International Journal of Innovative Research in Computer Science and Technology (IJIRCST)
ISSN (Online): 2347-5552, Volume-14, Issue-4, July 2026
DOI: https://doi.org/10.55524/ijircst.2026.14.4.2
Article ID IRP-1776, Pages 10-18
www.ijircst.org

A Deep Learning-Based Framework for Marathi OCR and Automated Braille Translation

Madhav A. Kankhar[1], and C Namrata Mahender[2]

[1] Department of Computer Science & IT, Dr. Babasaheb Ambedkar Marathwada University, Chhatrapati Sambhajinagar, Maharashtra, India
[2] Department of Computer Science & IT, Dr. Babasaheb Ambedkar Marathwada University, Chhatrapati Sambhajinagar, Maharashtra, India

Correspondence should be addressed to Madhav A. Kankhar; madhavkankhar.07@gmail.com

Received: 1 June 2026; Revised: 16 June 2026; Accepted: 29 June 2026

ABSTRACT- This paper presents a deep learning-based framework for converting printed Marathi textbook content into Braille to improve accessibility for visually impaired readers. The proposed system integrates image pre-processing, optical character recognition, character segmentation, convolutional neural network-based classification, Unicode reconstruction, and automated Braille translation into a unified pipeline. Before any character recognition can happen, the scanned pages of Marathi textbooks go through a series of preparation steps. The images are first converted to grayscale, then cleaned up to remove noise, made black and white through Binarization, straightened if they're slightly tilted, and finally sharpened for better clarity. Only after this groundwork is done does the actual character extraction and recognition begin. Once the characters are identified, they're pieced back together into proper Marathi Unicode text. From there, a rule-based system maps each character to its corresponding Braille symbol keeping the process structured and predictable. When tested on a real set of scanned textbook pages, the system performed well, achieving strong character recognition accuracy and producing Braille output that's genuinely reliable for printed educational content. The combination of deep learning with Unicode-aware post-processing turned out to make a meaningful difference in how accurately the Marathi text is read and converted. What makes this work particularly promising is that it's not just accurate it's also scalable. It offers a practical path toward building accessible learning materials, and with some further development, the same approach could be adapted for other Indian languages or applied to documents with more complex layouts down the line.

KEYWORDS- Marathi Braille, CNN, OCR, Text-to-Braille Conversion, Devanagari Script, Machine Learning, Deep Learning

I. INTRODUCTION

Braille is a tactile writing method used by people who are blind or visually challenged. It was made by Louis Braille in the early 1800s. Braille is a writing system designed to help blind individuals read and write through touch. Instead of printed letters, it uses patterns of raised dots that can be felt with the fingertips, and is typically embossed onto paper or other surfaces. As shown in **Figure 1**, each character or symbol is represented by a unique arrangement of dots within a six-dot grid. This simple but clever system is flexible enough to represent not just letters and numbers, but also punctuation and even musical notation. The system was created by Louis Braille, a Frenchman who lost his sight in a childhood accident. At just 15 years old, in 1824, he developed a code for the French alphabet that marked a significant leap forward from earlier attempts at tactile writing, essentially achieving overnight what others had struggled to accomplish before him [1].

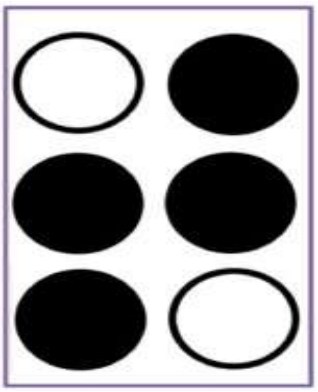


Figure 1: Braille six dot cell

Braille developed his system over many years, and by 1829, it had evolved to include musical notation a remarkable achievement that extended its usefulness far beyond the written word. Meanwhile, on the other side of the world, the Devanagari script tells a very different but equally fascinating story. Rooted in the ancient Brahmi script, it has been in continuous use for well over 1,500 years; quietly evolving into one of the most widely recognized writing systems across South Asia. Today, it forms the backbone of several major languages Hindi, Marathi, Nepali, and Sanskrit among them making it an integral part of daily life for hundreds of millions of people [2]. Devanagari belongs to a category of writing systems called an abugida something that doesn't quite fit the boxes we usually think of, like alphabets or syllabic scripts. What makes it unique is that every consonant already comes with a vowel baked

in. Writers then nudge that vowel around using small diacritical marks, or silence it altogether, rather than spelling out vowels and consonants as separate, independent pieces. Everything folds into one compact unit. The script has a satisfying internal logic to it, though anyone who sits down to learn it quickly realizes there's more going on beneath the surface than your typical alphabet. The most immediately recognizable thing about it the detail that catches the eye before anything else is that horizontal line cutting across the tops of the characters. It isn't decorative. It actively links letters together within a word, giving Devanagari that distinctive shelved, continuous appearance that sets it apart at a glance [3]. Written from left to right, Devanagari consists of vowels, consonants, diacritical marks, ligatures, and conjunct forms where multiple consonants merge into a single shape. This complexity makes digital text processing particularly challenging, as characters frequently change form when combined with others, and words are often written without explicit spacing between them, further complicating automated text recognition and processing tasks. [4]. Standard braille refers to the method of producing printed materials where raised dots are embossed onto thick sheets of paper, each arrangement mimicking what is called a braille cell. Each cell follows a simple but consistent structure three dots running vertically, forming a column, and two dots running horizontally, forming a row. 64 distinct designs can be made by arranging six dots in this way. A dot arrangement with at least one raised dot and up to six raised dots is called a cell [5]. Reading and writing in Indian languages can be challenging for both students and teachers due to the country's multilingual nature. Scripts like Hindi, Bangla, Gujarati and Marathi lack uppercase-lowercase distinctions and often use complex compound characters. The large sets of consonants and vowels further add to the difficulty, especially for visually impaired learners [6].

Marathi is a particularly demanding language when it comes to text recognition, mainly because of how complex the Devanagari script really is. Unlike simpler alphabets, Devanagari brings together a whole mix of vowels, consonants, modifiers, ligatures, and compound characters, all of which need to be handled with great care. On top of that, differences in font styles, varying document quality, and inconsistent scanning conditions make it even harder to accurately recognize characters and piece the text back together correctly. All of these factors combined mean that standard approaches simply don't cut it. You need specialized preprocessing and recognition methods that can actually hold the language together properly before even thinking about converting it into Braille. To tackle these issues head on, this study puts forward a framework built on deep learning that brings several key processes together into one smooth pipeline. This includes enhancing the input images, segmenting individual characters, running recognition through a convolutional neural network, rebuilding the text in Unicode and finally translating everything automatically into Braille, all working together in a single, connected system. The system is designed to process scanned Marathi textbook pages and generate accurate Braille representations suitable for accessible educational materials.

II. RELATED WORK OVERVIEW OF BRAILLE CHARACTER IN INDIAN LANGUAGE

Several Indian languages are represented in braille in India. Every language comes with its own Braille script, carefully designed to capture the unique sounds, characters, and linguistic nuances of that particular language. When it comes to Indian languages, this is especially fascinating each script has been thoughtfully adapted to reflect the rich diversity of the subcontinent's linguistic heritage. Here's a brief look at the Braille scripts used across some of the most widely spoken Indian languages (See **table 1**).

Table 1: Overview of Braille character in Indian language

Sr. No.	Technique / Dataset Used	Observation	Result & Discussion
1	CNN with transfer learning based on MobileNetV2 for translating Kannada text to Braille [7]	For reliable training, a dataset of 734 Kannada characters was expanded to 49,912 pictures.	Scalability for multilingual Braille systems was demonstrated by achieving 99.49% accuracy and effectively mapping identified characters to Unicode and Braille representations.
2	OCR research literature review using PRISMA (2015–2025) [8]	97 OCR research publications on scene text recognition, multilingual processing, and AI models were examined.	identified issues with handwritten OCR and low-resource languages; suggested further research on self-supervised learning, multimodal AI, AutoML, and TinyML.
3	Dataset of handwritten characters in Bangla [9]	59,892 samples of compound, primary, vowel, and consonant characters were included in the collection.	Offers a thorough baseline for deep learning models to be trained for precise recognition of Bangla characters.
4	Tamil-DB handwritten dataset [10]	For postal automation, almost 500 authors provided handwritten Tamil city names.	Captures a variety of writing styles and acts as a practical standard for handwriting recognition and Tamil OCR systems.
5	The Tamil-Kannada multipurpose dataset and the Kannada Handwritten Text Database (KHTD) [11]	Comprises 26,115 words, 4,298 text lines, and 204 documents; another dataset has almost 100,000 words in 600 classifications.	Uses unified models for South Indian scripts to support cross-script research and reliable handwritten text recognition.

6	Handwritten numeral dataset with multiple scripts [12]	Include numerals gathered from thousands of writers in Bangla, Devanagari, and Oriya.	Allows for extensive handwriting variability in multilingual numerical recognition research.
7	PBOK benchmark dataset at the page level [13]	Covers Kannada, Oriya, Bangla, and Persian with annotations at the pixel and content levels.	Contains more than 423,980 characters and 707 pages, making it a useful baseline for OCR and segmentation tasks.
8	Word-level dataset of numbers [14]	Include numeric strings from 43 authors in Bangla, Devanagari, Roman, and Urdu.	Offers multilingual numerical graphics that are helpful for assessing OCR systems in four popular scripts.
9	PHDIndic_11 handwritten dataset with many scripts [9]	Comprises 1,458 pages of images from 11 Indian scripts written by 463 authors.	Incorporates a variety of regional writing styles and makes it easier to construct generalized multi-script handwriting recognition systems.
10	Scene-text dataset for railway station signboards [15]	Includes 500 semi-automatically annotated signboard photos in Hindi, Odia, Urdu, Telugu, and English.	It serves as a useful baseline for scene text recognition by introducing real-world OCR problems such illumination variations, occlusion, and perspective distortion.

III. MAJOR CHALLENGES FOR INDIAN LANGUAGES

As highlighted in NEP 2020 [16], learning in one's mother tongue plays a vital role in building stronger understanding and cognitive growth and this holds especially true for students who are visually impaired. When it comes to Marathi-speaking learners, the need for well-designed assistive tools to support Braille education becomes even more pressing. These students face a unique set of hurdles: the complexity of the Marathi script itself, a shortage of appropriate learning materials, limited access to technology, socio-economic challenges, and in many cases, a general lack of awareness around available support systems. Add to this the cognitive demands of learning Braille, and the gap in accessible education becomes hard to ignore. Addressing these barriers isn't just important it's necessary if we truly want to build learning solutions that are both inclusive and effective for this community.

A. Socio - economics challenges

Despite real progress, visually impaired individuals in India still face social stigma, limited awareness of welfare schemes, and financial barriers to education. Many families hesitate to enrol their children, and supporting institutions often lack Braille materials, trained teachers, and proper infrastructure. Marathi-speaking learners face an added disadvantage due to the dominance of English-medium instruction, while insufficient vocational training further limits their employment prospects and financial independence. Addressing these gaps through greater awareness, accessible institutions, and stronger support systems remains essential for enabling inclusive education and self-reliance [17].

B. Gaps in Implementations of Government policies

The NEP 2020 takes a strong step toward inclusive education by ensuring that visually impaired students get fair and equal opportunities to learn. To make this possible, it supports the use of Braille textbooks, audio-based resources, assistive technologies, and ICT-enabled learning tools [18]. On top of that, programs like Samagra Shiksha Abhiyan, Sugamya Bharat Abhiyan, dedicated teacher training, and resource rooms have gone a long way in making education more accessible for these students. That said, the ground reality still tells a different story. Limited awareness among communities, weak skill development support, financial and social barriers, hesitation from parents and inconsistent implementation across regions continue to hold back the true potential of these efforts. If we genuinely want inclusive education to reach every child, there is a real need to strengthen execution at the grassroots level and ensure that assistive resources are available not just on paper, but in practice.

C. Psycholinguistic and cognitive challenges

Learning Marathi Braille is no simple task, and much of that difficulty comes down to how our minds process language and symbols. Devanagari script, which forms the foundation of Marathi, is inherently complex it carries consonant clusters, vowel modifiers, and a range of tactile symbols that can feel remarkably similar under the fingertips. Reading Braille sequentially, letter by letter, puts a heavy load on working memory, which not only slows down reading speed but can also lead to confusion and mental fatigue over time. On top of that, the shortage of quality learning materials makes it harder for learners to build vocabulary and develop real fluency. All of these points to a clear and pressing need simpler Braille representations, well-structured training programs, and learning tools that make smart use of technology could go a long way in making Marathi Braille more accessible and learnable for everyone.

D. Resource and Training Gaps in Marathi Braille

Finding good teachers and proper learning materials for Marathi Braille is still a real struggle, especially in villages and smaller towns where resources are already hard to come by. Walk into many schools and you'll notice the shelves are nearly bare few Marathi Braille textbooks, barely any reference books, and almost no tactile aids that visually impaired students actually need to learn through touch. Without these basics, students simply don't get enough exposure to their own language. Teacher training makes things worse in a quiet way most programs focus on English or Hindi Braille, and Marathi gets pushed to the side, so even well-meaning teachers end up with gaps in what they can actually teach. When Braille standards get updated or changed, that information rarely makes its way into any formal training, leaving teachers working with outdated knowledge. And on the technology front, things like digital Marathi Braille displays or proper learning software are largely out of reach for most schools. Taken together, all of

this quietly but seriously holds back Marathi-speaking visually impaired students from building the literacy and academic foundation they deserve.

IV. SYSTEM WORKFLOW

A. Input Image

Initially, Marathi data was collected from the Standard V Marathi E-Balbharati PDF [19]. Since the file ran into corruption issues during processing, it was converted into images and packed into a ZIP archive. From there, all 132 valid image files were pulled out and saved for the next step, which was OCR processing.

B. OCR Pre-Processing

Before the images could be run through any OCR system, they had to go through a fairly involved pre-processing routine. Each raw image was worked on step by step, with every stage building on the last; all to make sure the final output was as clean and readable as possible for the recognition engine to handle accurately.

i) Convert to Grayscale
All extracted colour images were put through a grayscale conversion using the PIL (Pillow) library, stripping away colour channels and keeping only the brightness information needed for OCR, with the final outputs saved to the grayscale_images directory.

ii) Apply Thresholding
Grayscale images were converted to pure black and white by applying a binary threshold using OpenCV, which cleanly separated the text from the background, and the resulting images were then saved to the binarized_images directory.

iii) Remove Noise
Morphological Opening through OpenCV's morphologyEx function using the MORPH_OPEN operation was applied to remove small specks and noise from the binarized images while keeping the text structure intact, with the cleaned images then saved into the denoised_images directory.

iv) Correct Skew & Enhance Contrast
Skewness was detected and corrected by applying canny edge detection, Hough Line Transform, and affine transformation the text was straightened and the results were saved to the deskewed_images directory. Contrast was then improved through CLAHE, which boosted local contrast to make the text more prominent for OCR, with the final images saved to the processed_images directory.

Finally, contrast enhancement was applied using CLAHE through OpenCV's cv2.createCLAHE function, which worked by boosting local contrast across different regions of the image rather than treating the entire image uniformly. This made the text noticeably more prominent and legible, setting it up well for accurate OCR. Once fully processed, the images were saved to the processed_images directory.

C. Character Segmentation

Optical Character Recognition was carried out using Tesseract, which was set up with a Marathi language model to make sure the text pulled from the pre-processed images was as accurate as possible. On the Python side, pytesseract acted as the bridge to Tesseract, keeping everything connected smoothly within the pipeline. But the tool did more than just lift text off the page — it also picked up detailed bounding box data, meaning it tracked exactly where each character, word, and line sat within the image. That spatial information turns out to be quite important when you need to make sense of how a document is actually laid out and structured. The raw extracted text output was saved as text files in the ocr_output_text directory for further linguistic processing, snapshot 1 show the sample ocr output text.

```
11 आम्ही, भारताचे लोक, भारताचे एक सार्वभोम
12 समाजवादी धर्मनिरपेक्ष लोकशाही गणराज्य घडविण्याचा
13 व त्याच्या सर्व नागरिकांस:
14 सामाजिक, आर्थिक व राजनैतिक न्याय;
```

Snapshot 1: Show the sample OCR output text

While the bounding box data containing positional and structural details was stored as structured CSV files in the ocr_output_boxes directory, making it easier to reference and analyse the recognized content in a tabular format, **Table 2** show sample structured CSV file in the OCR output boxes.

Table 2: Sample structured CSV file in the OCR output boxes.

level	page_num	block_num	par_num	line_num	word_num	left	top	width	height	conf	text
5	1	1	1	1	1	500	454	48	39	6	7,
5	1	2	1	1	1	490	504	115	38	95	भारताचे
5	1	3	1	1	1	790	522	3	8	0	ः
5	1	4	1	1	1	793	547	9	8	0	८>
5	1	5	1	1	1	765	571	15	22	45	छे
5	1	6	1	1	1	393	681	68	26	77	आम्ही,
5	1	6	1	1	2	471	680	68	24	96	भारताचे
5	1	6	1	1	3	545	680	51	27	96	लोक,
5	1	6	1	1	4	606	680	68	23	96	भारताचे
5	1	6	1	1	5	680	688	32	19	92	एक

D. CNN Based OCR Recognition

i) Character Dataset Creation for CNN

This pipeline begins with the creation of a character dataset for CNN training, where individual character and word images are cropped from processed images using bounding box data obtained from OCR. Each cropped image is associated with its recognized Unicode character as a label, and the resulting images are stored in the character_dataset directory alongside a master CSV file called character_dataset.csv, snapshot 2 show the character dataset creation for CNN, that maps each image path to its corresponding label.

```
Extracting and saving character/word images to 'character_dataset'...

Successfully extracted 33441 character/word images.
Dataset metadata saved to 'character_dataset/character_dataset.csv'.
Cropped images are in the 'character_dataset' directory.
```

Snapshot 2: Illustrates the character dataset creation process for the CNN model

ii) Dataset Splitting for CNN Training

Once the dataset is ready, it gets split into three parts training, validation, and test sets roughly following an 80/10/10 ratio. To make sure each character class is fairly represented across all three subsets, stratified splitting is applied based on the character labels. This splitting is done using sklearn's train_test_split function, and the resulting files train_dataset.csv, val_dataset.csv, and test_dataset.csv are saved in the same directory. Snapshot 3 illustrates how this dataset splitting is set up for CNN training.

```
Total samples in dataset: 33441
Samples remaining after initial filtering of single-occurrence labels: 22706
Samples remaining in temp_df after secondary filtering: 2894

Training set samples: 18164
Validation set samples: 1447
Test set samples: 1447
```

Snapshot 3: Illustrates the label encoding process used for CNN training

iii) Image Pre-processing & Label Encoding for CNN Training

After splitting the dataset, the images from each subset go through a preprocessing stage before being fed into the model. Every image gets loaded and resized to a fixed dimension of 32×32 pixels, ensuring all inputs share a consistent shape. The pixel values are then scaled down to fall between 0 and 1, and an extra channel dimension is added to meet the input format that CNNs expect. On the label side, character labels are first converted into numbers through a LabelEncoder, and then transformed into one-hot encoded vectors using to_categorical. This entire process produces the final training, validation, and test arrays X_train, y_train, X_val, y_val, X_test, and y_test along with a label_classes.npy file that can later be used to decode the model's predictions. Snapshot 4 illustrates the label encoding process carried out for CNN training.

```
--- Label Encoding ---
Shape of X_train: (18164, 32, 32, 1)
Shape of y_train: (18164, 3869)
Shape of X_val: (1447, 32, 32, 1)
Shape of y_val: (1447, 3869)
Shape of X_test: (1447, 32, 32, 1)
Shape of y_test: (1447, 3869)
Number of unique classes (Marathi characters/words): 3869
Label classes saved to 'character_dataset/label_classes.npy'.
```

Snapshot 4: Illustrates the label encoding process used for CNN training.

iv) CNN Model Definition & Training

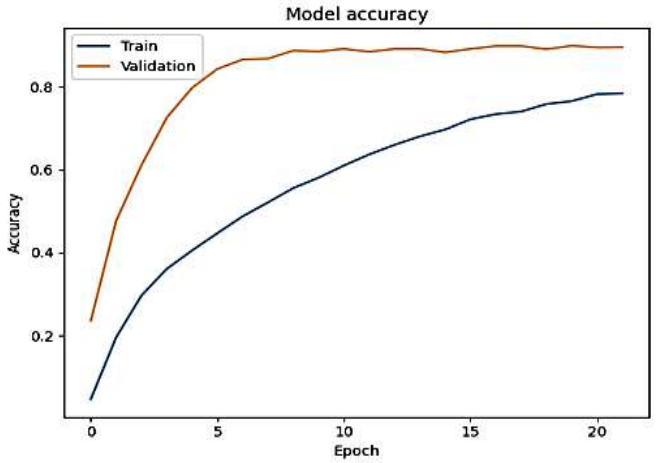
A Convolutional Neural Network is then defined using TensorFlow/Keras, bringing together convolutional, pooling, flatten, dense, and dropout layers into a cohesive architecture. Once the structure is in place, the model is compiled with the Adam optimizer paired with a categorical cross-entropy loss function, making it ready for the training phase. It is then trained on the prepared data with both early stopping and model check pointing active in the background, ensuring that the best performing version of the model is preserved throughout the process. This ultimately produces a best_cnn_model.h5 file alongside a history object that captures all the training metrics along the way, and snapshot 5 illustrates the CNN Model Training Completion report in detail.

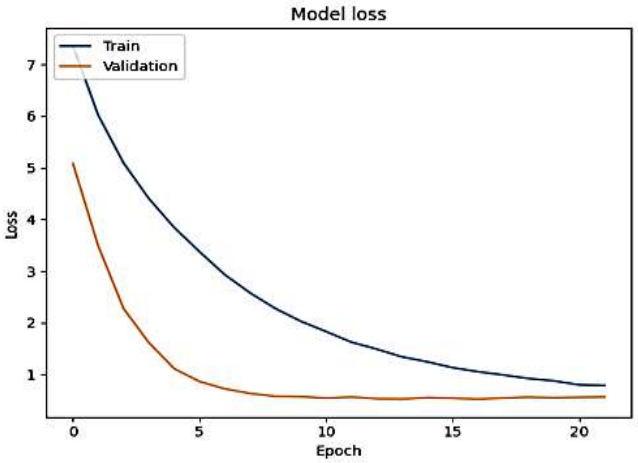
```
Epoch 22: val_accuracy did not improve from 0.89910
568/568 ━━━━━━━━━━━━━━━━━━━━ 32s 56ms/step - accuracy: 0.7842 - loss: 0.7869

Model training complete.
Best model saved to 'best cnn model.h5'.
```

Snapshot 5: Shows the successful completion of CNN model training.

v) CNN Model Evaluation

After training, the model's performance is evaluated by visualizing accuracy and loss curves over epochs using matplotlib, followed by evaluating the model on the unseen test set to obtain final loss and accuracy scores. A detailed classification report is also generated using sklearn.metrics.classification_report. Snapshot 6 shows the CNN Training Model Accuracy & Loss.






```
Evaluating model on the test set...
Test Loss: 0.5735
Test Accuracy: 0.8970

Generating classification report...
46/46 ━━━━━━━━━━━━━━━━━━━━ 1s 18ms/step
Classification Report (summarized for brevity if many classes):
```

Snapshot 6: Illustrates the accuracy and loss curves of the CNN model.

vi) Text Reconstruction from OCR Output

Finally, the word-level bounding box data obtained earlier from OCR is used to reconstruct the original text of each page in proper reading order by sorting detected words by their vertical and then horizontal positions. Individual page text files are saved, and all content is consolidated into a single full_document_reconstructed.txt file that contains the entire book's reconstructed text in sequence, snapshot 7 shows the Text Reconstruction from OCR Output.

फडके ते सी. (१८९४ १९७८) ना. :
लोकप्रिय कादंबरीकार, लघुनिबंधकार. 'कलेकरिता सातत्याने मांडणारे *लघुनिबंध' मराठीतील समीक्षक. किंवा 'गुजगोष्टी' वाङ्मयप्रकाराचे आद्यप्रवर्तक ते ओळखले वेधक व्यक्तिरेखाटन, संवादचातुर्य, रचनेच्या बांधेसूदपणा, तालित्यपूर्ण वाचकांशी संवाद साधण्याचे त्यांच्या लेखनाची वैशिष्ट्ये. त्यांचे कौशल्य ही 'दौल ; *अखेरचे बंड' शंभराहून अधिक कादंबऱ्या 'गुजगोष्टी', 'धूम्रचलये' हे लघुनिबंधसंग्रह; 'प्रतिभासाधन' *प्रतिभाविलास' हे लेखन प्रसिद्ध आहे. रत्नागिरी येथे भरलेल्या अखिल साहित्य संमेलनाचे ते साली भारत होते. सरकारने किताब देऊन त्यांचा केला आहे. साली गौरव 'पद्मभूषण' १९६२ भारत हा

Snapshot 7: Illustrates the text reconstruction process from the OCR output.

vii) Braille Conversion Pipeline

The process begins with the creation of a custom Marathi-to-Braille mapping system, where a dedicated dictionary called marathi_to_braille_map was built to associate individual Marathi characters, matras, and conjuncts with their corresponding Braille dot patterns. A function named marathi_to_braille was then developed to convert Unicode Marathi text into its Braille representation. To ensure accuracy, it uses a longest-match strategy, which means the function always tries to match the most complex sequence first whether that's a conjunct, a multi-character combination, or a vowel sign before falling back to a single character. Snapshot 8 illustrates this conversion process in action, showing how the function handles the nuances of Marathi script by prioritizing these longer, more intricate patterns over simpler single-character matches, resulting in a more faithful and precise Braille output.

```
--- Demonstrating Marathi to Braille Conversion ---
Marathi: मराठी
Braille: [1, 2, 5] [1, 2, 3, 5] [4] [1, 2, 4, 5, 6] [3, 5]

Marathi: मराठी कुमारभारती दहावी इयत्ता  संविधान भारताचे भाग क ४ |
Braille: [1, 2, 5] [1, 2, 3, 5] [4] [1, 2, 4, 5, 6] [3, 5]
```

Snapshot 8: Illustrates the process of converting Marathi text into Braille.

The output of this step was a file named full_document_braille.txt, snapshot 9 show the output of braille conversion, which contains the complete Braille representation of the entire processed document.

full_document_braille.txt ×

```
1 [1, 2, 5] [1, 2, 3, 5] [4] [1, 2, 4, 5, 6] [3, 5]
  [1, 3, 4, 5, 6] [1, 3, 4, 5, 6] [1, 3, 4, 5] [3,
  3] [3, 5] [1, 4, 6] [1, 2, 3, 5] [3, 5] [1, 2, 5]
```

Snapshot 9: Show the sample output txt file of braille conversion

Following the conversion, a quality evaluation step was introduced to measure how accurately the Marathi text was translated into Braille. A function called count_braille_conversions was developed to analyze the conversion results page by page, counting how many text segments including individual characters, matras, and conjuncts were successfully mapped to a valid and non-empty Braille pattern. The results were organized into a Pandas DataFrame that clearly displayed the total number of segments, the number of correctly converted segments, and the overall conversion percentage first 5 pages page of the document, **Table 3** shows the braille conversion quality evolution.

Table 3: First 5 pages braille conversion quality evolution

index	File Name	Total Mappable Segments	Converted Segments	Conversion Percentage
0	Marathi 10 Class Book_page-0001.txt	29	26	89.65517241379311
1	Marathi 10 Class Book_page-0002.txt	912	862	94.51754385964912
2	Marathi 10 Class Book_page-0003.txt	227	210	92.51101321585902
3	Marathi 10 Class Book_page-0004.txt	1520	1417	93.22368421052632
4	Marathi 10 Class Book_page-0005.txt	533	489	91.74484052532833

viii) Braille Conversion Accuracy Evaluation

Together, these steps form a complete workflow that takes raw image data through computer vision processing, OCR-based text extraction, CNN-based character recognition, and finally converts the reconstructed Marathi text into Braille, with built-in accuracy evaluation to measure the quality of the conversion at every stage.

In below **Table 3** summarizes the results of a text conversion or mapping process applied to a collection of 133 book pages stored in a .txt file. Out of a total of 1,81,3,820 segments identified across all the pages, 1,38,1,290 segments were successfully converted, yielding a conversion rate of approximately 76.15%. This means that roughly three-quarters of the content was processed and converted successfully, while the remaining ~24% of segments just over 432,000 could not be converted, possibly due to formatting inconsistencies, unsupported characters, encoding issues, or other anomalies in the source text.

Table 3: Summarizes the results of a text conversion or mapping process

File Name	Total 133 Book Pages in .txt
Total Mappable Segments	1,81,3820
Converted Segments	1,38,1290
Conversion Percentage	76.153643

ix) Database Explanation

To build the dataset for this study, we started with a Marathi textbook and went through it page by page scanning and extracting images from 132 pages in total. These images were then cleaned up and prepared for OCR processing. Using Tesseract OCR, we pulled out the text and mapped out where each character and word appeared on the page. Those coordinates helped us crop out 33,441 individual images of Marathi characters and words; each one tied to its correct label and saved neatly into a CSV file so the data would be ready for supervised learning.

We then split this collection into three parts: 18,164 images for training, and 1,447 each for validation and testing. Before feeding any of it into the model, every image was resized to 32 × 32 pixels, normalized, and encoded to suit CNN training requirements. This carefully prepared dataset sits at the heart of the entire OCR and Braille conversion pipeline making it possible to accurately read Marathi text and then transliterate it into Braille using NLP techniques.

V. RESULT AND DISCUSSION

To test how well the proposed system works, we ran scanned pages from Marathi textbooks through the entire pipeline starting with image pre-processing, moving through OCR and CNN-based character recognition, and finishing with Unicode reconstruction and Braille translation. Building the dataset was no small task; we ended up extracting and labelling over 33,000 character and word images in total. Once cleaned up, these were split into roughly 18,000 samples for training, with about 1,400 each set aside for validation and testing. Every image was resized to 32 × 32 pixels and normalized so the neural network would receive consistent input throughout.

When we tested the trained CNN on data it hadn't seen before, it held up quite well landing at around 89.70% accuracy with a test loss of 0.5735. Given how complex the Marathi script is, with its many compound characters and visually similar glyphs, that's a solid result. The errors that did occur were mostly down to characters that look alike, variations in font styles, and the occasional segmentation issue that's hard to avoid with printed documents.

We also put the full OCR-to-Braille pipeline through its paces using actual text pages. Across the first five sample pages, conversion rates ranged from about 89% to nearly 95%, which showed the system was performing consistently well. Scaling up to the full set of 133 pages, the system worked through over 1.8 million Mappable text segments and successfully converted around 1.38 million of them into Braille giving an overall success rate of 76.15%.

The roughly 24% that didn't convert cleanly came down to a mix of factors: OCR misclassifications, compound characters the system wasn't equipped to handle, Unicode normalization hiccups, and some formatting quirks in the source documents. That said, the results as a whole show the framework is a genuinely viable approach to Marathi-to-Braille conversion. With better CNN architecture, more diverse training data, and improved handling of compound characters and Unicode post-processing, there's good reason to expect both the recognition accuracy and Braille translation quality to improve meaningfully going forward.

VI. CONCLUSION

This study set out to solve a meaningful problem making printed Marathi documents accessible to visually impaired readers and the approach taken was both thoughtful and technically grounded. Rather than relying on a single tool, the researchers built an end-to-end pipeline that stitches together several components: image pre-processing, optical character recognition, a CNN for character recognition, Unicode reconstruction, and finally, Braille translation. Marathi script is notoriously tricky to work with

computationally, given its complex characters, modifiers, and compound symbols, so the fact that the system handles all of this in an automated way is genuinely noteworthy. When it came time to test how well everything worked, the results were encouraging. The CNN model landed at a test accuracy of 89.70% with a test loss of 0.5735 solid numbers for a script as nuanced as Marathi. The Braille translation side of things also held up well, with high conversion rates on individual sample pages and an overall success rate of 76.15% across the full dataset. In practical terms, this means the system can take a printed Marathi document and reliably produce Braille output that visually impaired users can actually read and benefit from.

That said, the system isn't without its rough edges. OCR misclassifications still slip through, some compound characters aren't yet supported, Unicode normalization occasionally causes hiccups, and documents with unusual formatting can trip things up. The researchers are well aware of these gaps and have a clear roadmap for addressing them think larger and more varied training datasets, more sophisticated deep learning architectures, better Unicode post-processing, and a more comprehensive Braille mapping ruleset. With those improvements in place, the system has real potential to become a robust, scalable tool for Marathi language accessibility.

ACKNOWLEDGMENT

The authors gratefully acknowledge the Chhatrapati Shahu Maharaj Research Training & Human Development Institute (SARTHI), Pune, for the fellowship support that enabled the successful completion of this study. This work was also supported by the Rajiv Gandhi Science and Technology Commission, Government of Maharashtra (RGSTC No. RGSTC-11/2022-2023/368–70), and by the PMUSHA Research Grant (Sanction No. PMUSHA/Research Grant/2025-26/132–36). The authors sincerely thank the Computational and Psycholinguistic Research Lab Facility for their excellent support. Additionally, the Department of Computer Science & Information Technology, Dr. Babasaheb Ambedkar Marathwada University, Chh. Sambhajinagar, Maharashtra, India, is acknowledged for providing essential resources and institutional support.

CONFLICTS OF INTEREST

The author declare that they have no Conflicts of Interest.

REFERENCES

[1] Braille, "Braille - Wikipedia, the free encyclopedia." Available from: http://www.Wikipedia.com/Braille

[2] S. Arora, L. Malik, S. Goyal, D. Bhattacharjee, M. Nasipuri, and O. Krejcar, "Devanagari character recognition: A comprehensive literature review," IEEE Access, vol. 13, pp. 1249-1284, 2024. Available from: https://doi.org/10.1109/ACCESS.2024.3520248

[3] N. Gautam, S. S. Chai, and M. Gautam, "The dataset for printed Brahmi word recognition," in Micro-Electronics and Telecommunication Engineering: Proceedings of 3rd ICMETE 2019, Singapore: Springer, 2020, pp. 125-133. Available from: https://doi.org/10.1007/978-981-15-2329-8_13

[4] S. Singh, N. K. Garg, and M. Kumar, "Feature extraction and classification techniques for handwritten Devanagari text recognition: A survey," Multimedia Tools and Applications, vol. 82, no. 1, pp. 747-775, 2023. Available from: https://doi.org/10.1007/s11042-022-13318-9

[5] Acharya, "Introduction to Braille," Multilingual Computing for Literacy and Education. Available from: https://www.acharya.gen.in/

[6] N. B. Jariwala and B. C. Patel, "Multi-Oriented Gujarati Characters Recognition: A Review," International Journal of Research in Computer Science and Information Technology, vol. 2, no. 2(A), pp. 1-5, Mar. 2014. Available from: https://tinyurl.com/8sn5npxx

[7] N. Bhaskar, K. P. Ashritha, P. Tupe-Waghmare, and K. S. Varadaraj, "A Deep Learning-based Text-to-Braille Translation Framework: A Case Study on an Indic Script," in Proc. 6th Int. Conf. Image Processing and Capsule Networks (ICIPCN), Jan. 2026, pp. 771-776. Available from: https://doi.org/10.1109/ICIPCN67432.2026.11439046

[8] N. Khan, A. A. H. Ab Rahman, S. M. Rizvi, I. Y. Alshareef, M. N. Marsono, M. P. Bakht, et al., "Systematic literature review of machine learning models and applications for text recognition," IEEE Access, 2025. Available from: https://doi.org/10.1109/ACCESS.2025.3618109

[9] S. M. Obaidullah, C. Halder, K. C. Santosh, N. Das, and K. Roy, "PHDIndic_11: Page-level handwritten document image dataset of 11 official Indic scripts for script identification," Multimedia Tools and Applications, vol. 77, no. 2, pp. 1643-1678, 2018. Available from: https://doi.org/10.1007/s11042-017-4373-y

[10] S. Thadchanamoorthy, N. D. Kodikara, H. L. Premaretne, U. Pal, and F. Kimura, "Tamil handwritten city name database development and recognition for postal automation," in Proc. 12th Int. Conf. Document Analysis and Recognition (ICDAR), Aug. 2013, pp. 793-797. Available from: https://doi.org/10.1109/ICDAR.2013.162

[11] Nethravathi, C. P. Archana, K. Shashikiran, and R. AG, "Creation of a huge annotated database for Tamil and Kannada OHR," in Proc. Int. Conf. Frontiers in Handwriting Recognition (ICFHR), Nov. 2010, pp. 415-420. Available from: https://doi.org/10.1109/ICFHR.2010.71

[12] U. Bhattacharya and B. B. Chaudhuri, "Databases for research on recognition of handwritten characters of Indian scripts," in Proc. 8th Int. Conf. Document Analysis and Recognition (ICDAR), Aug. 2005, pp. 789-793. Available from: https://doi.org/10.1109/ICDAR.2005.84

[13] Alaei, U. Pal, and P. Nagabhushan, "Dataset and ground truth for handwritten text in four different scripts," International Journal of Pattern Recognition and Artificial Intelligence, vol. 26, no. 4, Art. no. 1253001, 2012. Available from: https://doi.org/10.1142/S0218001412530011

[14] S. M. Obaidullah, C. Halder, N. Das, and K. Roy, "A corpus of word-level offline handwritten numeral images from official Indic scripts," in Proc. 2nd Int. Conf. Computer and Communication Technologies (IC3T), New Delhi, India: Springer, Sep. 2015, pp. 703-711. Available from: https://doi.org/10.1007/978-81-322-2517-1_67

[15] M. Verma, N. Sood, P. P. Roy, and B. Raman, "Script identification in natural scene images: A dataset and texture-feature based performance evaluation," in Proc. Int. Conf. Computer Vision and Image Processing (CVIP), Singapore: Springer, Dec. 2016, pp. 309-319. Available from: https://doi.org/10.1007/978-981-10-2107-7_28

[16] N. Khatun and M. N. Baidya, "Transforming Education in India: NEP 2020 and Inclusiveness," Journal of Research & Innovations in Education, vol. 10, no. 1, pp. 13-26, 2024. Available from: https://publications.cukashmir.ac.in/index.php/jrie/article/view/122

[17] M. A. Kankhar and C. N. Mahender, "A Comprehensive Study of Tactile Education System for Visual Impaired People," in Proc. Int. Conf. Recent Advancements and Modernisations in Sustainable Intelligent Technologies and Applications (RAMSITA), May 2025, pp. 88-99. Available from:

https://www.atlantis-press.com/proceedings/ramsita-25/126011501
[18] Ministry of Education, Government of India, "National Education Policy (NEP)." Available from: https://www.education.gov.in/en/nep-new
[19] Balbharti, "Balbharti eBook Library for Marathi Language." Available from: https://ebooks.ebalbharati.in/pdfs/501000541.pdf