MAR IVANIOS COLLEGE (AUTONOMOUS)

Mar Ivanios Vidya Nagar, Nalanchira

Thiruvananthapuram – 695015

B.Sc. Computer Science Major Project Report

SIGN LANGUAGE LEARNING SYSTEM

A report submitted in partial fulfilment of the requirement for the award of

BSc. Computer Science

SUBMITTED BY

Sharon Varghese	**2220809**
Abhijith M Nair	**2220810**
Vishnu H	**2220829**

Under the guidance of

Mrs. Tinu C Philip

DEPARTMENT OF COMPUTER SCIENCE

BSc Computer Science

2025

MAR IVANIOS COLLEGE (AUTONOMOUS)

Mar Ivanios Vidya Nagar, Nalanchira

Thiruvananthapuram – 695015

DEPARTMENT OF COMPUTER SCIENCE

<u>CERTIFICATE</u>

This is to certify that the project entitled "**SIGN LANGUAGE LEARNING SYSTEM**" is a bonafide record of the work done by **SHARON VARGHESE** (2220809), **ABHIJITH M NAIR** (2220810), **VISHNU H** (2220829), in partial fulfilment of the requirements for the award of the Degree of Bachelor of Science in Computer Science by the University of Kerala.

INTERNAL GUIDE HEAD OF THE DEPARTMENT

EXTERNAL EXAMINERS

1.

2.

ACKNOWLEDGEMENT

With the profound gratitude to the **ALMIGHTY,** I take this chance to thank people who helped us to complete this project.

We take this as a right opportunity to say **THANKS** to our **PARENTS** who are there to stand with us always with the words "YOU CAN".

We are thankful to Principal **Dr. MEERA GEORGE**, **Mar Ivanios College** who gave us the platform to establish ourselves to reach greater heights.

We earnestly thank **Dr. K. OOMMACHAN**, Director, who always encourages us to do novel things and providing all facilities.

We express our sincere gratitude to **Mrs. TINU C PHILIP**, Head, Department of Computer Science, for her valuable guidance and support to execute all incline in learning.

It is our delight to thank our project guide **Mrs. TINU C PHILIP**, Assistant Professor, Department of Computer Science for her help, support, encouragement, suggestions and guidance throughout the development phases of the project.

We convey our gratitude to all the faculty members of the department who extended their support through valuable comments and suggestions during the reviews.

A great note of gratitude to friends and people who are known and unknown for us and helped in carrying out this project work a successful one.

SHARON VARGHESE

ABHIJITH M NAIR

VISHNU H

ABSTRACT

This project focuses on creating a Real-time Sign Language Recognition system that uses a camera to capture and interpret hand gestures, translating them into text instantly. The system aims to recognize hand signs corresponding to all standard letters of the American Sign Language (ASL) alphabets, from A to Z, as well as numbers from 0 to 9. By leveraging advanced computer vision and machine learning techniques, the model ensures accurate and efficient recognition, enabling smooth real-time interaction. Beyond the technical model, the project focuses on integrating the recognition system into a web application designed to be accessible and engaging, particularly for children and individuals who are interested in learning and practicing ASL sign alphabets. Through this web application they can learn sign language which consists of all the standard ASL letters from A to Z and also the numbers from 0 to 9. The application will also consist of a blog page which contains the latest articles and news related to sign language and deafness. The user can click on the corresponding articles or links to visit their respective pages and read the desired contents. The application will serve as both an educational tool and a bridge for communication, providing users with a way to understand and learn sign language effectively. The goal is to bridge the communication gap between sign language users and those who are unfamiliar with it by offering a solution that supports learning the ASL sign alphabets and numbers. This initiative hope to address a significant gap by combining real-time technological innovation with a practical application tailored for real-world use.

LIST OF FIGURES

LIST OF ABBREVIATIONS

HTML	Hypertext Markup Language
SSD	Solid State Drive
RAM	Random Access Memory
CPU	Central Processing Unit
GUI	Graphical User Interface
CSS	Cascading Style Sheets
AJAX	Asynchronous JavaScript and XML
YOLO	You Only Look Once
NAS	Neural Architecture Search

TABLE OF CONTENTS

CHAPTER 1: INTRODUCTION

Sign language is an expressive visual language that enables communication through a combination of hand gestures, facial expressions, and body movements, instead of using spoken words. It serves as a crucial tool for individuals who are deaf, dumb, or have hearing impairments, enabling them to convey their emotions and communicate effectively. However, while sign language is significant for the deaf and hard-of-hearing community, it poses a great challenge for those who are unfamiliar with it creating barriers between them to communicate and connect with those who rely on sign language as their source of communication.

To address this problem, we propose a real time sign language recognition system, which will interpret the sign language gestures and convert them into text in real-time. This system aims to bridge the communication gap between sign language users and non-users. By using computer vision and machine learning techniques, the system is capable of capturing hand movements through a camera, processing these inputs in real-time to recognize the specific signs accurately. This ensures a seamless translation of converting visual language into a textual format that is universally understandable.

This proposed system is then implemented into an educational web application which is designed to be easily accessible particularly for children and individuals who are interested in learning and practicing ASL sign alphabets. The web application provides a learning page which consists of all the standard ASL letters and the numbers from 0 to 9. It helps in empowering children who are deaf and dumb to learn and practise sign language effectively and also to familiarize themselves with ASL sign alphabets.

The application will also consist of a blog page which contains the latest articles and news related to sign language. The users can click on the corresponding articles or links to visit their respective pages. Through real-time recognition, learning and translation capabilities, this system has the potential to transform how people perceive and engage with sign language.

1.1 PROJECT CATEGORY

The proposed Sign Language Learning System is a Research-Based Project. A research-based project is a project that is designed to investigate a particular topic or question using research methods and techniques. In a research-based project, the focus is on gathering and analyzing data to answer a research question or test a hypothesis. This project aims to develop and refine a system that is capable of accurately recognizing and interpreting sign language gestures in real-time.

1.2 OBJECTIVE

The objective of this project is to create a real-time sign language recognition system that uses a camera to capture and interpret hand gestures, translating them into text in real time. The system aims to recognize hand signs corresponding to all standard letters of the American Sign Language (ASL) alphabets as well as the numerals from 0 to 9.

The system serves as an educational platform by implementing it into a web application which is designed to be user friendly particularly for children and individuals who wish to learn and practice ASL sign alphabets. The application provides a learning page which consists of all the standard ASL letters, from A to Z, and the numbers from 0 to 9, offering children a platform to learn the language.

The application will also provide a blog page which contains the latest articles and news related to sign language. Overall, the objective is to offer a solution to efficiently translate ASL sign alphabets into text in real time and also to empower children who have hearing impairments to learn and practise ASL alphabets.

1.3 SCOPE OF THE PROJECT

The project aims to develop a real-time sign language recognition system using computer vision and machine learning. It involves collecting a comprehensive dataset of sign language signs, preprocessing the data to enhance quality, and training models to accurately recognize signs in real-time. This system is then implemented into an educational web application which is designed to be easily accessible particularly for children and individuals who are interested in learning and practicing ASL sign alphabets. A user-friendly interface is designed to provide intuitive and simple interactions for the children to learn the

ASL signs. Our main aim is to help children with speech impairments learn ASL sign language and break through communication barriers in social life.

1.4 IDENTIFICATION OF NEED

- **Promote Sign Language Education:** Offers a platform for children to learn and practise sign language by offering a learning page which includes real-time translation.
- **Bridge Communication Gaps:** Focuses on breaking down language barriers between sign language users and non-users.
- **Foster Awareness and Sensitivity:** Raising awareness about the importance of sign language and its role in connecting people through a specially designed blog page.
- **Empowers Deaf and Hard of Hearing Individuals:** Provides a tool that translates ASL sign language into text, enhancing communication with non-signers.

1.5 UNIQUE FEATURES OF THE SYSTEM

- **Environmental Robustness:** The system offers effective operations in diverse lighting conditions and against various backgrounds irrespective of the hand shape, size and skin tone of the users.
- **User-friendly interface**: The system offers an intuitive and user friendly interface where the individuals and children can easily access the different options and learn and practice ASL alphabets in real-time.
- **Learning and Practice Modes:** The proposed system offers the facility to learn ASL sign language by providing corresponding images to each alphabet and numerals from 0 to 9 of ASL and also to practise them in real time.

CHAPTER 2: LITERATURE SURVEY

This paper discusses a gesture-to-text translation system using YOLO NAS and deep learning combines computer vision and NLP to enable real-time sign language recognition and text conversion, aiding the deaf and hard-of-hearing community. This approach highlights the potential for inclusive communication but requires further refinement for broader real-world application. Despite its potential, challenges such as gesture movements and variation in signing styles persist. This method's transformative potential for inclusive communication, while also emphasizing the need for further optimization to improve usability and scalability in diverse real-world scenarios [1].

The study highlighted the importance of focusing on the convolutional neural network layer as the detector, suggesting that more attention should be paid to this aspect for improved performance. The dataset used in the study may have contributed to the observed outcomes, with variations in backdrop lighting leading to accuracy issues. The researchers noted that these lighting variations could be leveraged as a benefit for machine learning training, emphasizing the importance of training on the correct features [2].

This study explores real-time hand gesture recognition for Turkish Sign Language using an optimized YOLO algorithm. Transfer learning further accelerates training, enabling efficient detection and recognition of static hand signs. This enhanced YOLO offers a high-performance solution for real-time Turkish Sign Language detection, demonstrating advancements in speed [3].

This paper proposes a Python-based software that converts hand gestures into natural languages, such as English, by processing live camera feeds. Utilizing convolutional neural networks (CNNs), pre-trained on large or custom datasets, the system classifies gestures into corresponding alphabets or numbers from the American Sign Language set. This tool aims to bridge communication gaps, enabling accurate and seamless understanding of sign language gestures in real time [4].

This study utilises a Skeleton-based graph techniques to identify the signs. It includes a Sign Language Graph Convolution Network (SL-GCN) to capture skeleton dynamics and a Separable Spatial-Temporal Convolution Network (SSTCN) to refine skeleton features.

Late-fusion GEM integrates skeleton-based predictions with RGB and depth modalities, enhancing global information for accurate SLR predictions [5].

This study developed a dataset and a Convolutional Neural Network-based sign language interface system to interpret gestures of sign language and hand poses to natural language. The neural network used in this study is CNN, which enhances the predictability of the American Sign Language alphabet (ASL). The dataset is unique because it includes varied conditions like different lighting and distances. The results show that this system works well and could be used in medical applications and other areas needing sign language recognition [6].

This paper focuses on creating a system that translates sign language into voice and text, helping bridge the communication gap between sign language users and others. The system analyzes video sequences to extract both spatial and temporal features. It uses MediaPipe Holistic to detect facial, hand, and body landmarks for spatial features. For temporal features, it trains models like LSTM (Long Short-Term Memory) and GRU (Gated Recurrent Unit). The results show that combining MediaPipe Holistic with GRU or LSTM works well for real-time sign language recognition. This research aims to support the development of intelligent sign language recognition systems and provide guidance for future improvements [7].

The proposed project aims to provide a more individualized learning experience that empowers children to take control of their learning process, making them more independent and confident in their abilities. This project seeks to bridge the gap between technology and inclusivity by creating a tool that not only facilitates the learning of sign language but also real time translation of the corresponding sign letters bridging the communication gap between diverse groups of people.

CHAPTER 3: SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

- **Hardware Constraints:** Many existing systems rely on complex hardware, such as motion-capture gloves, multiple cameras, or depth sensors. This makes the technology expensive and inaccessible to many users.

- **Lack of integration into an Educational platform:** The existing systems in this field have only developed a sign language recognition model, not integrating it into an educational web application which would be easily accessible to the children.
- **Lack of Learning possibilities:** While most of the systems have successfully developed a recognition model, they have not provided options for learning the corresponding sign language.
- **Accuracy and Latency issues:** Many existing systems are still facing the issues of low accuracy of sign language detection in real-time applications. Achieving a system with low latency is still challenging, especially for computers running on less powerful hardware.

3.2 PROPOSED SYSTEM

- **Web Application for Sign Language Recognition model:** The proposed system is a web application designed to efficiently and accurately detect the ASL signs in real time, regardless of hand shapes, sizes, or skin tones.
- **Educational Platform:** The proposed system is integrated into an educational platform that offers the facility to learn ASL sign language by providing corresponding images to each alphabet and numerals from 0 to 9 of ASL.
- **User-friendly interface**: The system offers an intuitive and user friendly interface where the individuals and children can easily access the different options and learn and practice ASL alphabets in real-time.
- **Dedicated Blog page:** The proposed system will also consist of a dedicated blog page which consists of latest articles and news related to sign language and deafness.

The users can click on the corresponding articles or links to visit their respective pages.

- **Accessibility and Affordability:** Designed for broad accessibility, the web application works across multiple platforms, ensuring ease of use in various environments. By leveraging open-source technologies, the application remains affordable for institutions as well as individuals**.**

3.3 FEASIBILITY STUDY

Before the project is established, a feasibility study is conducted to determine the chances of success for the proposed Sign Language Learning System. A feasibility study is necessary to determine whether creating a new or improved system is compatible with costs, benefits, operation, and technology.

3.3.1 Technical Feasibility

- **Dataset Feasibility:** The dataset used in the proposed system is the ASL Letters dataset which was curated by *David Lee,* a data scientist. It is available publicly through the Roboflow platform.
- **Algorithm Feasibility:** The proposed system utilizes You Only Look Once-Neural Architecture Search [YOLO NAS] machine learning algorithm for gesture recognition. It is a Convolutional Neural Network (CNN) based algorithm known for its efficiency and real time performance in object detection.
- **Scalability:** Python libraries such as TensorFlow and OpenCV support scalable solutions and helps in real-time computer vision operations. These libraries play a huge role in integrating the proposed system into a web application.

3.3.2 Social Feasibility

- **Improved Accessibility:** This system enhances communication for individuals with hearing impairments by bridging the gap between sign language users and non-signers. By translating sign gestures into text in real time, it enables seamless accessibility and interaction in personal, educational and professional purposes.

- **Educational Benefits:** The system serves as a learning tool, by helping deaf and dumb children to learn and improve their sign language skills while fostering confidence and diminishing the gap between them and non-signers.
- **Wider impact:** While the system is primarily designed for accurately recognizing the ASL alphabets in real time, it can also be beneficial in other fields such as in academic purposes for teaching the special students ASL alphabets or in crime investigation fields to properly interpret the sign alphabets used by culprits in real time.
- **Awareness and Social Inclusion:** The project aims to promote inclusivity and awareness by breaking down communication barriers in schools, workplaces, and public spaces. It allows hearing-impaired individuals to participate more fully in various environments, ensuring they can access opportunities and services independently and confidently.

3.3.3 Operational Feasibility

- **Ease of use for end users:** The user interface will be designed in a manner that is simple and easy to use for children without the need of parent guidance. The process of navigating through the learning and practicing page will be seamless and efficient
- **Scalability and Compatibility:** The system is scalable in the sense that it can recognize the hand gestures regardless of the hand shape, size and skin tone. The system that is integrated into a web application is also compatible with computer systems that are running on low or less powerful hardware and software.

3.3.4 Economic Feasibility

- **Low Development Costs**: Since the system uses the default webcam of the computing device for sign recognition, the cost of an external camera, sensors or devices for sign recognition can be avoided, therefore making it economically feasible. Also the machine learning model, datasets etc... used in this system do not require an expensive license making the solution more affordable.
- **Affordable for different use cases:** The system can be easily operated on homely devices such as a laptop or a computer making it more affordable for people who are interested in learning ASL from their homes rather than going to an external institution. The system also helps in eliminating the cost of hiring human interpreters

or sign language teachers, by properly interpreting the sign gestures in real-time making it more affordable for different scenarios.

CHAPTER 4: SYSTEM REQUIREMENTS SPECIFICATION

4.1 SOFTWARE REQUIREMENTS

4.1.1 Django

Django is a high-level, open-source web framework written in Python that is well-suited for building robust and scalable web applications, making it an excellent choice for a real-time sign language recognition project. Its ability to seamlessly integrate with Python-based machine learning libraries such as TensorFlow, PyTorch, and OpenCV allows developers to easily incorporate the core recognition model into the platform. Django's flexibility ensures smooth integration of AI-driven functionalities while maintaining efficient performance. The framework's built-in features, such as its ORM (Object-Relational Mapping) system, simplify database management, making it easier to store user data, learning progress, and system outputs.

Django is designed for scalability, ensuring that the platform can handle increasing user loads as adoption grows, whether in educational institutions, workplaces, or public spaces. It also accelerates development by providing pre-built components like authentication systems, admin interfaces, and URL routing, enabling developers to focus on the unique features of the project. Security is another strength of Django, with protections against common vulnerabilities such as SQL injection and cross-site scripting, ensuring a safe and reliable environment for users. Additionally, its active community and comprehensive documentation offer extensive support for troubleshooting and implementing advanced features. For real-time functionalities, Django can be extended with tools like Django Channels, enabling live feedback and interaction essential for gesture recognition. Overall, Django's robust architecture, ease of use, and compatibility with machine learning workflows make it an ideal framework for this project.

4.1.2 AJAX

AJAX (Asynchronous JavaScript and XML) will be used on the frontend to facilitate seamless communication between the user and the server. With AJAX, the web application will be able to exchange data with the server without requiring a full page reload, making the user experience faster and more interactive. It allows the client application to capture gesture data, such as image frames and send it to the server for processing. The server, equipped with machine learning models, analyzes the data to recognize the gestures and translates them into text. AJAX retrieves the translated output from the server and dynamically updates the user interface, displaying the recognized sign language in real time. This ensures a seamless experience for users as the translation appears instantly without any disruption to the application flow.

The ability to update parts of a webpage asynchronously is one of the key advantages of using AJAX. It ensures that users can interact with the application in real-time without disruptions. Its integration with other front-end technologies like JavaScript, HTML, and CSS ensures a cohesive user experience, where dynamic content updates are seamlessly woven into the application. AJAX is an essential component of a real-time sign language recognition system, ensuring asynchronous, efficient, and dynamic communication between the client and server. It enables gesture data processing, real-time translation and resource fetching, creating an interactive and user-friendly web application.

4.1.3 Testing Frameworks

Pytest is a powerful testing framework that can be seamlessly integrated with Django to test the backend functionality. It provides a simple and flexible structure for writing and running tests, including those for Django models, views, and APIs. Pytest can be used to automate tests for server-side components, such as verifying the correctness of gesture processing algorithms or ensuring the server responds to AJAX requests efficiently. Test cases could cover scenarios like validating the recognition of specific sign language gestures, checking that the correct text output is generated, and ensuring that error handling mechanisms (such as unrecognized gestures) work as expected. With PyTest, we can automate the testing process, run test cases with different inputs, and easily handle any potential edge cases.

4.2 HARDWARE REQUIREMENTS

- **Camera:** A default webcam or an external camera is required.
- **CPU:** A processor with sufficient processing power to handle concurrent user requests.
- **GPU:** A minimum of 4 GB to ensure smooth performance is preferred.
- **RAM:** A minimum of 4 GB of RAM to ensure smooth performance is preferred.
- **Storage:** An SSD (Solid-State Drive) or HDD (Hard-Disk Drive) storage for fast data access is preferred.

4.3 LANGUAGE DESCRIPTION

4.3.1 HTML

HTML is the standard markup language for creating web pages and web applications. It defines the structure and layout of web content using elements and tags. HTML is not a programming language but rather a markup language for structuring web documents. HTML documents consist of a collection of elements, each represented by tags that denote the beginning and end of the element. These elements can include headings, paragraphs, images, links, forms, lists, and more. Tags are crucial in defining the relationships between different parts of the content and specifying how they should be rendered on the user's browser. Web developers use HTML as a foundation for creating responsive and engaging websites and web applications. The language's simplicity, along with its ability to integrate seamlessly with other technologies, makes it an essential tool for building a wide range of digital experiences. As the internet continues to evolve, HTML remains a cornerstone for shaping the structure and presentation of online content.

4.3.2 CSS (Cascading Style Sheets)

CSS is used for styling web pages and controlling the presentation of HTML elements. It allows developers to define the colours, fonts, spacing, and layout of web content. CSS is crucial for creating visually appealing and responsive web designs. Selectors are a key component of CSS, specifying which HTML elements are targeted by a particular style rule. CSS provides a wide range of selectors, including element selectors, class selectors, and ID selectors, allowing for fine-grained control over styling.

Media queries in CSS enable responsive web design by allowing developers to apply different styles based on characteristics such as screen size, resolution, or device orientation. This ensures a consistent and optimal user experience across various devices and screen sizes.

4.3.3 JavaScript

JavaScript is a versatile programming language primarily used for adding interactivity and behaviour to web pages. It can be executed in web browsers, making it ideal for creating dynamic web applications. JavaScript is an essential component of modern web development. Being an interpreted language, JavaScript executes code on-the-fly, without the need for a compilation step. This characteristic allows for quick development cycles and immediate feedback during the coding process. Additionally, JavaScript is dynamically typed, meaning variables can hold values of any type without explicit type declarations. While this flexibility can lead to concise code, developers need to be mindful of potential type-related issues.

JavaScript's event-driven and asynchronous nature is another notable feature. It responds to various events triggered by user interactions, such as clicks or keyboard input. The language's asynchronous capabilities, facilitated by features like callbacks and promises, enable the execution of non-blocking code. This is particularly valuable for handling tasks like fetching data from servers without freezing the user interface.

Beyond the browser, JavaScript has expanded its reach through environments like Node.js, allowing developers to use the language for server-side programming. This versatility has made JavaScript a prominent language in full-stack development, where it can be employed on both the client and server sides, providing a unified language stack for building modern web applications. Overall, JavaScript's adaptability, coupled with its vibrant ecosystem of libraries and frameworks, has solidified its position as a foundational technology in the world of web development.

4.3.4 Python

Python is a high-level, interpreted, and general-purpose programming language known for its simplicity, readability, and versatility. It is one of the most widely used programming languages across various domains, including web development, data analysis, machine learning, artificial intelligence, automation, and scientific computing. Python’s design philosophy emphasizes code readability, making it easier for developers to write clean and maintainable code.

Python supports multiple programming paradigms, including object-oriented, imperative, functional, and procedural programming. It provides a rich standard library, which offers modules and packages that handle various tasks such as file I/O, system operations, and even web development. Python’s syntax is clean and straightforward, with a strong emphasis on indentation rather than braces or semicolons, which makes code easier to understand and maintain.

One of Python's key strengths is its large ecosystem of third-party libraries and frameworks. For example, in the context of this project, Python can be used for building the backend server using Flask or Django, handling image processing using libraries like OpenCV, and integrating machine learning frameworks such as TensorFlow or PyTorch for the AI model. Python also supports integration with other languages like C, C++, and Java, allowing developers to leverage high-performance code when needed.

Python is dynamically typed, meaning you don’t need to declare variable types explicitly. This makes development faster, but it also requires more attention during testing and debugging. Python is interpreted, meaning that the code is executed line by line, which helps during the development process but might impact performance in computationally heavy applications. However, Python's vast libraries, efficient memory management, and ease of use make it suitable for rapid prototyping and large-scale application development alike. Due to its ease of learning, large community support, and widespread adoption, Python is often chosen for both small and large-scale projects. Its flexibility and powerful libraries make it an excellent choice for a wide range of applications, from simple scripts to complex AI systems.

4.4 DATASETS AND ALGORITHMS

4.4.1 American Sign Language Letters Dataset

The dataset used in the proposed system is the American Sign Language [ASL] letters dataset which was curated by *David Lee,* a data scientist and is publicly available through the Roboflow platform. This dataset consists of hand signs for each alphabet of the ASL making it valuable for developing and evaluating algorithms in real time sign detection fields. It can be easily downloaded through the Roboflow platform.

Dataset Description:

- The dataset consists of a total number of 1728 images.
- It includes sign images of all 26 letters of ASL and 10 digits [0 to 9], resulting in approximately 36 classes.

Data Source:

- The ASL Letters dataset was curated and released by *David Lee*, a data scientist. It is publicly available on the Roboflow platform.
- This dataset provides diverse images of hand gestures representing individual letters, and numbers making it a valuable resource for training the YOLO NAS model.

Data Collection:

- The data collection process involves collecting the dataset from the Roboflow platform and annotating each data image with a bounding box that marks the area of fingers or hand position.
- The annotations of the dataset can be done by using Labellmg software. It provides an interface to draw bounding boxes and assign class labels to objects in the image.

Data Preprocessing:

- Data preprocessing involves the steps that is needed to prepare the ASL Letters dataset for YOLO NAS model training. Roboflow provides tools to manage and preprocess this dataset to improve model performance.
- Some of the dataset preprocessing steps include:
    i. **Resizing:** Images are resized to a standard dimension to maintain consistency during model training.

ii. **Normalization:** Pixel values are normalized to a range of 0 to 1 to improve the ease of training the machine learning model.

iii. **Data Augmentation:** Additional augmentations techniques like rotation, scaling, cropping and lighting adjustments are applied to increase dataset variability and robustness.

iv. **Bounding Box Adjustment:** Ensuring that the bounding boxes are correctly aligned with the hand gestures in the dataset images.

Data Features:

- The ASL Letters dataset provides some important features for detecting and recognizing ASL sign letters. Some of the features are:

  i. **Bounding Boxes:** Highlight the location of the hand gesture, making it easier for the model to learn relevant features.

  ii. **Hand Shape and Finger Position:** Information on the hand's shape and finger position, essential for distinguishing between different ASL letters.

  iii. **Consistency:** Each image consistently focuses on a single ASL letter, which helps in building a model specifically to ASL letter recognition.

4.4.2 You Only Look Once-Neural Architecture Search [YOLO NAS]

In this project, YOLO NAS (You Only Look Once-Neural Architecture Search) is the algorithm that is used to train the ASL dataset. YOLO NAS, a Convolutional Neural Network (CNN) based algorithm, is well-known for its efficiency and real-time performance in object detection. It processes images in a single forward pass through its architecture, which is optimized for speed and accuracy. By employing Neural Architecture Search, the YOLO NAS model automatically refines its network architecture, enhancing its capability to detect the sign images with a high level of precision while maintaining computational efficiency.

This algorithm is particularly suited for this project as it provides a balance between performance and resource usage, making it ideal for training on the ASL Letters Dataset, which includes a range of hand signs representing different ASL letters and numbers. During real-time operation, image frames captured by a webcam are processed by YOLO NAS, which identifies and localizes the hand gestures within each frame and classifies them into the correct ASL letter.

Its optimized architecture ensures faster inference, making it suitable for real-time applications, even on devices with limited computational resources. This approach allows for effective recognition of ASL signs in real-time, leveraging YOLO's advanced object detection capabilities. Its scalability and efficiency make it ideal for integrating the model into a web-based applications, enhancing accessibility for the deaf and hard-of-hearing communities.

4.5 METHODOLOGY

4.5.1 INPUT PROCESS

The process begins with acquiring image frames of the signer's hand in real time through a camera or webcam. These input frames will serve as the starting point to recognize the correct ASL hand signs signed by the child. These frames are then processed to detect and isolate relevant regions, such as fingers and finger joints which are crucial for interpreting the corresponding ASL sign.

4.5.2 PRE-PROCESSING

This step is crucial as it prepares the input frame for accurate sign detection by cleaning and optimizing it. Preprocessing steps are applied to enhance the quality of the input frame captured by the camera.

- **Image Frame Extraction:** Extracting individual frames from the real time video input to properly analyze hand gestures effectively.
- **Image Resizing:** Resize the image frames to match the input dimensions required by YOLO NAS model for consistent detection performance.
- **Normalization:** It is used to scale the pixel values to a standard range (e.g., 0 to 1) to improve model efficiency and reduce computation time. Normalization ensures that all input data has a similar scale, which helps the YOLO NAS algorithm to process the images more efficiently and reduces the impact of variations in lighting conditions.
- **Data Augmentation:** Apply data augmentation techniques such as flipping, rotation, brightness adjustment to make the model robust against variations in input.
- **Grayscale Conversion:** Grayscale conversion is used to simplify the input data by reducing it to a single intensity channel by converting each pixel into a shade of grey.

Grayscale images will make the processing faster while retaining essential features like edges, shapes, and textures, which are important for gesture recognition.

- **Keypoint Detection:** It is used to extract keypoints such as fingertips, joints or wrist from the image frame. This data helps the recognition model to interpret gestures more accurately, as it focuses on meaningful features instead of processing the entire image.

4.5.3 RECOGNIZING USING YOLO NAS

The pre-processed image frame is then passed into the YOLO NAS model for ASL gesture recognition. YOLO NAS is effective in extracting patterns from images, making it suitable for properly recognizing the hand signs in real time.

The YOLO NAS performs the following tasks:

- **Input:** The pre-processed image frame is passed as an input into the model.
- **Backbone Layer:** This layer will detect features from the input image frame such as edges, strokes and textures specific to each ASL sign. It consists of several convolutional layers depending upon the features extracted.
- **Neck Layer:** This Layer is used to process the detected features using the inbuilt algorithms already present in the YOLO NAS model such as FPN and PAN
    1. **FPN:** It is used to process the image features at different scales. It works by creating a multi-scale feature representation from the input image. It combines high-resolution features with low-resolution, high-level semantic features. This helps the model detect both small and large objects effectively, which is crucial for recognizing gestures in sign language.
    2. **PAN:** PAN is used to improve the flow of information between different network layers by refining features in both top-down and bottom-up directions. This ensures that the model effectively utilizes the features at all levels both fine details, like finger positions, and broader contextual features, like wrist movements. PAN helps the model better understand complex gestures by enabling richer feature representations.

- **Head Layer:** The Head layer is used to make the final object detection predictions based on the features processed from the Neck layer. The Head layers mainly focuses on doing the three following tasks.
    1. **Bounding Box Predictions:** It determine the position of the detected objects such as hands or fingers in sign language within the processed image frame by mapping bounding boxes around them.
    2. **Class Prediction:** It will classify the detected object frame into specific categories or classes. In the proposed system, it will predict the different sign language gestures into distinct classes, such as alphabets [A to Z] or numerals [0 to 9]. The class predictions tell the YOLO NAS model which gesture or sign the object frame corresponds to.
    3. **Objectness Score:** The Objectness score indicates the likelihood that a detected region in the image frame contains an object. A higher score indicates that the model is more confident that the bounding box contains an actual object such as a hand or finger.
- **NMS Post Processing:** NMS or Non-Maximum Suppression is a post processing technique in YOLO NAS model that is used to eliminate duplicate bounding boxes for the same detected object. NMS eliminates redundant boxes by keeping only the one with the highest Objectness score and removing the others. It ensures that the overlapping boxes are reduced to one final detection per sign gesture.
- **Output Layer:** It will finally translate the detected sign gesture in a textual format with a single bounding box drawn around the sign.

CHAPTER 5: SYSTEM DESIGN

5.1 ARCHITECTURE DIAGRAM

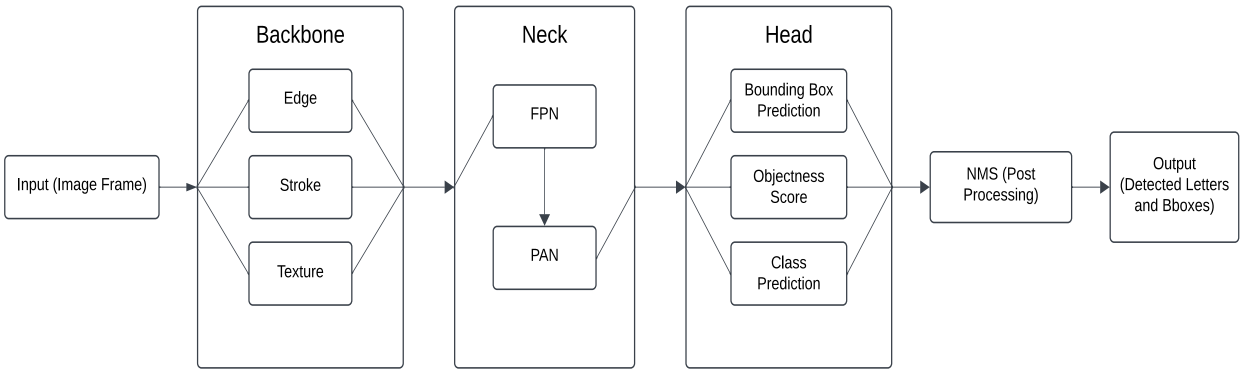


Figure 5.1 Architecture Diagram

5.2 USE CASE DIAGRAM

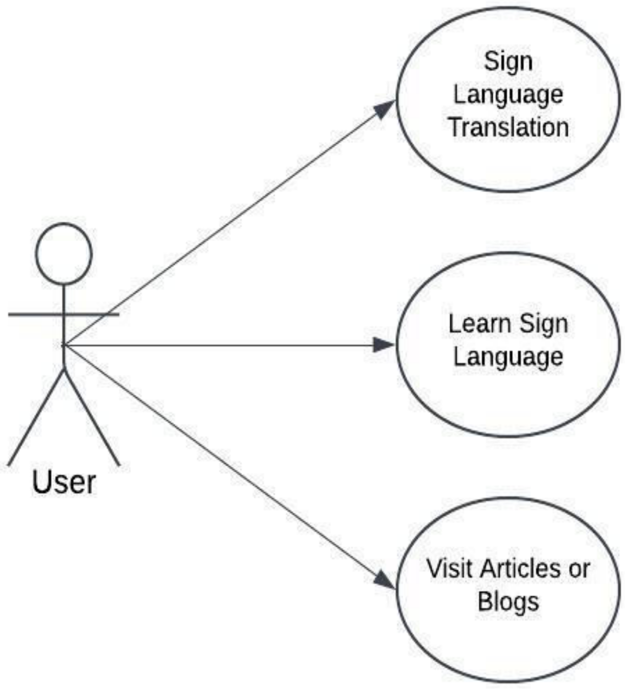


Figure 5.2 Use Case Diagram

CHAPTER 6: SYSTEM IMPLEMENTATION

6.1 TRAINING THE MODEL - LETTERS

```
import keras

import numpy as np

import pandas as pd

import cv2

from keras.models import Sequential

from keras.layers import Conv2D, MaxPooling2D, Dense, Flatten, Dropout

from keras.datasets import mnist

from keras.optimizers import SGD

import os

from pydrive.auth import GoogleAuth

from pydrive.drive import GoogleDrive

from google.colab import auth

from oauth2client.client import GoogleCredentials

train = pd.read_csv('train.csv')

test = pd.read_csv('test.csv')

y_train = train['label'].values

y_test = test['label'].values

X_train = train.drop(['label'], axis=1)

X_test = test.drop(['label'], axis=1)

X_train = np.array(X_train.iloc[:, :])

X_train = np.array([np.reshape(i, (28, 28)) for i in X_train])

X_test = np.array(X_test.iloc[:, :])

X_test = np.array([np.reshape(i, (28, 28)) for i in X_test])

num_classes = 26

y_train = np.array(y_train).reshape(-1)
```

```
y_test = np.array(y_test).reshape(-1)

y_train = np.eye(num_classes)[y_train]

y_test = np.eye(num_classes)[y_test]

X_train = X_train.reshape((27455, 28, 28, 1))

X_test = X_test.reshape((7172, 28, 28, 1))

classifier = Sequential()

classifier.add(Conv2D(filters=8, kernel_size=(3, 3), strides=(1, 1), padding='same',
input_shape=(28, 28, 1), activation='relu', data_format='channels_last'))

classifier.add(MaxPooling2D(pool_size=(2, 2)))

classifier.add(Conv2D(filters=16, kernel_size=(3, 3), strides=(1, 1), padding='same',
activation='relu'))

classifier.add(Dropout(0.5))

classifier.add(MaxPooling2D(pool_size=(4, 4)))

classifier.add(Dense(128, activation='relu'))

classifier.add(Flatten())

classifier.add(Dense(26, activation='softmax'))

classifier.compile(optimizer='SGD', loss='categorical_crossentropy', metrics=['accuracy'])

classifier.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

classifier.fit(X_train, y_train, epochs=50, batch_size=100)

accuracy = classifier.evaluate(x=X_test, y=y_test, batch_size=32)

print("Accuracy: ", accuracy[1])

classifier.summary()

from keras.utils.vis_utils import plot_model

plot_model(classifier, to_file='model_plot.png', show_shapes=True, show_layer_names=True)

!apt install graphviz

!pip install pydot pydot-ng

!echo "Double check with Python 3"

!python -c "import pydot"

plot_model(classifier, show_shapes=True, show_layer_names=True, to_file='model.png')

from IPython.display import Image
```

```
Image(retina=True, filename='model.png')

classifier.save('CNNmodel.h5')

import cv2

import numpy as np

from keras.models import load_model

model = load_model('CNNmodel.h5')

import pandas as pd

import keras

def keras_predict(model, image):

    pred_probab = model.predict(image)[0]

    pred_class = list(pred_probab).index(max(pred_probab))

    return max(pred_probab), pred_class

def keras_process_image(img):

    image_x = 28

    image_y = 28

    img = cv2.resize(img, (image_x, image_y), interpolation=cv2.INTER_AREA)

    img = img.astype('float32')

    img = np.expand_dims(img, axis=-1)

    img = np.expand_dims(img, axis=0)

    return img

train = pd.read_csv(r"C:\Users\shiha\OneDrive\Desktop\New
folder\ALphabet_Sign_language\sign_mnist_train.csv")

test = pd.read_csv(r"C:\Users\shiha\OneDrive\Desktop\New
folder\ALphabet_Sign_language\sign_mnist_test.csv")

train_X = train.drop(['label'], axis=1)

test_X = test.drop(['label'], axis=1)

train_Y = train['label']

test_Y = test['label']

train_X = train.drop(['label'], axis=1)

test_X = test.drop(['label'], axis=1)
```

```
print(test_X.shape)
print(train_X.shape)
train_X = train_X.values.reshape(27455, 784)
test_X = test_X.values.reshape(7172, 784)
train_Y = keras.utils.to_categorical(train_Y, 26)
test_Y = keras.utils.to_categorical(test_Y, 26)
img = test_X[25]
test_img = img.reshape((1, 28, 28, 1))
img_class = model.predict(test_img)
prediction = img_class[0]
classname = img_class[0]
print("Class: ", classname)
np.argmax(img_class)
img_class
import matplotlib.pyplot as plt
img = img.reshape((28, 28))
plt.imshow(img)
plt.title(alphabets[np.argmax(img_class)])
plt.show()
import cv2
import numpy as np
from keras.models import load_model
import string
import mediapipe as mp
model = load_model('CNNmodel.h5')
alphabets = list(string.ascii_uppercase)
def prediction(pred):
    return alphabets[pred]
def keras_predict(model, image):
    pred_probab = model.predict(image)
    x = np.argmax(pred_probab)
    pred_class = alphabets[x]
    return x, pred_class
def keras_process_image(img):
    image_x = 28
    image_y = 28
    img = cv2.resize(img, (image_x, image_y), interpolation=cv2.INTER_AREA)
    img = img.astype('float32')
    img = np.expand_dims(img, axis=-1)
    img = np.expand_dims(img, axis=0)
    return img
def crop_image(image, x, y, width, height):
    return image[y:y + height, x:x + width]
def main():
    mp_hands = mp.solutions.hands
    hands = mp_hands.Hands(static_image_mode=False, max_num_hands=1,
min_detection_confidence=0.5, min_tracking_confidence=0.5)
    mp_drawing = mp.solutions.drawing_utils
    cam_capture = cv2.VideoCapture(0)
    while True:
        ret, image_frame = cam_capture.read()
        if not ret:
            break
        image_rgb = cv2.cvtColor(image_frame, cv2.COLOR_BGR2RGB)
        results = hands.process(image_rgb)
        if results.multi_hand_landmarks:
            for hand_landmarks in results.multi_hand_landmarks:
                x_min, y_min = int(hand_landmarks.landmark[0].x * image_frame.shape[1]),
int(hand_landmarks.landmark[0].y * image_frame.shape[0])
```

```
                x_max, y_max = x_min, y_min
                for lm in hand_landmarks.landmark:
                    x, y = int(lm.x * image_frame.shape[1]), int(lm.y * image_frame.shape[0])
                    x_min, y_min = min(x_min, x), min(y_min, y)
                    x_max, y_max = max(x_max, x), max(y_max, y)
                margin = 20
                x_min = max(0, x_min - margin)
                y_min = max(0, y_min - margin)
                x_max = min(image_frame.shape[1], x_max + margin)
                y_max = min(image_frame.shape[0], y_max + margin)
                cropped_img = image_frame[y_min:y_max, x_min:x_max]
                image_grayscale = cv2.cvtColor(cropped_img, cv2.COLOR_BGR2GRAY)
                image_grayscale_blurred = cv2.GaussianBlur(image_grayscale, (15, 15), 0)
                processed_img = keras_process_image(image_grayscale_blurred)
                pred_probab, pred_class = keras_predict(model, processed_img)
                cv2.putText(image_frame, pred_class, (x_min, y_min - 10),
cv2.FONT_HERSHEY_COMPLEX, 1.0, (255, 255, 255), lineType=cv2.LINE_AA)
                cv2.rectangle(image_frame, (x_min, y_min), (x_max, y_max), (255, 255, 0), 3)
                mp_drawing.draw_landmarks(image_frame, hand_landmarks,
mp_hands.HAND_CONNECTIONS)
        cv2.imshow("frame", image_frame)
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break
    cam_capture.release()
    cv2.destroyAllWindows()
if __name__ == '__main__':
    main()
```

6.2 TRAINING THE MODEL – NUMBERS

```
import os, cv2, math
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
from keras.preprocessing.image import ImageDataGenerator
from keras.preprocessing import image
from sklearn.model_selection import train_test_split
from shutil import copyfile
from tqdm import tqdm
import matplotlib.pyplot as plt
import numpy as np
nrow, ncol = 2, 5
plt.rcParams['figure.figsize'] = (ncol*3, nrow*3)
for row in range(nrow):
   for col in range(ncol):
      img_index = row*ncol+col
      img = image.load_img('Sign-Language-Digits-Dataset/Examples/example_' + str(img_index) +
'.JPG', target_size=(64, 64))
      plt.subplot(nrow, ncol, img_index + 1)
      plt.imshow(img)
      plt.title(img_index)
      plt.axis('off')
DATASET_PATH = 'Sign-Language-Digits-Dataset/Dataset/'
train_set = {}
validation_set = {}
test_set = {}
for cat in os.listdir(DATASET_PATH):
   cat_dir = os.path.join(DATASET_PATH, cat)
   cat_files = os.listdir(cat_dir)
```

```
    train_list, test_list = train_test_split(cat_files, test_size=0.3)
    validation_list, test_list = train_test_split(test_list, test_size=0.5)
    train_set[cat] = train_list
    validation_set[cat] = validation_list
    test_set[cat] = test_list
for cat in tqdm(train_set.keys()):
    cat_dir = os.path.join(DATASET_PATH, 'training_set', 'class_0' + str(cat))
    os.makedirs(cat_dir)
    for file in train_set[cat]:
        src = os.path.join(DATASET_PATH, cat, file)
        dest = os.path.join(cat_dir, file)
        copyfile(src, dest)
for cat in tqdm(validation_set.keys()):
    cat_dir = os.path.join(DATASET_PATH, 'validation_set', 'class_0' + str(cat))
    os.makedirs(cat_dir)
    for file in validation_set[cat]:
        src = os.path.join(DATASET_PATH, cat, file)
        dest = os.path.join(cat_dir, file)
        copyfile(src, dest)
for cat in tqdm(test_set.keys()):
    cat_dir = os.path.join(DATASET_PATH, 'test_set', 'class_0' + str(cat))
    os.makedirs(cat_dir)
    for file in test_set[cat]:
        src = os.path.join(DATASET_PATH, cat, file)
        dest = os.path.join(cat_dir, file)
        copyfile(src, dest)
for i in range(10):
    train_size = len(train_set[str(i)])
    validation_size = len(validation_set[str(i)])
    test_size = len(test_set[str(i)])

    print("0{} : Training size({}) Validation size({}) Test size({})".format(i, train_size, validation_size, test_size))

train_datagen = ImageDataGenerator(rescale=1./255, shear_range=0.2, zoom_range=0.2, horizontal_flip=True)

validation_datagen = ImageDataGenerator(rescale=1./255)

test_datagen = ImageDataGenerator(rescale=1./255)

training_data = train_datagen.flow_from_directory(os.path.join(DATASET_PATH, 'training_set'), target_size=(64, 64), batch_size=32, class_mode='categorical')

validation_data = validation_datagen.flow_from_directory(os.path.join(DATASET_PATH, 'validation_set'), target_size=(64, 64), batch_size=32, class_mode='categorical')

test_data = test_datagen.flow_from_directory(os.path.join(DATASET_PATH, 'test_set'), target_size=(64, 64), batch_size=32, class_mode='categorical')

classifier = Sequential()

classifier.add(Conv2D(32, (3, 3), input_shape=(64, 64, 3), activation='relu'))

classifier.add(MaxPooling2D(pool_size=(2, 2)))

classifier.add(Dropout(0.25))

classifier.add(Conv2D(32, (3, 3), activation='relu'))

classifier.add(MaxPooling2D(pool_size=(2, 2)))

classifier.add(Dropout(0.25))

classifier.add(Conv2D(32, (3, 3), activation='relu'))

classifier.add(MaxPooling2D(pool_size=(2, 2)))

classifier.add(Dropout(0.25))

classifier.add(Flatten())

classifier.add(Dense(units=128, activation='relu'))

classifier.add(Dense(units=10, activation='softmax'))

classifier.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

history = classifier.fit_generator(training_data, steps_per_epoch=math.ceil(training_data.n / training_data.batch_size), epochs=100, validation_data=validation_data, validation_steps=math.ceil(validation_data.n / validation_data.batch_size))

plt.plot(history.history['accuracy'])

plt.plot(history.history['val_accuracy'])
plt.title('Model Accuracy')

plt.ylabel('accuracy')

plt.xlabel('epoch')

plt.legend(['Training', 'Validation'])

test_loss, test_accuracy = classifier.evaluate_generator(test_data, math.ceil(test_data.n / test_data.batch_size))

print("Accuracy on test set : {}".format(test_accuracy))

nrow, ncol = 2, 5

plt.rcParams['figure.figsize'] = (ncol*3, nrow*3)

for row in range(nrow):

   for col in range(ncol):

      img_index = row*ncol+col

      img = image.load_img('Sign-Language-Digits-Dataset/Examples/example_' + str(img_index) + '.JPG', target_size=(64, 64))

      test_image = image.img_to_array(img)

      test_image = np.expand_dims(test_image, axis=0)

      result = classifier.predict(test_image).argmax()

      plt.subplot(nrow, ncol, img_index + 1)

      plt.imshow(img)

      plt.title("Actual({}) Predicted({})".format(img_index, result))

      plt.axis('off')

model_json = classifier.to_json()

with open("model.json", "w") as json_file:

   json_file.write(model_json)

classifier.save_weights("model.h5")

print("Saved model to disk")

classifier.save("model_complete.h5")

import cv2

import numpy as np

import tensorflow as tf
```

```
model = tf.keras.models.load_model('model_complete.h5')
def preprocess_frame(frame):
    img = cv2.resize(frame, (64, 64))
    img = img / 255.0
    img = np.expand_dims(img, axis=0)
    return img
def overlay_predictions(frame, predictions):
    label = np.argmax(predictions)
    confidence = predictions[0][label]
    cv2.putText(frame, f"Label: {label}, Confidence: {confidence:.2f}", (10, 30), 
cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2)
    return frame
cap = cv2.VideoCapture(0)
if not cap.isOpened():
    print("Error: Could not open webcam.")
    exit()
print("Press 'q' to quit.")
while True:
    ret, frame = cap.read()
    if not ret:
        print("Error: Unable to capture frame.")
        break
    processed_frame = preprocess_frame(frame)
    predictions = model.predict(processed_frame)
    output_frame = overlay_predictions(frame, predictions)
    cv2.imshow("Sign Language Detection", output_frame)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
cap.release()
```

```
cv2.destroyAllWindows()

import mediapipe as mp

import time

model = tf.keras.models.load_model('model_complete.h5')

mp_hands = mp.solutions.hands

hands = mp_hands.Hands(static_image_mode=False, max_num_hands=1, 
min_detection_confidence=0.7, min_tracking_confidence=0.5)

def preprocess_frame(frame):

    img = cv2.resize(frame, (64, 64))

    img = img / 255.0

    img = np.expand_dims(img, axis=0)

    return img

cap = cv2.VideoCapture(0)

if not cap.isOpened():

    print("Error: Could not open webcam.")

    exit()

print("Press 'q' to quit.")

prev_time = 0

try:

    while True:

        ret, frame = cap.read()

        if not ret:

            print("Error: Unable to capture frame.")

            break

        frame = cv2.flip(frame, 1)

        rgb_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)

        results = hands.process(rgb_frame)

        predictions = None

        if results.multi_hand_landmarks:
```

```
        for hand_landmarks in results.multi_hand_landmarks:
            mp_drawing.draw_landmarks(frame, hand_landmarks, mp_hands.HAND_CONNECTIONS)
            h, w, _ = frame.shape
            x_min, y_min = w, h
            x_max, y_max = 0, 0
            for lm in hand_landmarks.landmark:
                x, y = int(lm.x * w), int(lm.y * h)
                x_min = min(x_min, x)
                y_min = min(y_min, y)
                x_max = max(x_max, x)
                y_max = max(y_max, y)
            padding = 20
            x_min = max(x_min - padding, 0)
            y_min = max(y_min - padding, 0)
            x_max = min(x_max + padding, w)
            y_max = min(y_max + padding, h)
            hand_crop = frame[y_min:y_max, x_min:x_max]
            processed_frame = preprocess_frame(hand_crop) if hand_crop.size > 0 else np.zeros((64, 64, 3))
            predictions = model.predict(processed_frame)
            label = np.argmax(predictions)
            confidence = predictions[0][label]
            cv2.rectangle(frame, (x_min, y_min), (x_max, y_max), (0, 255, 0), 2)
            label_text = f"Label: {label}, Confidence: {confidence:.2f}"
            label_box_y = y_min - 35 if y_min - 35 > 20 else y_min + 35
            cv2.rectangle(frame, (x_min, label_box_y - 5), (x_max, label_box_y + 25), (0, 255, 0), -1)
            cv2.putText(frame, label_text, (x_min + 5, label_box_y + 15), cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 0), 2)
    else:
            cv2.putText(frame, "No hand detected", (10, 30), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255), 2)

        curr_time = time.time()

        fps = 1 / (curr_time - prev_time)

        prev_time = curr_time

        cv2.putText(frame, f"FPS: {fps:.2f}", (10, 70), cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 0, 0), 2)

        cv2.imshow("Hand Sign Detection", frame)

        if cv2.waitKey(1) & 0xFF == ord('q'):

            break

finally:

    cap.release()

    cv2.destroyAllWindows()
```

6.3 CODE FOR DETECTION - LETTERS

```
{% load static %}

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">

    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.15.4/css/all.min.css">

    <title>Sign Language Detection</title>

    <link rel="stylesheet" href="style.css">

</head>

<style>

    :root { --primary-color: #3b82f6; --danger-color: #ef4444; --warning-color: #f59e0b; --text-color: #1f2937; --bg-color: #f9fafb; --border-color: #e5e7eb; --hover-color: #2563eb; }
```

```
* { margin: 0; padding: 0; box-sizing: border-box; }

body::-webkit-scrollbar { display: none; }

body { background: linear-gradient(rgba(0, 0, 0, 0.5), rgba(0, 0, 0, 0.5)), url("{% static 'images/background5.webp' %}"); background-color: #1e1e1e; background-size: cover; background-position: center; background-attachment: fixed; background-repeat: no-repeat; color: #e0e0e0; overflow-x: hidden; }

.container { display: grid; grid-template-columns: 1fr 1fr; gap: 2rem; max-width: 1400px; margin: 0 auto; padding: 2rem; min-height: 100vh; }

h1, h2 { margin-bottom: 1.5rem; font-weight: 600; text-align: center; }

h1 { font-size: 2rem; color: var(--text-color); }

h2 { font-size: 1.5rem; }

.video-section { display: flex; flex-direction: column; gap: 1.5rem; }

.video-wrapper { position: relative; }

.video-placeholder { position: absolute; inset: 0; display: flex; align-items: center; justify-content: center; color: white; font-size: 1.25rem; }

#video { width: 100%; height: 100%; object-fit: cover; }

.controls { display: flex; gap: 1rem; justify-content: center; }

.btn { display: flex; align-items: center; gap: 0.5rem; padding: 0.75rem 1.5rem; border: none; border-radius: 0.5rem; font-weight: 500; cursor: pointer; transition: all 0.2s; }

.btn-primary { background-color: var(--primary-color); color: white; }

.btn-danger { background-color: var(--danger-color); color: white; }

.btn-warning { background-color: var(--warning-color); color: white; }

.btn:hover:not(:disabled) { transform: translateY(-1px); filter: brightness(110%); }

.btn:disabled { opacity: 0.6; cursor: not-allowed; }

.letters-section { background: white; border-radius: 1rem; padding: 1.5rem; box-shadow: 0 4px 6px -1px rgba(0, 0, 0, 0.1); }

.letters-grid { max-height: 100px; overflow-y: auto; padding-right: 0.5rem; }

.letters-list { display: grid; grid-template-columns: repeat(auto-fill, minmax(60px, 1fr)); gap: 0.75rem; }

.letter { aspect-ratio: 1; display: flex; align-items: center; justify-content: center; background: var(--bg-color); border-radius: 0.5rem; font-size: 1.25rem; font-weight: 500; cursor: pointer; transition: all 0.2s; }

.letter:hover { background: var(--primary-color); color: white; transform: translateY(-2px); }

.active { background: var(--primary-color); color: white; transform: translateY(-2px); }
```

```
.preview-section { background: white; border-radius: 1rem; padding: 1.5rem; margin-top: 1.5rem;
box-shadow: 0 4px 6px -1px rgba(0, 0, 0, 0.1); color: black; }

.preview-container { aspect-ratio: 4/2; display: flex; align-items: center; justify-content: center;
background: var(--bg-color); border-radius: 0.5rem; overflow: hidden; }

#letter-image { height: 300px; object-fit: contain; }

.tooltip-container { position: relative; min-height: 2rem; }

.tooltip { position: absolute; left: 50%; transform: translateX(-50%); background: rgba(0, 0, 0, 0.8);
color: white; padding: 0.5rem 1rem; border-radius: 0.25rem; font-size: 0.875rem; display: none; }

@media (max-width: 1024px) { .container { grid-template-columns: 1fr; } .video-section { order: -
1; } }

.navbar { background-color: #333; }

.navbar-brand, .nav-link { color: #e0e0e0; }

.navbar-brand:hover, .nav-link:hover { color: #007bff; }

.navbar-nav .dropdown-menu { background-color: #2c2c2c; border: 1px solid #3a3a3a; color:
#e0e0e0; white-space: nowrap; }

.navbar-nav .dropdown-menu .dropdown-item { color: #e0e0e0; }

.navbar-nav .dropdown-menu .dropdown-item:hover { background-color: #007bff; color: #fff; }



</style>

<body>

  <nav class="navbar navbar-expand-lg navbar-dark bg-dark-custom">

    <a class="navbar-brand" href="{% url 'dashboard' %}">Sign Language Learning System</a>

            <button class="navbar-toggler" type="button" data-toggle="collapse" data-
target="#navbarNav" aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle
navigation">

      <span class="navbar-toggler-icon"></span>

    </button>

    <div class="collapse navbar-collapse" id="navbarNav">

      <ul class="navbar-nav ml-auto">

        <li class="nav-item">

          <a class="nav-link" href="{% url 'dashboard' %}">Home</a>

        </li>
```

```
          <li class="nav-item"></li>
            <a class="nav-link" href="{% url 'blogs' %}">Articles</a>
          </li>
                                          <li class="nav-item">
            <a class="nav-link" href="{% url 'test' %}">Letters</a>
          </li>
          <li class="nav-item">
            <a class="nav-link" href="{% url 'number_test_view' %}">Numbers</a>
          </li>
        </ul>
      </div>
    </nav>
    <div class="container">
          <div class="video-section">
              <div class="video-container">
          <div class="video-wrapper">
            <img id="video" src="" alt="Video Stream" style="display: none;">
            <div class="video-placeholder"></div>
          </div>
        </div>
        <div class="controls">
          <button id="predict-btn" class="btn btn-primary">
            <span class="btn-icon">▶</span>
            Start Prediction
          </button>
          <button id="stop-btn" class="btn btn-danger" disabled>
            <span class="btn-icon"> ⏸ </span>
            Pause
          </button>
        <button id="restart-btn" class="btn btn-warning">
            <span class="btn-icon">■</span>
            Stop Prediction
        </button>
    </div>
    <div class="tooltip-container">
        <p id="tooltip" class="tooltip"></p>
    </div>
</div>
        <div class="content-section">
    <div class="preview-section">
        <h2>Preview</h2>
        <div class="preview-container">
            <p id="default-message">Select any letter to view the sign</p>
            <img id="letter-image" src="" alt="Sign language preview" style="display: none;">
        </div>
        <div class="letters-section">
            <h2>Letters</h2>
            <div class="letters-grid">
                <div class="letters-list">
                    <div class="letter" id="letter-A">A</div>
                    <div class="letter" id="letter-B">B</div>
                    <div class="letter" id="letter-C">C</div>
                    <div class="letter" id="letter-D">D</div>
                    <div class="letter" id="letter-E">E</div>
                    <div class="letter" id="letter-F">F</div>
                    <div class="letter" id="letter-G">G</div>
                    <div class="letter" id="letter-H">H</div>
                    <div class="letter" id="letter-I">I</div>
```

```
                        <div class="letter" id="letter-J">J</div>
                        <div class="letter" id="letter-K">K</div>
                        <div class="letter" id="letter-L">L</div>
                        <div class="letter" id="letter-M">M</div>
                        <div class="letter" id="letter-N">N</div>
                        <div class="letter" id="letter-O">O</div>
                        <div class="letter" id="letter-P">P</div>
                        <div class="letter" id="letter-Q">Q</div>
                        <div class="letter" id="letter-R">R</div>
                        <div class="letter" id="letter-S">S</div>
                        <div class="letter" id="letter-T">T</div>
                        <div class="letter" id="letter-U">U</div>
                        <div class="letter" id="letter-V">V</div>
                        <div class="letter" id="letter-W">W</div>
                        <div class="letter" id="letter-X">X</div>
                        <div class="letter" id="letter-Y">Y</div>
                        <div class="letter" id="letter-Z">Z</div>
                    </div>
                </div>
            </div>
        </div>
    </div>
  </div>
  <script>
const startButton = document.getElementById('predict-btn');
const stopButton = document.getElementById('stop-btn');
const restartButton = document.getElementById('restart-btn');
const videoElement = document.getElementById('video');
const tooltip = document.getElementById('tooltip');
const letterImage = document.getElementById('letter-image');
const defaultMessage = document.getElementById('default-message');
const letters = document.querySelectorAll('.letter');
function showTooltip(text) {
  tooltip.innerText = text;
  tooltip.style.display = 'block';
}
function hideTooltip() {
  tooltip.style.display = 'none';
}
startButton.onclick = function() {
  startButton.disabled = true;
  stopButton.disabled = false;
  videoElement.style.display = 'block';
  document.querySelector('.video-placeholder').style.display = 'none';
  videoElement.src = "{% url 'video_feed' %}";
};
stopButton.onclick = function() {
  stopButton.disabled = true;
  startButton.disabled = false;
  videoElement.style.display = 'none';
  document.querySelector('.video-placeholder').style.display = 'flex';
  fetch("{% url 'stop_video_feed' %}");
};
restartButton.onclick = function() {
  alert('Prediction is stopping...');
  fetch("{% url 'stop_and_restart_view' %}")
    .then(response => response.json())
    .then(data => {
            if (data.status === 'restarting') {
                checkServerStatus();
            }
        });
};
function checkServerStatus() {
    fetch("/dashboard")
        .then(response => {
            if (response.ok) {
                window.location.reload();
            } else {
                setTimeout(checkServerStatus, 2000);
            }
        })
        .catch(() => {
            setTimeout(checkServerStatus, 2000);
        });
}
startButton.addEventListener('mouseenter', () => showTooltip('Click to start prediction'));
startButton.addEventListener('mouseleave', hideTooltip);
stopButton.addEventListener('mouseenter', () => showTooltip('Click to pause'));
stopButton.addEventListener('mouseleave', hideTooltip);
restartButton.addEventListener('mouseenter', () => showTooltip('Click to stop and restart
prediction'));
restartButton.addEventListener('mouseleave', hideTooltip);
letters.forEach(letter => {
    letter.addEventListener('click', () => {
        const letterValue = letter.textContent;
        letterImage.src = `{% static 'images/' %}${letterValue.toLowerCase()}.jpg`;
        letterImage.style.display = 'block';
        defaultMessage.style.display = 'none';
        letters.forEach(l => l.classList.remove('active'));
        letter.classList.add('active');
    });
});
document.addEventListener('DOMContentLoaded', () => {
    stopButton.disabled = true;
    videoElement.style.display = 'none';
});
    </script>
</body>
</html>
```

6.4 CODE FOR DETECTION – NUMBERS

```
{% load static %}
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/5.15.4/css/all.min.css">
    <title>Sign Language Detection</title>
    <link rel="stylesheet" href="style.css">
```

```
</head>

<style>

    :root { --primary-color: #3b82f6; --danger-color: #ef4444; --warning-color: #f59e0b; --text-
color: #1f2937; --bg-color: #f9fafb; --border-color: #e5e7eb; --hover-color: #2563eb; }

* { margin: 0; padding: 0; box-sizing: border-box; }

body::-webkit-scrollbar { display: none; }

body { background: linear-gradient(rgba(0, 0, 0, 0.5), rgba(0, 0, 0, 0.5)), url("{% static
'images/background5.webp' %}"); background-color: #1e1e1e; background-size: cover;
background-position: center; background-attachment: fixed; background-repeat: no-repeat; color:
#e0e0e0; overflow-x: hidden; }

.container { display: grid; grid-template-columns: 1fr 1fr; gap: 2rem; max-width: 1400px; margin:
0 auto; padding: 2rem; min-height: 100vh; }

h1, h2 { margin-bottom: 1.5rem; font-weight: 600; text-align: center; }

h1 { font-size: 2rem; color: var(--text-color); }

h2 { font-size: 1.5rem; }

.video-section { display: flex; flex-direction: column; gap: 1.5rem; }

.video-wrapper { position: relative; }

.video-placeholder { position: absolute; inset: 0; display: flex; align-items: center; justify-content:
center; color: white; font-size: 1.25rem; }

#video { width: 100%; height: 100%; object-fit: cover; }

.controls { display: flex; gap: 1rem; justify-content: center; }

.btn { display: flex; align-items: center; gap: 0.5rem; padding: 0.75rem 1.5rem; border: none;
border-radius: 0.5rem; font-weight: 500; cursor: pointer; transition: all 0.2s; }

.btn-primary { background-color: var(--primary-color); color: white; }

.btn-danger { background-color: var(--danger-color); color: white; }

.btn-warning { background-color: var(--warning-color); color: white; }

.btn:hover:not(:disabled) { transform: translateY(-1px); filter: brightness(110%); }

.btn:disabled { opacity: 0.6; cursor: not-allowed; }

.letters-section { background: white; border-radius: 1rem; padding: 1.5rem; box-shadow: 0 4px 6px
-1px rgba(0, 0, 0, 0.1); }

.letters-grid { max-height: 100px; overflow-y: auto; padding-right: 0.5rem; }

.letters-list { display: grid; grid-template-columns: repeat(auto-fill, minmax(60px, 1fr)); gap:
0.75rem; }
```

```
.letter { aspect-ratio: 1; display: flex; align-items: center; justify-content: center; background: var(--
bg-color); border-radius: 0.5rem; font-size: 1.25rem; font-weight: 500; cursor: pointer; transition:
all 0.2s; }

.letter:hover { background: var(--primary-color); color: white; transform: translateY(-2px); }

.active { background: var(--primary-color); color: white; transform: translateY(-2px); }

.preview-section { background: white; border-radius: 1rem; padding: 1.5rem; margin-top: 1.5rem;
box-shadow: 0 4px 6px -1px rgba(0, 0, 0, 0.1); color: black; }

.preview-container { aspect-ratio: 4/2; display: flex; align-items: center; justify-content: center;
background: var(--bg-color); border-radius: 0.5rem; overflow: hidden; }

#letter-image { height: 300px; object-fit: contain; }

.tooltip-container { position: relative; min-height: 2rem; }

.tooltip { position: absolute; left: 50%; transform: translateX(-50%); background: rgba(0, 0, 0, 0.8);
color: white; padding: 0.5rem 1rem; border-radius: 0.25rem; font-size: 0.875rem; display: none; }

@media (max-width: 1024px) { .container { grid-template-columns: 1fr; } .video-section { order: -
1; } }

.navbar { background-color: #333; }

.navbar-brand, .nav-link { color: #e0e0e0; }

.navbar-brand:hover, .nav-link:hover { color: #007bff; }

.navbar-nav .dropdown-menu { background-color: #2c2c2c; border: 1px solid #3a3a3a; color:
#e0e0e0; white-space: nowrap; }

.navbar-nav .dropdown-menu .dropdown-item { color: #e0e0e0; }

.navbar-nav .dropdown-menu .dropdown-item:hover { background-color: #007bff; color: #fff; }



</style>

<body class="bg-dark text-light">

  <nav class="navbar navbar-expand-lg navbar-dark bg-dark-custom">

    <a class="navbar-brand" href="{% url 'dashboard' %}">Sign Language Learning System</a>

    <button class="navbar-toggler" type="button" data-toggle="collapse" data-
target="#navbarNav" aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle
navigation">

      <span class="navbar-toggler-icon"></span>

    </button>

    <div class="collapse navbar-collapse" id="navbarNav">
```

```
        <ul class="navbar-nav ml-auto">
            <li class="nav-item">
                <a class="nav-link" href="{% url 'dashboard' %}">Home</a>
            </li>
            <li class="nav-item"></li>
                <a class="nav-link" href="{% url 'blogs' %}">Articles</a>
            </li>
            <li class="nav-item">
                <a class="nav-link" href="{% url 'test' %}">Letters</a>
            </li>
            <li class="nav-item">
                <a class="nav-link" href="{% url 'number_test_view' %}">Numbers</a>
            </li>
        </ul>
    </div>
</nav>
<div class="container">
    <div class="video-section">
        <div class="video-container">
            <div class="video-wrapper">
                <img id="video" src="" alt="Video Stream" style="display: none;">
                <div class="video-placeholder">
                </div>
            </div>
        </div>
        <div class="controls">
            <button id="predict-btn" class="btn btn-primary">
                <span class="btn-icon">▶</span>
                Start Prediction
            </button>

            <button id="stop-btn" class="btn btn-danger" disabled>

                <span class="btn-icon"> ⏸ </span>

                Pause

            </button>

            <button style="display: none;" id="restart-btn" onclick="location.reload()" class="btn
btn-warning">

                <span class="btn-icon">■</span>

                Stop Prediction

            </button>

            <button onclick="location.reload()" class="btn btn-warning">

                <span class="btn-icon">■</span>

                Stop Prediction

            </button>

        </div>

        <div class="tooltip-container">

            <p id="tooltip" class="tooltip"></p>

        </div>

    </div>

    <div class="content-section">

        <div class="preview-section">

            <h2>Preview</h2>

            <div class="preview-container">

                <p id="default-message">Select any number to view the sign</p>

                <img id="letter-image" src="" alt="Sign language preview" style="display: none;">

            </div>

            <div class="letters-section">

                <h2>Numbers</h2>

                <div class="letters-grid">
```

```
                    <div class="letters-list">
                        <div class="letter" id="number-0">0</div>
                        <div class="letter" id="number-1">1</div>
                        <div class="letter" id="number-2">2</div>
                        <div class="letter" id="number-3">3</div>
                        <div class="letter" id="number-4">4</div>
                        <div class="letter" id="number-5">5</div>
                        <div class="letter" id="number-6">6</div>
                        <div class="letter" id="number-7">7</div>
                        <div class="letter" id="number-8">8</div>
                        <div class="letter" id="number-9">9</div>
                    </div>
                </div>
            </div>
        </div>
    </div>
  </div>
  <script>
const startButton = document.getElementById('predict-btn');
const stopButton = document.getElementById('stop-btn');
const restartButton = document.getElementById('restart-btn');
const videoElement = document.getElementById('video');
const tooltip = document.getElementById('tooltip');
const letterImage = document.getElementById('letter-image');
const defaultMessage = document.getElementById('default-message');
const letters = document.querySelectorAll('.letter');
function showTooltip(text) {
  tooltip.innerText = text;
  tooltip.style.display = 'block';
```

```
}
function hideTooltip() {
    tooltip.style.display = 'none';
}
startButton.onclick = function() {
    startButton.disabled = true;
    stopButton.disabled = false;
    videoElement.style.display = 'block';
    document.querySelector('.video-placeholder').style.display = 'none';
    videoElement.src = "{% url 'video_feed_tf' %}";
};
stopButton.onclick = function() {
    stopButton.disabled = true;
    startButton.disabled = false;
    videoElement.style.display = 'none';
    document.querySelector('.video-placeholder').style.display = 'flex';
    fetch("{% url 'stop_video_feed' %}");
};
restartButton.onclick = function() {
    alert('Prediction is stopping...');
    fetch("{% url 'stop_and_restart_view' %}")
        .then(response => response.json())
        .then(data => {
            if (data.status === 'restarting') {
                checkServerStatus();
            }
        });
};
function checkServerStatus() {
    fetch("/dashboard")
        .then(response => {
            if (response.ok) {
                window.location.reload();
            } else {
                setTimeout(checkServerStatus, 2000);
            }
        })
        .catch(() => {
            setTimeout(checkServerStatus, 2000);
        });
}
startButton.addEventListener('mouseenter', () => showTooltip('Click to start prediction'));
startButton.addEventListener('mouseleave', hideTooltip);
stopButton.addEventListener('mouseenter', () => showTooltip('Click to pause'));
stopButton.addEventListener('mouseleave', hideTooltip);
restartButton.addEventListener('mouseenter', () => showTooltip('Click to stop and restart
prediction'));
restartButton.addEventListener('mouseleave', hideTooltip);
letters.forEach(letter => {
    letter.addEventListener('click', () => {
        const letterValue = letter.textContent;
        letterImage.src = `{% static 'images/numbers/' %}${letterValue.toLowerCase()}.jpg`;
        letterImage.style.display = 'block';
        defaultMessage.style.display = 'none';
        letters.forEach(l => l.classList.remove('active'));
        letter.classList.add('active');
    });
});
```

```
document.addEventListener('DOMContentLoaded', () => {
  stopButton.disabled = true;
  videoElement.style.display = 'none';
});
  </script>
</body>
</html>
```

CHAPTER 7: SYSTEM TESTING

Software testing is a critical element of software quality assurance and represent the ultimate review of specification, design and coding. System testing is actually a series of different test whose purpose is to fully exercise the computer-based system. Although each has a different purpose, all of them work to verify that all system elements have been properly integrated and all of them perform allocated functions. If the test is conducted successfully, it will uncover errors in the software. A second benefit is that the software is appearing to be working according to specification and that performance requirements appear to have been met. System testing is an inexpensive but critical process that can take as much as 50% of budget for development, the view of testing holds by users that is performed to prove that there is no error in the program. However, this is virtually impossible since analysis cannot prove that software is free and clear to errors. Testing is the process of executing a program with explicit intensions of finding errors.

7.1 LEVELS OF TESTING

7.1.1 Unit Testing

Unit testing alone cannot verify the functionality of a piece of software, but rather is used to assure that the building blocks the software uses work independently of each other. The unit testing approach for this project involved testing each component of the software individually to ensure its functionality, accuracy, and reliability. The testing focused on verifying the behaviour of the Sign Language Detection component, of both letters and numbers.

7.1.2 System Testing

System testing is actually a series of different tests, whose purpose is to test the completed system in it's entirely as a whole exercise. System testing involves security testing, performance testing and recovery testing. After output testing the whole system is tested in different lighting conditions and backgrounds for testing the overall functionality and user interface of the system in various environment. This testing is done after completing all the testing, ie system testing is the final phase of the testing process. Performance testing measured system response time and latency in hand detection to ensure a seamless experience without tags or delays. To ensure a consistent user experience compatibility testing assessed the system across various backgrounds and hand sizes.

7.1.3 Validation Testing

Validation testing was performed to confirm that Sign Language Recognition model handles both valid and invalid inputs appropriately. The system was tested using clear hand sign images to ensure correct detections and responses. Ul and responsiveness were also tested to ensure display and usability.

7.1.4 Test Cases

Test ID	Test Scenario	Test Steps	Test Data	Expected Result	Actual Result	Status
1	Valid Hand Detection	1) Click on start prediction 2) Position hand within the frame	Valid hand sign image	System should detect the hand	Hand was detected successfully	Pass
2	Hand sign partially out of frame	1) Click on start prediction 2) Position a part of the hand inside the frame	Invalid hand sign image	System should not detect any hand	System didn't detect any hand	Pass
3	Hand sign detection in low lighting condition	1) Click on start prediction 2) Position hand within the frame.	Valid hand sign image in low lighting condition	Hand should be detected	Hand is not detected	Pass
4	No hand sign within the camera frame	1) Click on start prediction 2) User sits idle in front of camera	No hand sign image	No hand should be detected	Hand is not detected	Pass
5	Hand sign detection using both hands	1) Click on start prediction 2) Position both hands within the frame	Valid hand sign image of both hands	Both hands should be detected	Both hands are detected	Pass

ASL Alphabets						
6	Detecting ASL sign alphabet "A"	1) Click on start prediction 2) Position the "A" hand sign within the frame	Valid image of ASL sign alphabet "A"	"A" hand sign should be recognized	"A" hand sign is recognized	Pass
7	Detecting ASL sign alphabet "B"	1) Click on start prediction 2) Position the "B" hand sign within the frame	Valid image of ASL sign alphabet "B"	"B" hand sign should be recognized	"B" hand sign is recognized	Pass
Numbers						
8	Detecting hand sign of number "1"	1) Click on start prediction 2) Position the hand sign of number "1" within the frame	Valid hand sign image of number "1"	System should recognize number "1" sign	System recognized number "1" sign	Pass
9	Detecting hand sign of number "2"	1) Click on start prediction 2) Position the hand sign of number "2" within the frame	Valid hand sign image of number "2"	System should recognize number "2" sign	System should recognized number "2" sign	Pass

CHAPTER 8: CONCLUSION

In conclusion, the proposed Sign Language Learning System represents a significant leap toward creating a more inclusive and accessible world. By harnessing the power of advanced technologies such as computer vision and machine learning, the system bridges communication gaps between sign language users and non-users while serving also as a valuable educational tool for children and individuals seeking to learn and practice sign language. Its ability to provide seamless translation, foster learning, and promote inclusivity ensures its broad impact across educational, social, and professional contextual fields.

This system is not just a technological innovation; it is a step toward fostering understanding, empathy, and connection among diverse communities. By making sign language more accessible to everyone, it empowers individuals with hearing impairments and paves the way for greater societal integration and equality. Ultimately, this project embodies the vision of a world where communication knows no barriers and inclusivity becomes the standard.

CHAPTER 9: FUTURE WORK

In the future, this project can be improved by adding more sign languages like Australian, British, and Indian Sign Language to make it more inclusive for users from different regions. The user interface can be made more interactive, and fun especially for children to make learning more enjoyable. Creating a mobile app version will help more people access the system anytime and anywhere, especially those with hearing or speech difficulties. Other useful additions could be showing helpful tips and giving them appropriate feedbacks. Compared to other projects, this one already offers more advanced features, and these future updates will make it even more user-friendly, educational, and accessible for everyone.

CHAPTER 10: REFERENCE

[1] R.Thirumahal , “Sign Language Detection Using Yolo Nas”,2024.

[2] Anubrolu Umesh Chowdary, “Sign Language Recognition Using Machine Learning”,2023.

[3] Elek Alaftekin, “Real Time Sign Language Recognition based on Yolo”,2024.

[4] Aishwarya Girish, “Sign Language Recognition using CNN”,2021.

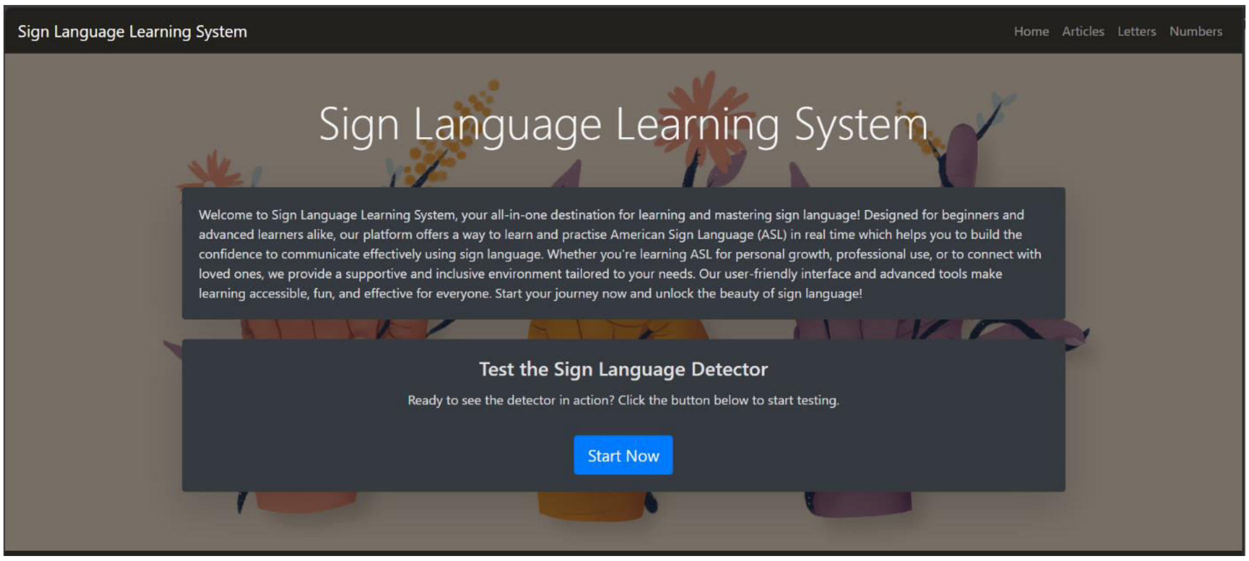
[5] Songwao Jiang, “Sign Language Recognition”,2021.

[6] Tanaji M Dudhane, T.R. Chentil, “A CNN based Human Interface for American Sign Language Recognition for Hearing-Impaired Individuals”,2022.

[7] Subhalaxmi Chakraborthy, Prayosi Paul, “Sign Language recognition using Landmark Detection, GRU and LSTM”,2023.

CHAPTER 11: APPENDIX

11.1 HOME PAGE


Sign Language Learning System

Home Articles Letters Numbers

Sign Language Learning System

Welcome to Sign Language Learning System, your all-in-one destination for learning and mastering sign language! Designed for beginners and advanced learners alike, our platform offers a way to learn and practise American Sign Language (ASL) in real time which helps you to build the confidence to communicate effectively using sign language. Whether you're learning ASL for personal growth, professional use, or to connect with loved ones, we provide a supportive and inclusive environment tailored to your needs. Our user-friendly interface and advanced tools make learning accessible, fun, and effective for everyone. Start your journey now and unlock the beauty of sign language!

Test the Sign Language Detector

Ready to see the detector in action? Click the button below to start testing.

Start Now

Figure 11.1 Home Page

11.2 ARTICLES PAGE

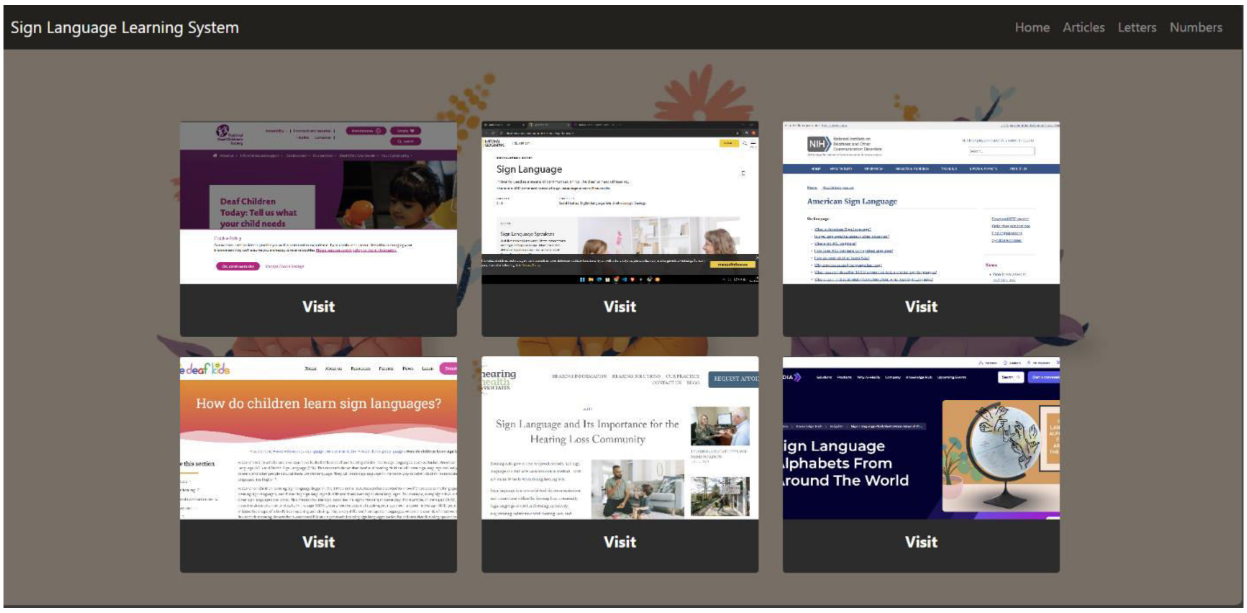

Sign Language Learning System
Home Articles Letters Numbers
Deaf Children Today: Tell us what your child needs
Visit
Sign Language
Visit
American Sign Language
Visit
How do children learn sign languages?
Visit
Sign Language and Its Importance for the Hearing Loss Community
Visit
ign Language lphabets From round The World
Visit

Figure 11.2 Articles Page

11.3 LETTERS PAGE

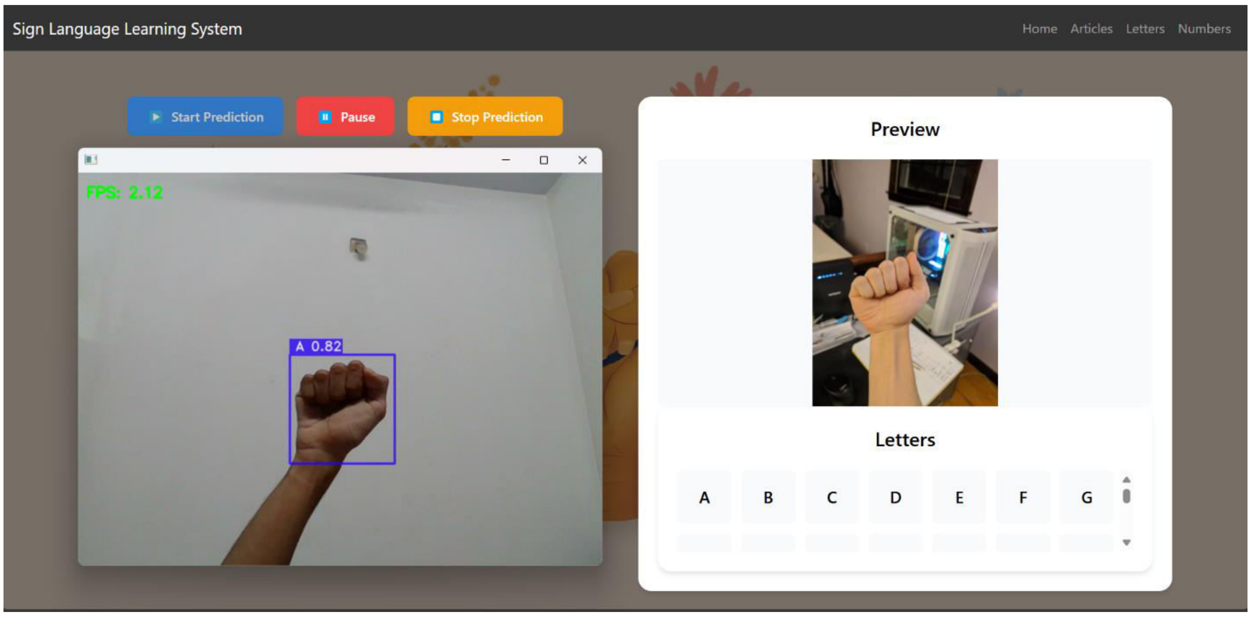


Figure 11.3 Letters Page

11.4 NUMBERS PAGE

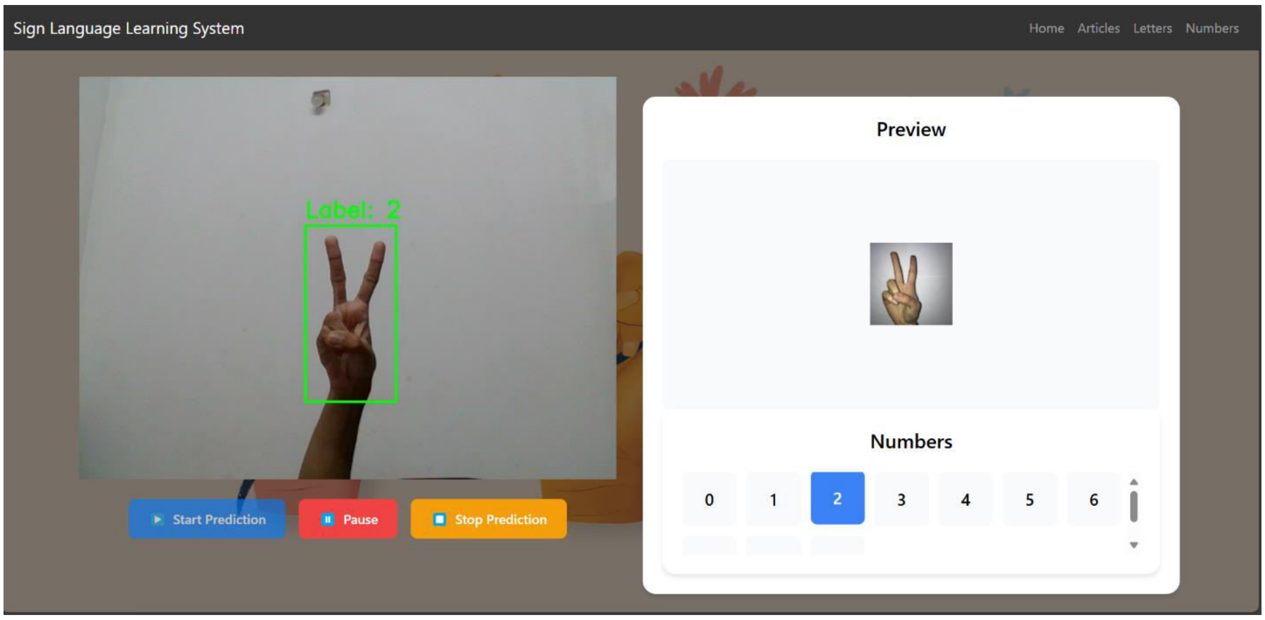


Figure 11.4 Numbers Page