

BRaille TO SPEECH USING CNN

Submitted in partial fulfillment of the requirements for the award of
Bachelor of Engineering degree in Computer Science and Engineering

By

UPADHYE ADITYA GAURAV (Reg. No - 40111367)

THARUNNESH R (Reg. No - 40111331)



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SCHOOL OF COMPUTING

SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

Category-1 University by UGC

Accredited with Grade "A++" by NAAC | 12B Status by UGC | Approved by AICTE

**JEPPIAAR NAGAR, RAJIV GANDHI SALAI,
CHENNAI - 600119**

APRIL - 2024



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Category - I University by UGC

Accredited "A++" by NAAC | Approved by AICTE

www.sathyabama.ac.in

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **UPADHYE ADITYA GAURAV (40111367)** and **THARUNNESH R (40111331)** who carried out the Project entitled "**BRAILLE TO SPEECH USING CNN**" under my supervision from November 2023 to April 2024.

Internal Guide

Dr. J. JESLIN SHANTHAMALAR, M.E., Ph. D.

HEAD OF THE DEPARTMENT
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)

Head of the Department
Jeppiyaar Nagar, P. V. V. Gandhi Salai,
Chennai - 600 119.

Dr. L. LAKSHMANAN, M.E., Ph.D.

Submitted for Viva voce Examination held on

15/04/2024

Internal Examiner

External Examiner

DECLARATION

I, **UPADHYE ADITYA GAURAV** (Reg. No - 40111367), hereby declare that the Project Report entitled "**BRAILLE TO SPEECH USING CNN**" done by me under the guidance of **Dr. J. JESLIN SHANTHAMALAR, M.E., Ph.D.**, is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in **Computer Science and Engineering**.

DATE: 15/04/2024

PLACE: Chennai

A handwritten signature in blue ink, appearing to read 'Aditya', enclosed within a circular stamp.

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management** of **Sathyabama Institute of Science and Technology** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. Sasikala, M.E., Ph.D., Dean**, School of Computing, **Dr. L. Lakshmanan, M.E., Ph.D.**, Head of the Department of Computer Science and Engineering for providing me necessary support during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr. J. Jeslin Shanthamalar, M.E., Ph.D.**, for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

ABSTRACT

Braille serves as a globally recognized system designed for the reading and writing needs of individuals with visual impairments. Rooted in a tactile approach, it enables users to perceive and communicate through touch. Its fundamental structure comprises specific arrangements of raised dots, forming letters, numbers, and punctuation symbols. Leveraging the advancements in technology, particularly in the domain of artificial intelligence, Braille content can now be seamlessly converted from images to text using sophisticated methodologies. One such approach involves employing Convolutional Neural Networks (CNN), a class of deep learning algorithms known for their effectiveness in image recognition tasks. By utilizing CNNs, the model can discern the intricate patterns and configurations of Braille dots within an image, deciphering them into their corresponding textual representations. This process facilitates the transformation of Braille content into coherent words and sentences, enhancing accessibility and comprehension for visually impaired individuals. In the implementation of this system, the Keras library plays a pivotal role, offering a versatile framework for conducting various operations essential for machine learning tasks. These include data generation, dataset partitioning, and model evaluation, streamlining the development and assessment process. Following the extraction of text from Braille images, the obtained textual output undergoes further processing to generate synthesized speech. The conversion from text to speech is accomplished using Google Text to Speech technology, which synthesizes natural-sounding audio output from textual input. By integrating this capability into the system, the Braille content is effectively rendered into audible form, enabling individuals with visual impairments to access and comprehend the information presented. Through rigorous training with a substantial number of training epochs, the model achieves a commendable accuracy rate of approximately 96.30%, underscoring its efficacy in accurately converting Braille content into spoken language.

TABLE OF CONTENT

CHAPTER NO.	TITLE	PAGE NO.
	ABSTRACT	i
	LIST OF TABLES	iv
	LIST OF FIGURES	v
	LIST OF ABBREVIATIONS	vi
1	INTRODUCTION	1
2	LITERATURE SURVEY	4
	2.1 Inferences from Literature Survey	6
	2.2 Open problems in Existing System	7
3	REQUIREMENT ANALYSIS	8
	3.1 Feasibility Studies/Risk Analysis of the Project	8
	3.2 Requirements Specification	9
	3.3 Libraries Used	10
	3.3.1 Numpy	10
	3.3.2 Pandas	10
	3.3.3 Matplotlib	10
	3.3.4 Tensorflow	11
	3.3.5 Keras	11
4	DESCRIPTION OF THE PROPOSED SYSTEM	12
	4.1 Selected Methodology or Process Model	12
	4.2 Architecture / Overall Design of Proposed System	15
5	IMPLEMENTATION DETAILS	18
	5.1 Development and Deployment Setup	19
	5.2 Algorithm	20
	5.2.1 Convolutional Neural Network (CNN)	20
6	RESULTS AND DISCUSSIONS	21
	6.1 Presentation of Findings	21
	6.2 Data Analysis and Interpretation	21

7	CONCLUSION	25
	7.1 Conclusion	25
	7.2 Future Work	25
	REFERENCES	26
	APPENDIX	29
	A. SOURCE CODE	29
	B. SCREENSHOTS	43
	C. RESEARCH PAPER	47

LIST OF TABLES

TABLE NO.	TABLE NAME	PAGE NO.
2.1	Results Obtained by various Researchers	6
4.1	Layers in BT-CNN	13
5.1	Comparative Analysis of Existing Methods vs Proposed Method	25

LIST OF FIGURES

FIG. NO.	FIGURE NAME	PAGE NO.
4.1	Image Augmentation	13
4.2	Architecture Diagram	15
5.1	Braille Characters	18
5.2	Convolutional Neural Network	20
6.1	Dataset Used	21
6.2	Creation of BT-CNN Model	22
6.3	Graph of accuracy at each epoch	22
6.4	Graph of loss function at each epoch	23
6.5	Accuracy of BT-CNN Model	23
6.6	Picture showing the braille text conversion for an image	23
6.7	Picture showing the braille text conversion for another image	24

LIST OF ABBREVIATIONS

ABBREVIATION	EXPANSION
ANN	Artificial Neural Network
CNN	Convolutional Neural Network
OBR	Optical Braille Recognition
OCR	Optical Character Recognition
RCSA	Radio Character Segmentation Algorithm
YOLO	You Only Look Once

CHAPTER 1

INTRODUCTION

Traditionally, Braille is used by blind, partially sighted, and deafblind people. Louis Braille created braille in France during the 1820s. The spatial units used to form braille symbols are called braille cells. Imagine the number 6 on a dice. A complete braille cell is made up of six raised dots organized in two parallel vertical rows of three dots each. The digits one through six represent the locations of the dots. When one or more of these six dots are used, there are 64 possible combinations. Both filled and empty cells are also used. Punctuation, numbers, alphabet letters, and even entire phrases can be represented by a group of one or more dots positioned inside the cell.

The significance of Braille extends beyond its role as a means of reading and writing. It serves as a fundamental tool for literacy and education, empowering individuals with visual impairments to access knowledge, pursue academic studies, and participate fully in society. By providing a tactile representation of written language, Braille facilitates the development of language skills, comprehension, and critical thinking among blind and visually impaired individuals, fostering intellectual growth and empowerment.

Furthermore, Braille promotes inclusivity and accessibility in various aspects of daily life, including employment, leisure activities, and social interactions. It allows individuals with visual impairments to access printed materials independently, from books and documents to labels, signs, and menus. Moreover, Braille signage and labeling in public spaces, transportation systems, and facilities contribute to greater independence and autonomy for blind individuals, enabling them to navigate their surroundings with confidence and dignity.

Despite the advent of digital technologies and assistive devices, Braille remains a vital tool for communication and literacy within the blind and visually impaired community. While advancements such as screen readers and voice-to-text software have expanded access to information, Braille continues to offer unique benefits, including tactile feedback, spatial representation, and the ability to read and write

discreetly without reliance on auditory cues. As such, Braille continues to play a pivotal role in promoting independence, inclusion, and equal opportunities for individuals with visual impairments in today's society.

Through raised dots, it whispers the secrets of the written word, providing a unique means to encode text. But, as remarkable as Braille is, it does come with its own set of challenges. Its full potential often remains locked because of the specialised skills and knowledge required for reading and writing. However, the proposed model is here to change that. It's not just about algorithms and data; it's about empowering individuals. It's like a digital bridge that connects Braille with spoken language, making the art of reading accessible to all.

There are several ways to produce Braille, including:

- Braille Typewriter: This is a special typewriter with just six keys, one for each of the dots in a Braille cell. When a key is pressed, a metal stylus embossed the dots for that character onto the paper.
- Braille Printer: This is a kind of printer that embossed (puts raised dots on-) paper to create Braille characters.
- Refreshable Braille Display: This is a device containing row upon row of mechanical or electronic Braille cells. Each cell has multiple pins that can be raised or lowered to form any Braille characters.

A Braille to Speech conversion tool whispers the secrets of the written word through raised dots, providing a unique means to encode text. But, remarkable as Braille is, it comes with its own set of challenges. The specialized knowledge required for reading and writing often leaves it an enigma. Not just about algorithms and data, braille to speech conversion brings a means to empower individuals with the most important of gifts -- the written word. It is a digital bridge, one that could make the art of reading available to everyone.

One of the key advantages of Braille to speech conversion lies in its ability to provide real-time access to written information for individuals who are blind or visually impaired. By converting Braille text into spoken language, these systems empower users to access a wide range of printed materials, including books, documents, signage, and electronic displays, without the need for tactile interpretation. This

enhances accessibility across various domains, including education, employment, recreation, and social interactions, enabling individuals with visual impairments to engage more fully with the world around them.

Moreover, Braille to speech conversion serves as a catalyst for independence, autonomy, and empowerment among individuals with visual impairments. By offering a seamless transition from tactile to auditory communication, these systems enable users to navigate and comprehend written information with greater ease and efficiency. Whether reading textbooks, browsing websites, or accessing public transportation schedules, individuals can rely on Braille to speech conversion to access the information they need in a format that suits their preferences and abilities.

For people with visual impairment, the first and foremost advantage of Braille to Speech conversion is accessibility. It allows users to hear printed material, signages and other written information, which promotes independence and inclusion with the help of audio output. Another key advantage of a Braille to Speech translation model is its ability to translate Braille text into spoken language in real-time. Using a well-designed Braille to Speech model, a blind person would be able to immediately hear spoken output of any Braille text they place their fingers on.

In an educational context, Braille to Speech conversion provides an important form of auditory reinforcement for Braille students. This reinforcement can help students to develop a firm association between new Braille characters and their corresponding spoken language in a faster manner than trial and error alone would allow. The main components of the Braille to Speech system include image preprocessing techniques, character segmentation, and CNN models for character recognition. When a braille character is recognized, the system converts it to its corresponding text form and then uses text-to-speech synthesis to produce natural, audible speech.

CHAPTER 2

LITERATURE SURVEY

Using preprocessing, image segmentation, and feature extraction approaches, M. Hanumanthappa et al. demonstrated that several researchers' efforts were still unsuccessful in achieving 100% accuracy with a high translation rate. In his research, Chollet demonstrated how Inception modules acted as a midway point, situated at opposite ends of a discrete spectrum, between convolutions and depthwise separable convolutions. Kirill Smelyakov et al. used Optical Braille Character Recognition system using Artificial Neural Networks. K M Naimul Hassan et al. developed a dual-purpose system, in 2019, which can help to detect objects around them and a refreshable braille display to display text in the form of Braille Characters. A. A. Choudhury et al. introduced a novel technique that uses the YOLO (You Only Look Once) algorithm to enhance real-time recognition of Braille characters. Known for its speed and recognition accuracy, Yolo was modified to recognize Braille characters, with the aim of improving accessibility and independence for the visually impaired.

A useful method for pronouncing and converting scenes to text was proposed by S. M. Qaisar et al. in 2019. This method can aid visually impaired people in understanding written text found in everyday objects such as caution notices, medicine cartons, and basic food goods. V Jha and K Parvathi proposed a transliteration system, in 2019, which converts Hindi characters into their corresponding Braille Characters using Optical Braille Recognition (OBR) Braille dot pattern. C. Park et al. proposed a 3-stage method using R-CNN as the first stage for banknote images captured in disturbed environments using a smartphone camera. A device was demonstrated by G. C. Bettelani et al. that uses an electromagnetic field to actuate six independently controlled dots. The alphabet letters can be encoded with this device in accordance with the Braille system. Additionally, the blind participants said the gadget was well-accepted, practical, economical, and intuitive. B.-M. HSU used the Radio Character Segmentation Algorithm (RCSA) to recognize complex Braille characters and translate them into

plain English. In a pivotal look, P. Kaur et al. harnessed both conventional and contemporary strategies to broaden a laptop program that converts Hindi Braille into spoken language. By combining those methods, they created a gadget that complements accessibility for the visually impaired.

Shokat et al. did groundbreaking research in assistive technology for visually impaired individuals in 2020. Their use of the GoogleNet Inception model achieved an impressive 95.8% accuracy in predicting Braille characters. Mao developed a model that converts Chinese Braille into spoken language, increasing communication accessibility and inclusion. This pioneering work extends beyond text, providing a natural understanding of the sounds of Braille letters, and contributing to universal inclusion and accessibility. In 2021, T. Kausar et al.'s innovative approach included replacing modules in traditional convolutional neural networks (CNNs) with techniques designed to significantly reduce computational costs. This system addresses a fundamental challenge in Braille character recognition by streamlining computational requirements, especially in critical or real-time. A method to develop a low-cost, portable communication device that can translate any Braille input into alphanumeric text format was presented by Ramachandran et al.

A mechanism that prints the Braille alphabet on a sheet after receiving an alphabet voice command was proposed by M. A. Hussain. In the same year, Bhatia et al. Automatic Arabic language has contributed to the translation system in Braille and contributed to the simple technology, that "Convolutional Neural Network Real Time Arabic Speech Recognition to Arabic Braille" is known as the requirement of the classification, that in real time in Arabic for translation capabilities. Leveraging the power of convolutional neural networks (CNNs), this revolutionary technology empowers individuals with sensory impairment, providing effective communication and access to information, exemplifying that they are committed to being inclusive and improving their lives. Ramani RG and Shanthamalar JJ have conducted extensive prior research on feature extraction methods utilizing full-reference and no-reference image quality measures for picture quality classification. This work presents an automatic feature extraction system for fundus images utilizing image

processing techniques, followed by an automatic image quality classification system employing different classification algorithms. A camera embedded with hand gloves with OCR for conversion to Braille and speech for blinds have been created by Seung-Bin Park and Bonghyun Kim to improve the quality of life.

Table 2.1 Results Obtained by various Researchers

Sl. No.	Researcher	Algorithms	Accuracy(%)
1	K. Smelyakov (2018)	ANN	95.00
2	S. Shokat (2020)	GoogleNet	95.80
3	J Mao (2021)	Prosody Prediction	94.42
4	T Kausar (2021)	DenseNet201	95.20
5	M. A. Hussain (2022)	VGG16	92.42

2.1 INFERENCES FROM LITERATURE SURVEY

- Despite efforts in preprocessing, image segmentation, and feature extraction, achieving 100% accuracy in Braille translation remains challenging, highlighting the complexity of the task and the need for further research.
- Chollet's research demonstrated the significance of Inception modules in enhancing neural network architectures, serving as a critical component between convolutions and depth wise separable convolutions.
- Various approaches, including Optical Braille Character Recognition systems and dual-purpose systems, have been developed by researchers like Kirill Smelyakov et al. and K M Naimul Hassan et al., respectively, to improve accessibility and functionality for individuals with visual impairments.
- Novel techniques such as the utilization of the YOLO algorithm by A. A. Choudhury et al. have been introduced to enhance real-time recognition of Braille characters, aiming to enhance accessibility for the visually impaired.
- Researchers like S. M. Qaisar et al. and V Jha and K Parvathi have proposed methods for converting visual scenes to text and transliterating Hindi characters into Braille, expanding access to information for individuals with visual impairments.

2.2 OPEN PROBLEMS IN EXISTING SYSTEM

- The development of innovative techniques, such as the use of Inception modules and the YOLO algorithm, demonstrates progress in real-time recognition of Braille characters. However, there is still a need to explore more efficient algorithms and methodologies to enhance recognition accuracy and speed further.
- While various transliteration systems and dual-purpose devices have been proposed to improve accessibility for visually impaired individuals, there is a need to address usability issues and enhance the practicality and effectiveness of these systems in real-world scenarios.
- Despite achieving high accuracy rates with models like GoogleNet Inception and innovative approaches like reducing computational costs in traditional CNNs, there are still challenges in scaling these solutions for broader implementation and addressing potential limitations in different application scenarios.
- The development of low-cost, portable communication devices for translating Braille input into alphanumeric text format is a significant advancement. However, there is a need for further research to optimize the design, functionality, and affordability of such devices to make them more accessible and widely available.

CHAPTER 3

REQUIREMENT ANALYSIS

3.1 FEASIBILITY STUDIES / RISK ANALYSIS OF THE PROJECT

Data Quality and Quantity:

Risk: Inadequate or low-quality Braille datasets can lead to inaccurate character recognition and speech synthesis.

Mitigation: Focus on data collection and cleaning efforts. Consider collaborating with organisations or institutions with Braille expertise.

Model Complexity:

Risk: Overly complex CNN models may require significant computational resources and time for training.

Mitigation: Choose model architectures that strike a balance between accuracy and efficiency. Optimise hyperparameters.

User Acceptance:

Risk: If the system is not user-friendly or does not meet the needs of the visually impaired community, adoption may be low.

Mitigation: Involve potential users in the design and testing phases. Collect feedback and make iterative improvements.

Integration Challenges:

Risk: Difficulties in integrating the Braille to Speech system with existing assistive technologies or hardware may impede deployment.

Mitigation: Engage IT professionals and conduct compatibility tests during the development process.

Regulatory and Ethical Compliance:

Risk: Legal and ethical issues related to data privacy, accessibility, and consent may lead to compliance challenges.

Mitigation: Consult with legal experts to ensure that the project adheres to relevant laws and ethical guidelines.

Resource Constraints:

Risk: Budget and resource limitations may hinder the timely development and deployment of the system.

Mitigation: Carefully manage project resources and explore partnerships or grants to support the initiative.

Model Robustness:

Risk: The CNN model may not perform well in recognizing Braille characters in diverse real-world conditions.

Mitigation: Extensive testing and validation, including real-world scenarios, can help identify and address weaknesses in the model.

3.2 REQUIREMENTS SPECIFICATION

Hardware Specification:

Processor	: Intel Core i5 or better
RAM	: 8 GB or more
Hard Disk	: 160 GB or more

Software Specification:

Operating System	: Windows 8 and above
IDE	: Jupyter Notebook / Google Colab
Technology	: Python 3.6

3.3 LIBRARIES USED

The libraries used in this project are:

- Numpy
- Pandas
- Matplotlib
- Tensorflow
- Keras

3.3.1 Numpy

- NumPy, short for Numerical Python, is a library targeted on multidimensional array objects and consists of a set of routines for array processing.
- With NumPy, you can perform mathematical and logical operations on arrays.
- NumPy is an open source numerical Python library.

```
import numpy as np
```

3.3.2 Pandas

- Pandas is a popular Python library on data analysis.
- It is directly related to Machine Learning. As all we have the dataset to be prepared before the training.
- In this case, Pandas are handy as a developed library to data extraction and preparation.
- It provides the implementation of high-level find data structures and a wide-variety tools for data analysis.
- It provides many methods for groping, combining and filtering data.

```
import pandas as pd
```

3.3.3 Matplotlib

- Matplotlib, a versatile Python plotting library, empowers users with a diverse array of visualization options, including line charts, scatter plots, and histograms.
- Its customization features allow precise control over plot appearance, ensuring informative and visually appealing representations.
- Matplotlib's seamless integration with other Python libraries, such as NumPy and Pandas, further solidifies its position as a fundamental tool for data

visualization in the Python ecosystem.

```
import matplotlib.pyplot as plt
```

3.3.4 Tensorflow

- TensorFlow stands as a freely available and open-source software library committed to machine learning.
- Its skills span a wide range of tasks, with a particular emphasis on training and inference for deep neural networks.
- TensorFlow operates as a symbolic math library, relying on fast and differentiable programming standards.
- It helps GPU/CPU computing, permitting the same code to execute seamlessly on each architecture.
- It is renowned for its scalability, TensorFlow excels in distributed computation throughout machines and managing big datasets.

```
import tensorflow as tf
```

3.3.5 Keras

- Keras, a broadly used Python Machine Learning library, affords a high-level neural networks API that seamlessly runs on TensorFlow, CNTK, and Theano.
- Keras is mainly user-pleasant for Machine Learning novices, allowing them to without problems design and build neural networks.
- One of Keras' splendid strengths is its assist for short and simple prototyping

```
from tensorflow import keras
```

CHAPTER 4

DESCRIPTION OF THE PROPOSED SYSTEM

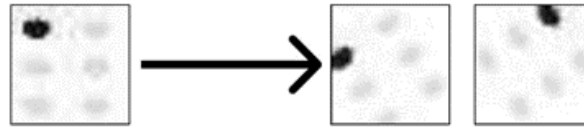
In our project, we will use the "Braille Character Dataset" obtained from Kaggle as the basis for training our model. For efficiency, we will preprocess the images to improve their quality and accuracy to be used as input in the model. The implementation of the model involves classifying the entire Braille image to determine the necessary structure of six connecting points in the Braille alphabet. After that, our model will perform the conversion process, and finally, we will have an audio of the Braille word or phrase given as input. This approach underscores our commitment to increasing accessibility and connectivity for visually impaired individuals.

4.1 SELECTED METHODOLOGY OR PROCESS MODEL

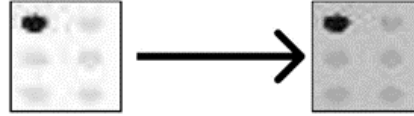
In the research, the Braille Character Dataset is used, which contains 1560 images, to train the model which will do the recognition process of the braille character. Keras, a deep learning library which runs on top of Tensorflow, is used. Tensorflow is one of the most used and user-friendly Deep Learning libraries. It is an open-source library used for creating the convolutional neural network. Keras library is used to perform several operations such as generating image data(using image augmentation), splitting the dataset into training and validation data of 80% (1248 images) and 20% (312 images), respectively, fitting the training data and also to evaluate the model's accuracy.

The images of the training dataset are augmented by applying image augmentation techniques (such as rotating, brightness, width-height shifting etc), thus increasing the dataset further and significantly increasing the diversity of the images available for training the model thus generalizing the model.

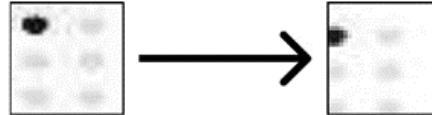
After Image Augmentation, the training dataset is splitted into the training(80%) and the validation(20%) data. The model architecture is defined and compiled using the Adam optimizer and categorical cross-entropy loss function for multi-class classification tasks.



(a) Rotation



(b) Brightness



(c) Width-Height Shifting

Fig 4.1 Image Augmentation

Table 4.1 Layers in BT-CNN

Layer (type)	Output Shape	Parameter
Input	28, 28, 3	0
Separable Convolution	26, 26, 64	283
Max Pooling	13, 13, 64	0
Separable Convolution	11, 11, 128	8896
Max Pooling	5, 5, 128	0
Separable Convolution	4, 4, 256	33536
Global Max Pooling	256	0
Dense	256	65792
LeakyReLU	256	0
Dense	64	16448
LeakyReLU	64	0
Dense	26	1690
Output	26	0

Training and validation data generators were used to train the model across 100 epochs. The training process is monitored and optimised by the use of callbacks, which include model checkpointing, learning rate decrease, and early termination.

Epoch, in machine learning, is referred to one-time passing of the entire training data through the model or algorithm. After a certain number of epochs, if there isn't significant increase in the model's accuracy, the early-stopping callback method executes to stop the training process. The best version is saved. Then, the accuracy of the model is evaluated using a validation data generator. The achieved model accuracy, computed using the evaluation generator, was printed with four decimal places for precision. The `evaluate_generator` function is used to return a list of metric values, and extract the accuracy value.

The `model.predict` method is used to generate predictions for the input data provided by `val_generator`. This typically occurs after the training phase when the model has learned to extract relevant features from the training data. In the proposed work, `plt.imshow(img)` uses Matplotlib to plot the image represented by the `img` variable. The `image.img_to_array(img)` function is used to convert the image into a NumPy array. Then, the dimensions of an array are found using the `shape` function and this is the shape of the array - (28, 28, 3). Preprocessing of input character is done by expanding its dimensions and then use it to make predictions using the proposed model. The predicted output(desired English Alphabet) was obtained by finding the index of the highest probability prediction.

As the final step in this system, the model is designed to generate an audio file that aligns with the Braille word or sentence first of all supplied as an image. In order to streamline this process, Google Text to Speech is used. Once trained, the network can synthesize speech for new input text by:

- Text pre-processing: The input text is analyzed and segmented into phonemes (basic units of sound).
- Feature prediction: The network predicts the acoustic features corresponding to each phoneme based on the surrounding context.
- Waveform generation: The predicted acoustic features are used to generate a speech waveform that resembles the target speaker's voice.

4.2 ARCHITECTURE / OVERALL DESIGN OF PROPOSED SYSTEM

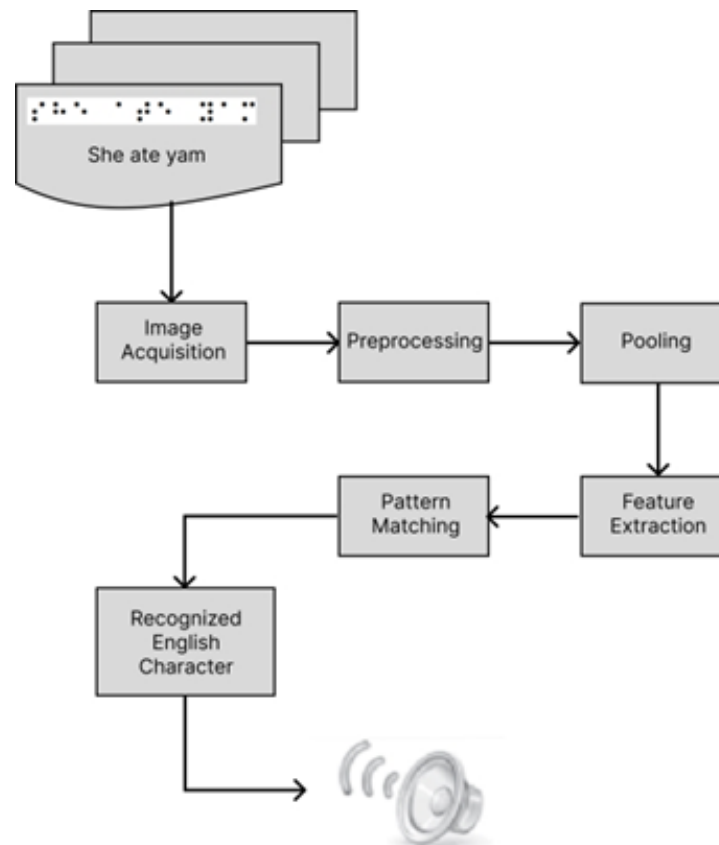


Fig 4.2 Architecture Diagram

Figure 4.2 presents a visible depiction of the proposed version, illustrating its structure and functionality within the system of changing Braille characters into an audio file. This graphical illustration serves as a schematic evaluation, capturing the important components and connections concerned in the version's operation. The discern conveys a clean visual interpretation of the way the proposed structure is hired to facilitate the transformation of Braille characters, highlighting the sequential steps and interactions that contribute to the remaining output—an audio report. This visible illustration aids in comprehending the model's design and its position in the revolutionary approach to Braille character conversion, imparting a visible reference for readers to better hold close the method hired inside the study.

The first and essential step in the digital representation of visual data is image acquisition. It involves special devices such as cameras, scanners and sensors to capture visible information from the physical world. These devices convert analog information such as light and electromagnetic radiation into digital images. The

quality of these acquired images is extremely important as it directly affects the success of image production and subsequent analysis. Factors such as lighting, camera settings, and sensor sensitivity play an important role in ensuring the accuracy and realism of digital images. Image acquisition is widely used in a variety of industries, including photography, medical imaging, surveillance, and computer vision. It serves as the basis for such vital processes as medical diagnosis, discovery, and independent guidance. By converting real-world visual information into a digital format, image acquisition enables computers and digital systems to make sense of and interact with the physical world, enabling product delivery of many practical and new uses.

In Image Processing, a comprehensive image records preprocessing strategy is implemented with the usage of two key strains. TensorFlow's Keras library imports the image module, which contains functions for loading and preprocessing images. This ensures finest formatting for superior version performance. Concurrently, the ImageDataGenerator module line employs the ImageDataGenerator module for dynamic dataset augmentation all through schooling, enhancing the model's adaptability and generalization throughout numerous scenarios.

To make the proposed version even more reliable, the plan is putting preprocessing strategies into play, refining the process of entering pictures before they undergo analysis. The operational method of proposed version entails a meticulous dissection of the complete photograph into smaller fragments, accomplished by way of figuring out the feature six dots that compose a Braille character.

The pooling process is implemented using the L.MaxPooling2D layers. Specifically, the lines $x = \text{L.MaxPooling2D}((2,2))(x)$ after the separable convolution layers downsample the spatial dimensions of the input feature maps through max-pooling. A downsampled (pooled) feature map is produced by the pooling process known as "max pooling," which determines the maximum value for each patch in a feature map.

In the proposed work, the feature extraction process occurs in the layers of the neural network model, specifically within the SeparableConv2D and GlobalMaxPooling2D layers. These layers are tasked with capturing essential

features from the input images. The `SeparableConv2D` layers perform convolution operations to capture spatial hierarchies of features, while the `GlobalMaxPooling2D` layer aggregates the most important features by taking the maximum value across spatial dimensions. Therefore, the lines involving `SeparableConv2D` and `GlobalMaxPooling2D` contribute to the feature extraction process in this model. The extraction method happens when the use of the `model.predict(x)` line. This line sends the enter records `x` through the skilled version, and the output represents the extracted capabilities or predictions primarily based at the discovered patterns from the input. The `np.expand_dims(x, axis=zero)` line is used to adjust the shape of `x` by adding a further dimension, which is a common preprocessing step, but it does not perform the characteristic extraction itself.

CHAPTER 5

IMPLEMENTATION DETAILS

In the method employed, the intention is to capitalize on the resources available inside the "Braille Character Dataset" from Kaggle to train our version effectively. Each image is a 28x28 image in BW Scale. Each image name consists of the character alphabet and the number of the image and the type of data augmentation it went through. (i.e WHS - width height shift, rot - Rotation, dim - brightness). Dataset composition: 26 characters * 3 Augmentations * 20 different images of different augmentation values (i.e different shift, rotational and brightness values).

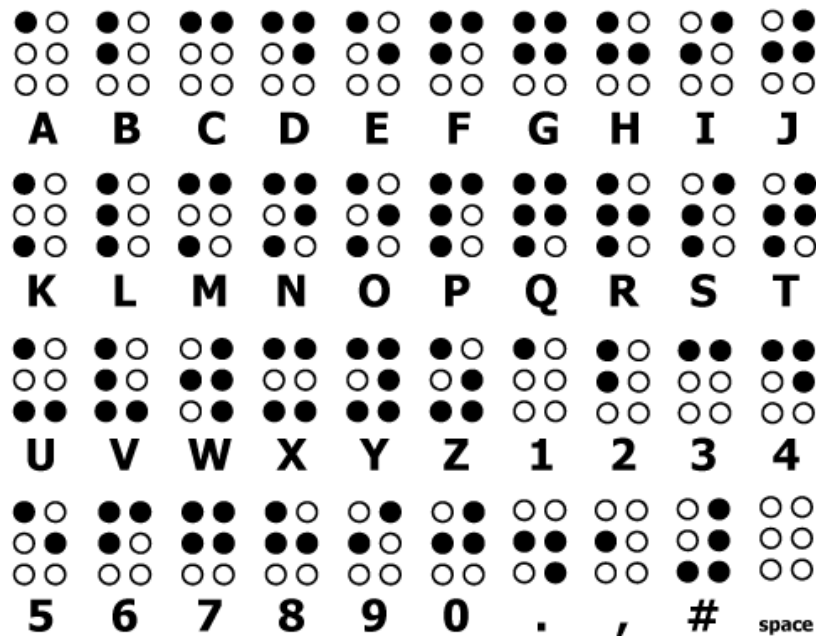


Fig 5.1 Braille Characters

To optimize the performance of the proposed version, the plan is to put in force preprocessing strategies, refining the process of entering pictures before they undergo analysis. The operational method of the proposed system entails a meticulous dissection of the complete photograph into smaller fragments, accomplished by way of figuring out the feature six dots that compose a Braille character.

5.1 DEVELOPMENT AND DEPLOYMENT SETUP

To develop and deploy a braille to speech conversion system using Convolutional Neural Network (CNN), you can follow the steps below:

1) Image Acquisition

- Involves special devices such as cameras, scanners, etc to capture visible information.
- These devices convert analog information such as light and electromagnetic radiation into digital images.
- The quality of these acquired images is extremely important as it directly affects the success of subsequent analysis.

2) Image Preprocessing

- Images of the training dataset are augmented by applying image augmentation techniques (such as rotating, brightness, width-height shifting etc)
- This is done to enhance the model's adaptability and generalization throughout numerous scenarios

3) Pooling

- The pooling process is implemented using the max-pooling layers.
- A downsampled(pooled) feature map is produced by the pooling process known as "max pooling," which determines the maximum value for each patch in a feature map.

4) Feature Extraction

- The Separable Convolution and Max-Pooling layers are responsible for extracting relevant features from the input images.

5) Pattern Matching

- After extracting the relevant features from the input images, the proposed model matches those features with features it has extracted from the images taken from dataset.

6) Recognize English Character

- After the pattern matching process, the braille character is recognized to its equivalent English character.
- These characters eventually form words and sentences.

7) Speech Synthesis

- By using Google Text to Speech, the words and sentences are then converted into an Audio speech file.
- Hence, the input image is converted into its equivalent Audio file.

5.2 ALGORITHM

5.2.1 Convolutional Neural Network (CNN):

Convolutional Neural Network (CNN) is like an intelligent computer system that is really good at understanding images and videos. It can see and recognize different things in images, similar to how humans see and understand things. CNNs are used for many applications such as recognizing objects in images, reading medical images, and even understanding words in text. They're like the super detectives of the computer world for visuals.

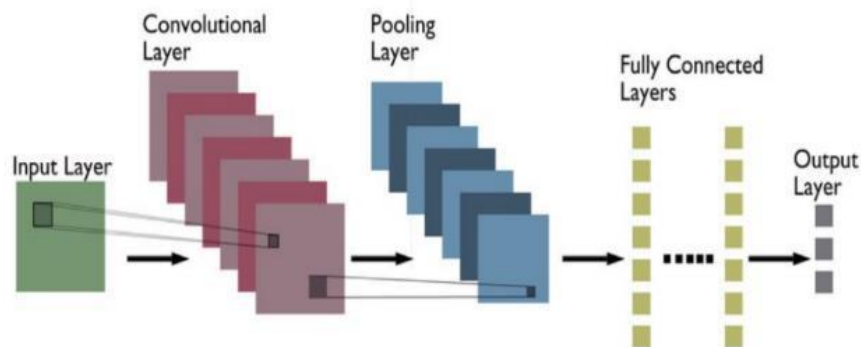


Fig 5.2 Convolutional Neural Network

At its core, a CNN comprises multiple layers, each with a specific function in the feature extraction process. The primary layers include convolutional layers, pooling layers, and fully connected layers. In convolutional layers, the network learns to detect various features by convolving input images with a set of learnable filters or kernels. These filters are small matrices that slide over the input image, capturing local patterns such as edges, textures, and shapes.

Overall, CNNs have revolutionized the field of computer vision, enabling significant advancements in tasks such as image recognition, object detection, and image generation. Their ability to automatically learn hierarchical representations from raw pixel data makes them indispensable tools in various domains, including healthcare, autonomous vehicles, and surveillance.

CHAPTER 6

RESULTS AND DISCUSSIONS

6.1 PRESENTATION OF FINDINGS

We have used Braille Character Dataset which contains 1560 images of Braille Characters. We have used CNN for Training and Testing our custom model named BT-CNN. We splitted the data into Training and Testing sets in the 80:20 ratio, which means 80% of the images used as Training set and the rest is Testing set. The highest accuracy achieved by the proposed model is 96.30%.

6.2 DATA ANALYSIS AND INTERPRETATION

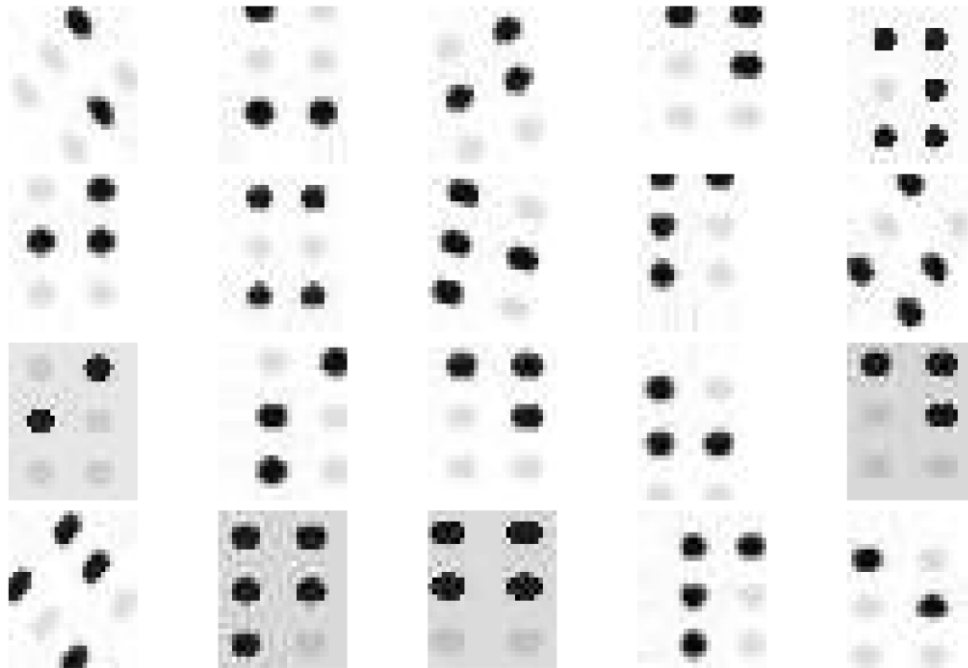


Fig 6.1 Dataset Used

We have used Braille Character Dataset, which contains 1560 images of Braille Characters.

Model: "model"

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 28, 28, 3)]	0
separable_conv2d (SeparableConv2D)	(None, 26, 26, 64)	283
max_pooling2d (MaxPooling2D)	(None, 13, 13, 64)	0
separable_conv2d_1 (SeparableConv2D)	(None, 11, 11, 128)	8896
max_pooling2d_1 (MaxPooling2D)	(None, 5, 5, 128)	0
separable_conv2d_2 (SeparableConv2D)	(None, 4, 4, 256)	33536
global_max_pooling2d (GlobalMaxPooling2D)	(None, 256)	0
dense (Dense)	(None, 256)	65792
leaky_re_lu (LeakyReLU)	(None, 256)	0
dense_1 (Dense)	(None, 64)	16448
leaky_re_lu_1 (LeakyReLU)	(None, 64)	0
dense_2 (Dense)	(None, 26)	1690

=====
Total params: 126645 (494.71 KB)
Trainable params: 126645 (494.71 KB)
Non-trainable params: 0 (0.00 Byte)

Fig 6.2 Creation of BT-CNN Model

Here we have our BT-CNN Model's Accuracy and Loss Curves from our Training and Testing steps.

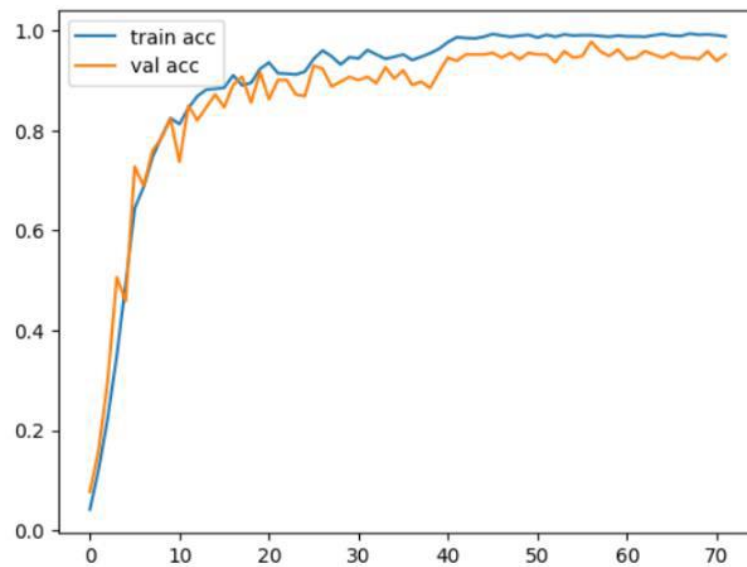


Fig 6.3 Graph of accuracy at each epoch

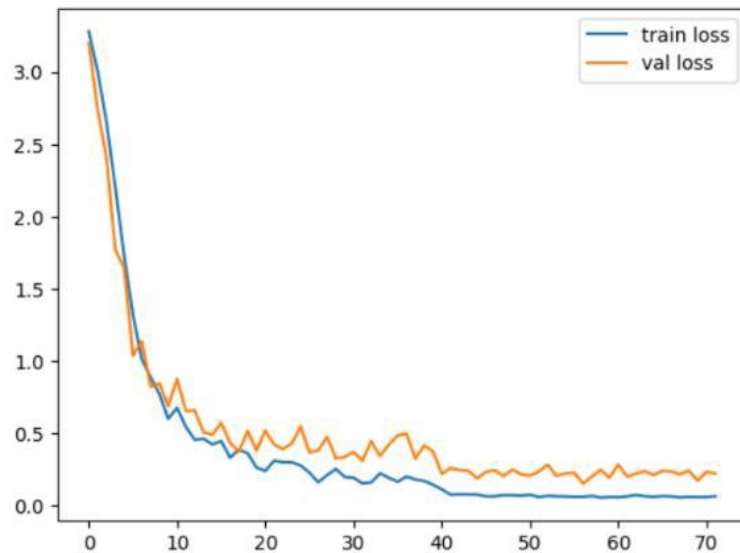


Fig 6.4 Graph of loss function at each epoch

```

model = load_model('BrailleModel.h5')
acc = model.evaluate(val_generator)[1]
print('model accuracy: {}'.format(round(acc,4)))

10/10 [=====] - 1s 36ms/step - loss: 0.2162 - accuracy: 0.9630
model accuracy: 0.9630

```

Fig 6.5 Accuracy of BT-CNN Model

We achieved an accuracy of 96.30% for the BT-CNN Model after Training and Testing.

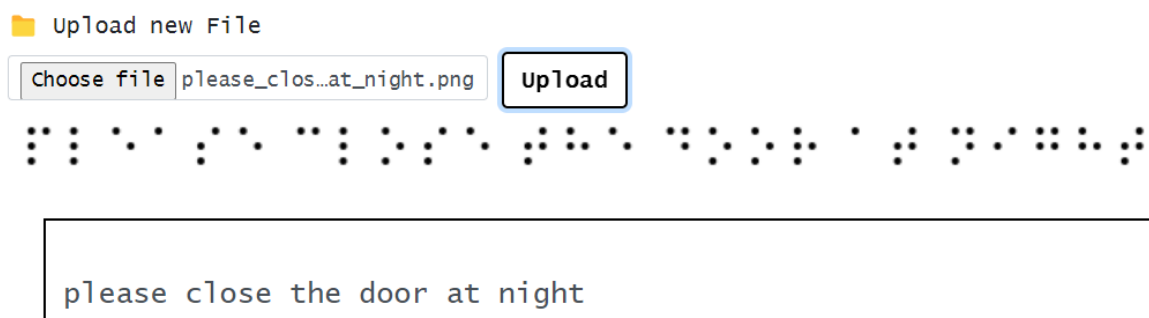


Fig 6.6 Picture showing the Braille text conversion for an image

The above picture shows the conversion of a Braille image into English language, which means “Please close the door at night”.



Fig 6.7 Picture showing the Braille text conversion for another image

The above picture shows the conversion of a Braille image into English language, which means “Please close the door at night”.

Table 6.1 Comparative Analysis of Existing Methods vs Proposed Method

SI. No.	Researcher	Algorithms	Accuracy(%)
1	K. Smelyakov (2018)	ANN	95.00
2	S. Shokat (2020)	GoogleNet	95.80
3	J Mao (2021)	Prosody Prediction	94.42
4	T Kausar (2021)	DenseNet201	95.20
5	M. A. Hussain (2022)	VGG16	92.42
6	Proposed Work	BT-CNN	96.30

A comparison between the proposed system and recent methods is shown in Table 6.1. Our system evaluated the overall performance with existing approaches, however it is challenging to compare them because they were trained on different datasets. J Mao and M. A. Hussain's approaches convert Braille text to only Chinese and Bangla languages, respectively, which caters to a relatively small set of people as compared to the Internationally accepted English language. S. Shokat's study does not mention incorporating feedback from visually impaired individuals during the development and evaluation of the system. Involvement of end-users in the design process could provide valuable insights into their needs, preferences, and usability challenges, ultimately leading to a more user-centric and effective solution. While T. Kausar's preprocessing stage involves the application of Principal Component Analysis (PCA) to enhance feature representation, they also add complexity to the system and may require additional computational resources.

CHAPTER 7

CONCLUSION

7.1 CONCLUSION

In this study, efforts were made to use Deep learning for braille to speech conversion and got accuracy of 96.30%. This system can be proved as a major breakthrough in the field of automation and innovative technologies and can be used in various websites and places with diverse set of people and can further be incorporated with systems which can help all the visually challenged people as well as normal people around the world to understand and visualize their respective surroundings and communicate with people in a simpler and efficient manner.

7.2 FUTURE WORK

This study can further be enhanced in the future using different and more accurate, high-resolution datasets or using different, more complex methods like Extreme Machine Learning (EML), or Classification and Regression Trees (CART) for better performance of the model which may require higher computation power and much advanced softwares for efficient working.

REFERENCES

- [1] A. A. Choudhury, R. Saha, S. Z. Hasan Shoumo, S. Rafsun Tulon, J. Uddin and M. K. Rahman, "An Efficient Way to Represent Braille using YOLO Algorithm," 2018 Joint 7th International Conference on Informatics, Electronics & Vision (ICIEV) and 2018 2nd International Conference on Imaging, Vision & Pattern Recognition (icIVPR), Kitakyushu, Japan, 2018, pp. 379-383.
- [2] B.-M. Hsu, "Braille recognition for reducing asymmetric communication between the blind and non-blind," *Symmetry*, vol. 12, no. 7, p. 1069, Jun. 2020.
- [3] Bhatia, Surbhi & Devi, Ajantha & Alsuwailem, Razan & Mashat, Arwa, Convolutional Neural Network Based Real Time Arabic Speech Recognition to Arabic Braille for Hearing and Visually Impaired, *Frontiers in Public Health*, 2020.
- [4] Chollet, "Xception: Deep Learning with Depthwise Separable Convolutions," *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1800-1807, Jul. 2017.
- [5] C. Park, S. W. Cho, N. R. Baek, J. Choi and K. R. Park, "Deep Feature-Based Three-Stage Detection of Banknotes and Coins for Assisting Visually Impaired People," in *IEEE Access*, vol. 8, pp. 184598-184613, 2020.
- [6] G. C. Bettelani, G. Averta, M. G. Catalano, B. Leporini and M. Bianchi, "Design and Validation of the Readable Device: A Single-Cell Electromagnetic Refreshable Braille Display," in *IEEE Transactions on Haptics*, vol. 13, no. 1, pp. 239-245, 1 Jan.-March 2020.
- [7] J. Mao, J. Zhu, X. Wang, H. Liu, and Y. Qian, "Speech synthesis of Chinese Braille with limited training data," in *Proc. IEEE Int. Conf. Multimedia Expo (ICME)*, Jul. 2021, pp. 1–6.
- [8] K. Smelyakov, A. Chupryna, D. Yeremenko, A. Sakhon and V. Polezhai, "Braille Character Recognition Based on Neural Networks," 2018 IEEE Second

International Conference on Data Stream Mining & Processing (DSMP), Lviv, Ukraine, 2018, pp. 509-513.

[9] K. M. Naimul Hassan, S. K. Biswas, M. S. Anwar, M. S. Iman Siam and C. Shahnaz, "A Dual-Purpose Refreshable Braille Display Based on Real Time Object Detection and Optical Character Recognition," 2019 IEEE International Conference on Signal Processing, Information, Communication & Systems (SPICSCON), Dhaka, Bangladesh, 2019, pp. 78-81.

[10] M. Hanumanthappa and V. V. Murthy, "Optical Braille recognition and its correspond-ence in the conversion of Braille script to text — A literature review," 2016 International Conference on Computation System and Information Technology for Sustainable Solutions (CSITSS), Bengaluru, India, 2016, pp. 297-301.

[11] M. A. Khan, P. Paul, M. Rashid, M. Hossain and M. A. R. Ahad, "An AI-Based Visual Aid With Integrated Reading Assistant for the Completely Blind," in IEEE Transactions on Human-Machine Systems, vol. 50, no. 6, pp. 507-517, Dec. 2020.

[12] M. A. Hussain, R. I. Rifat, S. B. Iqbal, S. Biswas, M. G. Rabiul Alam and M. T. Reza, "Deep Learning based Bangla Voice to Braille Character Conversion System," 2022 IEEE 13th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON), Vancouver, BC, Canada, 2022, pp. 0262-0267.

[13] P. Kaur, S. Ramu, S. Panchakshari, and N. Krupa, "Conversion of Hindi Braille to speech using image and speech processing," in Proc. IEEE 7th Uttar Pradesh Section Int. Conf. Elect., Electron. Comput. Eng. (UPCON), Nov. 2020, pp. 1–6.

[14] Ramani RG, Shanthamalar JJ. Automated image quality appraisal through partial least squares discriminant analysis. Int J Comput Assist Radiol Surg. 2022 July;1367-1377.

- [15] S. M. Qaisar, R. Khan and N. Hammad, "Scene to Text Conversion and Pronunciation for Visually Impaired People," 2019 Advances in Science and Engineering Technology International Conferences (ASET), Dubai, United Arab Emirates, 2019, pp. 1-4.
- [16] S. Shokat, R. Riaz, S. S. Rizvi, A. M. Abbasi, A. A. Abbasi, and S. J. Kwon, "Deep learning scheme for character prediction with position-free touch screen-based Braille input method," Hum.-centric Coput. Inf. Sci., vol. 10, no. 1, pp. 1–24, Dec. 2020.
- [17] S. Ramachandran, N. Rajan, K. N. Pallavi, J. Subashree, S. Suchithra and B. Sonal, "Communication Device for the Visual and Hearing Impaired Persons to Convert Braille Characters to English Text," 2021 International Conference on Emerging Smart Computing and Informatics (ESCI), Pune, India, 2021, pp. 587-592.
- [18] Seung-Bin Park and Bonghyun Kim. "Design of CNN-based Braille Conversion and Voice Output Device for the Blind" Journal of Internet of Things and Convergence 9, no.3 (2023) : 87-92.
- [19] T. Kausar, S. Manzoor, A. Kausar, Y. Lu, M. Wasif and M. A. Ashraf, "Deep Learning Strategy for Braille Character Recognition," in IEEE Access, vol. 9, pp. 169357-169371, 2021.
- [20] V Jha and K Parvathi, "Braille Transliteration of hindi handwritten texts using machine learning for character recognition", Int J Sci Technol Res, vol. 8.10, pp. 1188-1193, 2019.

APPENDIX

A. SOURCE CODE

1. Braille Model

```
import os
import numpy as np
import pandas as pd
from shutil import copyfile
from google.colab import drive
drive.mount('/content/drive')
os.mkdir('./images/')
alpha = 'a'
for i in range(0, 26):
    os.mkdir('./images/' + alpha)
    alpha = chr(ord(alpha) + 1)
rootdir = '/content/drive/MyDrive/Braille Dataset/'
for file in os.listdir(rootdir):
    letter = file[0]
    copyfile(rootdir+file, './images/' + letter + '/' + file)
from keras.preprocessing.image import ImageDataGenerator
datagen = ImageDataGenerator(rotation_range=20,
    shear_range=10,
    validation_split=0.2)
train_generator = datagen.flow_from_directory('./images/',
    target_size=(28,28),
    subset='training')
val_generator = datagen.flow_from_directory('./images/',
    target_size=(28,28),
    subset='validation')
from keras import backend as K
from keras import layers as L
from keras.models import Model,load_model
from keras.regularizers import l2
```

```

from keras.callbacks import ModelCheckpoint, ReduceLROnPlateau, EarlyStopping
K.clear_session()
model_ckpt = ModelCheckpoint('BrailleModel.h5', save_best_only=True)
reduce_lr = ReduceLROnPlateau(patience=8, verbose=0)
early_stop = EarlyStopping(patience=15, verbose=1)
entry = L.Input(shape=(28,28,3))
x = L.SeparableConv2D(64,(3,3),activation='relu')(entry)
x = L.MaxPooling2D((2,2))(x)
x = L.SeparableConv2D(128,(3,3),activation='relu')(x)
x = L.MaxPooling2D((2,2))(x)
x = L.SeparableConv2D(256,(2,2),activation='relu')(x)
x = L.GlobalMaxPooling2D()(x)
x = L.Dense(256)(x)
x = L.LeakyReLU()(x)
x = L.Dense(64, kernel_regularizer=l2(2e-4))(x)
x = L.LeakyReLU()(x)
x = L.Dense(26, activation='softmax')(x)
model = Model(entry,x)
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
history = model.fit_generator(train_generator,
validation_data=val_generator,
epochs=100,
callbacks=[model_ckpt, reduce_lr, early_stop],
verbose=1)
model.summary()
model = load_model('BrailleModel.h5')
acc = model.evaluate(val_generator)[1]
print('model accuracy: {}'.format(round(acc,4)))
import matplotlib.pyplot as plt
plt.plot(history.history['loss'], label='train loss')
plt.plot(history.history['val_loss'], label='val loss')
plt.legend()
plt.show()

```

```

plt.savefig('LossVal_loss')
# plot the accuracy
plt.plot(history.history['accuracy'], label='train acc')
plt.plot(history.history['val_accuracy'], label='val acc')
plt.legend()
plt.show()
plt.savefig('AccVal_acc')
pred = model.predict(val_generator)
pred
from tensorflow.keras.preprocessing import image
img=image.load_img('/content/drive/MyDrive/Braille Dataset/g1.JPG5whs.jpg')
plt.imshow(img)
x=image.img_to_array(img)
x
x.shape
x=np.expand_dims(x,axis=0)
model.predict(x)
a=np.argmax(model.predict(x), axis=1)
if a == [0]:
print("A")
if a == [1]:
print("B")
if a == [2]:
print("C")
if a == [3]:
print("D")
if a == [4]:
print("E")
if a == [5]:
print("F")
if a == [6]:
print("G")
if a == [7]:
print("H")

```

```
if a == [8]:
    print("I")
if a == [9]:
    print("J")
if a == [10]:
    print("K")
if a == [11]:
    print("L")
if a == [12]:
    print("M")
if a == [13]:
    print("N")
if a == [14]:
    print("O")
if a == [15]:
    print("P")
if a == [16]:
    print("Q")
if a == [17]:
    print("R")
if a == [18]:
    print("S")
if a == [19]:
    print("T")
if a == [20]:
    print("U")
if a == [21]:
    print("V")
if a == [22]:
    print("W")
if a == [23]:
    print("X")
if a == [24]:
    print("Y")
```

```
if a == [25]:  
    print("Z")
```

2. Flask Application

main.py

```
from flask import Flask, render_template, request, jsonify  
from PredictBraille import PredictBraille  
from sys import argv  
import pytesseract  
from PIL import Image  
from flask_cors import CORS  
app = Flask(__name__, template_folder='templates')  
app.secret_key = 'this is a very secure string'  
CORS(app)  
@app.route('/')  
def home():  
    return render_template('home.html')  
@app.route('/brailleImageTranslate', methods=['POST'])  
def brailleImageTranslate():  
    if 'image' in request.files:  
        uploaded_image = request.files['image']  
        image_path = 'temp_image.png' # Save the uploaded image to a temporary file  
        uploaded_image.save(image_path)  
        # Perform OCR on the uploaded image  
        predictor = PredictBraille()  
        result = predictor.predict(image_path)  
        return jsonify({'result': result})  
    else:  
        return jsonify({'error': 'No image file provided.'})  
if __name__ == '__main__':  
    app.run(debug=True)
```

PredictBraille.py

```
import cv2
import numpy as np
from keras.models import load_model
from OBR import BrailleImage, SegmentationEngine
import string

class PredictBraille:
    def __init__(self):
        self.prediction = ""
    def predict(self, input_image_path):
        print("Lets start")
        alphabet = list(string.ascii_lowercase)
        cur_pos = 0
        target = {}
        for letter in alphabet:
            target[letter] = [0] * 27
            target[letter][cur_pos] = 1
            cur_pos += 1
        target[' '] = [0] * 27
        target[' ')[26] = 1
        print("Load model")
        # Load the Braille model
        model = load_model('BrailleModel.h5')
        # Load the input image
        input_image = cv2.imread(input_image_path, cv2.IMREAD_GRAYSCALE)
        _, binary_image = cv2.threshold(input_image, 128, 255, cv2.THRESH_BINARY)
        braille_image = BrailleImage(input_image_path)
        # Create a SegmentationEngine instance
        segmentation_engine = SegmentationEngine(braille_image)
        prev_word=0
        prev_line=0
        # print("Predict")
        # Loop through segmented characters and predict their classes
```

```

for segmented_character in segmentation_engine:
    character_image = segmented_character.get_character_image()
    box = segmented_character.get_bounding_box()
    left,right,top,bottom = box
    width = int(right - left)
    height =int(bottom-top)
    if((left-prev_word)>width):
        self.prediction=self.prediction+" "
    prev_word=right
    if((top-prev_line)>height/4):
        self.prediction=self.prediction+"\n"
    prev_line=bottom
    # Preprocess the character image before passing it to the model
    resized_img = cv2.resize(character_image, (28, 28))
    rgb_character_image = resized_img.reshape(-1, 28, 28, 3)
    pred_img = rgb_character_image.astype(np.float32) / 255.0
    # Perform classification using the loaded model
    pred_lb = model.predict(pred_img)
    pred = ""
    for j in range(len(pred_lb[0])):
        if pred_lb[0][j] > 0.6:
            pred_lb[0][j] = 1.0
        else:
            pred_lb[0][j] = 0.0
    for key, value in target.items():
        if np.array_equal(np.asarray(pred_lb[0]), np.asarray(value)):
            pred = key
    if pred == "":
        pred = "!"
    self.prediction = self.prediction + pred
    # Mark the character's bounding box on the parent image
    segmented_character.mark()
    # print(self.prediction)
    # Get the final image with marked bounding boxes

```

```

final_image = braille_image.get_final_image()
# cv2.imshow("image", final_image)
cv2.waitKey()
cv2.destroyAllWindows()
return self.prediction

```

home.html

```

<!DOCTYPE html>
<html>
<head>
<title>Braille to Speech</title>
<link
rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.7.2/font/bootstrap-icons.css"
/>
<link
rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-
awesome.min.css"
/>
<link
rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@4.3.1/dist/css/bootstrap.min.css"
integrity="sha384-
ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1
T"
crossorigin="anonymous"
/>
<link rel="stylesheet" href="/static/css1/text2text.css" />
</head>
<body>
<div class="container">
<h2 class="mt-4 text-center">

```

```

<strong>Braille to Speech using CNN</strong>
</h2>
<br />
<form method="post" enctype="multipart/form-data">
<div class="custom-file" style="width: 30% !important">
<label for="formFile" class="form-label"> 📁 Upload new File</label>
<input
type="file"
id="imageInput"
class="form-control form-control-sm"
name="image"
accept="image/*"
required
/>
</div>
<button
type="submit"
class="btn"
id="upload"
style="
color: black;
font-weight: 600;
border: 2px solid black;
-webkit-backface-visibility: hidden;
backface-visibility: hidden; "
>
Upload
</button>
</form>
<div id="img-container"></div>
<br />
<div class="container">
<div style="display: flex">
<textarea

```

```

class="form-control"
id="braille-output-text"
rows="8"
readonly
placeholder="The Braille output will appear in this place"
style="
font-size: 20px;
height: 400px;
background-color: white;
border: 2px solid black;
border-radius: 0px;
margin-left: 10px;
"
></textarea>
</div>
<br />
<div class="d-flex justify-content-between">
<button
class="btn"
onclick="speak_Text()"
style="
color: black;
font-weight: 600;
border: 2px solid black;
-webkit-backface-visibility: hidden;
backface-visibility: hidden;
>
<svg
xmlns="http://www.w3.org/2000/svg"
width="16"
height="16"
fill="currentColor"
class="bi bi-music-player"
viewBox="0 0 16 16"

```

```

>
<path
d="M4 3a1 1 0 0 1 1-1h6a1 1 0 0 1 1 1v3a1 1 0 0 1-1 1H5a1 1 0 0 1-1-1V3zm1
0v3h6V3H5zm3 9a1 1 0 1 0 0-2 1 1 0 0 0 0 2z"
/>
<path
d="M11 11a3 3 0 1 1-6 0 3 3 0 0 1 6 0zm-3 2a2 2 0 1 0 0-4 2 2 0 0 0 0 4z"
/>
<path
d="M2 2a2 2 0 0 1 2-2h8a2 2 0 0 1 2 2v12a2 2 0 0 1-2 2H4a2 2 0 0 1-2-2V2zm2-
1a1 1 0 0 0-1 1v12a1 1 0 0 0 1 1h8a1 1 0 0 0 1-1V2a1 1 0 0 0-1-1H4z"
/>
</svg>

```

Click To Play Audio

```

</button>
<button
class="btn text-center"
onclick="clearTextFields()"
style="
color: black;
font-weight: 600;
border: 2px solid black;
-webkit-backface-visibility: hidden;
backface-visibility: hidden;"
>

```

Clear Text

```

</button>
</div>
<br />
<br />
</div>
</div>
<script type="text/javascript">
const imgContainer = document.getElementById("img-container");

```

```

function speak_Text() {
  if ("speechSynthesis" in window) {
    const utterance = new SpeechSynthesisUtterance();
    utterance.text = output.value;
    utterance.voice = speechSynthesis.getVoices()[0];
    speechSynthesis.speak(utterance);
  } else {
    console.log("Speech synthesis not supported in this browser.");
  }
}

function clearTextFields() {
  document.getElementById("braille-output-text").value = "";
  if (imgContainer) {
    imgContainer.innerHTML = "";
  }
}

const output = document.getElementById("braille-output-text");
const upload = document.getElementById("upload");
upload.addEventListener("click", function (event) {
  event.preventDefault();
  // Check if a file has been selected
  if (!imageInput.files || imageInput.files.length === 0) {
    alert("Please select an image before uploading.");
    return;
  }
  clearTextFields();
  output.value = "Translating....";
  const formData = new FormData();
  formData.append("image", imageInput.files[0]);
  // Fetch the image result from the Flask server
  fetch("/brailleImageTranslate", {
    method: "POST",
    body: formData, // Use FormData to send the form data
  })

```

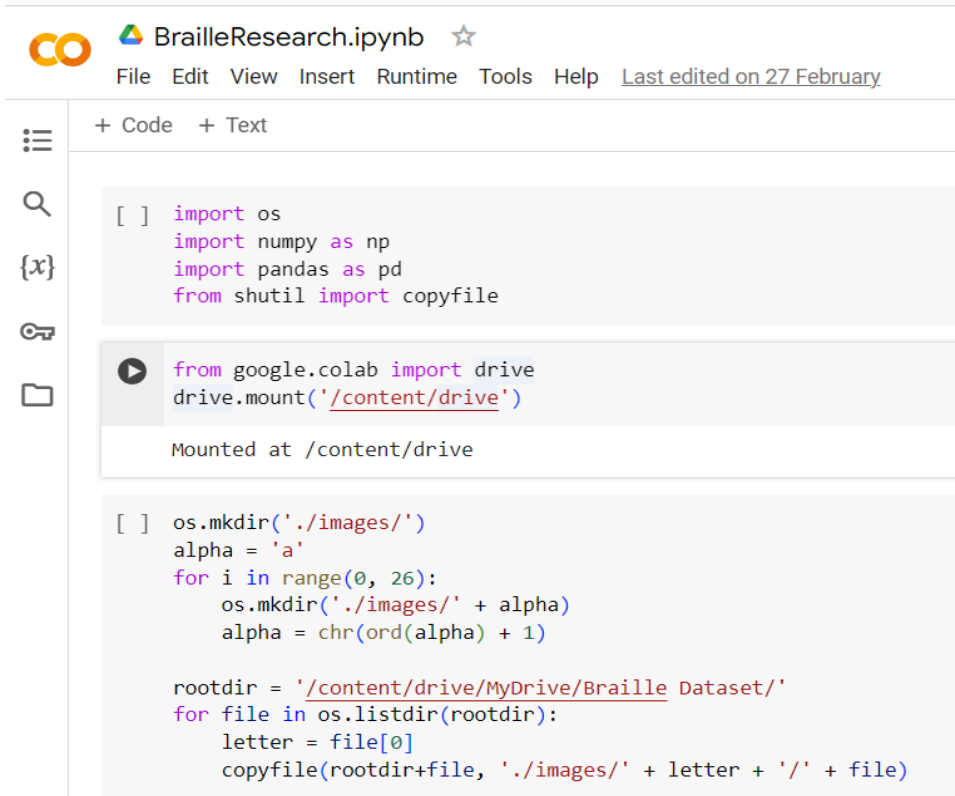
```

})
.then((response) => response.json()) // Parse the response as JSON
.then((data) => {
  // Display the image result in the image_result textarea
  const image = URL.createObjectURL(imageInput.files[0]);
  // Display the image in an <img> tag within the imgContainer
  const imgContainer = document.getElementById("img-container");
  const newimg = document.createElement("img");
  newimg.src = image;
  newimg.style.width = "800px";
  newimg.style.height = "auto";
  imgContainer.appendChild(newimg);
  output.value = data.result || data.error;
})
.catch((error) => {
  console.error("Error:", error);
});
});
</script>
<script
src="https://code.jquery.com/jquery-3.3.1.slim.min.js"
integrity="sha384-
q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo"
crossorigin="anonymous"
></script>
<script
src="https://cdn.jsdelivr.net/npm/popper.js@1.14.7/dist/umd/popper.min.js"
integrity="sha384-
UO2eT0CpHqdSjQ6hJty5KVphtPhzWj9WO1clHTMGa3JDZwrnQq4sF86dIHNDz
0W1"
crossorigin="anonymous"
></script>
<script
src="https://cdn.jsdelivr.net/npm/bootstrap@4.3.1/dist/js/bootstrap.min.js"

```

```
integrity="sha384-  
JjSmVgyd0p3pXB1rRibZUAYoIIy6OrQ6VrjlEaFf/nJGzlxFDsf4x0xIM+B07jRM"  
crossorigin="anonymous"  
></script>  
</body>  
</html>
```

B. SCREENSHOTS



```
[ ] import os
import numpy as np
import pandas as pd
from shutil import copyfile

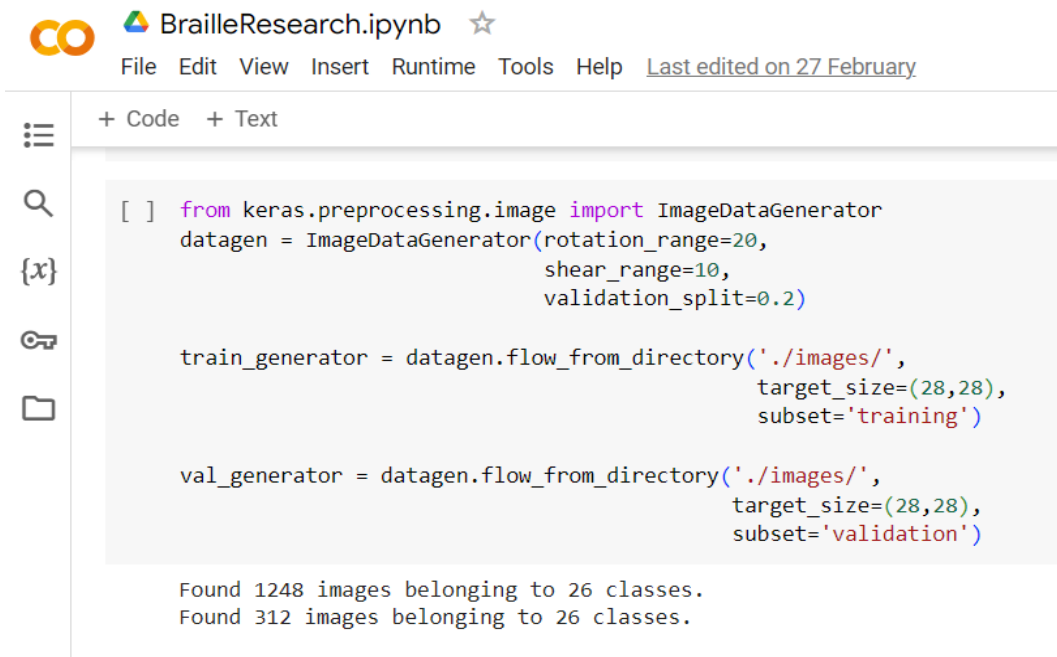
from google.colab import drive
drive.mount('/content/drive')

Mounted at /content/drive

[ ] os.mkdir('./images/')
alpha = 'a'
for i in range(0, 26):
    os.mkdir('./images/' + alpha)
    alpha = chr(ord(alpha) + 1)

rootdir = '/content/drive/MyDrive/Braille Dataset/'
for file in os.listdir(rootdir):
    letter = file[0]
    copyfile(rootdir+file, './images/' + letter + '/' + file)
```

The above image shows the imported libraries and basic setup of the project.



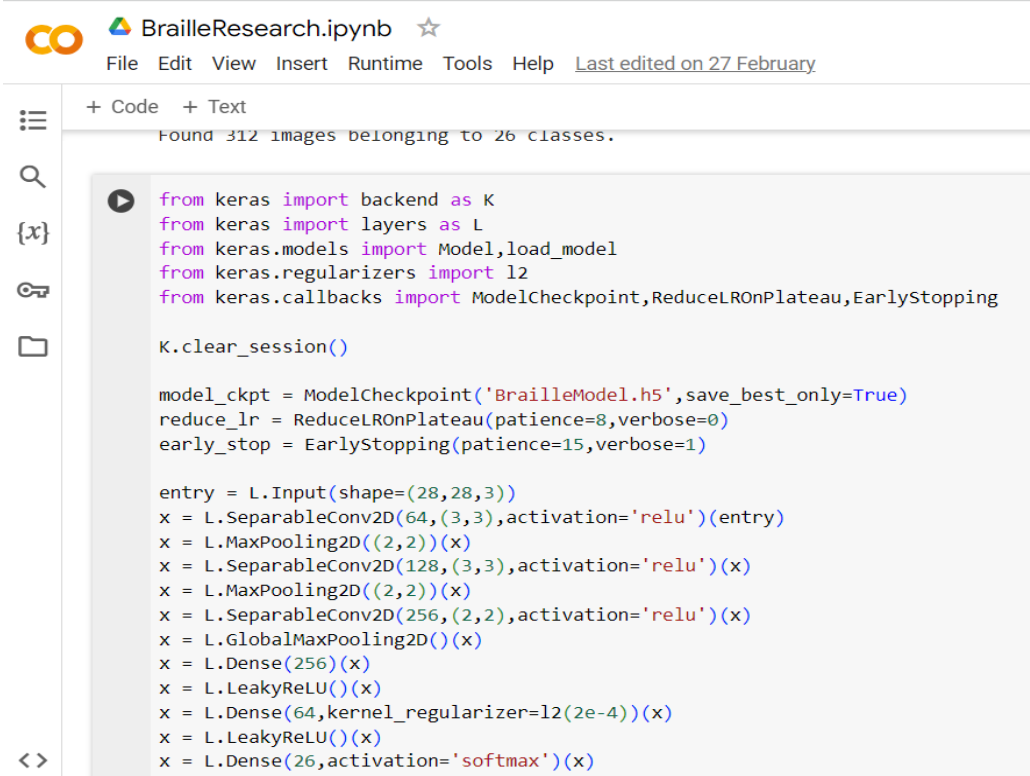
```
[ ] from keras.preprocessing.image import ImageDataGenerator
datagen = ImageDataGenerator(rotation_range=20,
                             shear_range=10,
                             validation_split=0.2)

train_generator = datagen.flow_from_directory('./images/',
                                              target_size=(28,28),
                                              subset='training')

val_generator = datagen.flow_from_directory('./images/',
                                           target_size=(28,28),
                                           subset='validation')

Found 1248 images belonging to 26 classes.
Found 312 images belonging to 26 classes.
```

The above image shows the data preprocessing and splitting of images as training and testing set.



```

Found 312 images belonging to 26 classes.

from keras import backend as K
from keras import layers as L
from keras.models import Model, load_model
from keras.regularizers import l2
from keras.callbacks import ModelCheckpoint, ReduceLROnPlateau, EarlyStopping

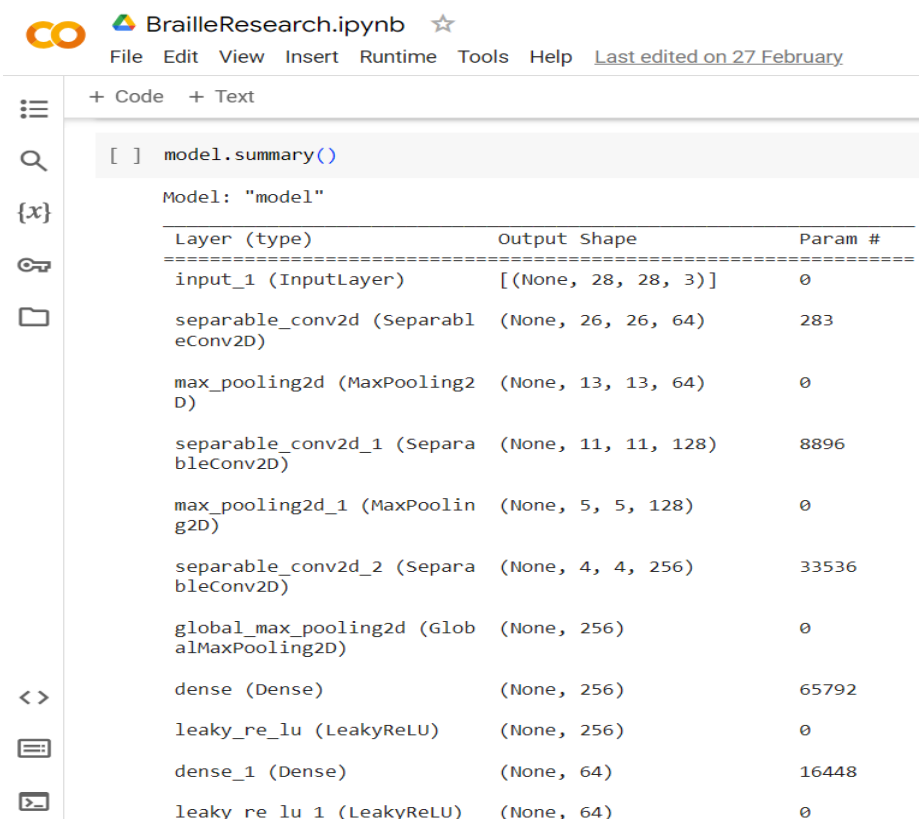
K.clear_session()

model_ckpt = ModelCheckpoint('BrailleModel.h5', save_best_only=True)
reduce_lr = ReduceLROnPlateau(patience=8, verbose=0)
early_stop = EarlyStopping(patience=15, verbose=1)

entry = L.Input(shape=(28, 28, 3))
x = L.SeparableConv2D(64, (3, 3), activation='relu')(entry)
x = L.MaxPooling2D((2, 2))(x)
x = L.SeparableConv2D(128, (3, 3), activation='relu')(x)
x = L.MaxPooling2D((2, 2))(x)
x = L.SeparableConv2D(256, (2, 2), activation='relu')(x)
x = L.GlobalMaxPooling2D()(x)
x = L.Dense(256)(x)
x = L.LeakyReLU()(x)
x = L.Dense(64, kernel_regularizer=l2(2e-4))(x)
x = L.LeakyReLU()(x)
x = L.Dense(26, activation='softmax')(x)

```

The above image shows the creation of BT-CNN model.



```

[ ] model.summary()

```

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 28, 28, 3)]	0
separable_conv2d (SeparableConv2D)	(None, 26, 26, 64)	283
max_pooling2d (MaxPooling2D)	(None, 13, 13, 64)	0
separable_conv2d_1 (SeparableConv2D)	(None, 11, 11, 128)	8896
max_pooling2d_1 (MaxPooling2D)	(None, 5, 5, 128)	0
separable_conv2d_2 (SeparableConv2D)	(None, 4, 4, 256)	33536
global_max_pooling2d (GlobalMaxPooling2D)	(None, 256)	0
dense (Dense)	(None, 256)	65792
leaky_re_lu (LeakyReLU)	(None, 256)	0
dense_1 (Dense)	(None, 64)	16448
leaky_re_lu_1 (LeakyReLU)	(None, 64)	0

The above image shows that the BT-CNN model has successfully been created.

```

BrailleResearch.ipynb
File Edit View Insert Runtime Tools Help Last edited on 27 February

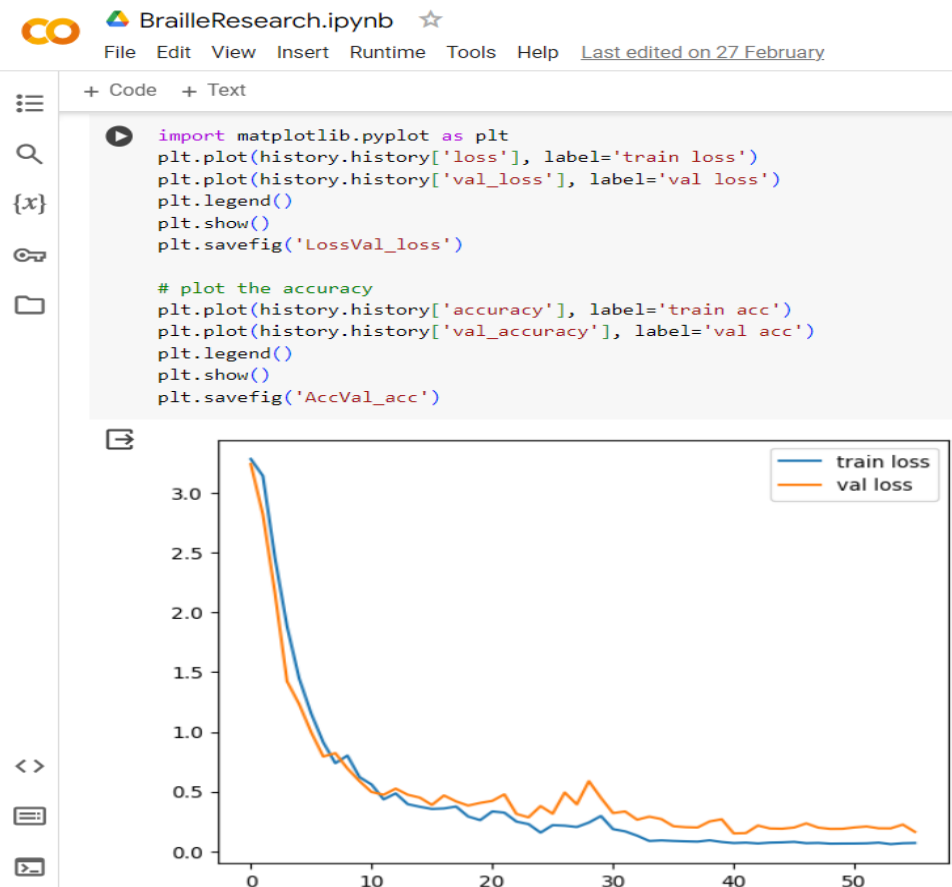
+ Code + Text

history = model.fit_generator(train_generator,
                             validation_data=val_generator,
                             epochs=100,
                             callbacks=[model_ckpt,reduce_lr,early_stop],
                             verbose=1)

Epoch 1/100
39/39 [=====] - 6s 88ms/step - loss: 3.2842 - accuracy: 0.0304 - val_loss: 3.2423 - val_accuracy: 0.0609 - lr: 0.0010
Epoch 2/100
1/39 [.....] - ETA: 4s - loss: 3.2169 - accuracy: 0.0625/usr/local/lib/python3.10/dist-packages/keras/src/engine/training
saving_api.save_model(
39/39 [=====] - 4s 100ms/step - loss: 3.1421 - accuracy: 0.1082 - val_loss: 2.8195 - val_accuracy: 0.2404 - lr: 0.0010
Epoch 3/100
39/39 [=====] - 4s 113ms/step - loss: 2.4569 - accuracy: 0.2853 - val_loss: 2.1625 - val_accuracy: 0.3590 - lr: 0.0010
Epoch 4/100
39/39 [=====] - 3s 77ms/step - loss: 1.8833 - accuracy: 0.4720 - val_loss: 1.4231 - val_accuracy: 0.6250 - lr: 0.0010
Epoch 5/100
39/39 [=====] - 3s 69ms/step - loss: 1.4484 - accuracy: 0.5817 - val_loss: 1.2324 - val_accuracy: 0.6186 - lr: 0.0010
Epoch 6/100
39/39 [=====] - 4s 99ms/step - loss: 1.1533 - accuracy: 0.6747 - val_loss: 0.9969 - val_accuracy: 0.7500 - lr: 0.0010
Epoch 7/100
39/39 [=====] - 3s 77ms/step - loss: 0.9121 - accuracy: 0.7284 - val_loss: 0.7933 - val_accuracy: 0.7660 - lr: 0.0010
Epoch 8/100
39/39 [=====] - 3s 77ms/step - loss: 0.7376 - accuracy: 0.7829 - val_loss: 0.8214 - val_accuracy: 0.7500 - lr: 0.0010

```

The above image shows that the BT-CNN model is being evaluated.



The above image shows the plotting of loss and accuracy curves of the BT-CNN Model.

Upload new File

Choose file

No file chosen

Upload

The Braille output will appear in this place

Click To Play Audio

Clear Text

Upload new File

Choose file

No file chosen

Upload

The Braille output will appear in this place

Braille to Speech using CNN



46

C. RESEARCH PAPER

Braille to Speech conversion using BT-CNN

Aditya Upadhye, Tharunnesh R, Jeslin Shanthamalar J, C. A. Daphine Desona Clemency, N. S. Usha

Department of Computer Science and Engineering

Sathyabama Institute of Science and Technology

Chennai, India

adityaupadhye47@gmail.com

imtharunr@gmail.com

jeslin.cse@sathyabama.ac.in

daphine.cse@sathyabama.ac.in

ushans.cse@sathyabama.ac.in

Abstract— Braille is a universally accepted system of writing and understanding scripts for blind people. It is a touch-based system for reading and writing. The alphabet is composed of specific dot configurations. An image containing a word or statement in Braille can be given to the model. Although there are various ways to implement it, the proposed model uses Convolutional Neural Network (CNN) to analyze the dots in the Braille image and convert them into letters, eventually forming words and statements. Keras library is to perform several operations like generating image data, splitting the dataset and evaluating the model's accuracy. This text is further processed by the model and the final output is an audio file which is a speech of the text obtained from the Braille Image. This text to speech conversion is done by using Google Text to Speech. By training the model with a large number of epochs, an accuracy of about 96.30% is achieved.

Keywords— Machine Learning, Deep Learning, Convolutional Neural Network, Speech Synthesis

I. INTRODUCTION

Traditionally, Braille is used by blind, partially sighted, and deafblind people. Louis Braille created braille in France during the 1820s. The spatial units used to form braille symbols are called braille cells. Imagine the number 6 on a dice. A complete braille cell is made up of six raised dots organised in two parallel vertical rows of three dots each. The digits one through six represent the locations of the dots. When one or more of these six dots are used, there are 64 possible combinations. Both filled and empty cells are also used. Punctuation, numbers, alphabet letters, and even entire phrases can be represented by a group of one or more dots positioned inside the cell.

There are several ways to produce Braille, including:

- **Braille Typewriter:** This is a special typewriter with just six keys, one for each of the dots in a Braille cell. When a key is pressed, a metal stylus embosses the dots for that character onto the paper.
- **Braille Printer:** This is a kind of printer that embosses (puts raised dots on-to) paper to create Braille characters.
- **Refreshable Braille Display:** This is a device containing row upon row of mechanical or electronic

Braille cells. Each cell has multiple pins that can be raised or lowered to form any Braille characters.

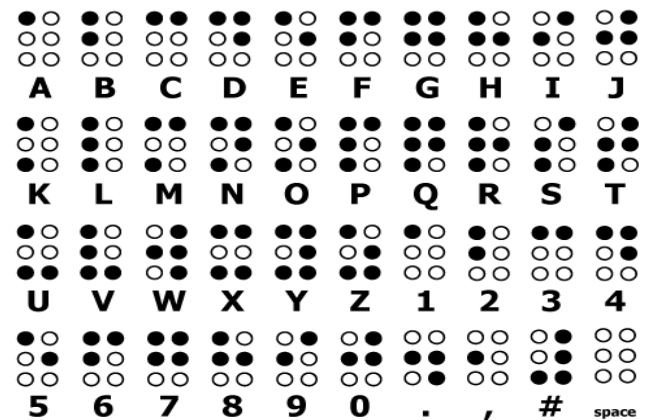


Fig. 1. Braille Characters

A Braille to Speech conversion tool whispers the secrets of the written word through raised dots, providing a unique means to encode text. But, remarkable as Braille is, it comes with its own set of challenges. The specialized knowledge required for reading and writing often leaves it an enigma. Not just about algorithms and data, braille to speech conversion brings a means to empower individuals with the most important of gifts -- the written word. It is a digital bridge, one that could make the art of reading available to everyone.

For people with visual impairment, the first and foremost advantage of Braille to Speech conversion is accessibility. It allows users to hear printed material, signages and other written information, which promotes independence and inclusion with the help of audio output. Another key advantage of a Braille to Speech translation model is its ability to translate Braille text into spoken language in real-time. Using a well-designed Braille to Speech model, a blind person would be able to immediately hear spoken output of any Braille text they place their fingers on.

In an educational context, Braille to Speech conversion provides an important form of auditory reinforcement for Braille students. This reinforcement can help students to develop a firm association between new Braille characters

and their corresponding spoken language in a faster manner than trial and error alone would allow.

II. RELATED WORK

Using preprocessing, image segmentation, and feature extraction approaches, M. Hanumanthappa et al. demonstrated that several researchers' efforts were still unsuccessful in achieving 100% accuracy with a high translation rate. In his research, Chollet demonstrated how Inception modules acted as a midway point, situated at opposite ends of a discrete spectrum, between convolutions and depthwise separable convolutions. Kirill Smelyakov et al. used Optical Braille Character Recognition system using Artificial Neural Networks. K M Naimul Hassan et al. developed a dual-purpose system, in 2019, which can help to detect objects around them and a refreshable braille display to display text in the form of Braille Characters. A. A. Choudhury et al. introduced a novel technique that uses the YOLO (You Only Look Once) algorithm to enhance real-time recognition of Braille characters. Known for its speed and recognition accuracy, Yolo was modified to recognize Braille characters, with the aim of improving accessibility and independence for the visually impaired.

A useful method for pronouncing and converting scenes to text was proposed by S. M. Qaisar et al. in 2019. This method can aid visually impaired people in understanding written text found in everyday objects such as caution notices, medicine cartons, and basic food goods. V Jha and K Parvathi proposed a transliteration system, in 2019, which converts Hindi characters into their corresponding Braille Characters using Optical Braille Recognition (OBR) Braille dot pattern. C. Park et al. proposed a 3-stage method using R-CNN as the first stage for banknote images captured in disturbed environments using a smartphone camera. A device was demonstrated by G. C. Bettelani et al. that uses an electromagnetic field to actuate six independently controlled dots. The alphabet letters can be encoded with this device in accordance with the Braille system. Additionally, the blind participants said the gadget was well-accepted, practical, economical, and intuitive. B.-M. HSU used the Radio Character Segmentation Algorithm (RCSA) to recognize complex Braille characters and translate them into plain English. In a pivotal look, P. Kaur et al. harnessed both conventional and contemporary strategies to broaden a laptop program that converts Hindi Braille into spoken language. By combining those methods, they created a gadget that complements accessibility for the visually impaired.

Shokat et al. did groundbreaking research in assistive technology for visually impaired individuals in 2020. Their use of the GoogleNet Inception model achieved an impressive 95.8% accuracy in predicting Braille characters. Mao developed a model that converts Chinese Braille into spoken language, increasing communication accessibility and inclusion. This pioneering work extends beyond text, providing a natural understanding of the sounds of Braille letters, and contributing to universal inclusion and accessibility. In 2021, T. Kausar et al.'s innovative approach included replacing modules in traditional convolutional neural networks (CNNs) with techniques designed to significantly reduce computational costs. This system addresses a fundamental challenge in Braille character recognition by streamlining computational requirements, especially in critical or real-time. A method to develop a low-cost, portable communication device that can translate any Braille input into

alphanumeric text format was presented by Ramachandran et al.

A mechanism that prints the Braille alphabet on a sheet after receiving an alphabet voice command was proposed by M. A. Hussain. In the same year, Bhatia et al. Automatic Arabic language has contributed to the translation system in Braille and contributed to the simple technology, that "Convolutional Neural Network Real Time Arabic Speech Recognition to Arabic Braille" is known as the requirement of the classification, that in real time in Arabic for translation capabilities. Leveraging the power of convolutional neural networks (CNNs), this revolutionary technology empowers individuals with sensory impairment, providing effective communication and access to information, exemplifying that they are committed to being inclusive and improving their lives. Ramani RG and Shanthamalar JJ have conducted extensive prior research on feature extraction methods utilising full-reference and no-reference image quality measures for picture quality classification. This work presents an automatic feature extraction system for fundus images utilising image processing techniques, followed by an automatic image quality classification system employing different classification algorithms. A camera embedded hand glove with OCR for conversion to Braille and speech for blinds have been created by Seung-Bin Park and Bonghyun Kim to improve the quality of life.

III. METHODOLOGY

A. Dataset

In the proposed approach, the idea is to use the resources available inside the "Braille Character Dataset" from Kaggle to train our model. In BW scale, each image is a letter and is 28x28 pixels. The image number, the alphabet of characters, and the kind of data augmentation applied to each image make up their names. The dataset consists of 26 characters, 3 augmentations, and 20 distinct images with varying augmentation values (i.e., distinct shift, rotational, and brightness values). 1560 images make up this dataset.

B. Convolutional Neural Network

Convolutional Neural Network (CNN) is like an intelligent computer system that is really good at understanding images and videos. It can see and recognize different things in images, similar to how humans see and understand things. CNNs are used for many applications such as recognizing objects in images, reading medical images, and even understanding words in text. They're like the super detectives of the computer world for visuals.

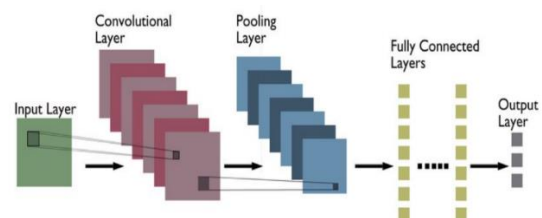


Fig. 2. Convolutional Neural Network

The first and essential step in the digital representation of visual data is image acquisition. It involves special devices

such as cameras, scanners and sensors to capture visible information from the physical world. These devices convert analog information such as light and electromagnetic radiation into digital images. The quality of these acquired images is extremely important as it directly affects the success of image production and subsequent analysis. Factors such as lighting, camera settings, and sensor sensitivity play an important role in ensuring the accuracy and realism of digital images. Image acquisition is widely used in a variety of industries, including photography, medical imaging, surveillance, and computer vision. It serves as the basis for such vital processes as medical diagnosis, discovery, and independent guidance. By converting real-world visual information into a digital format, image acquisition enables computers and digital systems to make sense of and interact with the physical world, enabling product delivery of many practical and new uses.

In Image Processing, a comprehensive image records preprocessing strategy is implemented with the usage of two key strains. TensorFlow's Keras library imports the image module, which contains functions for loading and preprocessing images. This ensures finest formatting for superior version performance. Concurrently, the ImageDataGenerator module line employs the ImageDataGenerator module for dynamic dataset augmentation all through schooling, enhancing the model's adaptability and generalization throughout numerous scenarios.

To make the proposed version even more reliable, the plan is putting preprocessing strategies into play, refining the process of entering pictures before they undergo analysis. The operational method of proposed version entails a meticulous dissection of the complete photograph into smaller fragments, accomplished by way of figuring out the feature six dots that compose a Braille character.

The pooling process is implemented using the L.MaxPooling2D layers. Specifically, the lines $x = \text{L.MaxPooling2D}((2,2))(x)$ after the separable convolution layers downsample the spatial dimensions of the input feature maps through max-pooling. A downsampled (pooled) feature map is produced by the pooling process known as "max pooling," which determines the maximum value for each patch in a feature map.

In the proposed work, the feature extraction process occurs in the layers of the neural network model, specifically within the SeparableConv2D and GlobalMaxPooling2D layers. These layers are tasked with capturing essential features from the input images. The SeparableConv2D layers perform convolution operations to capture spatial hierarchies of features, while the GlobalMaxPooling2D layer aggregates the most important features by taking the maximum value across spatial dimensions. Therefore, the lines involving SeparableConv2D and GlobalMaxPooling2D contribute to the feature extraction process in this model. The extraction method happens when the use of the model.predict(x) line. This line sends the enter records x through the skilled version, and the output represents the extracted capabilities or predictions primarily based at the discovered patterns from the input. The np.expand_dims(x, axis=zero) line is used to adjust the shape of x by adding a further dimension, which is a common preprocessing step, but it does not perform the characteristic extraction itself.

C. BT-CNN Model

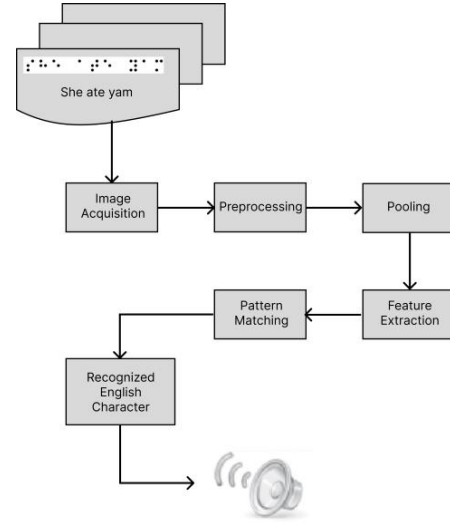


Fig. 3. Architecture Diagram

Figure 3 presents a visible depiction of the proposed version, illustrating its structure and functionality within the system of changing Braille characters into an audio file. This graphical illustration serves as a schematic overview highlights the important components and connections implicated in the operation of the model. The figure captivates a clear visual representation of how the proposed structure is employed to facilitate the transformation of Braille characters, illuminating the sequential steps and interactions that yield the final output—an audio clip. This visible representation aids in comprehending the design of the model and its role in the transformation process of Braille character conversion, by providing a visible reference for readers to better hold close the method hired inside the study.

A CNN architecture was employed, comprising several layers: separable convolutional layers followed by max pooling operations. The network further utilized global max pooling to reduce spatial dimensions. Dense layers with ReLU and Leaky ReLU activation functions, along with a softmax output layer, were implemented for feature extraction and classification. The final output layer consisted of 26 units, indicating a multi-class classification task.

TABLE I. LAYERS IN BT-CNN

Layer (type)	Output Shape	Parameter
Input	28, 28, 3	0
Separable Convolution	26, 26, 64	283
Max Pooling	13, 13, 64	0
Separable Convolution	11, 11, 128	8896
Max Pooling	5, 5, 128	0
Separable Convolution	4, 4, 256	33536
Global Max Pooling	256	0
Dense	256	65792
LeakyReLU	256	0
Dense	64	16448
LeakyReLU	64	0
Dense	26	1690
Output	26	0

The images of the training dataset are augmented by applying image augmentation techniques (such as rotating, brightness, width-height shifting etc), thus increasing the dataset further and significantly increasing the diversity of the images available for training the model thus generalizing the model.

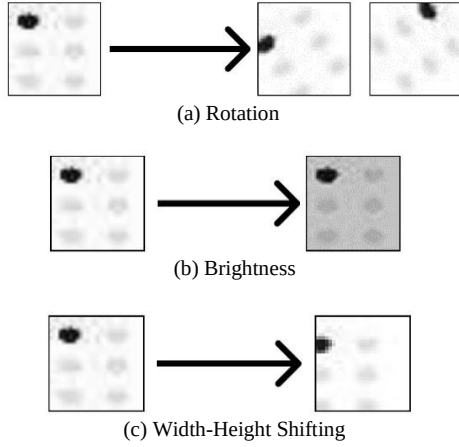


Fig. 4. Image Augmentation performed on an image taken from dataset

After Image Augmentation, the training dataset is splitted into the training(80%) and the validation(20%) data. The model architecture is defined and compiled using the Adam optimizer and categorical cross-entropy loss function for multi-class classification tasks.

The system specifications used for developing the proposed model were:

Processor: Intel Core i7

RAM: 16 GB

GPU: Intel Iris Xe

OS: Microsoft Windows 11

An alternative platform would be Google Colab that provides CPU and T4 GPU hardware runtime for free.

Training and validation data generators were used to train the model across 100 epochs. The training process is monitored and optimised by the use of callbacks, which include model checkpointing, learning rate decrease, and early termination. Epoch, in machine learning, is referred to one-time passing of the entire training data through the model or algorithm. After certain number of epochs, if there isn't significant increase in the model's accuracy, the early-stopping callback method executes to stop the training process. The best version Then, the accuracy of model is evaluated using a validation data generator. The achieved model accuracy, computed using the evaluation generator, was printed with four decimal places for precision. The evaluate_generator function is used to return a list of metric values, and extract the accuracy value.

The model.predict method is used to generate predictions for the input data provided by val_generator. This typically occurs after the training phase when the model has learned to extract relevant features from the training data. In the proposed work, plt.imshow(img) uses Matplotlib to plot the image represented by the img variable.

The image.img_to_array(img) function is used to convert the image into a NumPy array. Then, the dimensions of an array are found using the shape function and this is the shape of the array - (28, 28, 3). Preprocessing of input character is done by expanding its dimensions and then use it to make predictions using the proposed model. The predicted output(desired English Alphabet) was obtained by finding the index of the highest probability prediction.

As the final step in this system, the model is designed to generate an audio file that aligns with the Braille word or sentence first of all supplied as an image. In order to streamline this process, Google Text to Speech is used.

Once trained, the network can synthesize speech for new input text by:

- Text pre-processing: The input text is analyzed and segmented into phonemes (basic units of sound).
- Feature prediction: The network predicts the acoustic features corresponding to each phoneme based on the surrounding context.
- Waveform generation: The predicted acoustic features are used to generate a speech waveform that resembles the target speaker's voice.

IV. RESULTS

This paper deals with the image recognition of Braille characters and convert Braille Text into Audio Speech, a CNN approach is proposed which uses various different types of layers in the convolutional neural network to recognize the Braille character images and classify each into one of the 26 characters of the English alphabets. The highest accuracy obtained by the proposed model is 96.30%.

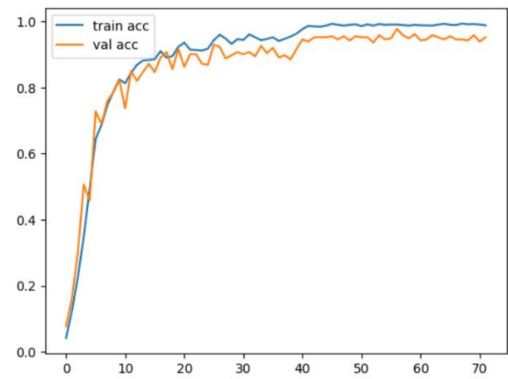


Fig. 5. Graph of accuracy at each epoch

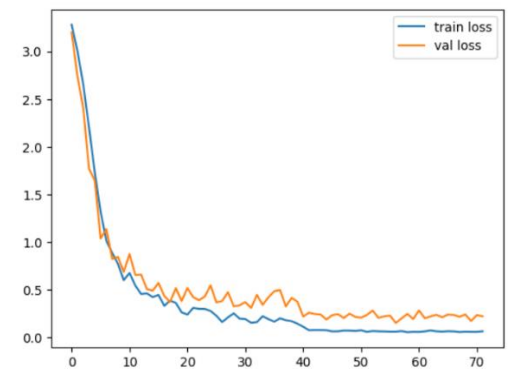


Fig. 6. Graph of loss function at each epoch

V. COMPARATIVE ANALYSIS

TABLE II. COMPARATIVE ANALYSIS OF EXISTING METHODS VS PROPOSED METHOD

No.	Researcher	Algorithm	Accuracy
1	K. Smelyakov (2018)	ANN	95.00%
2	S. Shokat (2020)	GoogLeNet	95.80%
3	J Mao (2021)	Prosody Prediction	94.42%
4	T Kausar (2021)	DenseNet201	95.20%
5	M. A. Hussain (2022)	VGG16	92.42%
6	Proposed Work	BT-CNN	96.30%

A comparison between the proposed system and recent methods is shown in Table 2. Our system evaluated the overall performance with existing approaches, however it is challenging to compare them because they were trained on different datasets. J Mao and M. A. Hussain's approaches convert Braille text to only Chinese and Bangla languages, respectively, which caters a relatively small set of people as compared to the Internationally accepted English language. S. Shokat's study does not mention incorporating feedback from visually impaired individuals during the development and evaluation of the system. Involvement of end-users in the design process could provide valuable insights into their needs, preferences, and usability challenges, ultimately leading to a more user-centric and effective solution. While T. Kausar's preprocessing stage involves the application of Principal Component Analysis (PCA) to enhance feature representation, they also add complexity to the system and may require additional computational resources.

VI. CONCLUSION AND FUTURE WORK

In this study, efforts were made to use Deep learning for braille to speech conversion and got accuracy of 96.30%. This system can be proved as a major breakthrough in the field of automation and innovative technologies and can be used in various websites and places with diverse set of people and can further be incorporated with systems which can help all the visually challenged people as well as normal people around the world to understand and visualize their respective surroundings and communicate with people in a simpler and efficient manner. This study can further be enhanced in the future using different and more accurate, high-resolution datasets or using different, more complex methods like Extreme Machine Learning (EML), or Classification and Regression Trees (CART) for better performance of the model which may require higher computation power and much advanced softwares for efficient working.

REFERENCES

- [1] M. Hanumanthappa and V. V. Murthy, "Optical Braille recognition and its correspond-ence in the conversion of Braille script to text — A literature review," 2016 International Conference on Computation System and Information Technology for Sustainable Solutions (CSITSS), Bengaluru, India, 2016, pp. 297-301, doi: 10.1109/CSITSS.2016.7779374.
- [2] Chollet, "Xception: Deep Learning with Depthwise Separable Convolutions," IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1800-1807, Jul. 2017.
- [3] K. Smelyakov, A. Chupryna, D. Yermenko, A. Sakhon and V. Polezhai, "Braille Character Recognition Based on Neural Networks," 2018 IEEE Second International Conference on Data Stream Mining & Processing (DSMP), Lviv, Ukraine, 2018, pp. 509-513, doi: 10.1109/DSMP.2018.8478615.
- [4] K. M. Naimul Hassan, S. K. Biswas, M. S. Anwar, M. S. Iman Siam and C. Shahnaz, "A Dual-Purpose Refreshable Braille Display Based on Real Time Object Detection and Optical Character Recognition," 2019 IEEE International Conference on Signal Processing, Information, Communication & Systems (SPICSCON), Dhaka, Bangladesh, 2019, pp. 78-81, doi: 10.1109/SPICSCON48833.2019.9065110.
- [5] A. A. Choudhury, R. Saha, S. Z. Hasan Shoumo, S. Rafsun Tulon, J. Uddin and M. K. Rahman, "An Efficient Way to Represent Braille using YOLO Algorithm," 2018 Joint 7th International Conference on Informatics, Electronics & Vision (ICIEV) and 2018 2nd International Conference on Imaging, Vision & Pattern Recognition (icIVPR), Kitakyushu, Japan, 2018, pp. 379-383, doi: 10.1109/ICIEV.2018.8641038.
- [6] S. M. Qaisar, R. Khan and N. Hammad, "Scene to Text Conversion and Pronunciation for Visually Impaired People," 2019 Advances in Science and Engineering Technology International Conferences (ASET), Dubai, United Arab Emirates, 2019, pp. 1-4, doi: 10.1109/ICASET.2019.8714269.
- [7] V. Jha and K. Parvathi, "Braille Transliteration of hindi handwritten texts using machine learning for character recognition", Int J Sci Technol Res, vol. 8.10, pp. 1188-1193, 2019.
- [8] C. Park, S. W. Cho, N. R. Baek, J. Choi and K. R. Park, "Deep Feature-Based Three-Stage Detection of Banknotes and Coins for Assisting Visually Impaired People," in IEEE Access, vol. 8, pp. 184598-184613, 2020, doi: 10.1109/ACCESS.2020.3029526.
- [9] G. C. Bettelani, G. Averta, M. G. Catalano, B. Leporini and M. Bianchi, "Design and Validation of the Readable Device: A Single-Cell Electromagnetic Refreshable Braille Display," in IEEE Transactions on Haptics, vol. 13, no. 1, pp. 239-245, 1 Jan.-March 2020, doi: 10.1109/TOH.2020.2970929.
- [10] M. A. Khan, P. Paul, M. Rashid, M. Hossain and M. A. R. Ahad, "An AI-Based Visual Aid With Integrated Reading Assistant for the Completely Blind," in IEEE Transactions on Human-Machine Systems, vol. 50, no. 6, pp. 507-517, Dec. 2020, doi: 10.1109/THMS.2020.3027534.
- [11] B.-M. Hsu, "Braille recognition for reducing asymmetric communication between the blind and non-blind," Symmetry, vol. 12, no. 7, p. 1069, Jun. 2020, doi: 10.3390/SYM12071069.
- [12] P. Kaur, S. Ramu, S. Panchakshari, and N. Krupa, "Conversion of Hindi Braille to speech using image and speech processing," in Proc. IEEE 7th Uttar Pradesh Section Int. Conf. Elect., Electron. Comput. Eng. (UPCON), Nov. 2020, pp. 1-6, doi: 10.1109/UPCON50219.2020.9376566.
- [13] S. Shokat, R. Riaz, S. S. Rizvi, A. M. Abbasi, A. A. Abbasi, and S. J. Kwon, "Deep learning scheme for character prediction with position-free touch screen-based Braille input method," Hum.-centric Coput. Inf. Sci., vol. 10, no. 1, pp. 1-24, Dec. 2020, doi: 10.1186/s13673-020-00246-6.
- [14] J. Mao, J. Zhu, X. Wang, H. Liu, and Y. Qian, "Speech synthesis of Chinese Braille with limited training data," in Proc. IEEE Int. Conf. Multimedia Expo (ICME), Jul. 2021, pp. 1-6, doi: 10.1109/icme51207.2021.9428160.
- [15] T. Kausar, S. Manzoor, A. Kausar, Y. Lu, M. Wasif and M. A. Ashraf, "Deep Learning Strategy for Braille Character Recognition," in IEEE Access, vol. 9, pp. 169357-169371, 2021, doi: 10.1109/ACCESS.2021.3138240.
- [16] S. Ramachandran, N. Rajan, K. N. Pallavi, J. Subashree, S. Suchithra and B. Sonal, "Communication Device for the Visual and Hearing Impaired Persons to Convert Braille Characters to English Text," 2021 International Conference on Emerging Smart Computing and Informatics (ESCI), Pune, India, 2021, pp. 587-592, doi: 10.1109/ESCI50559.2021.9396859.
- [17] M. A. Hussain, R. I. Rifat, S. B. Iqbal, S. Biswas, M. G. Rabiul Alam and M. T. Reza, "Deep Learning based Bangla Voice to Braille Character Conversion System," 2022 IEEE 13th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON), Vancouver, BC, Canada, 2022, pp. 0262-0267, doi: 10.1109/IEMCON56893.2022.9946619.
- [18] Bhatia, Surbhi & Devi, Ajantha & Alsuwailam, Razan & Mashat, Arwa. (2022). Convolutional Neural Network Based Real Time Arabic Speech Recognition to Arabic Braille for Hearing and Visually Impaired. Frontiers in Public Health. 10.3389/fpubh.2022.898355.

- [19] Ramani RG, Shanthamalar JJ. Automated image quality appraisal through partial least squares discriminant analysis. *Int J Comput Assist Radiol Surg*. 2022 Jul;17(7):1367-1377.
- [20] Seung-Bin Park and Bonghyun Kim. "Design of CNN-based Braille Conversion and Voice Output Device for the Blind" *Journal of Internet of Things and Convergence* 9, no.3 (2023) : 87-92.doi: 10.20465/KIOTS.2023.9.3.087.