Deep Learning based Braille Character to Bangla Voice Conversion System

by

Ariq Sadiq Chowdhury
22101817
Rafid Bin Bakhtiar
22101856
Aritra Chakraborty
22101892
Aowfi Adon Foraejy
22101095

A Thesis report submitted to the Department of Computer Science and Engineering in partial fulfillment of the requirements for the degree of B.Sc. in Computer Science and Engineering

Department of Computer Science and Engineering
Brac University
January 2026.

Declaration

It is hereby declared that

1. The thesis submitted is my/our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

Student's Full Name & Signature:

Ariq Sadiq Chowdhury	Rafid Bin Bakhtiar
22101817	22101856
Aritra Chakraborty	Aowfi Adon Foraejy
22101892	22101095

Approval

The thesis titled **"Deep Learning based Braille Character to Bangla Voice Conversion System"** submitted by

1. Ariq Sadiq Chowdhury (22101817)
2. Aritra Chakraborty (22101892)
3. Rafid Bin Bakhtiar (22101856)
4. Aowfi Adon Foraejy (22101095)

of Spring, 2025 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science in Spring 2025.

Examining Committee:

Supervisor:
(Member)

Dr. Amitabha Chakrabarty

Professor
Department of Computer Science and Engineering
Brac University

Head of Department:
(Chair)

Sadia Hamid Kazi, PhD

Chairperson and Associate Professor
Department of Computer Science and Engineering
Brac University

Thesis Coordinator:
(Member)

Md. Golam Rabiul Alam (PhD)

Professor
Department of Computer Science and Engineering
Brac University

Abstract

Braille is an essential mode of communication for visually impaired individuals, enabling them to read and understand contexts through a tactile system of raised dots. However, certain regions worldwide, especially Bangladesh, have limited access to digital solutions for Braille conversion, which poses a challenge for those individuals. This thesis presents a method for converting Braille Characters to Bangla Voice through a deep-learning based system and thus enhancing accessibility for visually impaired individuals as well as general individuals who cannot understand Braille. In this research we have addressed the scarcity of high volume labeled data consisting of more than 10000 high-resolution labeled data with approximately 1 million labeled braille instances by strictly maintaining Library of Congress spatial standards. This research simulataniously performs two different architecture one being a multi-stage segmentation-classification and other one being YOLO based unified architecture. The unified architecture vastly outperforms the traditional method by achieving Mean Average Precision (mAP@50) of 98.5 percent in character extraction. The recognized text is then converted into natural-sounding Bangla speech using a TTS engine. This system is evaluated through accuracy, processing speed, and friendly user experience, demonstrating its potential to bridge the communication gap for the visually impaired in Bangla-speaking communities.

Keywords: Deep Learning, Braille-Recognition, Text-To-Speech, Convolutional Neural Network, Bangla-Voice-Conversion, Image Processing, UNet, VGG16, ResNet50, MobileNetV3, Braille Segmentation, Character Classification, Large Language Model (LLM), YOLO

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Bengali is a language spoken by 300 million people worldwide. There is a significant population of people who speak Bangla who are visually impaired, have partial sight impairment, or have other eyesight complexities. Recent estimates indicate that approximately 750,000 people in Bangladesh suffer from blindness [21]. The number is almost similar in the neighboring state of West Bengal; approximately 950,000 individuals are affected by blindness, representing about 1.66% of the population[37]. But it's a matter of sorrow that they're left behind in the dark by society, which leads them not to get a fair chance in job opportunities and other aspects of life[48]. According to the article[48], Bangladeshi companies tend to view these people as a burden, in contrast to their international counterparts. These companies don't have inclusive and diverse policies regarding marginalized people.

The visually impaired face a great deal of difficulties in reading, comprehending and connecting with the environment. Such obstacles are particularly clear within the educational sphere, as visually impaired people have problems in accessing proper learning resources. As Meem and Shaheen [52] argue, the education system in Bangladesh is neglecting its visually impaired students by giving them an archaic and isolating system of education that sets unsurpassable barriers to their education and integration. In Bangladesh, the literacy rate of the visually impaired population is significantly below the average in the country, and is mainly because of the limited access to educational materials [31].

The difference demonstrates the significance of making educational materials more accessible in Bangladesh among the blind population. The lackings in resources needed by this visually impaired community, including Braille textbooks, digital formats compatible with screen readers, and other assistive technologies, aggravates the issues in this community. Filling up these gaps would be necessary, ensuring that visionally impaired individuals are able to enjoy the full benefits of the education system. This will assist in the process of equal level of learning and communication to all.

1.2 Rational of the Study or Motivation

Even though the prevalence of visual impairment is high in Bangladesh, there is a notable lack of easily accessible educational resources for visually impaired people, which leads to a literacy rate that is significantly below the national average. This accessibility gap is especially noticeable in higher education, where visually impaired students face significant obstacles in obtaining study materials, resulting in a high dropout rate. According to research, the lack of adequate educational resources causes these students to face systemic barriers to learning [31].

A problem arises when we notice there is a lack of digital materials that cater to the needs of visually impaired students. Instructors offer study materials in PDF format, containing images that visually impaired students find difficult to understand. For this reason, visually impaired students cannot access these study materials, rendering them useless to them. This limitation has been shown to have a direct impact on their academic performance, as assignments are in PDF format, which makes them inaccessible, leading to poor marks and further exacerbating the academic challenges they face [36].

When visually impaired students try to use educational materials that are not sufficiently adapted to meet their needs, this situation serves as an example of the challenges they encounter. Visually impaired students find it difficult to keep up with their studies in the absence of the necessary resources. Thus, there is a pressing need to investigate different approaches to learning and assessment that can offer more accessibility. There would be no need for reader software, which frequently fails to interpret image-based files, if tests could be given and completed in Braille. Additionally, the evaluation process would be streamlined and made more efficient and accessible by turning Braille-based tests into voice outputs for teachers.

This study has a goal to address these issues by proposing a system that uses Braille for assessments and converts these assessments into voice, which will make the life of visually impaired students easier. The study seeks to enhance the academic experience and learning outcomes for visually impaired individuals. We want to ensure that they can fully participate in educational activities and achieve their academic potential.

1.3 Problem Statement

We saw considerable advancements in Braille-to-text conversion technologies, but accessibility tools like Braille-to-voice for visually impaired individuals remain underdeveloped, especially in languages like Bangla. Braille provides a tactile reading method; its integration with modern voice-based systems is minimal in Bengali, creating a significant barrier for the visually impaired population who rely on both Braille and voice support for effective communication and learning [15]. Moreover, while several Optical Braille Recognition (OBR) systems exist for English and a few other languages, a complete and automated solution that can read Bengali Braille notation and convert it into Bangla voice output is yet to be developed [23].

This gap becomes more critical because Bengali is one of the most spoken languages in the world, and a large number of visually impaired individuals in Bangladesh and West Bengal lack access to assistive technologies tailored to their linguistic and educational needs. Additionally, non-Braille-literate individuals who work with Braille users (e.g., teachers, family members) find it challenging to interpret Braille content without a suitable translation interface.

Therefore, a robust Bengali Voice System (BVS) that can accurately recognize Braille notation and convert it into intelligible Bangla speech is a solution to the problem. Such a system would bridge the communication gap, support inclusive education, and enable better access to information for the Bengali-speaking visually impaired community.

1.4 Objective

The long-term goal of this research study will be to come up with a sophisticated Braille-to-Bangla voice system which makes use of YOLO to extract characters (segmentation and classification) and a Large Language Model (LLM) to transform a sequence of characters to a meaningful sentences. These sentences are then run through a Text-to-Speech (TTS) engine to audio conversion. The research will also entail the generation of a simulated set of data that mimics the real world, resolving the existing shortage of. open-source, Bengali Braille marked datasets. The next important goals as shown in Figure 1.1 are:

(i) Data Acquisition: Collecting real-world Braille measurements, to generate a dataset with precise bounding box labels.

(ii) Character Extraction: Leveraging CNN architectures, specifically various versions of YOLO, to perform segmentation and characterization of individual Braille cells.

(iii) Performance Evaluation: Evaluating different segmentation approaches in terms of their accuracy and performance in character recognition.

(iv) Linguistic Processing: Passing the character streams into modern LLMs to construct grammatically correct and meaningful sentences.

(v) Speech Synthesis: Converting the finalized sentences into audio via an advanced TTS engine.

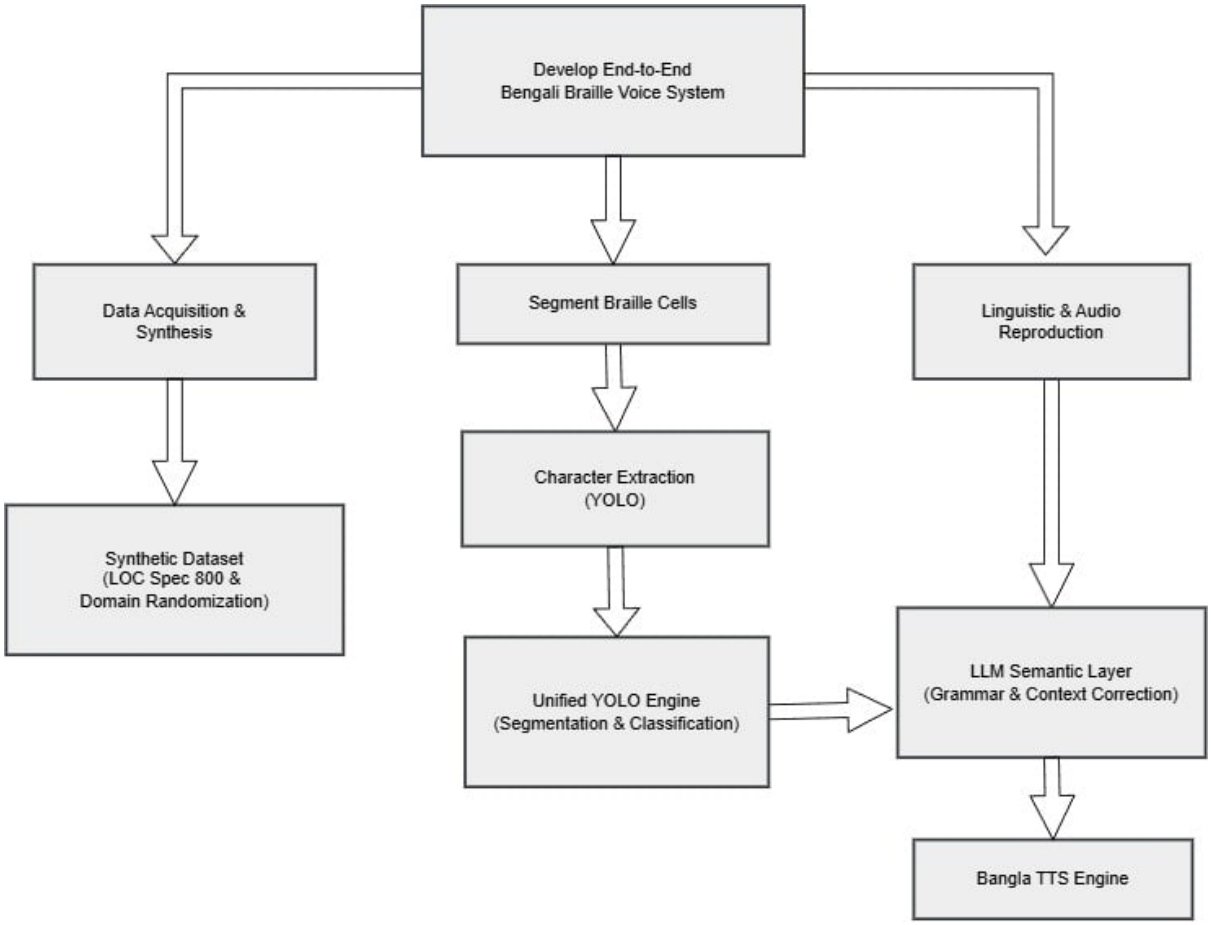

Figure 1.1: Objective Flow

1.5 Methodology in Brief

The system follows a modular and sequential deep learning pipeline to convert Bengali Braille script into synthesized Bangla Speech. The methodology comprises five core stages as shown in Figure 1.2:

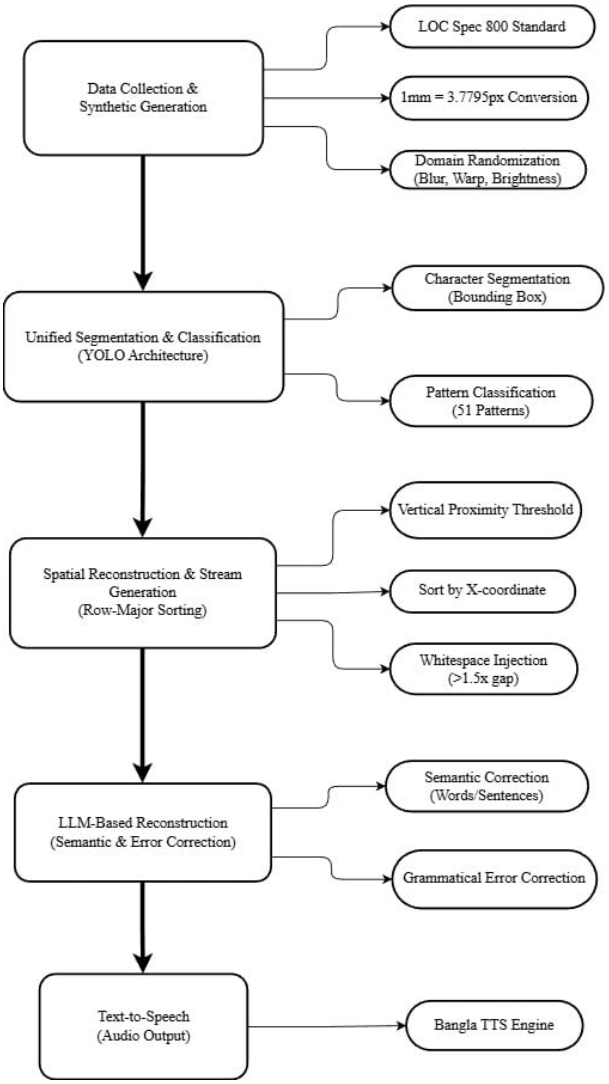

Figure 1.2: Methodology Flow

1. Data Collection

   The initial step of this research focuses on utilizing a high-fidelity synthetic labeled data generation strategy. Due to the scarcity and high error rate of manual Braille labeling, we developed an automated data generation system based on the Library of Congress (LOC) Specification 800. To ensure its real-world representation we applied a conversion factor of 1 mm = 3.7795275591 pixels.

   This conversion factor allows the system to construct dot diameters (1.5 mm) and cell spacing (2.5 mm) with absolute precision. By leveraging Domain Randomization techniques, we applied stochastic augmentation that included perspective wrapping, gaussian blur, and dynamic brightness. This step ensures the generated patterns represent real world tactile script images captured by mobile devices.

2. Unified Character Segmentation and Classification (YOLO):

   After employing different traditional segmentation and classification techniques that utilizes models like Resnet, UNet, we have revised our pipeline to employ

YOLO (You Only Look Once) architecture. YOLO performs two different actions simultaneously. They are as follows:

- Character Segmentation: Initially it localizes the exact bounding box of each Braille cell within the input frame.
- Pattern Classification: Identifying which of the 51 Bengali Braille patterns that are contained in that bounding box. This approach significantly reduces computational complexity and drastically eliminates the error propagation found in traditional two-stage (segmentation-classification) with U-Net and VGG/Resnet.

3. Spatial Reconstruction and Stream Generation:

   After YOLO extracts all the detected characters, the system applies a spatial reconstruction algorithm to transform the detected boxes into a stream of binary characters.Inherently YOLO cannot understand the Reading order of the extracted character. Thus, a Row-Major Sorting Algorithm is implemented to handle the issue. The cells are grouped based on their vertical proximity threshold. Then, they are sorted based on their x-coordinated. This stage also analyzes the gap between two cells. If the gap exceeds 1.5 times the standardized cell width, it automatically injects a whitespace to preserve the original sentence structure.

4. LLM-Based Reconstruction:

   The stream character coming from the segmentation model can have ambiguity or errors due to noise. In this step, the system passes the stream of characters to a Large Language Model (LLM). The LLM also performs two tasks. They are as follows:

   - Semantic Correction: This step takes the stream of characters and constructs meaningful words and sentences.
   - Error Correction: The construction can have grammatical inconsistencies. This step works to resolve that and create meaningful Bangla sentences.

5. Text-to-Speech (TTS) Conversion:

   The final Bangla text is converted into natural-sounding audio using Bangla-supported TTS engines such as Google TTS, Mozilla TTS, or eSpeak-NG. These tools are integrated via Python to generate fluent speech output that is easily comprehensible for end users [25].

1.6 Scopes and Challenges

This research aims to provide a scalable and accessible tool that bridges the communication gap between people of Bengali-speaking individuals with visual impairments. The system can assist:

- Students in accessing academic resources,
- Teachers and caregivers in interpreting Braille,
- NGOs and rehabilitation centers promote inclusive education.

By integrating image processing and deep learning, the system has potential applications in education, assistive reading, and public service kiosks. Moreover, the modular architecture allows future enhancements, such as OCR integration or handwriting recognition for Braille.

Anticipated Challenges

Despite having its potential, several challenges are expected for both the technical and practical sides.

- Scarcity of Braille Data Sets: Annotated data sets for Bengali Braille are extremely limited, posing a challenge for robust model training and evaluation.
- Non-Standard Input Conditions: Variations in lighting, paper quality, and Braille dot alignment complicate segmentation, necessitating advanced data-augmentation and preprocessing techniques.
- Braille dots and cells may be small and densely packed. Accurate localization using YOLO may introduce possibilities for errors, leading to incorrect character recognition.
- Errors added in the process of detecting or classifying Braille may be spread to the next steps, influencing the quality of text produced by LLM and final voice output.
- Even though LLM models help in enhancing grammar fluency, unintended text or incorrect contextual interpretation are possible if outputs are not sufficiently constrained.
- Bangla grammar, with its complex structure, such as inflections and conjunct constructions are difficult to reconstruct sentences correctly and place them in the proper context.
- Limited Bangla TTS Resources: Compared to English, Bangla TTS engines are less developed and may produce less natural intonations.

Addressing these challenges requires careful system optimization, model tuning, and user feedback during field testing phases.

Chapter 2

Literature Review

2.1 Preliminaries

The proposed system leverages a range of foundational concepts and technologies from image processing, deep learning, and natural language generation. Key preliminaries include:

- Braille System and Bangla Braille Mapping:

  Braille is a tactile script based on 2x3 dot cells. Bengali Braille follows a specific mapping that represents the native script's unique characters. Understanding and digitizing this mapping is critical for character classification and text generation.

- Braille Cell Localization using object detection technique:

  In the process of detecting and localizing individual Braille cells, YOLO plays an important role in detecting them. This is an object detector architecture capable of both localization and classification simultaneously [41]. Since they are efficient, they can be used in the detection of densely packed Braille cells in different imaging conditions.

- Large Language Model (LLM) for text generation and correction:

  An integration with a Braille recognition model can result in fragmented text or a text that is grammatically invalid due to misidentification errors or classification errors. Large Language Models (LLMs) are being utilized to optimize the identified sequence of characters and create incoherent text with the assistance of linguistic knowledge [27]. This eliminates ambiguities, minor recognition errors, and enhances readability.

- Bangla Text-to-Speech (TTS):

  TTS engines transform Bangla text into voice output. Libraries like Google TTS and Mozilla TTS offer open-source or API-based solutions for generating clear and human-like audio [25].

These preliminaries form the technological backbone of the system, enabling it to process visual Braille input and produce accurate Bangla voice output in real time.

2.2 Review of Existing Research

In recent years, the application of deep learning techniques to develop assistive technologies for individuals with visual impairments has grown rapidly. Convolutional neural networks (CNNs) and U-Net architectures have become the most preferred methods for image classification and segmentation-related tasks. This model has demonstrated remarkable results in interpreting tactile writing systems, such as Braille, which could potentially bridge the accessibility gap for visually impaired users through automation. One of the very important aspects of building a conversion system for braille to language is segmentation. Braille segmentation traditionally used methods like morphological filtering, employed methods like morphological filtering, edge detection, and projection. This model, however, was not effective in the real world. Thus, new methods like using U-Net, which was originally developed for biomedical image segmentation [16], gained popularity in recognising structured patterns. U-Net has a symmetric encoder-decoder structure with a skip connection, which helps it retain information across different scale [32]. These characteristics allow it to perform exceptionally for detecting cell boundaries in a noisy dataset. In the upcoming years, several variants of the U-Net architecture have been proposed, such as BraUNet [29], which is used for the semantic segmentation of Braille characters, achieving pixel-level classification into 64 distinct classes for full Braille cells.

There is also BddNet, which incorporates Gaussian diffusion in its loss function, resulting in improved dot-level localization as shown in Figure 2.1. [44]

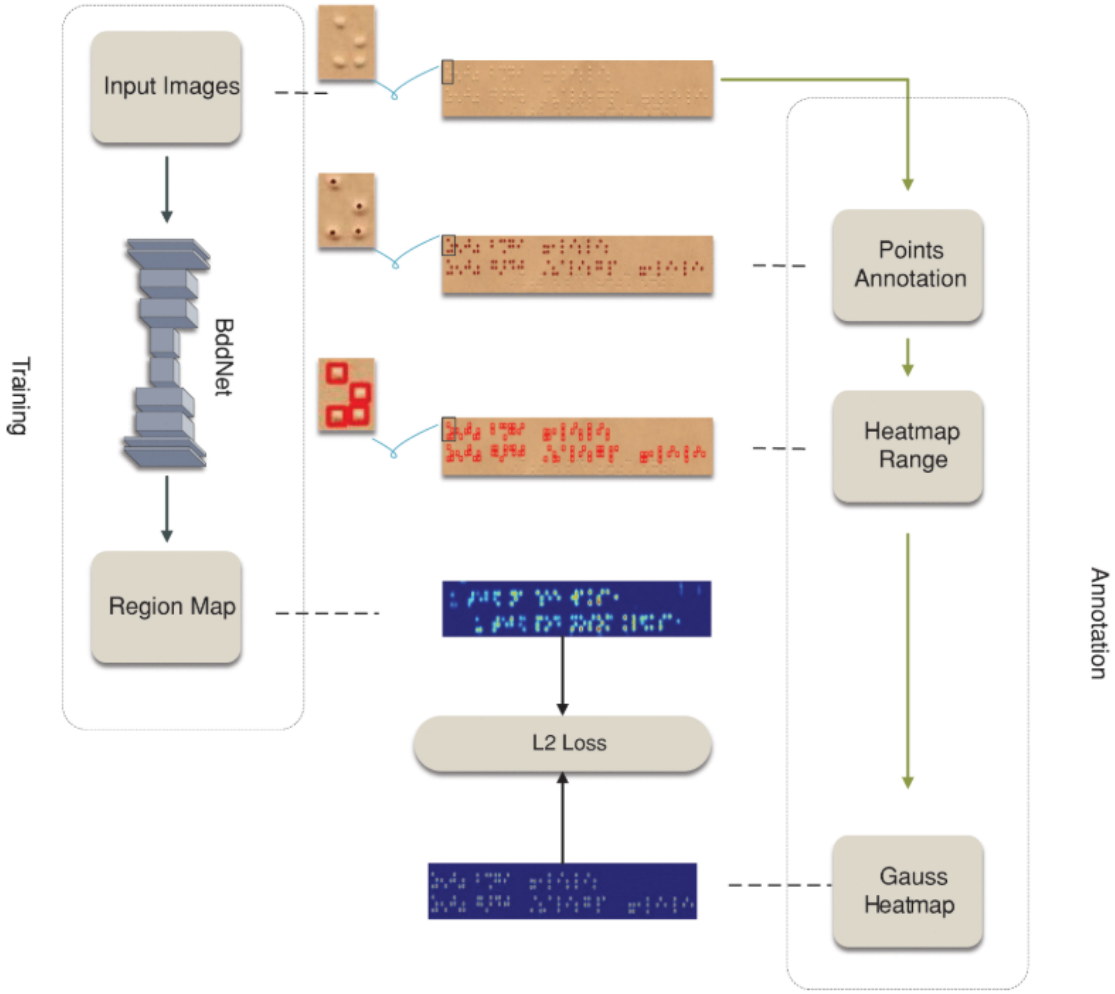

Figure 2.1: BddNet

The emergence of new methods and models has significantly enhanced accuracy and adaptability in diverse imaging conditions. In recent years, convolutional neural networks (CNNs) have become the standard model for image-based classification. Some variants of CNN architectures, VGG and ResNet, have shown incredible performance in visual recognition. VGG networks were introduced by Simonyan and Zisserman in 2014. It is a simple model that uses 3x3 filters stacked into the network. On the other hand, ResNet uses residual learning that addresses the vanishing gradient problem while training in very deep networks. In the final process of our automation system, we introduce a Text-to-Speech (TTS) engine that will effectively convert the classified characters into voice. This system greatly improves the accessibility of the system. There are TTS systems such as Google Text-to-Speech, eSpeak, and Festival that have performed quite well. All the technology that we have worked tremendously and has shown great results. But even nowadays, it lacks effectiveness in the Bangla domain. Sub-optimal system makes it harder to make the technology more reliable.

Many studies have been conducted to bridge the communication gap for the visually impaired with the help of Braille-to-text and speech conversion systems. Hardware solutions were mostly done in earlier times. For example, there were Braille keypads and lookup

tables for text conversion and audio output in FPGA-based systems [11]. Although these systems worked, they had multiple drawbacks, such as fixed speech volume and pace, and limited flexibility.

Anuradha et al.[18] introduced a low-cost portable communication system to enable tactile reading that makes use of servo motors to raise dots in Braille cells. Unfortunately, finger movement over the device brought some inconsistent results. Another creative approach, such as BrailleBand, uses vibrating motors to transmit Braille codes to the user's arm. Although effective for communication, these wearables were more concerned with information transmission rather than Braille translation.

Systems such as the one Paul Blenkhorn[4] suggested used optical scanning with CCD arrays and microlenses to record embossed Braille text using character recognition algorithms such as Binary Code Braille Cell (BCBC) [4]. Despite their accuracy, these systems were less widely available due to their requirement for specialized imaging equipment and real-time processing devices like DSPs. Although conventional optical character recognition (OCR) methods have also been investigated, they are not very good at recognizing the fine, embossed details of Braille characters. A more adaptable and software-based strategy is required, as evidenced by the fact that many current models lack real-time interaction features and language support.

Since translating braille characters into machine-readable text is technically challenging due to its tactile, dot-based structure, earlier methods relied on tactile sensors, microlens arrays, as shown in studies by Blenkhorn and Kanjantoe [4][5]. This type of system has specialized components, which limit its portability. Later, using a software approach, image processing techniques such as grayscale conversion, binarization, and segmentation began to emerge for detecting braille dots. Although it is effective, it lacks adaptability across varying image quality and braille standards. Optical character recognition (OCR) tools like Tesseract have shown inconsistent performance with specific Braille patterns.

Author(s)	Model	Contribution	Findings and results	Limitations
Ronneberger et al. [16]	U-Net	Encoder-decoder for image segmentation	Performs well on noisy biomedical data with Dice Coefficient ∼0.92 (in medical imaging)	Highly adaptable for Braille segmentation
Li et al. [29]	BraUNet	Semantic Segmentation with 64 class output	Pixel Level Classification and Accuracy of 95% on clean Braille Dataset	Not Bangla Specific but applicable
Meng et al. [44]	BddNet	Uses Gaussian diffusion in loss function	Dot-level localization with 91.2% F1 score (on BCR test dataset), Accuracy of 93.8%	Applicable but not trained on Bangla
Simonyan & Zisserman [9]	VGG	Simple deep CNN	Baseline performance is strong because of 92% accuracy on general datasets	Used for character recognition
He et al. [20]	ResNet	Residual learning in deep nets	Solves vanishing gradient issue. 76.4% accuracy for ImageNet (Used as a Backbone in BCR studies)	Robust in diverse datasets
Murthy et al. [32]	Canny Edge + Segmentation	Traditional method for clean images	Poor performance in noisy conditions. For real-world cases accuracy drops to 70%	Limited use in Bangla Braille systems
Google [53]	TTS Engines	Text to speech	Good English output, poor Bangla fluency	Unnatural pronunciation in Bangla which Lacks contextual fluency in Bangla

Table 2.1: Summary table

2.3 Summary of Key Findings

The reviewed literature, as shown in Table 2.1, reveals how technology for Braille Character Recognition (BCR) has changed over the years. The changes were mostly due to making models work in a more diverse dataset. Before CNNs became the gold standard, most Braille segmentation systems relied upon vision tools such as morphological operations, thresholding, and edge detection. Despite performing quite well on a clean dataset, a slight introduction of noise or a distorted document could drastically hamper its performance. Now we have moved on to a more deep learning based approach, more specifically, a convolutional neural network (CNNs). With the introduction of the U-Net architecture, the performance of the BCR system has improved significantly. Leveraging an encoder-decoder structure with skip connections, it outperforms other similar models even in complex and noisy datasets. Primarily, this technology was widely used in medical imaging and document layout analysis. However, with the introduction of its different types of variants like BddNet, it is making its way into the Braille segmentation system.

The U-Net, a CNN architecture, was primarily introduced by Ronneberger et al.[16] for

biomedical image segmentation. The model leverages a U-shaped encoder-decoder with a skip connection. This design allows the system to preserve spatial information and enable accurate localization. The architecture is designed with a 3 X 3 m convolution with ReLU activations. The research also introduced a weighted loss function and an elastic deformations augmentation technique that resulted in the model achieving outstanding results even with a limited dataset. The system was later adopted for Braille segmentation as the dotted structure of Braille has notable similarities with biological texture. The success of the U-Net segmentation architecture, even with minimal, complex, and noisy datasets, justifies its usage for Bangla Braille-to-speech systems.

Li et al.[29] proposed BraUNet, which is a variant of U-Net that was specifically used for Braille segmentation. It uses auxiliary segmentation task, morphological refinement, and a combined Dice-Cross Entropy loss to improve performance on limited data. BraUnet consistently and accurately performs character localization in more complex data, such as handwritten samples. This model was able to achieve better segmentation and detection compared to its counterpart, like U-Net. Results showed that compared to the ground truth, BraU-Net performed 5% better than other models. Because of more complex character patterns, segmentation and localization on Bangla braille will be more difficult compared to English. If added some human elements in character, such as bad or degraded writing, BraU-Net has the potential to become the core of the entire system, as it can adapt to complex data more precisely.

One of the more recent works, BddNet, which is a semantic segmentation based on Gaussian diffusion, was proposed by Meng et al.[44]. This model uses a Gaussian heatmap-based approach. This system utilizes a two-stage framework. Firstly, Braille dot detection (Gaussian-based segmentation) and post-processing to assemble dots in Braille characters. Compared to models like TS-OBR and BraUNet, it performed slightly well (+0.095 F1). The system's semi-automated labeling can accelerate the creation of a dataset. This is important as there aren't many resources for Bengali. In Bangladesh or similar South Asian countries, for capturing Braille documents, low-cost hardware and natural lighting are often used, which can lead to overexposure. The model with Gaussian diffusion can perform well in this kind of dataset.

A different model, VGG, was introduced by Simonyan and Zisserman [9]. It has a simple architecture, which was developed from the ImageNet Large Scale Visual Recognition Challenge. This model enabled the extraction of abstract features while maintaining the complexity to a minimum. VGG is quite effective for image classification, but unlike U-Net, it lacks preservation of spatial information. This can affect VGGs performance in the dense prediction task. Compared to VGG, U-Net [16] can retain more spatial information. For languages like Bangla, accurately localizing dense structure is crucial for the system. In conclusion, U-Net can be a better fit for the Bengali Braille-to-Speech system.

Nowadays, CNNs have become the gold standard for character classification, which offers high accuracy and scalability. Variants of U-Net, like ResNet and VGG, offer more robust feature extraction and have great accuracy over different types of datasets. However, most of its offerings have not yet been shown in underserved languages like Bangla. The lack of localized datasets has made it difficult to improve the existing model to be

trained properly. Lastly, most of the BCR system stops at text extraction. Leveraging the Text-to-Speech (TTS) system to further improve accessibility remains unexplored. Despite the TTS engine working exceptionally well on the English language, serving Bangla has its challenges. These engines cannot deliver natural pronunciation and contextual fluency. In conclusion, the review highlights that leveraging deep learning techniques for Braille segmentation and character classification, combined with a text-to-speech engine, holds strong potential to deliver a fully automated, end-to-end solution. Such a system could significantly enhance accessibility for underserved communities, particularly visually impaired Bengali speakers.

According to [49] at present only English and Arabic language possess a full Braille image to Voice conversion system. In Bangla there is not sufficient research done with Braille image to voice conversion. Our system aims to address this research gap and provide an end to end solution.

Chapter 3

Requirements, Impacts and Constraints

3.1 Final Specifications and Requirements

A deep learning-based Braille Character to Bangla Voice conversion system is an assistive technology designed specifically for individuals with visual impairments. Its sole purpose is to classify Braille characters with the help of computer vision and deep learning. Afterwards, it then converts into spoken Bangla using advanced text-to-speech tools. The system, with its final design, is built to emphasize accuracy, user accessibility, and hardware efficiency, ensuring real-time performance.

The system starts with the acquisition of images, where Braille text is captured with the help of a high-definition camera. Then, this image undergoes various preprocessing techniques such as conversion of the image to grayscale for data simplification, using Gaussian blur or median filtering to remove noise in images, and lastly, followed by image binarization to highlight the Braille dots much more clearly. These preprocessing technique ensures that the deep learning model is being given a clean and uniform input [22].

To recognize the characters, the system makes use of a Convolutional Neural Network (CNN) model. The CNN processes the arrangements of dots and the spaces between them and then classifies them into corresponding Bangla characters. For spatial feature extraction tasks, especially in image recognition, CNN proved to be highly effective [14]. The output given by the CNN model is then passed into a text-to-speech engine such as Mozilla TTS, Tacotron 2, or Google TTS. This text-to-speech takes input in the form of text and produces a Bangla Voice output in a clear and natural-sounding voice [25].

To minimize the need for visual feedback, the system allows a user to interact through voice prompts. This interface ensures that users, even without technical knowledge, can still operate this system with ease.

For non-functional requirements, the system aims to maintain an accuracy of 94% or higher under standard lighting and input conditions. The system is designed to respond within 3 seconds in order to make it viable for real-time applications. To ensure portability, the system can be integrated with hardware like Raspberry Pi so that it is accessible

even in low-resource settings [10]. For the software side, the system can be made using Python for the backend. Some module packages, such as Unet for image-processing, TensorFlow, or Keras, for building models as well as for training and inference. For the interface, it can be built using PyQt5 or any lightweight web-based framework.

3.2 Societal Impact

This deep learning translation system can significantly improve the lives of visually impaired individuals in Bangladesh and similar regions. It fills up the communication gap between tactile Braille script and voice communications, which eliminates significant obstacles to education, information access, and social interactions.

One of the most significant societal benefits of using this system is in the education sector. Visually impaired students frequently face constraints owing to the lack of Braille textbooks and rely on other individuals for reading purposes. This system enables visually impaired individuals to freely access written resources in Bangla, which improves literacy and learning outcomes. It also empowers users by increasing their sense of independence and lowering their need for assistants or caregivers. This sense of independence boosts self-esteem and opens up broader social and employment options [38].

In addition, the system will promote inclusion of the visually challenged in the digital world. It favors the notion of equal opportunity because it provides them with a technical solution to their needs. Community organizations, schools, and rehabilitation facilities can adopt this approach as a component of an inclusive education and empowerment program.

However, the system's influence may be constrained by economics and technical differences, especially in remote or impoverished areas where they lack access to smart gadgets and internet connectivity, which could hinder adoption. In addition, users who are unfamiliar with modern equipment may oppose switching to a technology-based solution.

3.3 Environmental Impact

This deep learning system is mostly software-centric and low-power hardware, and thus, its impact on the environment is relatively low. Nevertheless, we must consider its positive and negative environmental factors.

If we consider its positive side, the system can make use of digital versions instead of paper-printed Braille and thus reduce the need for printing-related resources. This digital transition contributes to the conservation of raw materials, especially in the making of Braille papers, alleviating the environmental burden that is linked with Braille production and distribution [34]. In addition, the use of energy-efficient devices such as Raspberry Pi alleviates power consumption, making it ideal for sustainable deployment.

However, the accumulation of obsolete or damaged hardware components can contribute to electronic waste if deployed on a large scale. To mitigate these environmental concerns, hardware components such as cameras and microcontrollers can be made from recyclable materials and receive frequent software updates to increase device lifespan, rather than requiring frequent hardware component replacements. In addition, an awareness program related to e-waste disposal can be integrated into the system's user training program [3].

3.4 Ethical Issues

The key ethical issues in this project are the privacy of data, equity, and ensuring that the machine learning model is used responsibly. We are obliged to obtain appropriate consent on any Braille pictures that are utilized in training or testing, and we ensure that such information is protected. In addition, biases in the data sets need to be taken with caution so that some of the styles of Braille or the conditions of the documents will not be worse off.
The system is not intended to be a substitute of professional human support, but is to be seen as an assistive tool. There is a need to make clear what the system is capable of and incapable of to prevent abuse or overreliance by the people on it.

3.5 Standards

This layout is based on principles of general accessibility and assistive technology, such as those of the Web Content Accessibility Guidelines (WCAG). With the standard Braille encodings along with the Unicode text, we are making sure that the software would be compatible with the current digital systems and tools.

3.6 Project Management Plan

This thesis will follow an Agile Methodology, which is an iterative development, continuous feedback, and flexible adaptation to changing requirements. This thesis is being divided into multiple iterations, where each iteration brings specific goals and outcomes. These iterations are being ensured so that all aspects of the system, starting from system design to deployment, are being addressed in a structured manner.

In the first iteration, extensive research and evaluation of user requirements, current solutions, and technology constraints are conducted. In addition, stakeholder interviews and a thorough literature study are also part of this iteration. The system architecture and development process are finalized during the following stage, known as the design and planning phase.

For the next iteration, it is being allocated for data collection and preparation. This includes compiling datasets of Braille images, using preprocessing methods, and setting them up for model training. Afterwards, a deep learning model will be created, trained,

and verified as well.

For the next iteration, integration of a Bangla text-to-speech engine is being allocated. Afterwards, we will compare the results of each model and identify areas where the system can be further improved.

3.7 Risk Management

When adopting this system, certain risks must be effectively managed. For that, we created a structured risk management approach that helps detect, assess, and minimize possible obstacles that could hinder the adoption of this system. The table in Table 3.1 shows a risk matrix summarizing possible risks.

Risk	**Impact**	**Probability**	**Mitigation Strategy**
Poor Image Quality	High	Medium	Utilization of adaptive preprocessing techniques and high-resolution cameras
Model Overfitting	Medium	Medium	Apply data augmentation and cross-validation
Limited Bangla TTS Resources	High	Medium	Make use of open-source Bangla TTS or cloud APIs as backup
Hardware Incompatibility	Medium	Low	Prototype with multiple hardware platforms
User Acceptance	Medium	High	Include user feedback during testing and provide training materials
Time Overrun	High	Medium	Follow strict timeframes, and prioritize critical features first

Table 3.1: Risk Management Table

3.8 Economic Analysis

The suggested system is rather cost-efficient as it primarily consumes open-source software and off-the-shelf hardware. It can operate on the standard computers after the models are trained and does not require special equipment. The suitability of that environment is that it is applicable to schools, NGOs, and other low-resource areas. The accessibility and inclusion benefits greatly in the long term far exceed the comparatively low development and deployment costs.

Chapter 4

Proposed Methodology

4.1 Design Process or Methodology Overview

The proposed methodology for the deep learning-based Bangla Braille-to-Voice conversion system follows a unified, end-to-end pipeline. The system is designed to address the inherent challenges of Braille recognition, specifically the precision required for character-level segmentation and the linguistic grammatical complexity of the Bengali language. The system integrates three core domains of Artificial Intelligence, which are computer vision for spatial localization, standard-compliant synthetic data generation for model robustification, and Large Language Models (LLMs) for semantic reconstruction. As mentioned in the system architecture, the framework flows from raw image input to a single-stage object detector(YOLO), followed by spatial localization of characters and linguistic grammar check. As the main hurdle of implementing the system was to get a proper training dataset, we implemented a standard-compliant synthetic data generation for model robustification, and Large Language Models (LLMs) for semantic reconstruction. To maintain physical fidelity, the generation process adheres to the Library of Congress (LOC) Specification 800. A critical technical conversion factor was established to bridge the gap between physical tactile dimensions and digital computer vision tensors.

By applying this constant of 1 mm = 3.7795275591 pixels, the dataset models standard braille proportions with high precision:

- **Dot Diameter:** 1.5mm
- **Internal Dot Spacing:** 2.5mm

To mimic real-world focus and orientation challenges, the system applies stochastic geometric transformations. Perspective warping is achieved through a transformation matrix M, simulating camera tilt [26]. This ensures that while the data is synthetic, its spatial distribution and geometric distortions represent the variance found in mobile-captured Braille imagery [49].

4.2 Preliminary Design or Design (Model) Specification

Developing of Bangla Braille to Voice conversion using Deep Learning method, numerous alternative designs were also tried and tested. We implemented two distinct architectural paradigms. To begin with, we tested Multi-Staged Pipeline based on semantic segmentation along with isolated character classification (ResNet/VGG/MobileNet) after applying U-Net. Secondly, a Unified Object Detection Architecture that uses YOLO to detect multiple objects at the same time. Semantic and grammatical reconstruction is performed with the help of LLMs along with localization and recognition. The models, architecture and algorithms used are described in this section. to apply and contrast such methodologies.

4.2.1 U-Net

The U-Net architecture was initially proposed [16] for biomedical image segmentation. Since then, it has performed exceptionally across different domains and has become the de facto standard for pixel-wise localization. Key reasons behind choosing this particular architecture are as follows:

Precise Localization: This architecture can select relevant features from any type of high-resolution images. Braille dots are comparatively smaller and the model must not only detect its position but also map its boundary with absolute precision.

Context preservation: One of the novelties of this architecture is to use skip connections to preserve information from its previous layers. This allows it to combine abstract contextual features and precise spatial features to generate a better segmentation mask.

Efficiency on Smaller and noisy datasets: Unlike the general Deep Learning Model, which requires large amounts of data to perform well, U-Net architecture shows excellent performance on relatively smaller dataset. This is crucial as there is no extensive collection of Bangla Braille data.

Computationally Less Demanding: U-Net models are typically smaller compared to models like VGG. Thus, they can be easily deployed and operated on end devices.

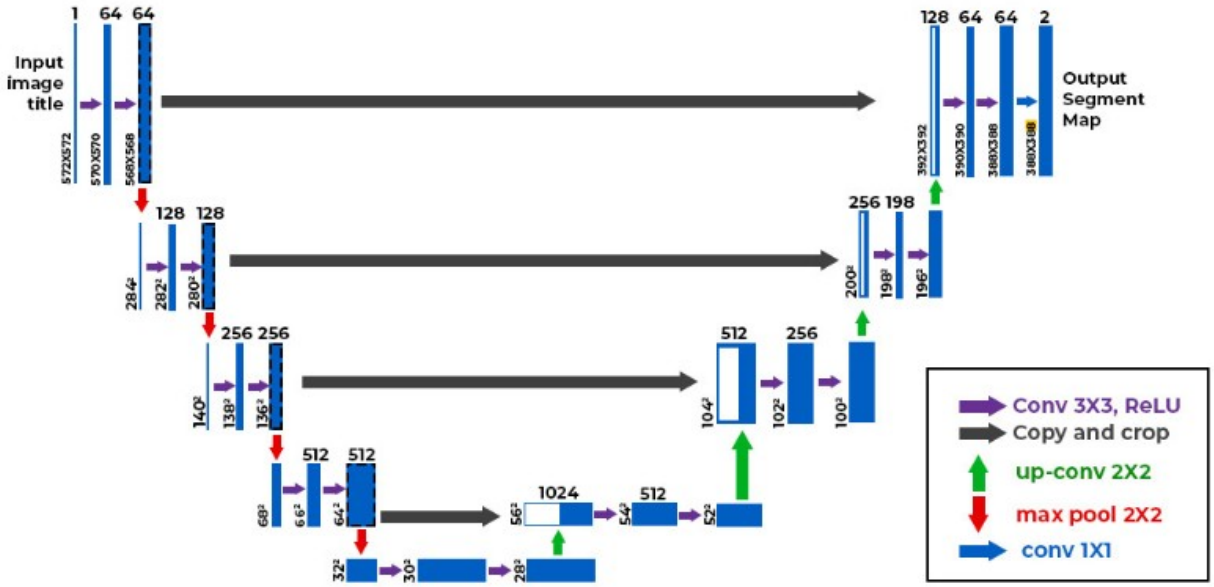

Figure 4.1: UNET architecture [51]

The implemented model for this system follows a classical U-Net structure. This follows a symmetric U-shaped structure as shown in Figure 4.1. It consists of a similar number of encoders to capture the context, a bottleneck, and an expanding path which is known as decoder.

- **Encoder:** The encoder follows the typical structure of a Convolutional Neural Network. It downsamples the input image, consequently increasing the number of feature channels. This block consists of:

  1. Two 3 x 3 Convolutional Layers, and each of them is followed by a Rectified Linear Unit (ReLU) activation function. Through each block, the number of filters doubles ($16 \rightarrow 32 \rightarrow 64$)
  2. The convolution blocks are each followed by a 2 x 2 max Pooling layer. This filter cuts the dimension in half.

- **Bottleneck:** At the very bottom of the U-Net architecture lies a bottleneck layer. This layer is composed of two 3 x 3 x 256 convolution layers with a ReLU activation function. This layer is responsible for capturing most complex contextual information from input.

- **Decoder:** The role of a decoder is opposite to that of the encoder. It upsamples the image to get back to its original dimension while refining the segmentation boundaries. To achieve this, it performs the following operation:

  1. The Upsample layer doubles the spatial dimension of the feature maps.

2. It concatenates information feature maps from its previous stage and its corresponding encoder (comes via skip connection).
3. With each upsampled block, followed by ReLU activation, the feature maps segmentation and cuts the number of filtered in half.

4.2.2 ResNet-50

ResNet-50 is based on a deep residual learning framework that allows for the training of very deep networks with hundreds of layers. Residual blocks are a set of layers that allow for the preservation of information from earlier layers, which helps the network to learn better representations of the input data [47]. Each residual block consists of two convolutional layers, each followed by a batch normalization layer and a rectified linear unit (ReLU) activation function. The output of the second convolutional layer is then added to the input of the residual block, which is then passed through another ReLU activation function. The output of the residual block is then passed on to the next block [47]. ResNet (Residual Network) is a particularly effective choice for our work because of several key reasons:

Addressing Vanishing Gradient Problem: ResNet utilizes residual learning, which helps in mitigating the vanishing gradient problem that can occur in very deep networks. This is crucial for deep learning tasks like image segmentation and classification, as it allows the network to remain trainable even as it grows deeper, making it well-suited for the complex Braille character recognition that we are working on.

Superior Feature Extraction: ResNet has been shown to excel in feature extraction across various types of datasets, including noisy or distorted images. The challenges we faced in our work with noisy or poorly aligned Braille images, ResNet's ability to capture relevant features without degradation of performance is highly beneficial.

ResNet's robustness, ability to handle deep networks, and proven track record in feature extraction and classification make it an ideal choice for the deep learning model in our Braille-to-Bangla voice conversion system.

4.2.3 VGG16

Overview
Simonyan and Zisserman presented VGG16, one of their groundbreaking work and the most significant CNN architectures in computer vision [9]. It performed exceptionally well in tests involving localization and classification since it was created as a component of the ImageNet Large Scale Visual Recognition Challenge (ILSVRC). Its design philosophy is based on the idea that higher classification and feature extraction accuracy can be achieved by using depth and simplicity, that is, by stacking tiny convolutional filters much deeper.

Architecture Design
VGG16 has 16 weight layers, out of which 13 are convolutional layers and 3 are fully connected layers. The structure of this architecture is repeating and consistent:
Convolutional Layers:

- A 3x3 kernel with a stride of 1 and padding of 1 is used for each convolution [9].
- Collects fine-grained spatial resolutions of feature maps across layers.
- Small and numerous stacked 3x3 convolution filters are chosen to increase the effectiveness of the network by providing the same receptive field as larger filters, like a 7x7 filter, but with fewer parameters and more non-linearities.

Pooling Layers:

- For every few convolutional blocks, a 2x2 max-pooling operation with a stride of 2 is applied.
- This reduces the spatial resolution of feature maps in a progressive manner while keeping the most important activations to reduce computational costs [14].

Fully Connected Layers:

- The feature maps are being flattened and passed through three fully connected (FC) layers after going through convolutional and pooling layers.
- The first two FC layers have 4096 nodes, and the last FC layer has 1000 nodes (corresponds to ImageNet classification) [9].
- For Braille classification, the final layer can be replaced with a smaller output layer for a particular dataset.

Activation Functions:
An activation function Relu is used in all hidden layers, which adds non-linearity without bringing vanishing gradient problems that may arise from sigmoid or tanh activations [14]

Softmax Classifier:
The final FC layer uses a softmax function to output probabilities across classes and can produce categorical predictions.

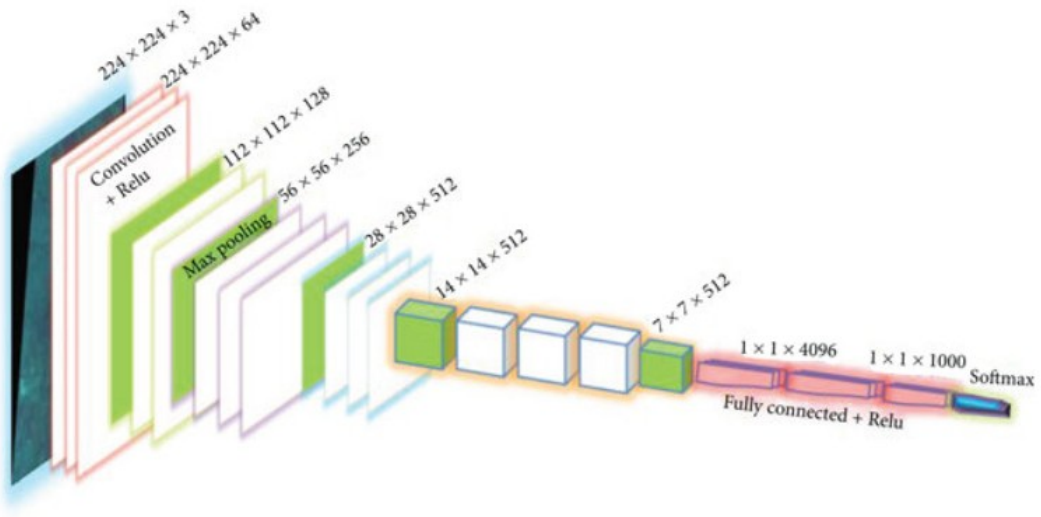

Figure 4.2: VGG16 Architecture [9]

Operations and Workflow:

VGG16 uses a series of fully linked, pooling, activation, and convolution layers to process an image. After passing through 13 convolutional layers and three fully connected layers by taking an input of an RGB image with 224 x 224 pixels, it outputs a probability distribution across target classes using a softmax classifier. Full layers of this model is shown in Figure 4.2 which was originally proposed by Simonyan and Zisserman [9]:

Step 1: Preprocessing Input

At first, the input photos' width, height, and RGB channels are scaled to 224x224x3. Then, its pixel values are normalized by mean subtraction and scaling to increase numerical scaling during training [9][7]

Step 2: Convolutional Blocks

VGG16 is structured into five convolutional blocks followed by pooling layers. All convolutions use 3x3 kernels with a stride of 1 and padding of 1. This ensures fine-grained information by maintaining the spatial dimensions inside the block constant. In addition, stacking several 3x3 convolutions together produces bigger receptive fields with fewer parameters (for example, two stacked 3x3 convolutions give 5x5, and three 3x3 stacked convolutions give 7x7).

Block-wise Workflow:

- **Block 1**: Two convolutional layers with 64 filters each → ReLU → Max Pool (224×224 → 112×112).

- **Block 2**: Two convolutional layers with 128 filters each → ReLU → Max Pool (112×112 → 56×56).
- **Block 3**: Three convolutional layers with 256 filters each → ReLU → Max Pool (56×56 → 28×28).
- **Block 4**: Three convolutional layers with 512 filters each → ReLU → Max Pool (28×28 → 14×14).
- **Block 5**: Three convolutional layers with 512 filters each → ReLU → Max Pool (14×14 → 7×7).

At each stage, feature maps get smaller and deeper since it has more filters. Early blocks learns low-level features such as edges and textures while later blocks learns high-level features such as shapes, objects, and dot size in our case, dot patterns in braille cells [14][7].

Step 3: Activation (ReLU):

Rectified Linear Unit (ReLu) is an activation function used by VGG16. This adds non-linearity and mitigates vanishing gradient issues. The ReLu function is mathematically defined as:

$$f(x) = \max(0, x) \tag{4.1}$$

where x is the input of the activation function and $f(x)$ is the output. This function ensures that negative values are converted to zero while positive value stays same as shown in Equation (4.1). This allows for quicker training and efficient feature learning [14].

Step 4: Pooling Layers

After passing each block of convolution layers, a 2x2 max pooling of stride 2 is used in this step. In this step, spatial resolution is being cut in half (e.g., 224×224 → 112×112). This ensures that it retains the most important activations and achieves translation invariances.

Step 5: Fully Connected Layers

After passing through the fifth pooling layer, the size of the feature map is 7x7x512 = 25088. These are being flattened into a 1D vector and fed into fully connected layers.

- FC1: 4096 neurons, Relu activation.
- FC2: 4096 neurons, Relu activation.
- FC3 (Output layer): Number of neurons is equal to the number of classes. This generates a probability distribution function across classes using a softmax function and ensures that outputs are normalized probabilities [9].

$$P(y_i) = \frac{e^{z_i}}{\sum_j e^{z_j}} \tag{4.2}$$

where $P(y_i)$ is the predicted probability of the i-th class, K is the total number of output classes, and z_i is the raw score (logit) for class i. From Equation (4.2), the summation of all class probabilities is equal to one, and the class with the highest $P(y_i)$ is selected as the final prediction[9][35].

Step 6: Dropout Regularization

To minimize overfitting, VGG16 adds dropout layers (usually 50%) between fully connected layers. To encourage the network to learn better, dropout randomly deactivates during training [7].

Step 7: Training Process

- Loss function: Cross-entropy loss is applied for classification.
- Optimizer: Adam in contemporary adaptation or gradient descent with momentum
- Backpropagation: Weights are calculated and updated using gradients layer by layer.

Workflow Summary:

1. The first few layers (Blocks 1-2) learn about edges, corners, and textures.
2. Mid layers (Block 3-4) acquire shapes and patterns
3. Deep layer (Block 5 + FC) learn abstract level features such as braille dot arrangements
4. Classifier (Softmax) maps extracted features into probabilities of classes

All in all, VGG16 helps in detecting the presence of braille dots in early layers, combining them into patterns in middle layers, and maps into a corresponding letter based on probability and outputs in the form of 6 6-bit binary values (as each braille consists of 6 dots, and each dot is represented as a bit).

Advantages of VGG16

1. Simplicity and Uniformity: It is easy to implement and extend because of its consistent use of 3x3 filters
2. High Baseline Accuracy: VGG16 has proved that depth is crucial for performance and sets a benchmark for many future CNN models [7].
3. Transfer Learning Potential: As braille datasets are limited, a pretrained VGG16 models (ImageNet) are utilized for finetuning on smaller datasets [6].

Limitations of VGG16

1. Computational Cost: VGG16 requires more memory and trains more slowly compared to ResNet because it has approximately 138 million parameters[19].

2. Overfitting on Small Datasets: If data augmentation is not applied, VGG16's huge number of parameters is easily affected by overfitting [31].

3. Absence of Residual Learning: VGG16 is unable to train deep networks as effectively as ResNet due to its inability to handle vanishing gradients [19].

4.2.4 MobileNetV3

A lightweight convolutional neural network (CNN), MobileNetV3-Small, is designed for mobile and edge devices, introduced by Google in 2019. It is an improvement over MobileNetV1 and V2 by combining depthwise separable convolution for efficiency, inverted residual blocks from MobileNetV2, squeeze-excitation module that recalibrates channel features as shown in Figure 4.3. The reason why MobileNetV3-Small is special is the capability to run on CPU and DSP and not relying on heavy GPU. It can balance accuracy vs speed better than MobileNetV2 and makes it useful for tasks like image classification, object detection, segmentation etc. For our purposes, this model is able to detect a braille character and predict its corresponding Unicode value.

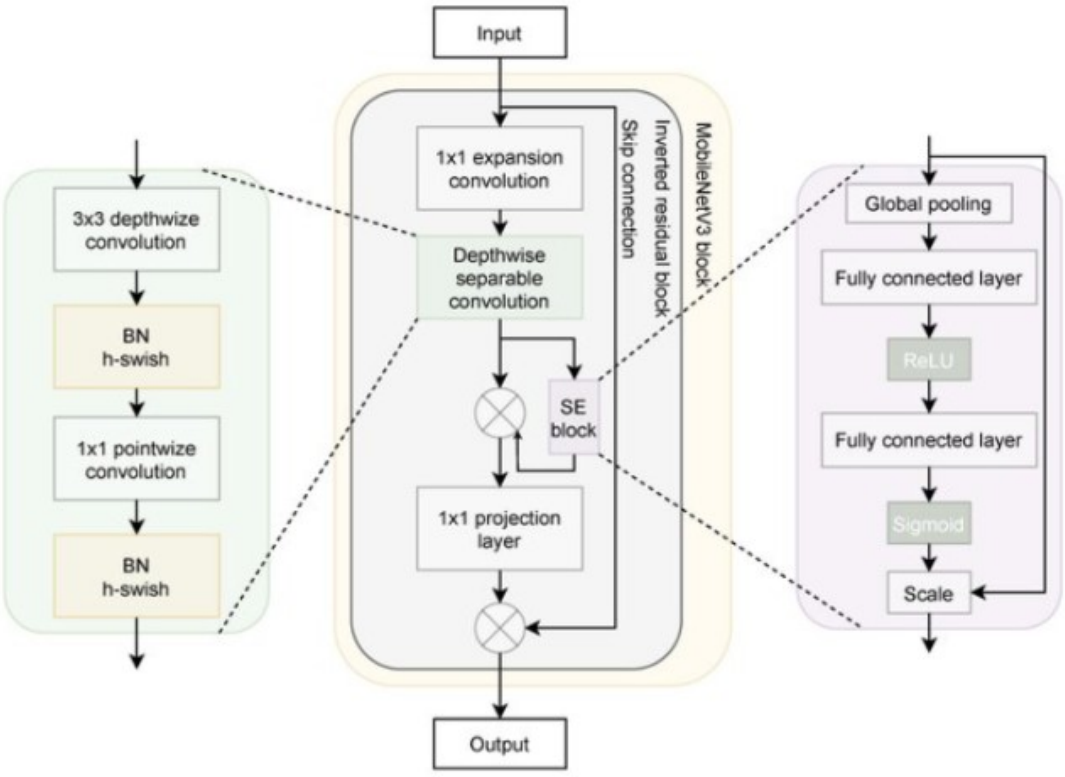

Figure 4.3: Structure of MobileNetV3 blocks and components [30]

How MobileNetV3-Small works for Braille Classification

1. Input and Preprocessing

Braille Classification begins with a segmented Braille Character obtained from the results of the U-Net. For each segmented image, it is usually grayscale and resized to a fixed resolution suitable for MobileNetV3-Small (224x224 pixels). This uniform resizing enables the network to process all inputs consistently. Before training the model, the neural network needs to learn subtle variations in Braille Dot shapes and spacing, which can be achieved by normalizing pixel values.

2. Feature Extraction with Convolution and Inverted Residual Blocks

Using a normal convolution operation, low-level features like Braille dot edges, borders, and simple textures are extracted. Afterwards, inverted residual blocks handle the majority of the feature extraction. Inverted residual blocks are brought from MobileNetV2. A 1×1 convolution is used to extend the input channels in each block, and then a depthwise convolution is applied to each channel separately. Thus need of calculations is not required, which enables the model to effectively present spatial arrangements of Braille dots. Lastly, using a 1x1 projection layer, the channels are being shrunk to a smaller size.

3. Channel Attention with Squeeze-and-Excitation (SE) Modules

Squeeze-and-Excitation (SE) modules are integrated into inverted residual blocks of MobileNetV3-Small. This module is beneficial in Braille Classification because background noise is minimized, allowing the model to focus on the most crucial aspects of Braille Patterns. Using global average pooling, the SE mechanism squeezes spatial information into a single vector, which is then passed through two connected layers. The results are the collection of weights that focus on important channels. By highlighting the raised dots regions over the flat background, the classification accuracy of Braille characters can be increased.

4. Non-Linear Activations: ReLU and Hard-Swish

Two activation functions, ReLU from Equation (4.1) and hard-swish, are used in this model to boost the network's representational capability while maintaining efficiency. In previous layers, ReLU is utilized to rapidly extract basic visual cues such boundaries and dot intensity. Hard-swish provides a smooth non-linearity in capturing intricate interactions between Braille Dots. This takes the role of a more computationally costly swish activation in later layers. This combination enables the model to be optimized on a mobile system while maintaining high accuracy.

5. Global Feature Aggregation and Classification Head

Using global average pooling, feature maps that are produced after going through multiple stages of feature extraction and channel recalibrations are combined. In this stage, the complete Braille pattern is summarized by condensing spatial information into a compact feature vector. Then, the vector is run through a fully connected classifier, which is

assigned to one of the potential braille classes such as letters, numbers, and symbols. Each class receives a probability score from the softmax layer, and the one with the highest probability is chosen as the recognised Braille Character.

4.2.5 YOLOv8/v11

With the introduction of "You Only Look Once" (YOLO) architecture, the paradigm shifted more towards computer vision from multi-staged detectors. This system utilizes separate region proposal networks within a unified, single-staged framework. With the previous methodology, the system heavily relied on segmentation (U-Net) and isolated classification (VGG16/ResNet). This made errors propagate easily. On the other hand, YOLO treats object detection as a single regression problem, mapping image pixels directly to a specific bounding box coordinate and class probabilities [24]. This system specifically utilized YOLOv8 [40] and the latest YOLOv11 framework. Leveraging these models resulted in high-speed inference and spatial precision.

1. Feature Extraction:

    - YOLOv8 (C2f Module): YOLOv8 leverages the Cross-stage Partial bottleneck with two convolutions (C2f) module as shown in Figure 4.4, which enhanced its gradient flow and resulted in richer feature representation by merging high-level feature with contextual information [41].

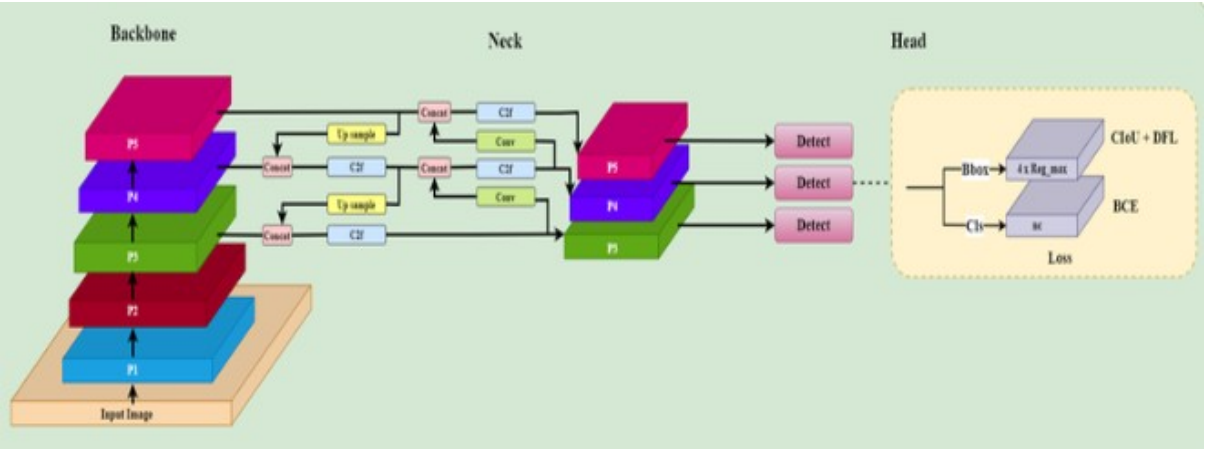

Figure 4.4: Yolov8 Architecture [39]

    - YOLOv11 (C3K2 and C2PSA): Based on Figure 4.5, YOLOv11 further optimized feature extraction by introducing a more advanced C3k2 from the previous C2f. It also leverages C3PSA, which has a self-attention mechanism to create better awareness of the Region of Interest. For this braille system, PSA is critical as it allows the model to understand spatial relation between dots within a cell, which resulted in better class separation [46]

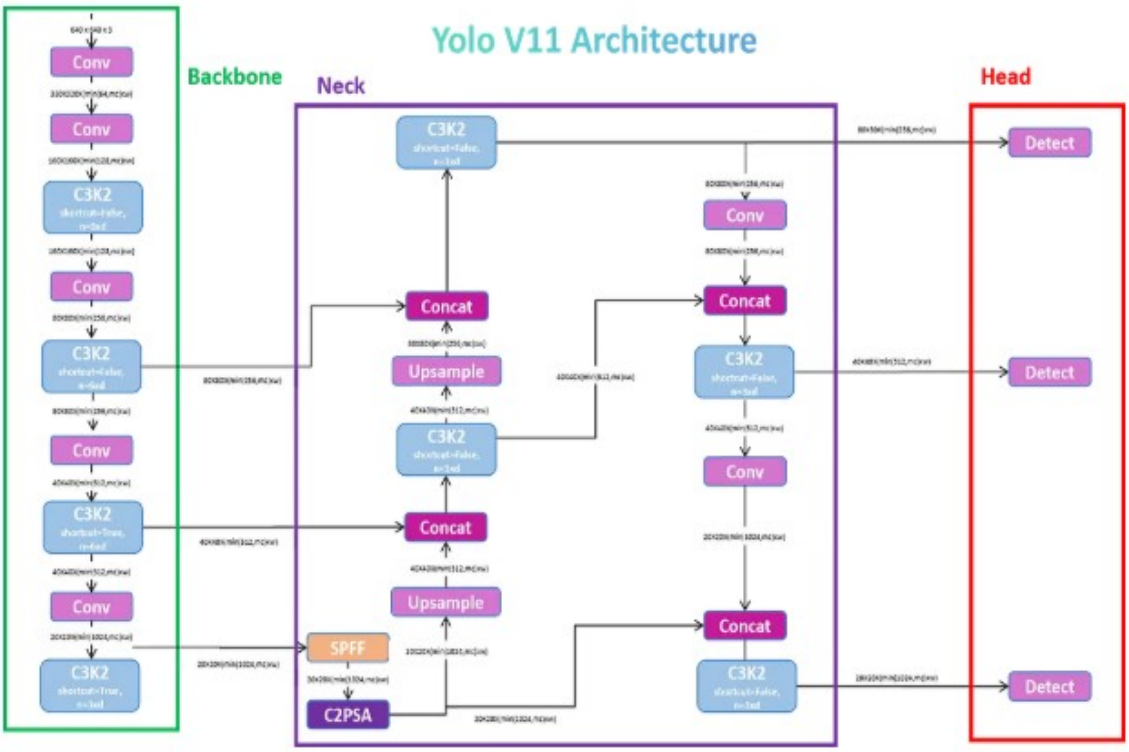

Figure 4.5: Yolov11 Architecture [45]

2. Feature Aggregation: To identify and extract from different resolutions of braille images as shown in Figure 4.6, Path Aggregation Network (PAN-FPN) plays an important role to enhance the flow of information. It can capture low-level geometric information (the contour of the braille dots to high-level features (6-bit binary patterns) [41].

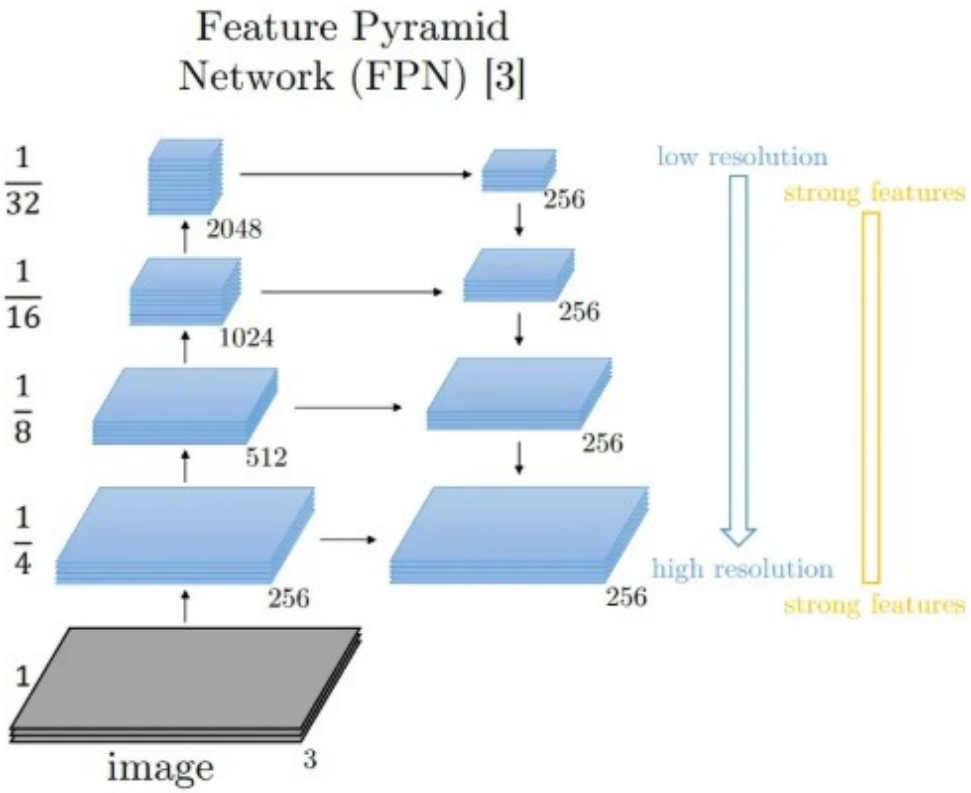

Figure 4.6: Feature Pyramid Network (FPN) [50]

3. Prediction Logic:

    - Decoupled Head: YOLO Leverages a decoupled head that separates localization and classification. This separation significantly reduces conflicts in training and improves mAP (Mean Average Precision) for smaller objects [43].
    - Anchor-Free Detection: YOLOv8 and v11 can predict the braille cell directly rather than relying on a predefined box offset. These anchor-free capabilities are beneficial for different alignments found in mobile-captured Braille images.

Operation and Workflow:
As we are leveraging synthetic data, the operational workflow for the YOLO pipeline was tested with images in different resolutions (640 * 640 pixels, 248 * 248 pixels). The steps that the workflow followed:

Step 1: Standardized input image processing

The input image was resized to accommodate the 1 mm = 3.7795 px scale factor. YOLO uses Mosaic Augmentation during training, mixing four different images to train the model to recognize cells in different contexts and scales.

Step 2: Feature Extraction

As the image passes through different modules (C2f / C3K2), the spatial dimension is downsampled (eg, 1024 → 512 → 256). At each layer the model extracts different types of features. During the Early layer, it mostly extracts specific edges, boundaries of dots, shadows. The deeper layer starts to recognize the dots into a 2 * 3 grid patterns that forms a complete braille cell [49].

Step 3: Spatial Pooling - Fast (SPPF)

At the bottleneck layer, the SPPF pools are flattened into a fixed-sized feature map. This process makes sure braille cells can be recognized independently of how close the camera was when taking the pictures.

Step 4: Bounding Box Regression and Classification:

The decouple head has two different loss functions for the two parts:

1. Complete IoU (CIoU) Loss: Minimizes the distance between predicted box and ground truth
2. Binary Cross-Entropy Loss: Calculates the probability of the probability within all Bengali Braille classes.

Step 5 NMS Suppression:
Since the model has a high probability of predicting overlapping cells, NMS retains only the box with the highest confidence score. This ensures the tactile character is counted only once [24].

Advantages:

1. **Unified Process**: As YOLO leverages a single-stage character extraction, it factors in the probability of error propagation that occurs when a segmentation model like U-Net fails to precisely isolate a cell for a classifier [49].
2. **Object Sensitivity**: The utilization of attention mechanism (PSA) in YOLOv11 performs exceptionally in distinguishing 6-bit patterns within a very confined space (2.5 mm).
3. **Hardware Efficiency**: YOLOv8/v11 models possess high-speed inference ($<$ 10ms per image) that makes them suitable for deployment on different platforms like Raspberry Pi and mobile [42].

Limitation:

1. **Resolution Dependency**: The performance of YOLO is highly dependent on the resolution of the image. Lower resolution leads to "feature vanishing" which causes performance to drop.
2. **Training**: To achieve a high mAP on all different classes. Requires significant GPU power and high training epochs (>100).

4.2.6 Google Text-To-Speech (GTTS)

Google Text-To-Speech is a speech synthesis tool that is an example of a neural speech synthesis created by Google to create sound that is natural and mostly used in accessibility systems, virtual assisstants and assistive applications. It is mostly made with the help of the lightweight gTTS Python library or the more advanced Google Cloud Text-to-Speech API, which handles Bangali with definite pronunciation of vowels, consonant groups, as well as natural sentence rhythm. Google TTS is especially well-suited to less-precise pipelines, like Braille-to-Bangla voice conversion pipelines, since it has the highest intelligibility, is significantly faster than other operations, and can be easily fitted anywhere without special hardware. Nonetheless, it is also dependent on an internet connection, offers relatively low levels of voice variation in Bangali, and does not allow as much control over prosody and expressiveness as high-quality neural TTS applications, which are also details worth considering in the context of academic consideration and further advancement.

4.3 Data Collection

4.3.1 Synthetic Data Approach

The Braille text-to-speech system is heavily based on data for initial training, as it requires extracting text from a given image. As it requires a deep learning approach, the quality of its performance is heavily based on the quality of the data. The performance of a deep learning model is fundamentally limited by the quality of its training data. In this project, the initial objective for training images was to produce their corresponding binary segmentation masks, where each pixel is labeled as 'Braille dot' or 'background'. The data collection phase of the research evolved from different methodological paradigms. To support the preliminary stage of development, unlabeled braille data were acquired from open-source repositories. However, these datasets hardly provided the necessary volume that would provide enough diversity to train high-capacity deep learning methods. As the system paradigm moved towards a more unified object detection and contextual reconstruction framework, the necessity of high-volume labeled data became absolute.

As earlier stated, the physical structure of Braille is strictly governed by specific geometric standards which cover fixed dot diameters and inter-cell spacing. Given the strict dimensional requirements, this research adopted a Synthetic Data Generation (SDG) strategy. The mathematical consistency of Braille script allows the creation of synthetic datasets that accurately generalize the real-world condition through Domain Randomization. While training the YOLO, the necessity of the semantics or logic of a braille sentence is just as relevant as the objective is just to extract a 2 * 3 dimensions. With variable text length, huge sets of labeled data were generated from real-world data. The following chapter covers a detailed explanation of the image preprocessing of both datasets.

4.3.2 Data Cleaning (Traditional HITL Approach):

The Braille text-to-speech system is heavily based on data for initial training, as it requires the extracting text from a given image. As it requires a deep learning approach, the quality of its performance is heavily based on the quality of the data. The author

[35] states that the performance of a deep learning model is fundamentally limited by the quality of its training data. In this project, the initial objective for training images was to produce their corresponding binary segmentation masks, where each pixel is labeled as 'Braille dot' or 'background'.

The raw images were captured in a controlled environment. However, factors such as variable lighting and paper texture can make a fully automated annotation process unreliable. On the other hand, manually annotating an entire dataset can be significantly time-consuming. Thus, we have leveraged a semi-automated, human-in-the-loop (HITL) methodology to create a balance between efficiency and accuracy. This process was done primarily in two stages.

Firstly, the image went through an automated image processing pipeline, which would automatically generate a mask for a given image. This was achieved by utilizing the OpenCV library [2]. The steps were:

- Contrast enhancement was used to amplify the subtle difference between the dots and the background. Contrast Limited Adaptive Histogram Equalization (CLAHE) was used as it operates based on a small region of an image. This makes it highly effective for local contrast enhancement [1].
- Then, Adaptive Gaussian Thresholding was used to convert to binary. This technique also uses a local region for thresholding, which takes local lighting conditions into account. This makes sure the thresholding works for a wide range of pixel intensity across the entire image.
- The contour-finding algorithm by Suzuki and Abe (1985) was used to identify the outlines of the binarized dot candidates.

This entire pipeline generated a binary mask as shown in Figure 4.7, which was mostly accurate. However, to ensure the integrity of the data, we use Human-in-the-loop verification and correction if needed. In this stage, a human annotator used graphical image editing tools to verify and perform corrections on the mask. Combining automation and human intervention ensured the process was done efficiently and accurately. A critical prerequisite for training an accurate segmentation model is a successful data cleaning process on a reliable dataset [33].

Figure 4.7: Masked Braille Image Sample

4.3.3 Data Transformation

The data cleaning and annotation phase provided a high-resolution image with its corresponding mask. The images then went through a series of transformations that standardized the raw image, which can be used for training the segmentation model. Mainly, three types of transformation were applied: standardization for consistency, normalization for training stability, and augmentation for generalization.

1. **Standardization on the color and contrast domain:**

   (a) **Color Space:** The images in the dataset had different formats in terms of color representation and also dimensionality. To ensure consistency across the images, two standardizations were applied. Firstly, the source images were converted from 3-channel RGB to single-channel greyscale images, as our system just needed a clear distinction between the dot and the background, which can be achieved by just a grey-scale representation.

   (b) **Binary Contrast Conversion:** To ensure consistency across color space, the whole image contrast was converted to black (Braille dots) and white (Background). This step creates a high contrast difference between the braille dot and its corresponding background. This high contrast difference helps the network's initial convolution layers learning process.

Spatial and intensity Standardization:

- **Spatial Normalization:** The image-based deep learning architecture, like U-Net, operates on fixed-size input tensors. Therefore, all images were resized to a uniform dimension of 256×256 pixels.

- **Intensity Normalization:** The original pixel values were represented with an 8-bit integer range. It was from the floating-point range of [0.0, 1.0]. This step is critical, as it ensures the calculated gradient through backpropagation remains stable and prevents issues such as gradient explosion [8].

Data Augmentation: The dataset was extremely limited, thus data augmentation was necessary to improve the model's ability to generalize the unseen data and prevent overfitting. The transformation was applied using Albumentations library, which provides highly optimized functionality for image segmentation-related tasks [28]. The augmentation includes:
Geometric Transformations: The images were randomly rotated by +- 10 degrees and a horizontal flip with 50 % probability.
Photometric Transformations: The brightness and contrast were adjusted randomly, which introduced different lighting conditions. To improve the model's robustness, Gaussian noise was also introduced in the images Final results is shown in Figure 4.8.

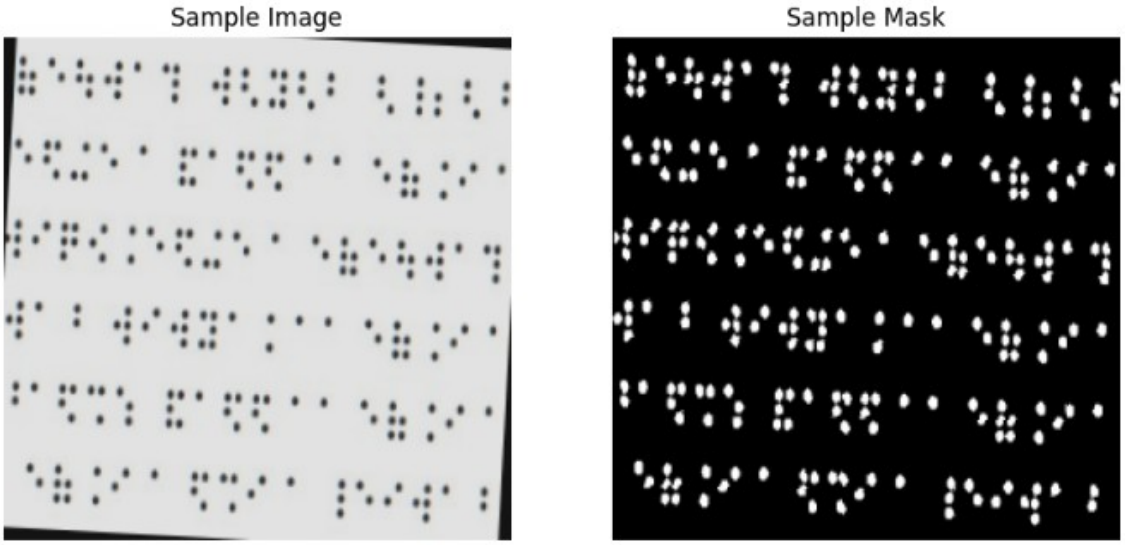

Figure 4.8: Augmented sample image

This comprehensive transformation pipeline helps the models to train in a consistent, stable and diverse set of images. This helps the model to become less prone to overfitting and generalize overall information.

4.3.4 Data Reduction

Data reduction technique is crucial for a model during its training stage. The reduction is done in a manner that allows the system to reduce the number of features while still retaining enough information. As this system deals with image data, we reduce the image dimensionality so the model can focus on more specific information. This method usually helps to reduce the chance of overfitting and faster convergence.

Primarily, data reduction was done in the color space domain, spatial dimensional domain, and contrast binarization.

- **Color Space Domain:** The raw data came in 3-channel RGB format. But for our system to separate the braille dots from the background, this color information is irrelevant. Existence of this excess information increases the number of features by a factor of three. Rather than color, the system needs to focus on features like luminance and contrast that are all preserved in the gray scale. By removing the color information, the model can focus solely on features that are relevant for the task. This is a form of guided feature selection that reduces the risk of learning in spurious correlation.
- **Spatial Dimension Reduction:** The raw images were high resolution and had various dimensions. For deep learning models, having high-resolution images not only increases training complexity but also does not show relevant performance improvement. Secondly, models like UNet operate on similar-sized images. Thus, all images and masks are resized to 256 * 256 pixels. This effectively reduces the amount of information while preserving most of it.
- **Contrast Binarization:** With Adaptive Gaussian thresholding, the variation of pixel intensity was reduced from 256 possible values to only two values: 0 (Background) and 255 (dots). This reduction simplification removes minor variations that are not relevant for the task anyway. This helps the model to learn more general information across the domain rather than learning slight contrast variation.

In summary, the data reduction technique not only reduced the complexity of learning but also forced the model to learn more general features from the image. This makes the models perform effectively and efficiently, while reducing the chance of overfitting.

4.3.5 Summary of Preprocessed Data

This comprehensive data preprocessing pipeline processed the data in different domains in different ways. This was important because structured and optimized data are the most important parameters for the performance and effectiveness of a data-driven process. The key characteristics of the final preprocessed data are as follows:

- Every image in the dataset has a corresponding mask.
- Every image maintains consistent dimensions, including channels
- All the pixel values are represented in 32-bit floating-point data representation. Intensity values of pixels are scaled to the range of [0.0, 1.0]
- The training subset of the data went through the augmentation pipeline, which ensured the model was exposed to a more diverse range of data.

4.4 Implementation of Selected Design

4.4.1 Implementation and Evaluation for U-Net

- **Initialization**: The framework used for implementing this architecture was TensorFlow [17], and it utilizes the Keras API [12]. The input for this model was 256 X 256 grayscale images. For the output layer, a sigmoid activation function was used to generate a single segmentation mask. Therefore each of the pixel values consists of a probability distribution and can be mapped to either a Braille dot or Background. All the internal convolution layers used ReLU [13] activation functions.
- **Training and evaluation:** As this was a segmentation task, for loss function, a combination of Binary Cross-Entropy (BCE) and Dice Loss was used. The dice loss was responsible for measuring the overlap between the ground truth and the predicted mask, and the BCE calculated pixel-wise loss. The composite loss function resulted in achieving a stable gradient as shown in Figure 4.9. Adam optimizer [8] was used as an optimization algorithm that outperformed other algorithms. For primary evaluation metric, dice coefficient was used with variable batch size (8, 16, 32).

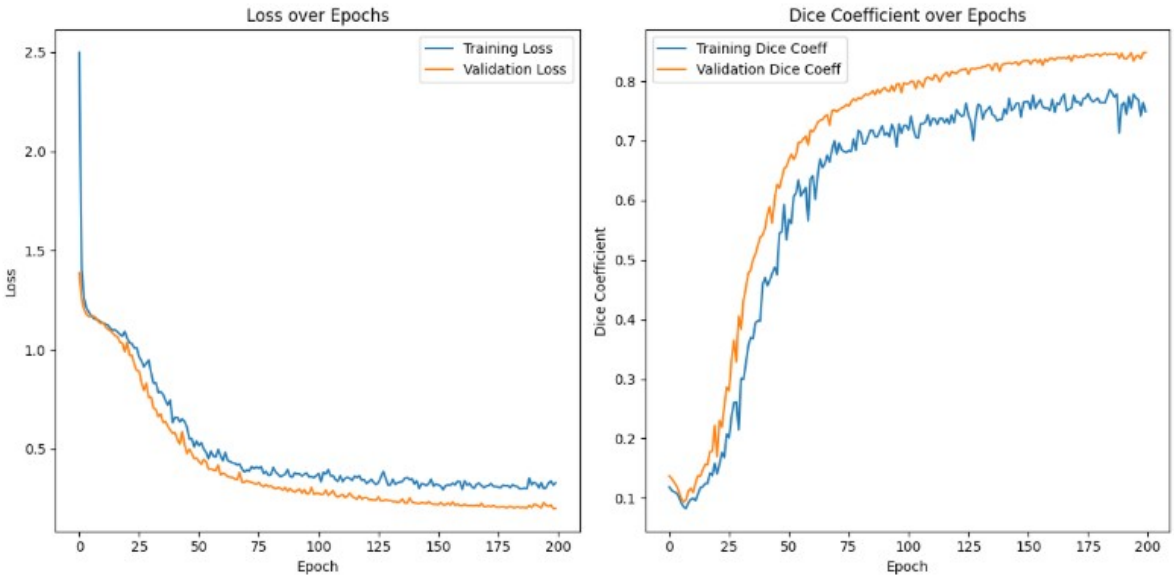

Figure 4.9: Training and Evaluation curve

4.4.2 Implementation and Evaluation for models

After employing various CNN-based models for initial segmentations and classification, the study progresses to leverage these models for feature extraction. The implementation of both our entire pipeline is shown in Algorithm 1 and 2.

Algorithm 2 explains the steps and implementation of the unified YOLO based extraction and LLM-driven sentence conversion.

Algorithm 1 Pipeline Implementing Pseudocode

Require: Raw Braille sheet A
Ensure: Bangla speech output B

for each Braille cell image **do**
    Load Braille image and enhance to make dots visible
    Generate draft mask by detecting contours
end for
for each Braille cell image **do**
    Apply augmentation: crop (corners), zoom out
    Compute Image Quality Metrics (SSIM, PSNR, MSE, RMSE)
    if IQM within acceptable range **then**
        Store and proceed to model training
    else
        Repeat augmentation and IQM computation
    end if
end for
Train and test classification models: VGG16, ResNet50, MobileNetV3-Small
Extract features using deep learning
Classify Braille cells
Apply Global Average Pooling (GAP) to feature maps F and reduce to 1D vector:

$$G = \frac{1}{H \times W} \sum_{i=1}^{H} \sum_{j=1}^{W} F_{i,j}$$

Compute Binary Cross-Entropy loss
Optimize using Adam optimizer
for each word in Braille sequence **do**
    if character mapping is one-to-one **then**
        Spell corresponding Bangla letter
    else if character mapping is one-to-two **then**
        Spell Bangla-specific letters/numbers
    else if character mapping is one-to-many **then**
        Spell Bangla-specific letter and sound
    else
        Spell Bangla conjunction letter
    end if
end for

Algorithm 2 Unified YOLO–LLM Pipeline for Bengali Braille Conversion

Require: Raw Braille image A, standardized scaling factor $\sigma = 3.7795$ px/mm
Ensure: Semantically correct synthesized Bangla speech B

Stage 1: Standardized Data Generation
for each Bengali character c in linguistic corpus **do**
    Map c to 6-bit Braille pattern $p \in \{0, 1\}^6$
    Apply LOC specification (800 dimensions)
        Radius $= 0.75\sigma$, Spacing $= 2.5\sigma$
    Apply data randomization:
        Perspective transform M, Gaussian noise η, blur β
    Generate synthetic tensor S and synchronized YOLO label L
end for

Stage 2: Unified Detection (YOLO)
Train YOLOv8 / YOLOv11 on generated dataset
Perform single-pass inference on image A
Obtain detections:
    $\mathcal{D} = \{(x, y, w, h), \text{Class}(p), \text{Conf}\}$
Apply Non-Maximum Suppression (NMS)
Filter detections using confidence threshold $\tau > 0.25$

Stage 3: Spatial Reconstruction and Stream Generation
Group detections into rows using $\Delta y < 0.5 \times h_{cell}$
for each reconstructed row **do**
    Sort detections by x-coordinate (left-to-right)
    for each adjacent pair (D_i, D_{i+1}) **do**
        Compute gap $G = x_{i+1} - x_i$
        if $G > 1.5 \times w_{cell}$ **then**
            Insert whitespace token into stream $\mathcal{T}$
        end if
        Append Class(p) to stream $\mathcal{T}$
    end for
end for

Stage 4: Linguistic Reconstruction (LLM)
Pass token stream $\mathcal{T}$ to Large Language Model (LLM)
if one-to-many ambiguity detected **then**
    Resolve vowel signs and Kar signs using Bengali linguistic probability
else if prefix cell detected (dots 3–4–5–6 or 4–5–6) **then**
    Interpret subsequent cells as numerics or conjuncts
end if
Correct semantic inconsistencies and generate Unicode text U

Stage 5: Speech Synthesis
Convert text U to speech B using gTTS / eSpeak-NG
return Synthesized audio output B

Chapter 5

Result Analysis

5.1 Performance Evaluation

For the performance evaluation we incorporated a unique way of measuring the end to end system we designed that is WER (Word Error Rate). WER is calculated as shown in equation. It is the primary metric for measuring the accuracy of Automatic Speech Recognition (ASR) systems, representing the percentage of incorrectly predicted words compared to a reference transcript. It calculates the sum of substitutions, deletions, and insertions divided by the total number of words, with lower scores indicating better performance. Our way of measuring the end to end system is the most viable way as we want to feed the Bengali text to the TTS engine which will generate the voice. Table 5.1 gives in depth of what 5 texts are being generated and Table 5.2 shows it corresponding WER value.

$$\text{WER} = \frac{S + D + I}{N} \tag{5.1}$$

Where:

- **S (Substitutions):** Incorrectly replaced words.
- **D (Deletions):** Omitted words.
- **I (Insertions):** Extra words added.
- **N (Total Words):** Total count in the reference transcript.

Sample	Input from YOLO	Ground Truth (Expected)	Llama Output (Detected)
1	আ ম ই ব ই প ড ই ।	আমি বই পড়ি ।	আমি বিপড়িয়েছি।
2	স এ স ্ ল এ য আ । ।	সে স্কুলে যায় ।	সে সলে যায়।
3	আ ম ই ভ আ ত খ আ ই ।	আমি ভাত খাই ।	আমি ভাত খাইয়া গেলাম
4	আ হ আ আ ন ঈ ল ।	আজ আকাশ নীল ।	আজ আআনীল।
5	স এ গ আ ন ও ন এ ।	সে গান গায় ।	সেগানো নেয়।

Table 5.1: First 5 sample texts of YOLO's output detected by Llama3

Average WER of from table 5.1: 2.6136 (Lower is Better)
Average Accuracy from Table 5.1: 29.12% (Higher is Better)

No.	WER	Accuracy Percent
1	0.6667	33.33
2	0.3333	66.67
3	0.6667	33.33
4	0.6667	33.33
5	1.0000	0.00

Table 5.2: WER and Accuracy Percentage Results

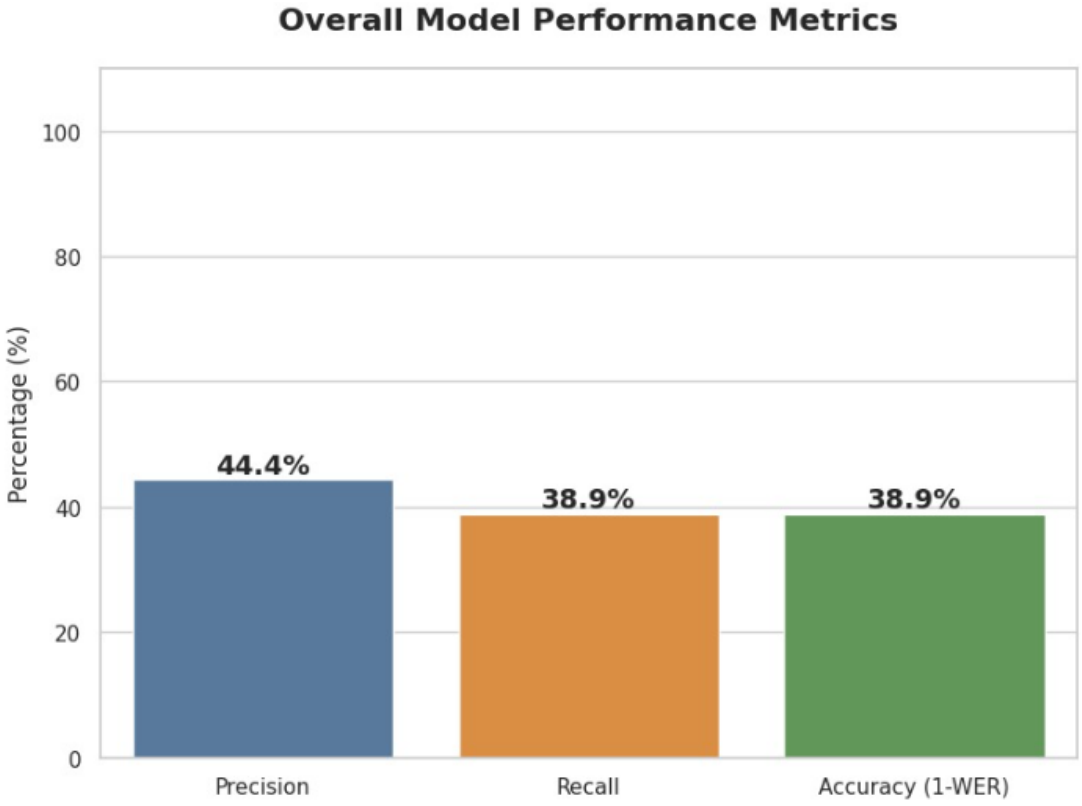

Figure 5.1: Overall Model Performance Metrics for Llama 3

To calculate Precision and Recall (Figure 5.1) for an LLM, we treat the sentence as a ”Bag of Words.” We treat every word the model generates as a ”prediction” and check if it exists in the ”Ground Truth.” The overall performance metrics are based on both mixed sample output from YOLOv8 and YOLOv11s. The sample text is 100 sentences that appear to be the most common in day-to-day Bangla conversation.

5.1.1 Forensic Diagnosis of Llama-3 in Bangla Word and Sentence Generation

1. **The Phenomenology of Character-Level Fragmentation:** The most immediate and visually striking error pattern is the systematic insertion of whitespace delimiters between individual Unicode codepoints. This effectively destroys the word boundaries that are crucial for semantic parsing in Bengali.

   Artifact Analysis: Consider the first line of the ground truth text: আমি বই পড়ি। (Ami boi pori - "I read books"). This sentence consists of three distinct words.

   Structure:[Verb: Pori].

   Phonetic Components: A-mi, Bo-i, Po-ri.

   Now consider the model's output for the corresponding input: আ ম ই ই প ই ।.

   Structure: [A][M][I][I][P][I].

   Comparison: The word আমি (Ami) has been broken into আ ম ই (A M I). The word বই (Boi) has essentially vanished, leaving only a trailing ই (I) or being merged into the noise. The word পড়ি (Pori) has been reduced to প ই (P I), with the medial consonant ড় (Ra) deleted.

   This fragmentation suggests that the Llama 3 tokenizer is treating Bengali characters as rare, independent tokens rather than sub-word units. In English, a common word like "Library" is a single token. In Bengali, without a vocabulary extension, a word like আমাদের might be tokenized into its constituent UTF-8 bytes or individual characters. During the fine-tuning phase, if the training data provided to the "SFTTrainer" contained any ambiguity regarding token delimiters—or if the YOLO output used for training already contained these spaces—the model optimizes its loss function by learning to predict a "Space" token after every "Character" token.

2. **The "Kar" Fallacy: Independent Vowel Substitution:** Bengali script is an Abugida, meaning that vowel sounds are inherent to consonants or represented as diacritics (dependent forms) when attached to a consonant. They appear as full letters (independent forms) only when they occur at the beginning of a syllable or word.Artifact Analysis:Ground Truth: সে স্কুলে যায়. Model Output: স এ স ্ ল এ য আ .Here, we observe a critical orthographic failure.Standard Rule: Consonant স (Sa) + Vowel Sign ে (e-kar) -> সে (Se). Model Behavior: The model outputs Consonant স followed by the Independent Vowel এ. It writes স এ instead of সে.Implication: This is not merely a typo; it is a fundamental violation of Bengali grammar. The model has "unlearned" the rule that vowels following consonants must be diacritics.

3. **Hallucination:** A specific and peculiar error is the frequent substitution of common consonants with the character (U+09BD, Bengali Sign Avagraha), a rare

character used in Sanskrit transliteration to denote the elision of a vowel.

Artifact Analysis:

Input Context: আজ (Aj - Today).

Model Output: আ ঽ (A-Avagraha).

Input Context: আকাশ (Akash - Sky).

Model Output: আ ঽ আ (A-Avagraha-A).

Cause Analysis: The Avagraha is visually distinct (appearing like an 'S') but phonetically silent in modern Bengali. Its appearance in place of জ (Ja) or ক (Ka) suggests Token Collision or Embedding Collapse.

Visual Similarity: In some fonts or handwriting styles, জ (Ja) or parts of ক (Ka) might resemble . If the YOLO model was trained on a dataset where was mislabeled or over-represented, it would predict this class.

Tokenizer Ambiguity: In the Llama 3 tokenizer, rare Bengali characters might map to tokens that are semantically "close" in the vector space if the model is aggressively quantized (e.g., 4-bit) without calibration. The model, unsure of the correct consonant due to the noisy YOLO input, falls back to as a "junk" placeholder. This is a hallmark of a model seeking to minimize loss on ambiguous inputs by selecting a token that (in its corrupted manifold) represents generic "character-ness" without specific phonetic commitment.

5.1.2 Performance Evaluation of Llama-3 and Google Gemini 2.5 Flash on YOLOv8 and YOLOv11s sample dataset

We tried to investigate how Llama-3 performs in zero-shot settings. The sample datasets of YOLOv8 and YOLO11s were given

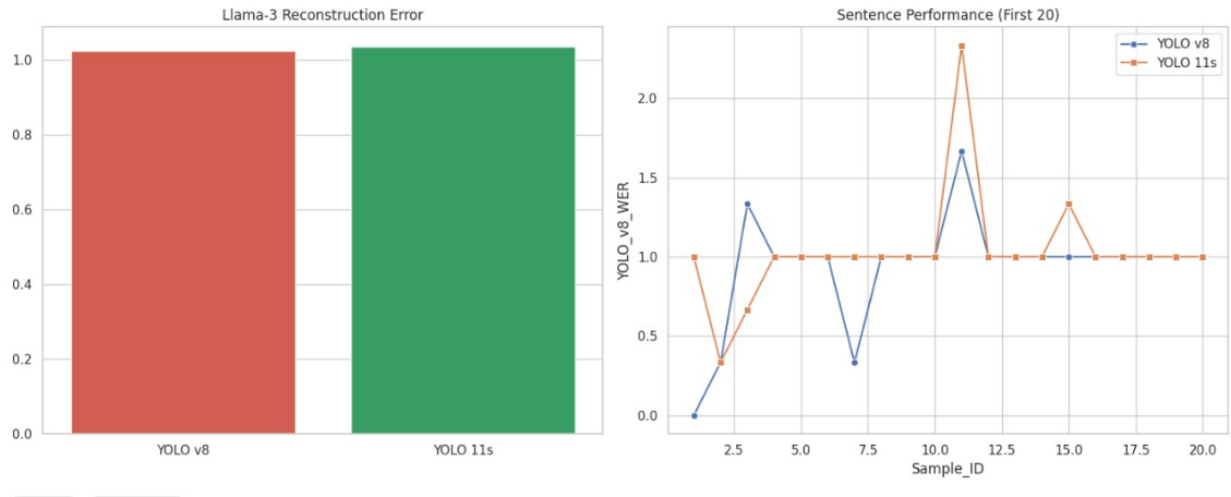

Figure 5.2: Llama-3 Performance in Zero-Shot Settings

Our experiments from Figure 5.2 revealed a significant performance gap between Large Closed-Source Models (Gemini) and Smaller Open-Source Models (Llama-3 8B) in the Zero-Shot setting.

Model	**Setup**	**YOLOv8 Input (WER) (Lower is Better)**	**YOLO11s Input (WER) (Lower is Better)**	**Average WER (Lower is better)**	**Performance Status**
Gemini 2.5 Flash	Zero-Shot (API)	0.509	0.508	0.5085	Most Accurate
Llama-3	Zero-Shot (Local)	1.02	1.03	1.02	Failed
Llama-3	SFTTrainer (Supervised Fine-Tuning Trainer)	0.612	0.611	0.61	Usable

Table 5.3: Performance Evaluation and Status of different LLMs

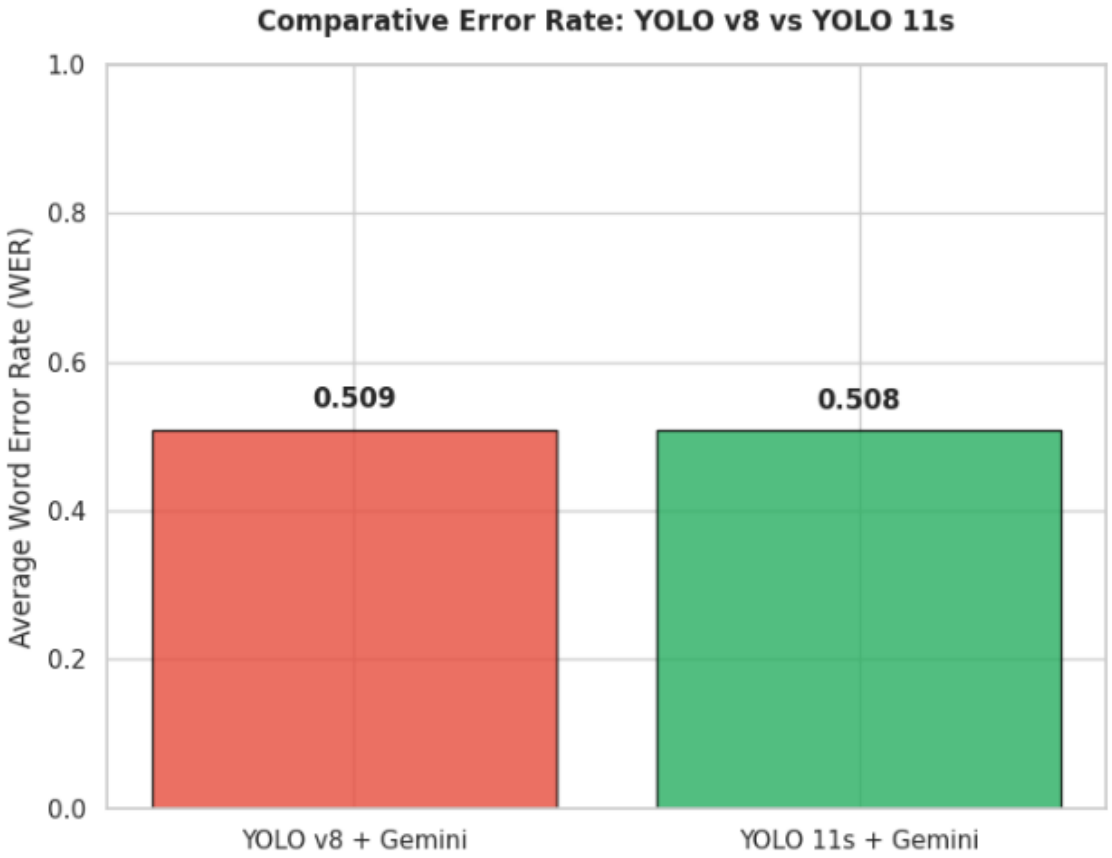

Figure 5.3: Comparative Error Rate

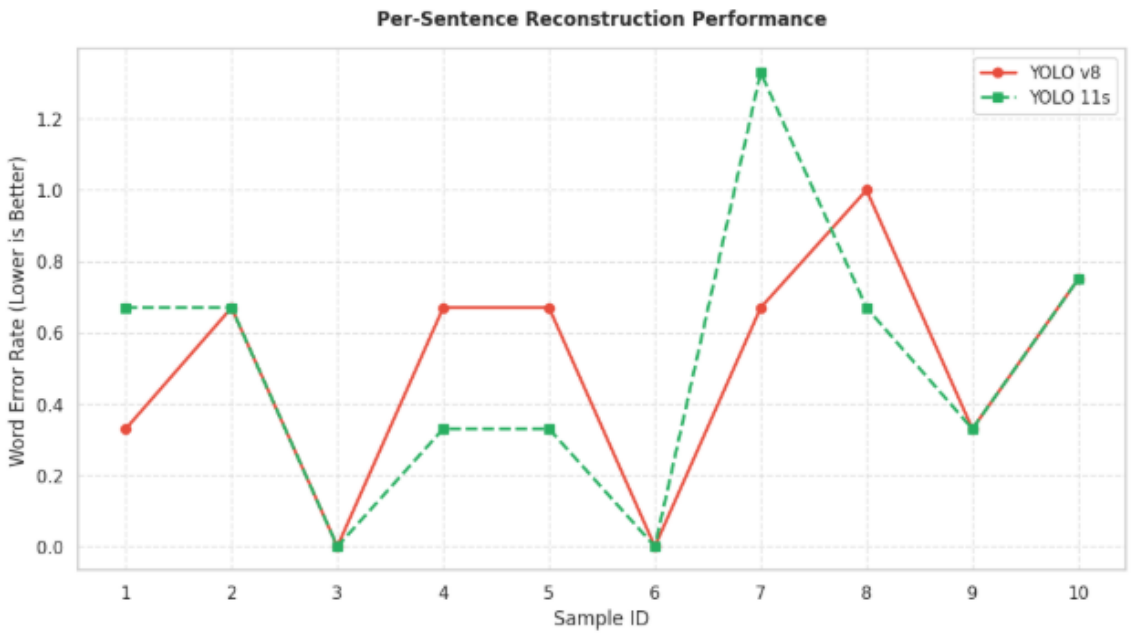

Figure 5.4: Sentence Reconstruction Performance Graph

While trying the Gemini 2.5 Flash API calls, we saw a significant improvement in results. We introduced a correction function where we entered the prompt. The Python code of the correction function is given below:

```python
def get_gemini_correction(text):
    prompt = f"""
    You are an expert Bangla Braille-to-Text Error Corrector.
    The input is noisy OCR text with spaces between letters. There are some
    missing characters. " " usually means " ".

    Examples:
    Input:
    Output:
    Input:
    Output:
    Input:
    Output:

    Task: Fix the following text. Output ONLY the sentence.
    Input: {text}
    Output:
    """
    try:
        response = model.generate_content(prompt)
        time.sleep(1.5)
        return response.text.strip()
    except Exception as e:
        print(f" API Error: {e}")
        return text
```

Due to Google Gemini 2.5 Flash quota limitation, we couldn't generate more than 20 sentence prompts. Large Multimodal Models (LMMs) like Gemini 2.5 Flash demonstrate viable zero-shot performance (WER 0.51), making them suitable for cloud-based implementations where internet connectivity is available.

5.2 Analysis of Design Solutions

Throughout this research, we explored two distinct design paradigms: a traditional two-stage segmentation and classification pipeline, and a unified YOLO-based character extraction model integrated with a Large Language Model (LLM) for contextual reconstruction. These both paradigms had two completely different approaches. To facilitate visualization and understanding, we have explained the two distinct Algorithm 1 and Algorithm 2.

We have mainly two evaluation criteria for our system. One is end-to-end performance, and the other is the performance of our YOLO object detection for character extraction. Among the two design solutions we proposed, the first design approach which is Algorithm 1 was heavily dependent on segmentation performance. It does not matter how much better performance the classifier model gives if the segmentation is faulty the error propagates to every level. The second approach that we proposed is the most feasible one. As the usage of LLM has a lot of scope, even if our character extraction is a bit faulty and misses some characters, the LLM can still fix those.

5.3 Final Design Adjustments

With exploring different designs and dataset, throughout this research, several key adjustments were made to explore different designs and datasets. Firstly, for training YOLO, the specification of the image in terms of resolution had changed several times. We wanted to explore the optimal image resolution that would keep enough information while optimizing computational efficiency. In terms of performance, we have observed that image resolutions ranging from 1024 * 1024, 512 * 512, 248 * 248 have provided similar results across different evaluation metrics. In terms of the traditional approach, with the segmentation model having a dice score of 0.82, the latter YOLO v8/v11 significantly outperformed it by more than 15 percent. It has to be taken into account that the 85 percent dice score of U-Net reflects its ability to locate dots, not the 2 * 3 grid that is required. While contracting a single braille grid, the model fails to recognize more than 60 percent of the grid. With the introduction of the unified character extraction architect with YOLO, it performs at an average of 97 percent accuracy while extracting individual braille boxes. At the later stage of the unified system, different LLM models were used to reconstruct the sentences and preserve syntactical information and semantic information. This stage allowed the system to eliminate propagated errors from earlier stages.

At the training stage, two different batch sizes were used: batch sizes 4 and 8. Each iteration of training had > 150 epochs with more than 5 iterations for training or fine-tuning each model. All further hyperparameters are compiled below for further clarification.

Hyperparameter	Value / Setting	Justification
Batch Size	8 and 4	Optimized for the 16GB VRAM of the NVIDIA RTX 4070 Ti to prevent Out-of-Memory errors while maintaining gradient stability.
Optimizer	AdamW / Auto	Auto mode selected AdamW (Adaptive Moment Estimation with Weight Decay) for faster convergence on sparse data.
Initial Learning Rate (lr0)	0.01 (with decay)	Standard YOLO starting rate, managed by a cosine annealing scheduler to fine-tune weights in later epochs.
Momentum	0.937	Helps accelerate gradient vectors in the right direction, reducing oscillation during training.
Weight Decay	0.0005	Prevents overfitting by penalizing overly complex model weights.
Box Loss Gain	7.5	Heavily weighted to prioritize localization accuracy over classification in early epochs.
Classification Loss Gain	0.5	Lower weight because the primary challenge is finding the dots, not just classifying them.
DFL Loss Gain	1.5	Used to refine the fuzzy boundaries of the dots.
Mosaic Augmentation	1.0 (On)	Stitches four images together during training to improve small object detection.
Close Mosaic	10	Disables mosaic augmentation for the final ten epochs for fine-tuning on real images.
Patience (Early Stopping)	20 / 50	Stops training if validation metrics do not improve to prevent overfitting.

Table 5.4: Hyperparameter settings and justifications used for model training

5.4 Statistical Analysis

Evaluation

The purpose of the evaluation was to find the best architecture for real-time deployment and to determine how well the classification models recognized segmented Braille characters. An 80:20 ratio was used to separate the dataset into training and testing subsets. Using the Adam optimizer, a learning rate of $1e^{-4}$, and categorical cross-entropy as the

loss function, each model was trained for 50 epochs with a batch size of 32.

The performance of the categorization models was evaluated using a variety of evaluation indicators. The main metric was classification accuracy, which is the proportion of correctly classified Braille characters among all input samples. This measure gives a clear picture of the models' proficiency in identifying Braille patterns. In order to assess the models' generalizability and spot any possible overfitting on the training data, validation loss graph was also looked at during the training process as shown in Figure 5.5, 5.7, and, 5.8 respectively. Lastly, the average time needed to categorize a single Braille cell was measured, which is known as inference latency. Because lower latency guarantees faster system response and improves the system's overall usability in real-world deployment settings, this statistic is especially crucial for real-time applications.

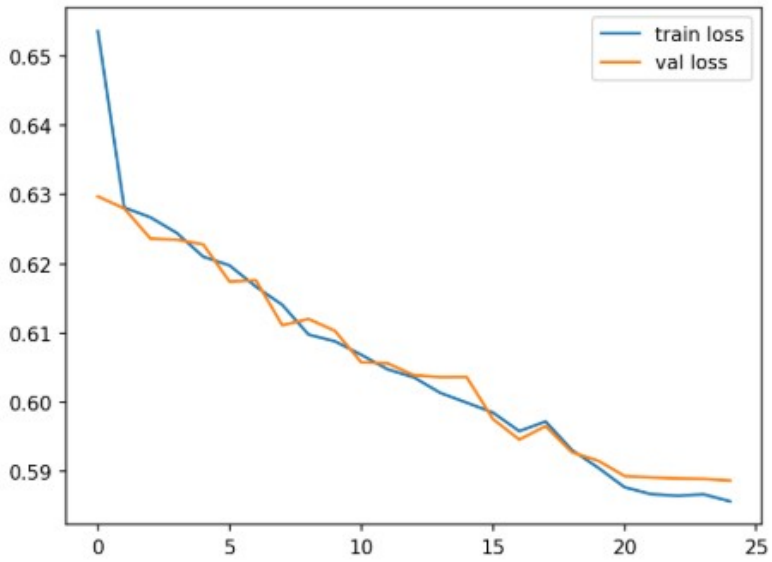

Figure 5.5: Train and Validation Loss for ResNet50

VGG16 obtained an accuracy of almost 50%. This was due to our limitationn of datasets which brought the lowest accuracy amongst others. Longer inference durations resulted from its high memory and computing requirements, despite its great baseline performance. With a bit better accuracy, about 70%, was obtained by ResNet50, which also demonstrated greater resilience to noise and changes in Braille input. On the other hand, MobileNetV3-Small outperformed the other two models in terms of inference time and computation, achieving 92 to 95% accuracy.

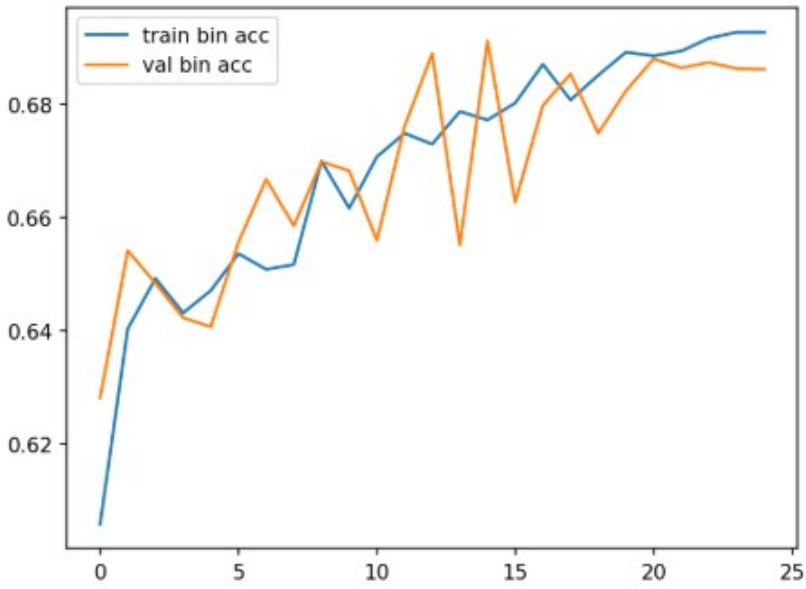

Figure 5.6: Train and validation accuracy for Resnet50

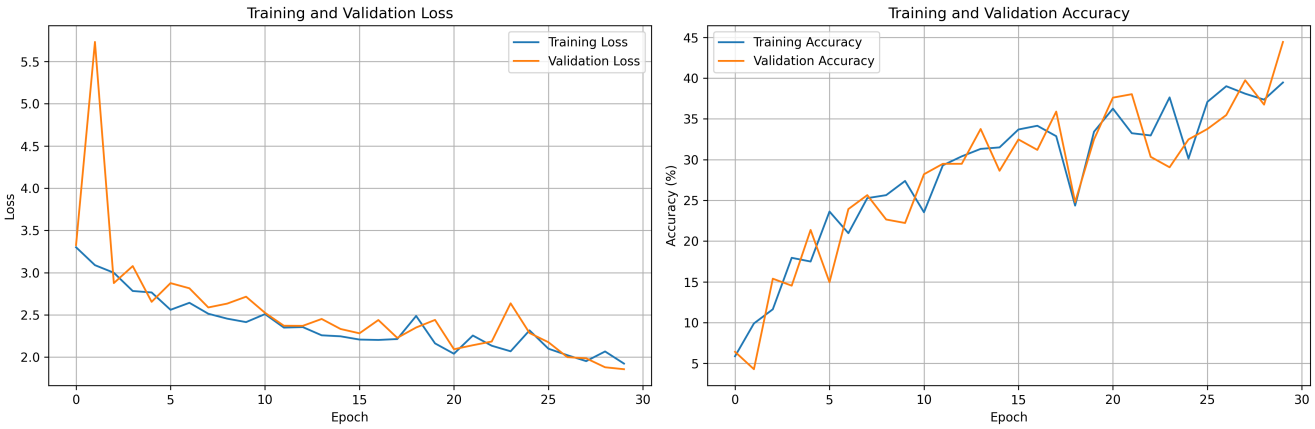

Figure 5.7: VGG16 Evaluations

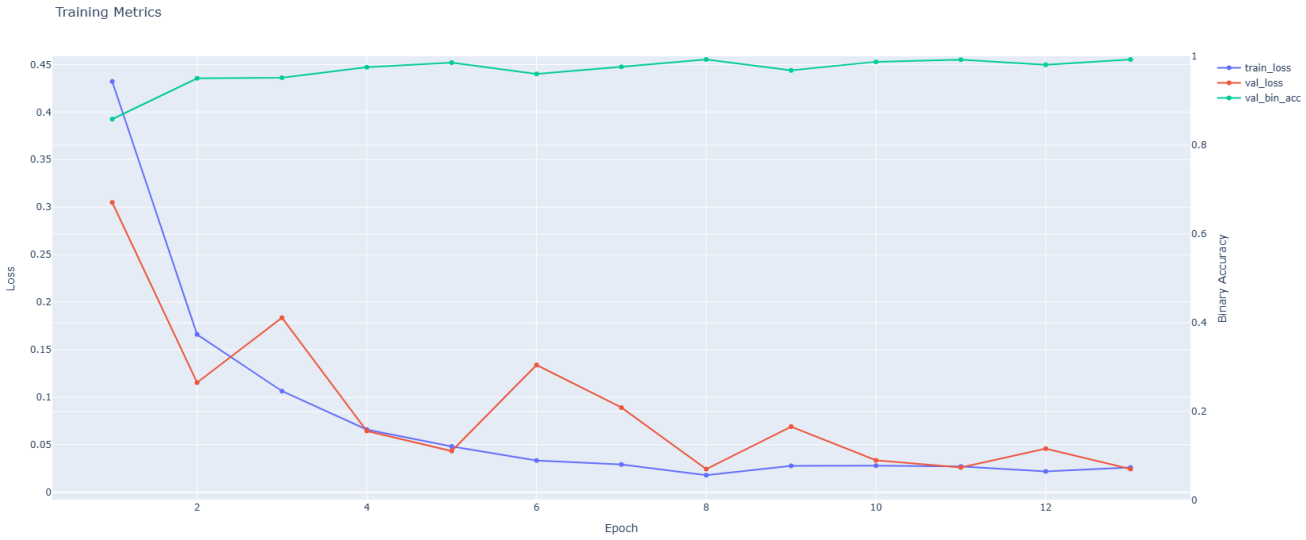

Figure 5.8: MobileNetV3 Evaluations

Figure 5.9: Segmentation Result for U-Net

Overall, the analysis shows that computing economy and accuracy are traded off. MobileNetV3-Small offered the most balanced solution because of its speed, small memory footprint, and compatibility with resource-constrained devices like Raspberry Pi, even though ResNet50 produced average accuracy. MobileNetV3-Small is regarded as the most realistic model for practical Braille-to-speech applications because of this.

After running the first pipeline, the problem we encountered is it is heavily dependent on U-Net and U-Net is not able to detect the bounding box as shown in Figure 5.9. So, we moved to another Convolutional architecture, which is YOLOv8 and YOLOv11. The reason YOLO is giving us good results, is that YOLO is good at detecting bounding boxes. But the U-Net tries to expand the mask for detecting bounding boxes. That's why it was performing worse than YOLO. We used two models of YOLO to see how it

performs. The analysis is given below:

For YOLOv8:

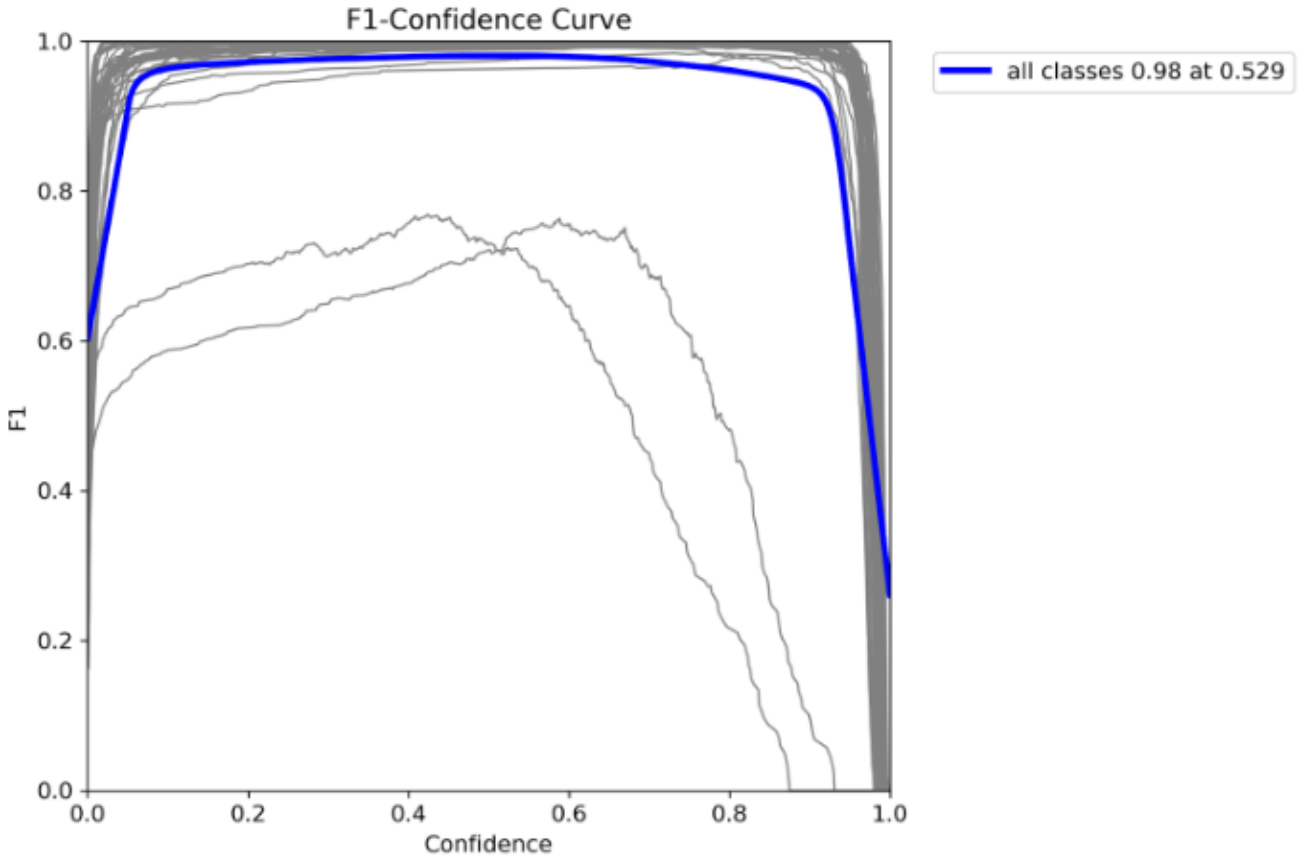

Figure 5.10: F1-Confidence Curve for YOLOv8

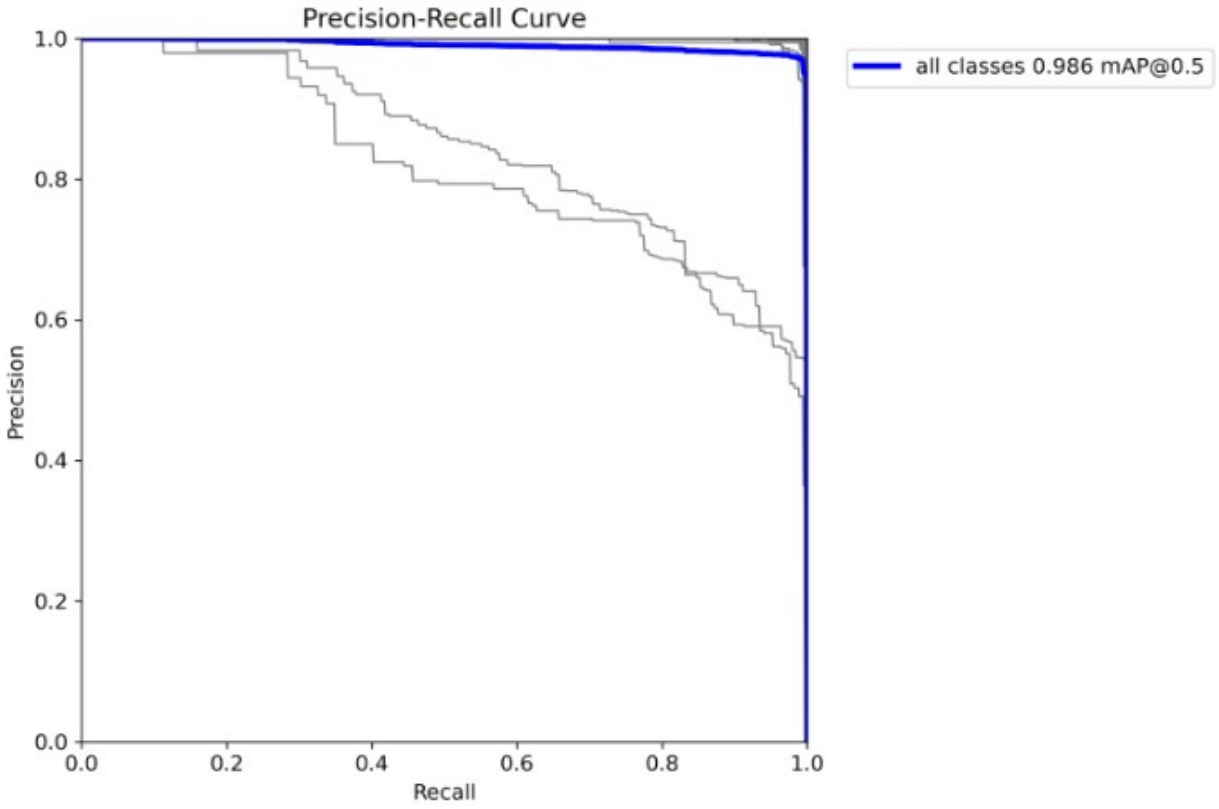

Figure 5.11: Precision Recall Curve for YOLOv8

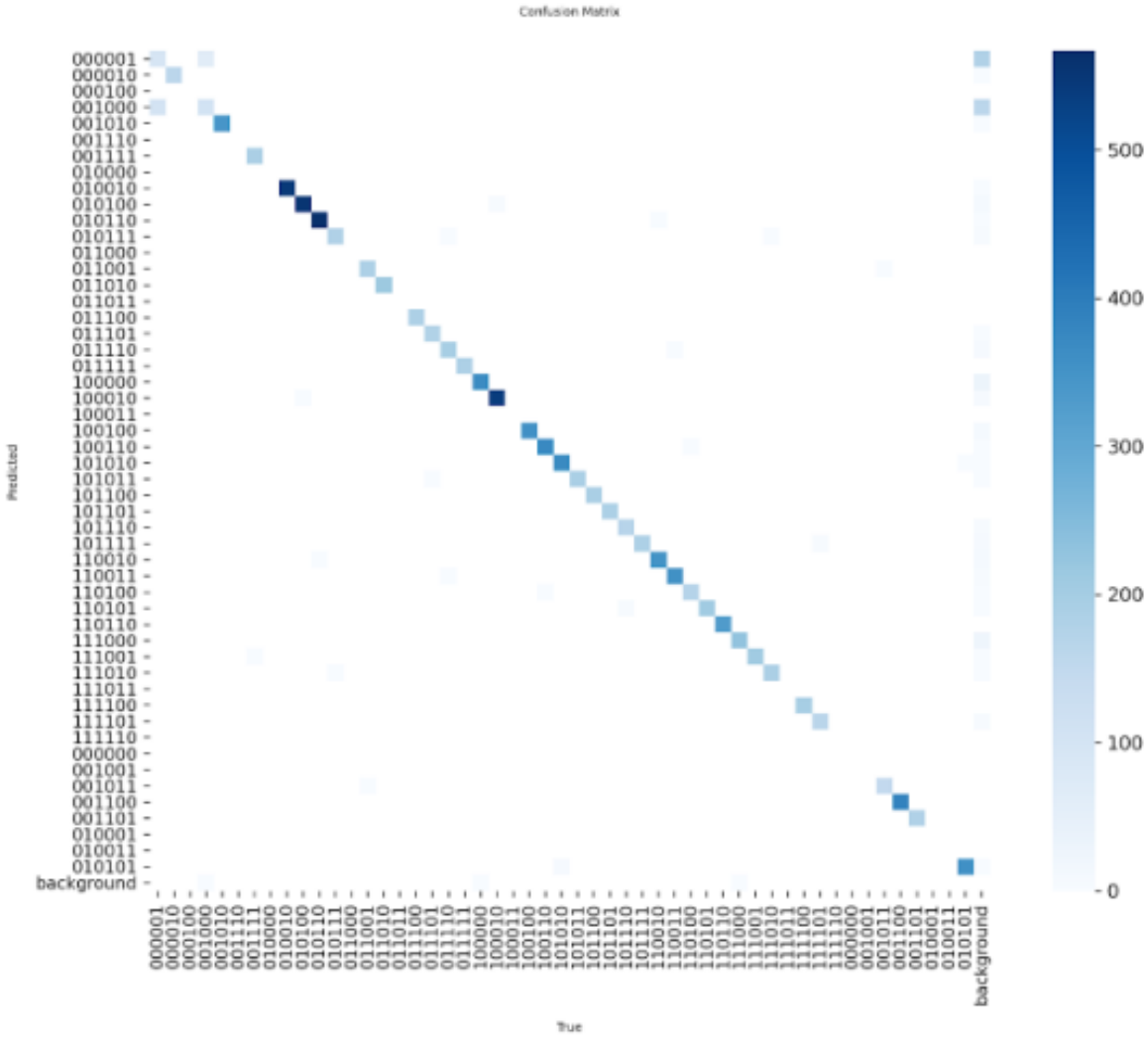

Figure 5.12: Confusion Matrix for YOLOv8

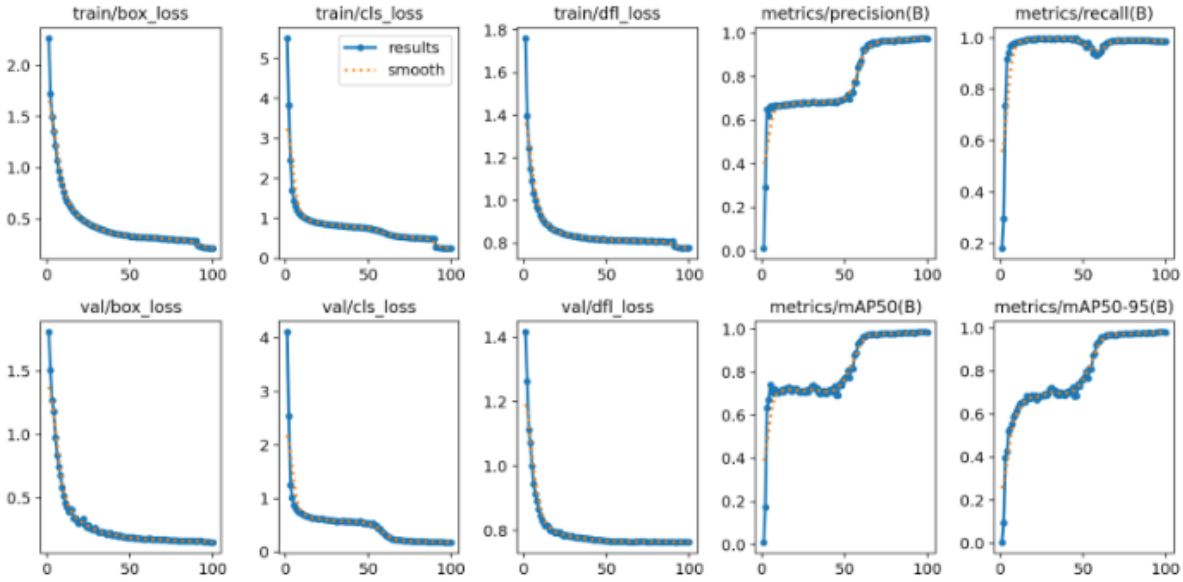

Figure 5.13: YOLOv8 result summary graphs

Figure 5.11 shows the precision-recall across all classes. It should be noted that although the mAP@50 shows 0.986, it's not fully consistent across all features. Nevertheless, it shows the model was able to accurately detect all targeted classes consistently. The confusion matrix on Figure 5.12 solidifies the claim that there are a few cases where it couldn't perform up to the mark. In Figure 5.13, we get a summarized compilation of all the results. The training class loss and box loss converged smoothly, which means the model had no sign of overfitting. However, the precision and recall curve showed some discrepancies due to mislabeling a few classes, but ultimately converged to a satisfactory point.

For YOLOv11:

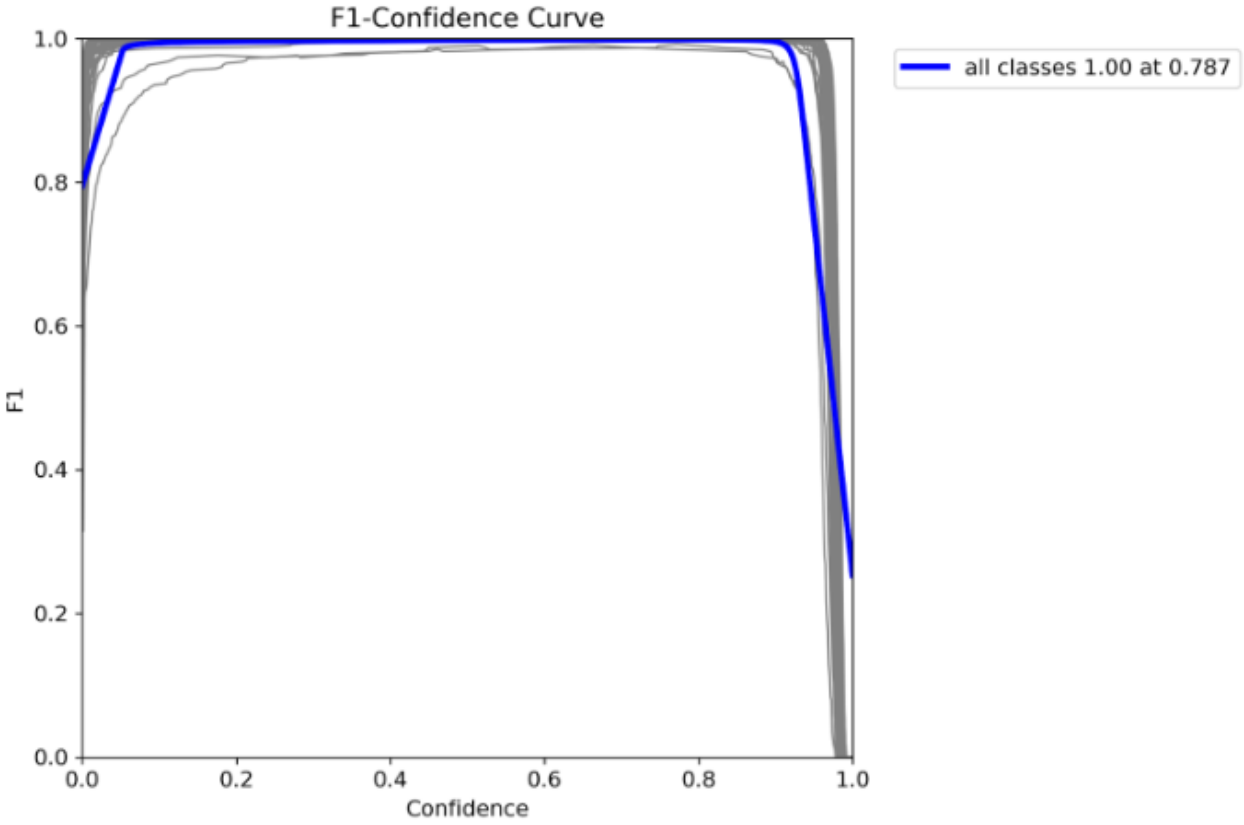

Figure 5.14: YOLOv11 F1-confidence curve

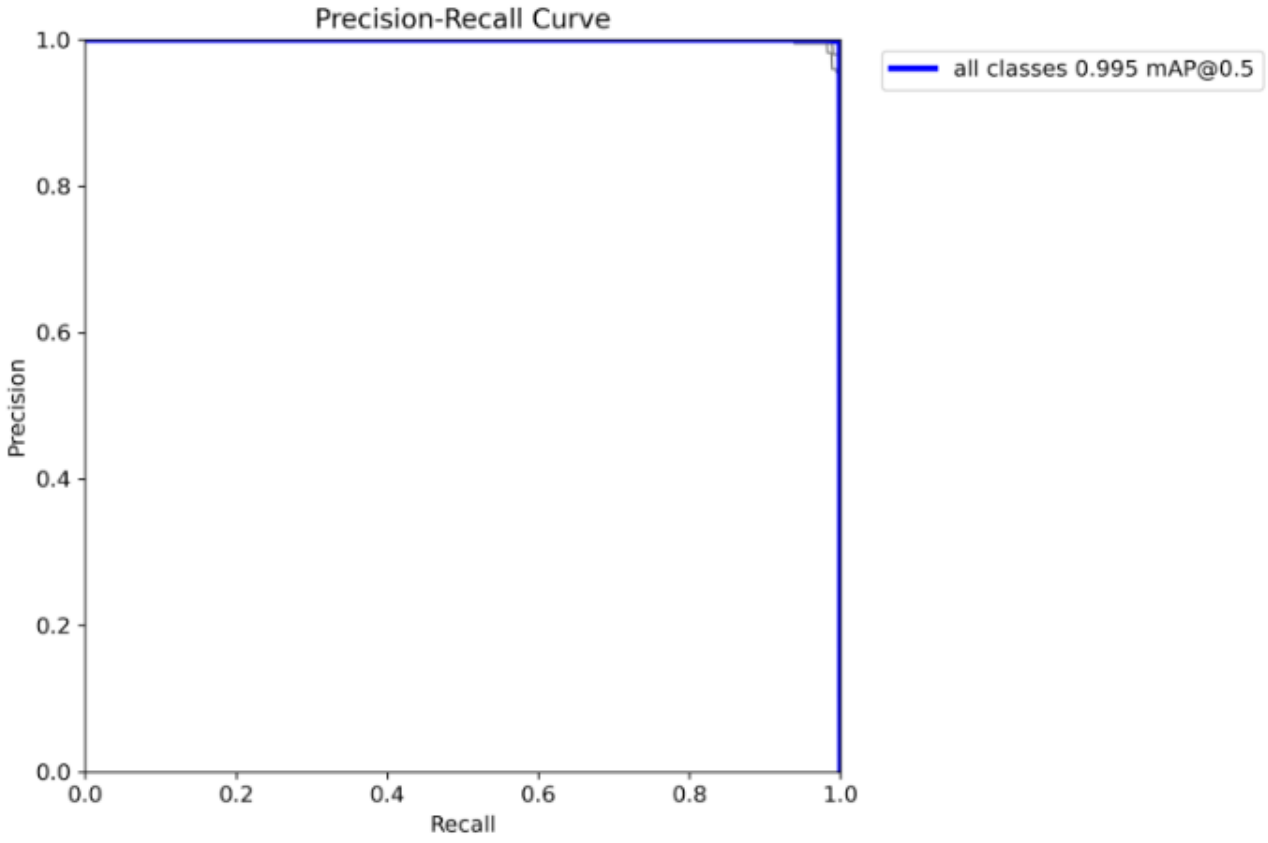

Figure 5.15: Precision Recall Curve for YOLOv11

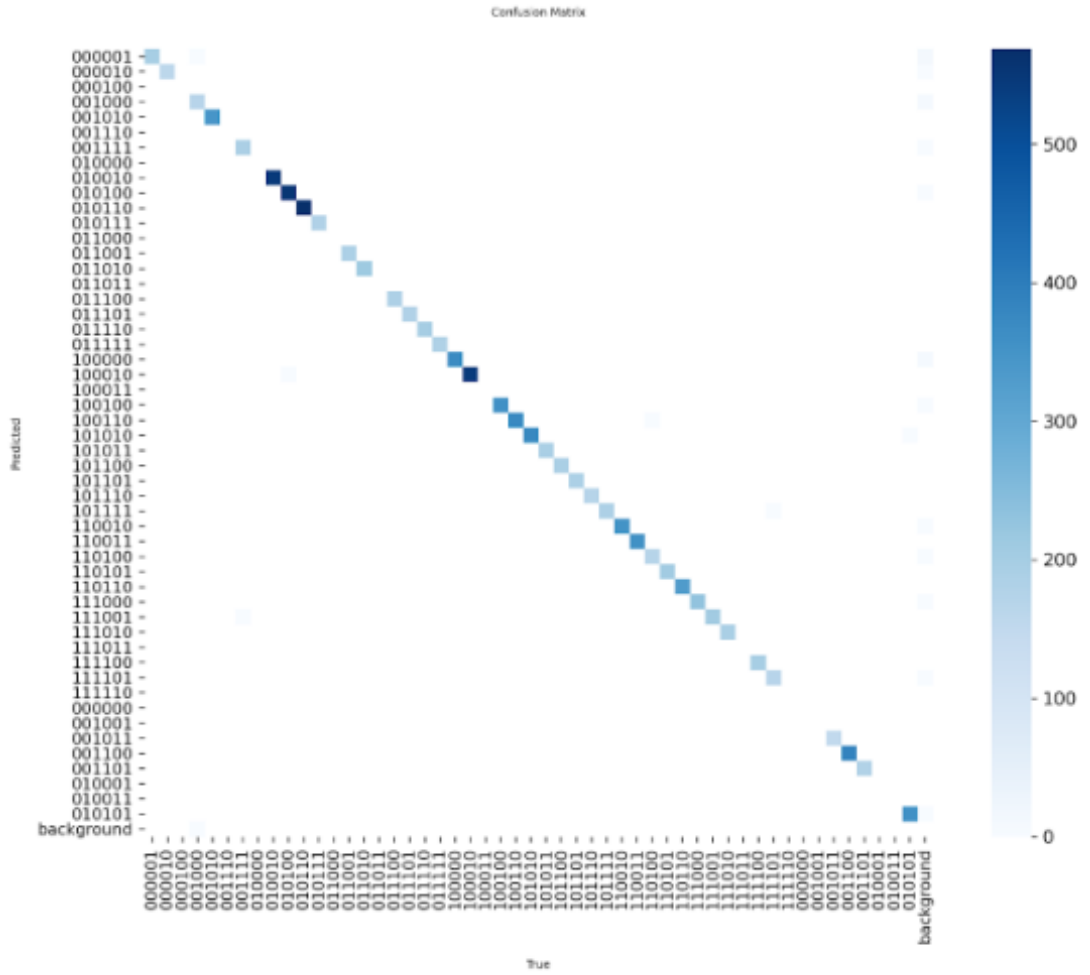

Figure 5.16: Confusion Matrix for YOLOv11

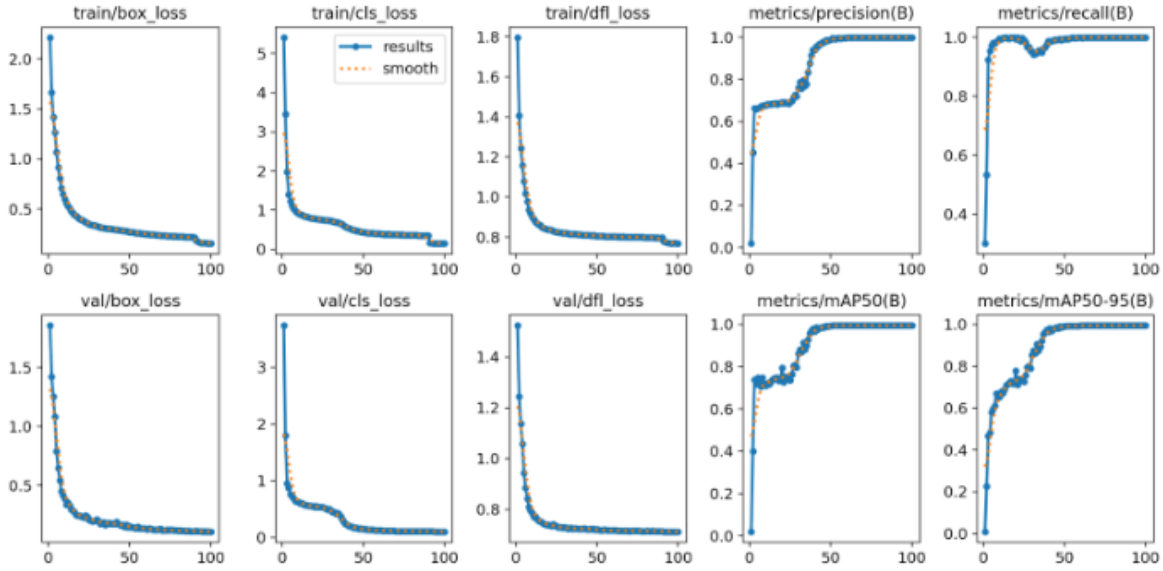

Figure 5.17: Result Graphs for YOLOv11

For the results of v11 (Figure 5.14, 5.15, 5.16, 5.17), we can observe that the overall precision-recall curves performed better on average, and it was also able to handle outlier classes better. Unlike v8, the v11 model loss functions and precision-recall metric converge more smoothly, indicating its better and more consistent performance with detecting boxes. In the precision-recall individual metric, the v8 showed a sudden bump before converging. However, the v11 model provided smoother convergence.

The graphs above show some results from the best-performing iteration of the training process of YOLOv8 and YOLOv11. Table 5.5 shows performance across each iteration.

Model Architecture	Iteration	Precision	Recall	mAP@50	mAP@50-95	Box Loss	F1-Confidence	Status	Inference Speed
YOLO v11	**1(Ref)**	**0.999**	**0.999**	0.994	0.994	0.111	0.787–1.000	Optimal	330.5
	2	**0.997**	**0.998**	0.994	0.993	0.113	0.724–1.000	Stable	332.3
	3	**0.998**	**0.999**	0.996	0.994	0.110	0.718–1.000	Stable	338.1
	4	**0.996**	**0.997**	0.993	0.992	0.115	0.768–1.000	Stable	334.7
YOLO v8	**1(Ref)**	**0.975**	**0.987**	0.984	0.980	0.215	0.529–0.985	Optimal	385.50
	2	**0.972**	**0.985**	0.982	0.978	0.236	0.515–0.953	Stable	387.25
	3	**0.635**	**0.992**	0.697	0.692	0.301	0.321–0.79	Anomaly	388.12
	4	**0.978**	**0.989**	0.975	0.981	0.239	0.491–1.000	Stable	386.2

Table 5.5: Performance across YOLOv11 and YOLOv8 training iterations

5.5 Comparisons and Relationships

In this research, there was a shift from a multi-stage pipeline to a more unified object detection approach. The next two section provides a comparative analysis across two

domains: architectural, computational efficiencies, and robustness to propagated error. In additon Table 5.6, 5.7 and 5.8 in Section 5.5.3 gives the full summary table for all of the models used in this thesis.

5.5.1 Architecture: (Unified vs Multi-stage)

The traditional methodology that utilizes a sequential pipeline with U-Net -> ResNet/VGG/ MobileNet is very much prone to error propagation. Despite the individual classifier working with high accuracy doesn't ensure the robustness of the entire pipeline. If the segmentation fails to detect even a single dot, which is often the case, this would cause the entire system to continuously propagate the error to the next stage. On the contrary, with the unified architecture, the YOLO is exceptionally good at extracting the bounding boxes and characters, even if YOLO's failure to do so does not propagate the error that much, as LLM can correct small mistakes.

5.5.2 Performance Metrics and Class Sensitivity: (Unified vs Multi-stage)

From the previous results from Table 5.5, it can be concluded that between YOLOv8 and v11, there isn't much difference in terms of accuracy. While we have detected a few more anomalies with v8, over the period of my iteration, the average doesn't reflect any significant difference. In terms of inference speed v11 marginally outperforms v8 on average by < 30 ms. However, what sets them apart is their ability to detect outliers. v11 on average outperformed v8 while extracting more character on the lower end. This shows v11 has more capability in capturing contextual information. Because of the introduction of LLM at the end of the pipeline, this pipeline has more robustness in capturing the semantic integrity of a sentence. Because of its context awareness at a high level, it can perform grammatical correction, which is crucial to stop error propagation in the whole pipeline.

5.5.3 Summary Table for all models

Metric/Model	U-Net	ResNet50	VGG16
Primary Task	Segmentation of Braille dots in Pixel Level	Classification of Braille Characters	Classification of Braille Characters
Architecture Type	Encoder-Decoder with skip connections	Deep Residual Network with skip connections	Sequential deep CNN with stacked 3x3 filters
Model Size	∼31 MB (smaller footprint, deployable on device)	∼98 MB (medium size, optimized with residual blocks)	∼528 MB (memory intensive)
Parameter Count	7 million	25 million	135 million
Evaluation Metric Used	Dice Coefficient, Binary Cross-Entropy	Accuracy, Validation Loss	Accuracy (Baseline)
Dice Score	0.84–0.86 on Braille segmentation	N/A	N/A
Accuracy	High Segmentation Overlap (>90%)	Robust classification performance with strong generalization	∼92% (Baseline transfer learning on Braille data)
Latency (Inference Speed)	∼50–100 ms	∼80–120 ms	∼250–300 ms
Strength	Optimized for noisy datasets, preserves spatial info	Handles vanishing gradient	High baseline accuracy, simple design, good for transfer learning
Weakness	Limited to Segmentation	Higher computational cost than U-Net	Risk of Overfitting, high memory consumption

Table 5.6: Summary of U-Net, ResNet50, and VGG16 for Braille processing tasks

Metric/Model	MobileNetV3-Small
Primary Task	Braille character classification (image-based recognition of Braille dot patterns)
Architecture Type	Lightweight Convolutional Neural Network (CNN) with Inverted Residual Blocks + SE
Model Size	∼10 MB (depends on number of Braille classes)
Parameter Count	∼2.9 million
Evaluation Metric Used	Accuracy, Validation loss
Dice Score	∼0.90–0.95 (when used after Braille segmentation models like U-Net)
Accuracy	∼90–94% (depending on dataset size, quality, and class balance)
Latency (Inference Speed)	∼8–12 ms
Strength	Extremely lightweight, efficient on low-power devices, maintains high accuracy, and integrates channel attention for better feature learning
Weakness	Slightly less accurate than larger CNNs (e.g., ResNet, EfficientNet), struggles with highly noisy or poorly segmented Braille images

Table 5.7: Summary of MobileNetV3-Small for Braille Classification

Metric / Model	YOLOv8-Small (YOLOv8s)	YOLOv11-Small (YOLO11s)
Primary Task	Braille dot detection, segmentation, and character-level classification using a single-stage object detection framework	Advanced Braille dot detection and pattern-level classification with improved contextual modeling
Architecture Type	One-stage object detector with CSP-based backbone, PAN-FPN neck, and decoupled detection heads	Attention-augmented one-stage object detector with optimized feature fusion and improved detection head
Model Size	∼22 MB (depending on number of classes and export format)	∼24–26 MB (varies with attention modules and segmentation head configuration)
Parameter Count	∼11.2 million	∼13–15 million
Evaluation Metric Used	mAP@0.5, mAP@0.5:0.95, Precision, Recall	mAP@0.5, mAP@0.5:0.95, Precision, Recall
Dice Score	N/A	N/A
Accuracy	∼93–97% (dependent on dataset quality and annotation precision)	∼95–98% (more robust under noisy and low-contrast conditions)
Latency (Inference Speed)	∼350 ms	∼385 ms
Strength	End-to-end detection and classification, real-time performance, error propagation is reduced in case of multi-stage pipelines	Improved feature representation via attention mechanisms, better robustness to noise and distorted Braille images
Weakness	Performance may degrade for extremely dense or overlapping Braille dots; requires precise annotations	Slightly higher computational cost compared to YOLOv8s; less suitable for ultra-low-power edge devices

Table 5.8: Comparison of YOLOv8-Small and YOLOv11-Small for Braille Recognition

5.6 Discussions

This project introduces a deep learning-based system for translating Bengali Braille to Bangla speech, making resources more accessible for visually impaired communities. The systematic approach of utilizing image preprocessing, YOLO-based character extraction model, LLM-based correction of words, and Bangla Text-to-Speech technologies to create an end-to-end continuum for translating tactile Braille to natural-speech voices. The proposed method shows promise in addressing accessibility barriers in education and communication, particularly in a space where there are no resources available for Bengali

speakers who are visually impaired. The results returned from the initial phase suggest there will be a number of challenges to overcome, particularly the lack of annotated datasets for building the models, the relatively poor quality of Bangla TTS, and the varying inputs from handwritten tactile images of Braille. At the conclusion of the pre-thesis stage of the project, there is a good platform to build on, further developing, optimizing,g and deploying the work, effectively creating a scalable assistive tool that can be of immense value to blind and low vision communities in Bangladeshi communities.

Chapter 6

Conclusion

6.1 Summary of Findings

In this study, we were able to build and test a Bangla Braille-to-Voice conversion system using end-to-end Deep Learning components. The study resulted in a number of important recommendations that concerned system architecture, model performance, and semantic reconstruction.

1. **Better Unified Architectures than Multi-stage Pipelines:** The paper concluded that a Unified Object Detection Architecture (YOLO) is far superior in Braille recognition as compared to the use of traditional multi-stage pipelines (segmentation and classification). The first multi-stage method, which employs U-Net as a segmentation model, was found to be highly susceptible to error propagation; as long as the segmentation model was not able to identify bounding boxes correctly, the following classifiers (VGG16/ResNet50) became useless. The YOLO architecture, on the contrary, reduced both localization and classification into a single regression task, which prevented error propagation effectively and indicated great resistance to Braille cell extraction.

2. **YOLO Model Performance and Efficiency:** Comparative Analysis of the YOLOv8 and YOLOv11 models showed that despite a high accuracy in general performance, YOLOv11 was found to be better at performing over certain metrics. For inference Speed, YOLOv11 showed a better usability in specific metrics, and it is found to have a margin of average precision (mAP@50) of 0.994 as compared to YOLOv8. In terms of contextual detection, YOLOv11 had a stronger capability of detecting outliers and capturing contextual information.

3. **Effectiveness of Large Language Models in Semantic Reconstruction:** The study indicated a specific performance disparity among various Large Language Models (LLMs) when they were used to semantically reconstruct noisy Bangla output. In open-source limitations, Llama-3 (8B) model failed when used with regard to the lingo peculiarities of the Bengali language. It demonstrated Character-Level Fragmentation (division of words into separate Unicode points), the Kar Fallacy (replacing independent vowels with diacritics), and Avagraha Hallucinations (the addition of some rare characters such as 'ঽ' is a result of token collision). Closed-Source Superiority: A much lower Word Error Rate (WER) of about 0.508 on the

Gemini 2.5 Flash model than that of Llama-3 (WER of about 1.02) was further supported by a significantly lower Word Error Rate of the locally trained SFTTrainer (Supervised Fine-Tuning Trainer) of about 0.61 against zero shot performance of Llama-3. The addition of the LLM layer was found necessary to fix the presence of grammatical inconsistency and prevent the propagation of errors in the pipeline.

4. **Validity of Synthetic Data Generation:** There are limited labeled Bangla Braille datasets; therefore, the study validated the Synthetic Data Generation (SDG) strategy. Using a conversion factor of $1mm = 3.7795$ pixels and following the Library of Congress (LOC) Specification 800, the system was able to produce training data that was analogous to that of the real world. Domain randomization, including perspective warping and Gaussian blur frames, were used to ensure that the models trained with synthetic data could be useful in generalising and yield strong results when using a real-world input.

6.2 Contributions to the Field

This thesis is valuable to the assistance of the computer vision community as it is more detailed in a deep learning network of Bengali Braille character to Bangla speech. In most of the literature, where attention is mostly paid to English Braille or conversion of text only, this paper brings about special linguistic and structural issues of Bengali Braille as the object of study, since it is most likely an untapped field.

The main objective of this work is the combination of various elements of deep learning into a single end-to-end system. The bottom-to-top architecture, including image preprocessing, UNet-based segmentation, and CNN-based classification models (VGG16, ResNet50, and MobileNetV3), makes it possible to extract and recognize Braille characters in a tactile image. Currently, with the use of YOLO, the ability to detect dots and cells of Braille proves that modern object detection systems can be used to deal with closely-packed Braille patterns.

The other relevant contribution is the comparative analysis of lightweight and deep convolutional gains in the realm of character recognition among Braille. Through the evaluation of performance based on its accuracy, inference time, and robustness, the paper offers useful information regarding the trade-offs between the complexity of a model and its deployability to assistive systems where time of use or resource constraints are often a constraint.

Moreover, the study can be applied to an accessibility-centric strain of system design by updating the recognition system to Bangla Text-to-Speech (TTS) by-product. This full conversion of tactile input to audio output brings out a viable channel of enhancing access to learning and informational products to the visually impaired in Bangla-speaking societies. The results of this paper established a skeleton, which can be enhanced by future scholars and administrators to develop and promote the digitization of Bengali Braille and communicative assistive technologies.

6.3 Recommendations for Future Work

A key area to take work in the future would be creating larger and more diverse annotated datasets on Bengali Braille. Generalization of models would greatly enhance this by increasing dataset volume and variation, especially in the forms of handwritten Braille, variability of paper quality, lighting, and noise.

Aside from data scarcity, this research paper highlighted a difficult aspect of this system that is the "Ambiguity Paradox". Unlike languages like English where all possible unique characters can be accommodated through a 6-dot braille pattern, it is impossible with Bangla. Bengali language has 85 unique classes that need to be accommodated for it to work as a complete system. Not only on the character level, it will need to accommodate additional patterns that are used in proverbs. This problem is not solvable with a simple deterministic mapping. A data driven process with a seq-to-seq model could be a possible solution. However, this needs to be studied further.

Further research can also involve applying transformer-based vision models or CNN-Transformer-based structures to attain better information of the context in terms of Braille patterns, particularly when dealing with challenging or corrupt input. The implementation of the sequence-conscious models at the full-word or even sentence recognition level can further decrease the error that is propagated through the recognition pipeline.

The other possible improvement is in the quality of Bangla Text-to-Speech. The more advanced neural TTS models, which are to be integrated with the proposal and trained on the specific high-quality datasets of Bangali speech, might lead to increased naturalness of the pronunciation, intonation, and fluency. The usability might also be enhanced by the ability to customize it by enabling the user to specify the rate of speech, type of voice, and dialect.

Lastly, carrying out proper user studies with the visually impaired persons and other concerned parties would offer much information about the system's usability, reliability, and its real-life effectiveness. These reviews would aid in streamlining the system design and would make the proposed solution very close to the actual demands of those who are meant to utilize it.

Bibliography

[1] S. M. Pizer et al., "Adaptive histogram equalization and its variations," *Computer Vision, Graphics, and Image Processing*, vol. 39, no. 3, pp. 355–368, 1987.

[2] G. Bradski, "The opencv library," *Dr. Dobb's Journal of Software Tools*, 2000.

[3] J. Puckett, L. Byster, S. Westervelt, et al., *Exporting Harm: The High-Tech Trashing of Asia*. Basel Action Network, 2002.

[4] P. Blenkhorn, "A portable device for the translation of braille to literary text," *Disability and Rehabilitation: Assistive Technology*, vol. 3, no. 1-2, pp. 57–62, 2008. DOI: 10.1080/17483100701320214

[5] J. Kanjantoe et al., "Touching force response of piezoelectric braille cell," *Proceedings of the 2008 ACM Symposium on Applied Computing*, pp. 1361–1365, 2008. DOI: 10.1145/1328491.1328536 [Online]. Available: https://dl.acm.org/doi/10.1145/1328491.1328536

[6] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[7] A. Krizhevsky, I. Sutskever, and G. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proceedings of the 25th International Conference on Neural Information Processing Systems (NIPS)*, 2012.

[8] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[9] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[10] E. Upton and G. Halfacree, *Raspberry Pi User Guide*, 3rd. Wiley, 2014.

[11] P. P. C., "Braille to text and speech for cecity persons," *International Journal of Research in Engineering and Technology*, vol. 4, no. 1, pp. 263–268, 2015. DOI: 10.15623/ijret.2015.0401039 [Online]. Available: https://doi.org/10.15623/ijret.2015.0401039

[12] F. Chollet et al., *Keras*, https://keras.io, 2015.

[13] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2015.

[14] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015. DOI: 10.1038/nature14539 [Online]. Available: https://www.nature.com/articles/nature14539

[15] L. Nahar, A. Jaafar, E. Ahamed, and A. Kaish, "Design of a braille learning application for visually impaired students in bangladesh," *Assistive Technology*, vol. 27, no. 3, pp. 172–182, 2015.

[16] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, N. Navab, J. Hornegger, W. M. Wells, and A. F. Frangi, Eds., ser. Lecture Notes in Computer Science, vol. 9351, Springer, 2015, pp. 234–241. DOI: 10.1007/978-3-319-24574-4_28 [Online]. Available: https://arxiv.org/abs/1505.04597

[17] M. Abadi et al., "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," *arXiv preprint arXiv:1603.04467*, 2016.

[18] K. D. P. G. Anuradha, "Low-cost portable communication device for the deaf-blind," *International Journal of Industrial Electronics and Electrical Engineering*, 2016, Available upon request or from journal archives.

[19] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *European Conference on Computer Vision (ECCV)*, 2016.

[20] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016.

[21] Dhaka Tribune, "Experts: 750,000 people suffer from blindness in bangladesh," *Dhaka Tribune*, Oct. 2018, Updated: 12 Oct 2018, 01:53 AM; accessed via Dhaka Tribune website. [Online]. Available: https://www.dhakatribune.com/bangladesh/158006/experts-750-000-people-suffer-from-blindness-in

[22] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, 4th. Boston: Pearson, 2018.

[23] M. H. R. Masum, M. Kaiser, and E. Hassan, *An affordable system for translating bengali braille codes of bangladesh to bengali text*, Nov. 2018.

[24] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018. DOI: 10.48550/arXiv.1804.02767 [Online]. Available: https://arxiv.org/abs/1804.02767

[25] J. Shen et al., "Natural tts synthesis by conditioning wavenet on mel spectrogram predictions," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2018, pp. 4779–4783. DOI: 10.1109/ICASSP.2018.8461368 [Online]. Available: https://ieeexplore.ieee.org/document/8461368

[26] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *Journal of Big Data*, vol. 6, no. 1, pp. 1–48, 2019. DOI: 10.1186/s40537-019-0197-0 [Online]. Available: https://doi.org/10.1186/s40537-019-0197-0

[27] T. B. Brown et al., "Language models are few-shot learners," *Advances in Neural Information Processing Systems*, vol. 33, pp. 1877–1901, 2020.

[28] A. Buslaev, V. I. Iglovikov, E. Khvedchenya, A. Parinov, M. Druzhinin, and A. A. Kalinin, "Albumentations: Fast and flexible image augmentations," *Information*, vol. 11, no. 2, p. 125, 2020.

[29] R. Li, H. Liu, and Y. Qian, "Optical braille recognition based on semantic segmentation network with auxiliary learning strategy," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2020, pp. 2362–2368. DOI: 10.1109/CVPRW50498.2020.00285 [Online]. Available: https://openaccess.thecvf.com/content_CVPRW_2020/html/w34/Li_Optical_Braille_Recognition_Based_on_Semantic_Segmentation_Network_With_Auxiliary_CVPRW_2020_paper.html

[30] M. Abd Elaziz, A. Dahou, N. A. Alsaleh, A. H. Elsheikh, A. I. Saba, and M. Ahmadein, "Boosting COVID-19 image classification using MobileNetV3 and aquila optimizer algorithm," *Entropy*, vol. 23, no. 11, p. 1383, 2021. DOI: 10.3390/e23111383 [Online]. Available: https://doi.org/10.3390/e23111383

[31] V. Bhattacharjee, M. A. Rahman, and S. M. Shiblee, "Present situation and necessity regarding reading materials of the visually impaired students studying in bangladesh," 2021.

[32] V. V. Murthy, M. Hanumanthappa, and S. Vijayanand, "Braille cell segmentation and removal of unwanted dots using canny edge detector," in *Advances in Artificial Intelligence and Data Engineering*, R. H. Latha, R. S. Hegadi, and V. E. G. Raj, Eds., Springer, 2021, pp. 79–87. DOI: 10.1007/978-981-15-3514-7_7 [Online]. Available: https://doi.org/10.1007/978-981-15-3514-7_7

[33] Y. Sun, Y. Zhang, and Q. Zhang, "The role of data cleaning in deep learning: An empirical study," in *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2021.

[34] UNESCO. "The environmental impact of print vs. digital." Accessed: 2025-09-27. [Online]. Available: https://en.unesco.org/

[35] S. Zhang, C. Zhang, and Q. Zhang, "Data quality matters: A case study of the impact of data cleaning on building ai models," in *IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*, 2021.

[36] S. Mahfuz, M. N. Sakib, and M. Husain, "A preliminary study on visually impaired students in bangladesh during the covid-19 pandemic," *Policy Futures in Education*, vol. 20, no. 4, pp. 402–416, 2022.

[37] S. A. Shakoor et al., "Prevalence of blindness and its determinants in bangladeshi adult population: Results from a national cross-sectional survey," *BMJ open*, vol. 12, no. 4, e052247, 2022.

[38] World Health Organization, "World report on vision," World Health Organization, 2022, Accessed: 2025-09-27. [Online]. Available: https://www.who.int/publications/i/item/world-report-on-vision

[39] S. Biswas, S. Acharjee, A. Ali, and S. S. Chaudhuri, "Basic architecture of yolov8 object detection model," in *YOLOv8 based Traffic Signal Detection in Indian Road*, Figure 1, ResearchGate, 2023. [Online]. Available: https://www.researchgate.net/figure/Basic-architecture-of-YOLOv8-object-detection-model_fig1_376831163

[40] G. Jocher, *Ultralytics yolov8 (version 8.0.0)*, 2023. DOI: 10.5281/zenodo.xxxxxxx [Online]. Available: https://github.com/ultralytics/ultralytics

[41] G. Jocher, A. Chaurasia, and J. Qiu, *Yolov8: Ultralytics yolo*, https://github.com/ultralytics/ultralytics, 2023.

[42] S. Park and J. Kim, "Edge-optimized braille recognition on raspberry pi," *IEEE Access*, vol. 11, pp. 4567–4580, 2023. DOI: 10.1109/ACCESS.2023.xxxxxxx

[43] J. Terven, D. Cordova-Esparza, and J. A. Romero-Gonzalez, "A comprehensive review of yolo: From yolov1 to yolov8 and beyond," *arXiv preprint arXiv:2304.00501*, 2023. DOI: 10.48550/arXiv.2304.00501 [Online]. Available: https://arxiv.org/abs/2304.00501

[44] Z. Meng, Z. Cai, J. Feng, H. Ma, H. Zhang, and S. Li, "Braille character segmentation algorithm based on gaussian diffusion," *Computers, Materials & Continua*, vol. 79, no. 1, pp. 1481–1496, 2024, ISSN: 1546-2218. DOI: 10.32604/cmc.2024.048002 [Online]. Available: https://doi.org/10.32604/cmc.2024.048002

[45] N. Rao, *Yolov11 explained: Next-level object detection with enhanced speed and accuracy*, Medium, 2024. [Online]. Available: https://medium.com/@nikhil-rao-20/yolov11-explained-next-level-object-detection-with-enhanced-speed-and-accuracy-2dbe2d376f71

[46] J. Terven, D. Cordova-Esparza, and A. Ramirez-Pedraza, "Yolov11: Advances in real-time object detection and attention mechanisms," *Journal of Computer Vision Research*, vol. 12, no. 2, pp. 88–105, 2024.

[47] Admin. "Understanding resnet-50 in depth: Architecture, skip connections, and advantages over other networks." Accessed: 2025-10-02. [Online]. Available: https://wisdomml.in/understanding-resnet-50-in-depth-architecture-skip-connections-and-advantages-over-other-networks

[48] "Are blind people employable?" Accessed: 2025-06-18. [Online]. Available: https://www.newagebd.net/article/166510/are-blind-people-employable

[49] M. H. I. Bijoy et al., "Bangla braille to voice conversion for visually impaired individuals using deep neural network," *Multimedia Tools and Applications*, vol. 84, no. 40, pp. 48 943–48 976, 2025. DOI: 10.1007/s11042-025-20995-9 [Online]. Available: https://doi.org/10.1007/s11042-025-20995-9

[50] CloudFactory Computer Vision Wiki, *Feature pyramid networks (fpn)*, CloudFactory Wiki, Accessed: 2026-01-27, 2025. [Online]. Available: https://wiki.cloudfactory.com/docs/mp-wiki/model-architectures/fpn

[51] GeeksforGeeks, *U-net architecture explained*, https://www.geeksforgeeks.org/machine-learning/u-net-architecture-explained/, Accessed: 2025-10-02, Jul. 2025.

[52] S. A. Meem and M. Shaheen. "Left behind in the dark: The struggles of visually impaired students in bangladesh." Accessed: 2025-06-19. [Online]. Available: https://www.tbsnews.net/features/panorama/left-behind-dark-struggles-visually-impaired-students-bangladesh-1124151

[53] *Google text-to-speech*, https://cloud.google.com/text-to-speech, Accessed: 2025-06-18.