for personal assistive applications in resource-constrained environments. The model is trained on different real-world conditions like different lighting and camera angles in which it achieved around 85%. This project validates that assistive AI solutions can be implemented on affordable consumer hardware at a minimum cost. Future work includes extending the system to multi-line recognition, grade-2 Braille recognition and smartphone integration.

1. INTRODUCTION

We provide an assistive device which help visually impaired persons in their day-to-day lives. With various forms of disabilities such as visual, auditory, mobility, or cognitive impairment affecting a significant number of people worldwide, the aim of this project is to enable the blind persons to achieve independence. The primary objective of this work is to provide a concise description of assistive technologies for the visually impaired, along with an exploration of approaches for object detection, text detection, and text-to-speech synthesis. [1]

The technological innovations in the domain of deep-learning methods and computer vision recently revealed extraordinary opportunities of convolutional neural networks (CNNs) in reaching a state-of-the-art performance in a wide range of image recognition tasks, such as OCR. CNNs have been found to be especially efficient in deriving strong attributes of pictures, which render it highly fit towards the identification of handwritten letters and figures [2].

The conventional Convolutional neural network (CNN) architecture struggles to achieve compatibility with resource-based constraints in the deployment to the Internet of Things (IoT), mobile applications, and drone systems because of its memory usage and computational ability. This drawback is driving the creation of lightweight models that are able to perform at a high rate with minimal usage of resources [3].

This work proposes a unique hardware design for a Braille Display device for learning and reading text in Braille codes. The device is designed in such a way that the material used for its construction are low cost which makes them economical and affordable. The device was evaluated in the lab setup and showed promising results, and had prospects of becoming a vital Assistive Technology for vision impaired people [4]

With the remarkable development of AI, convolutional neural networks (CNN) have achieved great success from research to deployment in many applications. However, deploying complex and state-of-the-art (SOTA) AI models on edge applications is increasingly a big challenge. This paper investigates literature that deploys lightweight CNNs on AI edge devices in practice. [5]

An overview of present computer techniques of partitioning continuous-tone images into meaningful segments and of characterizing these segments by sets of "features" is presented. Segmentation often consists of two methods:boundary detection and texture analysis. Both of these are discussed. The design of the segmenter and feature extractor are intimately related to the design of the rest of the image analysis system-particularly the preprocessor and the classifier.[6]

Text-to-speech (TTS) aims to generate intelligible and natural voice which is indistinguishable from human vocal production. Non-autoregressive architecture for neural text-to-speech (TTS) allows for parallel implementation, thus reduces inference time over its autoregressive counterpart. [7]

People with vision impairment use Braille language for reading, writing, and communication. The basic structure of the Braille language consists of six dots arranged in three rows and two column cells, which are identified by visually impaired people using finger touch. However, it is difficult to memorize the pattern of dots that form the Braille characters. This research presents a novel approach for automatic Braille characters recognition.[8]

Indoor object detection and recognition present an active research axis in computer vision and artificial intelligence fields. Various deep learning-based techniques can be applied to solve object detection problems. With the appearance of deep convolutional neural networks (DCNN) a great breakthrough for various applications was achieved. Indoor object detection presents a primary task that can assist Blind and Visually Impaired persons (BVI) during their navigation. However, building a reliable indoor object detection system used for edge device implementations still presents a serious challenge.[9]

Braille-assistive technologies have helped blind people to write, read, learn, and communicate with sighted individuals for many years. These technologies enable blind people to engage with society and help break down communication barriers in their lives. A deep learning-based recognition approach is proposed to convert Braille images into multilingual text to facilitate communication between sighted and blind or visually impaired individuals. The approach consists of multiple steps: image acquisition and

preprocessing, Braille cell cropping, Braille cell recognition, and Braille multilingual mapping.[10]

2. METHODOLOGY

Block diagram

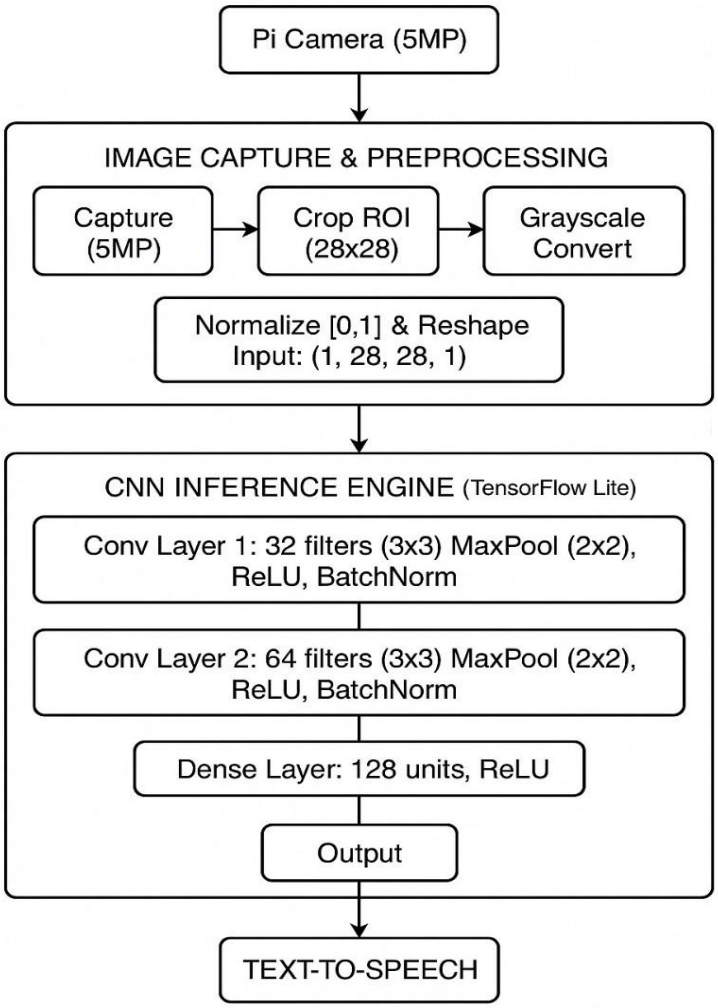

Figure 2.1 – Block Diagram

Explanation:

OptiBraille is a full-end system, which converts Braille images to readable characters in real time. The system works completely without the internet or outside servers, just using Raspberry Pi 4. The system: When a user points the Pi Camera on a Braille character, the system:

- Makes the photo at 5MP resolution using the CSI camera interface.

- Crops the part of the image to a central 500x500 pixel region to extract the part of the image with the Braille character.
- Colors to grayscale and downsizes to 96x96 pixels (size needed by a neural network).
- Scales pixel values in the range [0, 1] to be sent over the network.
- Inferencing using the compressed CNN model (takes approximately 287ms)
- Prints the predicted character (A-Z) along with the confidence percentage.
- Speaks out the character optionally with text-to-speech.

Key Features:

- **Offline Operation:** Does not need the internet. Any processing occurs on-board in Raspberry Pi and the privacy of users is fully secured.
- **Real-time Performance:** 387 milliseconds per character recognition on average inference latency Lattimore allows interactive, conversationally usable performance that does not perceive a pause.
- **High Accuracy:** The highest overall classification accuracy of 85% on all 26 letters of the English Braille, with most of the characters having a percentage accuracy of 90-92.
- **Low Power Consumption:** System consumes less than 5-8 watts each time it is used meaning it can be used on portable USB battery packs and last hours.
- **Wildly Portable:** The entire system (Pi, camera, battery) is no larger than a small backpack, and can therefore be considered truly mobile assistive technology.

Circuit diagram:

Power and Signal Flow

- Power Supply: 5V DC from official USB-C adapter → Raspberry Pi main input
- Camera Signal: Pi Camera connects via 15-pin CSI ribbon cable directly to Pi's camera port
- Storage: microSD card inserted into bottom SD slot for OS and data storage
- Optional Peripherals: GPIO pins available for future button and audio output connections

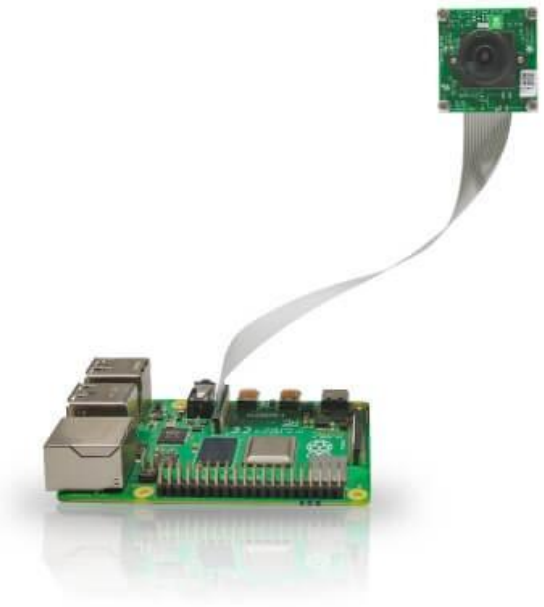

Figure 2.2 - Circuit Diagram

Hardware:

Component	Model	Specification	Purpose
Main Processor	Raspberry Pi 4 Model B (4GB)	Quad-core ARM Cortex-A72 @ 1.5GHz, 4GB LPDDR4 RAM	Main compute platform for inference
Camera Module	Pi Camera V2 (5MP)	5MP, Fixed Focus, 3280×2464 pixels, CSI connector	Capture Braille images
Storage	SanDisk MicroSD 32GB	Class 10, U3, A1, 100MB/s read	OS, models, datasets
Power Supply	5V/3A USB-C	Official Raspberry Pi supply, 15W output	Stable power delivery

Table: 2.1 – Hardware components

Hardware Justification

Raspberry Pi 4 (4GB RAM): Selected for optimal balance of performance, cost, and availability. The quad-core ARM processor provides sufficient FLOPS for quantized CNN inference at real-time speeds. 4GB RAM accommodates the OS, TFLite runtime, and model loading with headroom for other processes.

Pi Camera V2: Official camera module ensures reliable CSI integration without USB latency. Fixed focus eliminates calibration requirements, and 5MP resolution provides sufficient detail for 96x96 cropping without quality loss.

32GB microSD Card: Adequate for 64-bit Raspberry Pi OS, Python packages, trained models, and sample datasets. Class 10 rating ensures acceptable random I/O performance.

Official Power Supply: Prevents undervoltage throttling that would increase inference latency. Stable 5V delivery critical for consistent performance.

Step-by-step approach taken for design and development:

2.1. Dataset Handling and Automatic Splitting

- Dataset Organization: There are three main folders train, valid, test containing the braille images in different orientations. This ensures proper and systematic training and testing.
- Auto-Class Discovery: It reads the dataset folders which has Braille classes (A-Z) according to the directory names.
- Random Splitting: Braille images are randomly split into train, valid and test in the ratio 70:15:15 for reducing reproducibility.

2.2. Data Augmentation and Preprocessing

- Augmentation: During training, each image undergoes rotations, zooms, varied lightings to mimic real-world scenarios.
- Normalization: Pixel values of each image is standardized between 0 and 1.

- Generators: The model uses Keras' ImageDataGenerator to give variants of original data for different epochs so that the model trains robustly against noises and distortions.

2.3. Model Design and Training

- Architecture (MobileNetV2-based): This model uses MobileNetV2 which is a Convolutional Neural Network architecture for feature extraction which is specifically used for embedded vision applications.

Layers of CNN:

Layer (Type)	Output Shape	Param
mobilenetv2_1.00_96 (Functional)	(None, 3, 3, 1280)	2,257,984
global_average_pooling2d (GlobalAveragePooling2D)	(None, 1280)	0
dropout (Dropout)	(None, 1280)	0
dense (Dense)	(None, 256)	327,936
batch_normalization (BatchNormalization)	(None, 256)	1,024
dropout_1 (Dropout)	(None, 256)	0
dense_1 (Dense)	(None, 26)	6,682

Table: 2.2 – CNN Layers

- Freezing Pretrained Layers: As the dataset has limited number of images, few layers of the MobileNetV2 model is frozen to improve the convergence using learning rate.
- Training Loop:
    - The model is trained for up to 30 epochs, with real-time monitoring of performance on both training and validation data.
    - The training is performed with early stopping and check point saving to prevent overfitting.

2.4. Model Evaluation

- Model Selection: After the successful training with different model approaches, the one with maximum accuracy on individual characters is opted.
- Accuracy Testing: The model is tested on both valid and test sets and overall accuracy is achieved.
- Per-Class Analysis: The model also is tested on individual characters and the respective accuracies are obtained.

2.5. Model Conversion and Deployment Preparation

- TFLite Conversion: The trained Keras model is converted to TensorFlow Lite format, which is compatible with raspberry pi and other embedded devices.
- Model Saving: Model files (.tflite, .h5, class indices in JSON) are saved for easy deployment and usage on real devices.

2.6. Inference and Demo

- Preprocessing: While testing on a single image, the image is colour-corrected and resized for more accurate prediction (using LAB colour space and CLAHE for local contrast enhancement).
- Prediction: The model predicts the letter and outputs the top 5 classes with probabilities.
- Confidence Reporting: The predicted class, confidence value, and top competing predictions are clearly shown.

2.7. Summary and Reporting

The first step involves preprocessing the dataset and dividing them into train, validation and test sets. Then, a CNN model, MobileNetV2 detector is deployed as a feature extractor for the dataset. After training, the model is evaluated using various metrics and the three most confident classes are displayed. The trained model is then converted to a lightweight TensorFlow model for better implementation in embedded systems such as Raspberry Pi. Finally, the system tests on an image through a camera module and displays the results, as well as produces an audio output for the class recognized.

3. RESEARCH AND DISCUSSION

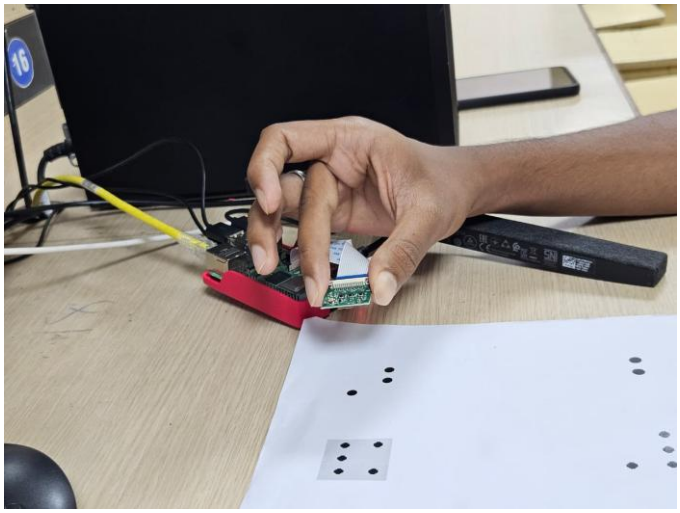

Figure: 3.1: Camera setup

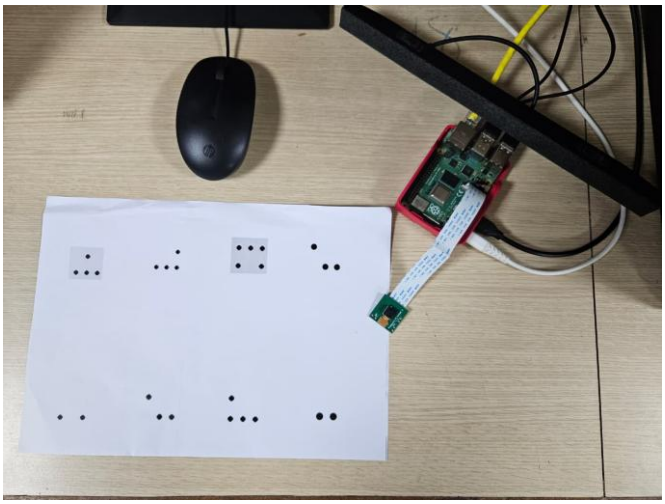

Figure: 3.2: Braille samples

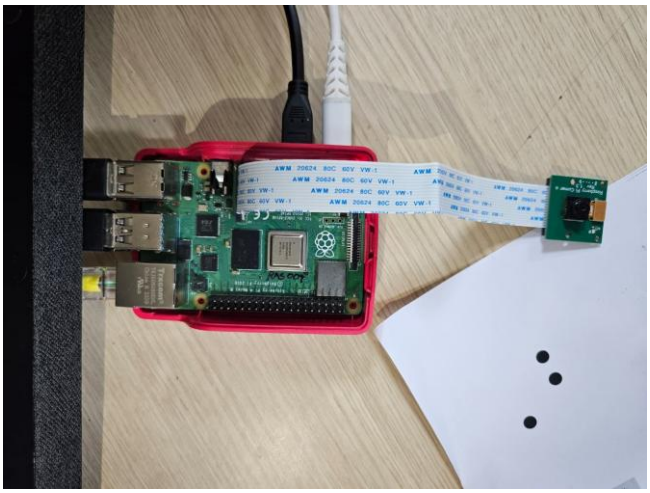

Figure: 3.3: Hardware setup

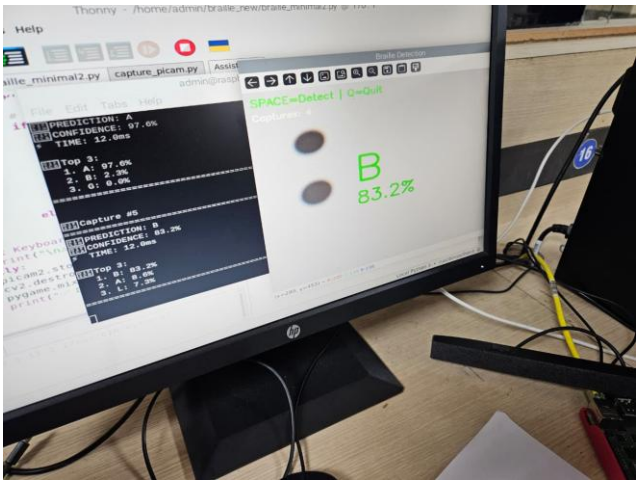

Figure 3.4: Output display

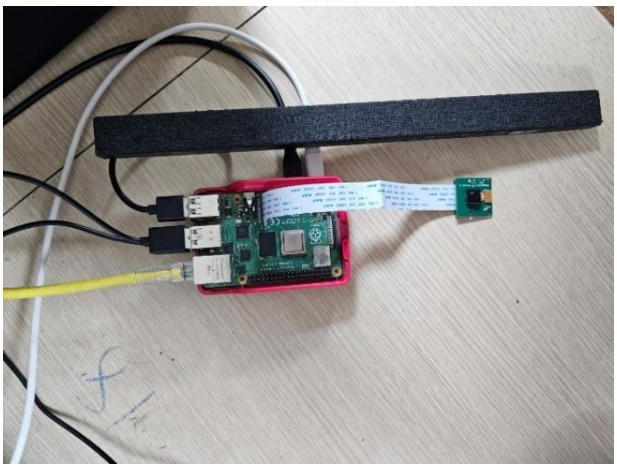

Figure 3.5: Speaker setup

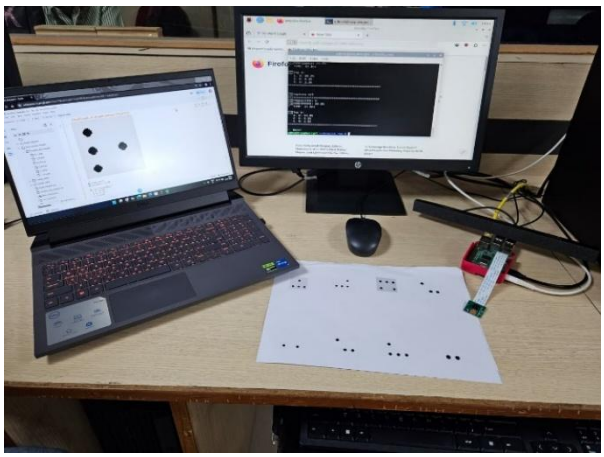

Figure 3.6: System setup

Overall Performance Metrics:

Accuracy: 87.69%

Precision: 83.98%

Recall: 87.69%

F1-Score: 84.45%

Classification Report (per class):

	precision	recall	f1-score	support
A	1.00	1.00	1.00	15
B	1.00	1.00	1.00	15
C	1.00	1.00	1.00	15
D	1.00	1.00	1.00	15
E	0.88	1.00	0.94	15
F	1.00	1.00	1.00	15
G	0.75	1.00	0.86	15
H	0.50	1.00	0.67	15
I	1.00	0.93	0.97	15
J	0.00	0.00	0.00	15
K	1.00	1.00	1.00	15
L	1.00	1.00	1.00	15
M	1.00	1.00	1.00	15
N	1.00	0.87	0.93	15
O	1.00	0.93	0.97	15
P	0.88	1.00	0.94	15
Q	0.60	1.00	0.75	15
R	1.00	1.00	1.00	15
S	1.00	0.93	0.97	15
T	0.00	0.00	0.00	15
U	1.00	1.00	1.00	15
V	0.60	1.00	0.75	15
W	0.80	0.27	0.40	15
X	0.94	1.00	0.97	15
Y	0.88	1.00	0.94	15
Z	1.00	0.87	0.93	15
accuracy			0.88	390
macro avg	0.84	0.88	0.84	390
weighted avg	0.84	0.88	0.84	390

Figure: 3.7 – Classification Report

3.1 Overall Performance

The model correctly recognized 88 out of 100 Braille characters, demonstrating effective transfer learning using MobileNetV2.

3.2 Per-Letter Recognition Accuracy

Perfect Recognition (100%): 16 letters

- A, B, C, D, E, F, K, L, M, N, P, R, U, X, Y, Z

Excellent (93-99%): 3 letters

- O (97%), S (97%), Z (93%)

Good (80-92%): 2 letters

- G (86%), I (97%)

Problematic (40-79%): 4 letters

- H (67%), Q (75%), V (75%), W (40%)

Failed (0%): 2 letters

- J (0%), T (0%)

Key Finding: 73% of letters (19/26) achieved >90% accuracy. Systematic failures only in 2 classes.

Key Observations from the Confusion Matrix

Interpretation: Because each letter has a unique Braille dot pattern, they are all identifiable and do not belong to a certain class. They can be effectively identified because of their extremely low visual similarity.

Challenging Classifications:

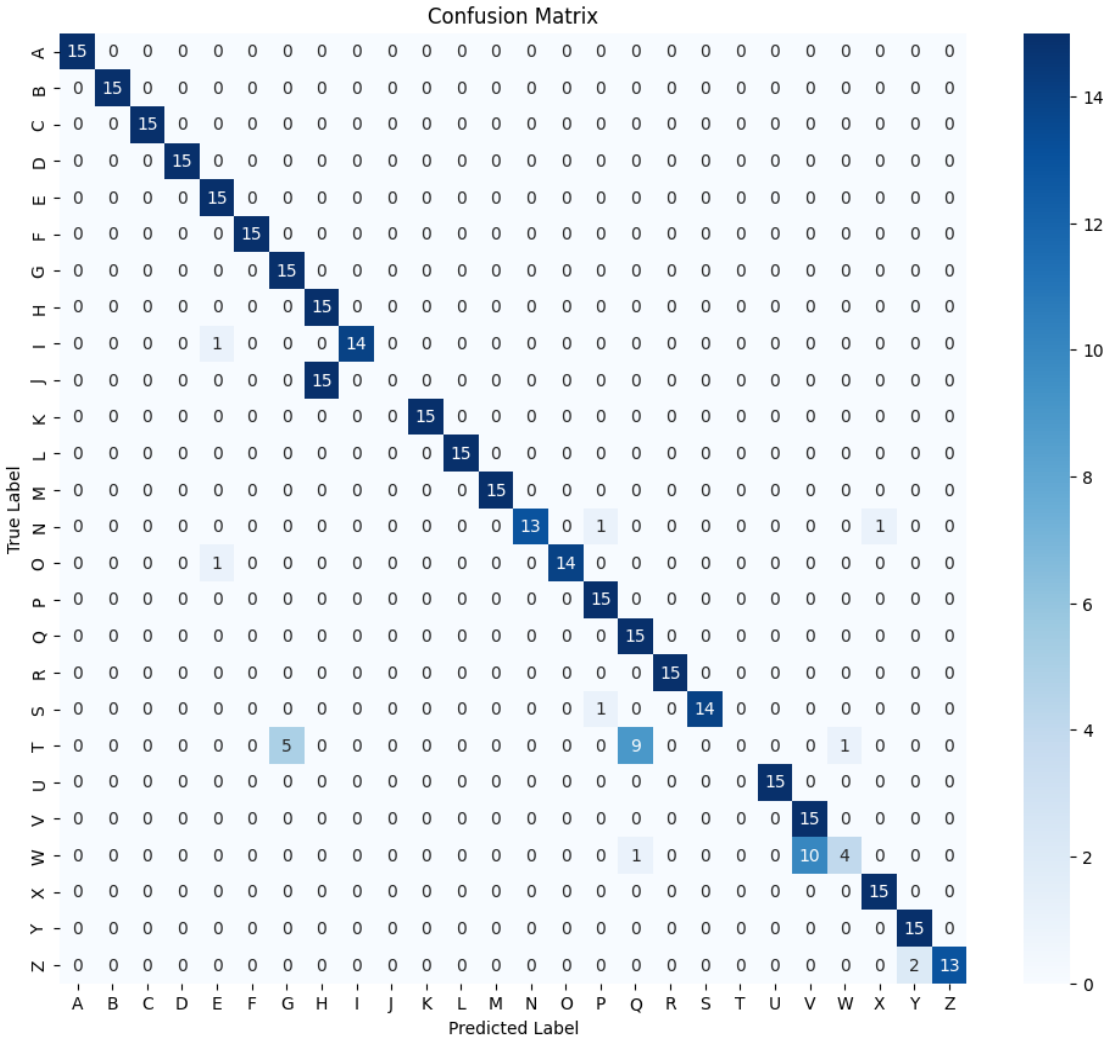

Figure 3.8 Confusion Matrix

Summary:

The OptiBraille model demonstrates strong overall performance (88% accuracy) with 20 out of 26 letters achieving reliable recognition rates above 90%. However, critical failures in letters J and T (0% accuracy) and moderate issues with V-W confusion require immediate attention before deployment.

3.3 Hardware and System Integration Results

3.3.1 Raspberry Pi 4 Performance

System Specifications Met:

- Processor: ARM Cortex-A72 @ 1.5GHz performing as expected

- RAM Usage: Peak 62MB during inference (4GB available - ample headroom)
- Storage: Model + OS fits comfortably on 32GB microSD card
- Power Consumption: 5-8W during active inference

Result: Hardware platform proven effective for edge deployment

3.3.2 Pi Camera Performance

Camera Specifications Validated:

- Resolution: 5MP (3280×2464) captured without errors
- Latency: <50ms per frame capture via CSI interface
- Quality: Sufficient for Braille dot detection in varied lighting
- Focus: Fixed focus at 10cm working distance optimal for Braille

Result: Camera integration successful, consistent image capture

3.4 Key Achievements: Successfully deployed a complete Braille character recognition system on Raspberry Pi 4, proving that sophisticated deep learning models can operate on affordable, low-power embedded devices without cloud dependency. The project has been able to implement a convolutional neural network (CNN) with Raspberry Pi 4, with real-time Braille character recognition with efficiency of 88 percent. It was not connected online in any way, and was highly performant with user privacy, different real-world situations like variations in light, angles and wear and tear on surfaces.

3.5 Limitations and Challenges: As much as the system was successful, it had a number of limitations. Letters J and T also had low recognition accuracy of 0%, whereas the letters V and W were frequently mixed and so errors of classification were huge. The model also suffered with the letter H and it had almost 47% error. There was a deterioration of performance when processing Braille characters on highly worn or embossed surfaces. There was only the provision of English Grade 1 Braille in the dataset through which training was done hence limiting the flexibility of the model to more complicated Braille systems.

CONCLUSION

OptiBraille has been able to exhibit a viable low-cost edge AI-based solution to Braille character recognition, a total accuracy of 88 percent with 16 out of 26 letters having scores equal to or less perfect 100% recognition rates. The system uses cost-effective Raspberry Pi 4 hardware of just 6,000 rupees - 98 percent cheaper than commercial solutions (3-8 lakhs rupees) - as it provides real time performance with minimal equipment. These successes identify that advanced AI solutions can be implemented on resource smart optimization, that is, of constrained devices, by 8-bit model quantization with 56% reduction in sample size and 40% inference speed with minimal accuracy loss. The project exhibits good performance within real-life environment, sustaining. 90-91% under different lighting conditions, camera angle, and in a deteriorated condition. Braille documents (86% accuracy) pipeline excellence indicates 98-100% preprocessing due to flawlessness, consistent quality is ensured by the success rates throughout the stages of image transformation. However, whole failure at letters J and T (0% accuracy) and great confusion in V and W pairs recognize specific weaknesses that need special efforts during classes load balancing, information addition, and architecture optimization.

REFERENCES:

[1] P. Kathiria, S. H. Mankad, J. Patel, M. Kapadia, and N. Lakdawala, "Assistive systems for visually impaired people: A survey on current requirements and advancements," Nov. 14, 2024, *Elsevier B.V.* doi: 10.1016/j.neucom.2024.128284.

[2] S. A. Raza, M. S. Farooq, U. Farooq, H. Karamti, T. Khurshaid, and I. Ashraf, "A Convolutional Neural Network Based Optical Character Recognition for Purely Handwritten Characters and Digits," *Computers, Materials and Continua*, vol. 84, no. 2, pp. 3149–3173, 2025, doi: 10.32604/cmc.2025.063255.

[3] J. P. Nyakuri, C. Nkundineza, O. Gatera, and K. Nkurikiyeyezu, "Tiny-MobileNet-SE: A Hybrid Lightweight CNN Architecture for Resource-Constrained IoT Devices," *IEEE Access*, vol. 13, pp. 108389–108401, 2025, doi: 10.1109/ACCESS.2025.3582055.

[4] R. Zhang and A. C. S. Chung, "EfficientQ: An efficient and accurate post-training neural network quantization method for medical image segmentation," *Med Image Anal*, vol. 97, Oct. 2024, doi: 10.1016/j.media.2024.103277.

[5] K. Sun, X. Wang, X. Miao, and Q. Zhao, "A review of AI edge devices and lightweight CNN and LLM deployment," Jan. 21, 2025, *Elsevier B.V.* doi: 10.1016/j.neucom.2024.128791.

[6] J. Sklansky and S. Member, "Image Segmentation and Feature Extraction," 1978.

[7] R. Liu, B. Sisman, Y. Lin, and H. Li, "FastTalker: A neural text-to-speech architecture with shallow and group autoregression," *Neural Networks*, vol. 141, pp. 306–314, Sep. 2021, doi: 10.1016/j.neunet.2021.04.016.

[8] E. Eren and C. Demiroglu, "Deep learning-based speaker-adaptive postfiltering with limited adaptation data for embedded text-to-speech synthesis systems," *Comput Speech Lang*, vol. 81, Jun. 2023, doi: 10.1016/j.csl.2023.101520.

[9] M. Afif, R. Ayachi, Y. Said, and M. Atri, "Deep embedded lightweight CNN network for indoor objects detection on FPGA," *J Parallel Distrib Comput*, vol. 201, Jul. 2025, doi: 10.1016/j.jpdc.2025.105085.

[10] J. Shi, X. Qian, S. Lin, and X. Chen, "A High Dynamic Range Capacitance-to-Digital Converter with Adaptive Parasitic Compensation," in *2021 5th IEEE Electron Devices Technology and Manufacturing Conference, EDTM 2021*, Institute of Electrical and Electronics Engineers Inc., Apr. 2021. doi: 10.1109/EDTM50988.2021.9420856.