

SIGN LANGUAGE TO TEXT/SPEECH TRANSLATION

A

MINOR PROJECT-II REPORT

Submitted partial fulfillment of the requirements

for the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE & ENGINEERING

By

GROUP NO. 3

Shruti Tiwari	0187CS221189
Siya Gajbhiye	0187CS221192
Sitvat Fatima	0187CS221191

Under the guidance of
Dr. Amit Kumar Mishra
(Associate Professor)



Department of Computer Science & Engineering
Sagar Institute of Science & Technology (SISTec), Bhopal (M.P.)

Approved by AICTE, New Delhi & Govt. of M.P.
Affiliated to Rajiv Gandhi Proudyogiki Vishwavidyalaya, Bhopal (M.P.)

JUNE -2025

Sagar Institute of Science & Technology (SISTec), Bhopal (M.P)

Department of Computer Science & Engineering



CERTIFICATE

We hereby certify that the work which is being presented in the B.Tech. Minor Project-II Report entitled **Sign Language to Text/Speech Translation**, in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology**, submitted to the Department of **Computer Science & Engineering**, Sagar Institute of Science & Technology (SISTec), Bhopal (M.P.) is an authentic record of our own work carried out during the period from Jan-2025 to Jun-2025 under the supervision of **Dr. Amit Kumar Mishra**.

Shruti Tiwari
0187CS221189

Siya Gajbhiye
0187CS221192

Sitvat Fatima
0187CS221191

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

Date:

Dr. Amit Kumar Mishra
Project Guide

Prof. Nargish Gupta
HOD, CSE

Dr. Manish Billore
Principal

ACKNOWLEDGMENT

We would like to express our sincere thanks to **Dr. Manish Billore, Principal, SISTec** and **Dr. Swati Saxena, Vice Principal SISTec** Gandhi Nagar, Bhopal for giving us an opportunity to undertake this project.

We also take this opportunity to express a deep sense of gratitude to **Prof. Nargish Gupta, HOD, Department of Computer Science & Engineering** for his kindhearted support.

We extend our sincere and heartfelt thanks to our guide, **Dr. Amit Kumar Mishra**, for providing us with the right guidance and advice at crucial junctures and for showing us the right way.

I am thankful to the **Project Coordinator, Prof. Mayank Kurchaniya**, who devoted his precious time in giving us information about various aspects and gave support and guidance at every point of time.

I would like to thank all those people who helped me directly or indirectly to complete my project whenever I found myself in any issue.

TABLE OF CONTENTS

TITLE	PAGE NO.
Abstract	i
List of Abbreviation	ii
List of Figures	iii
Chapter 1 Introduction	1
1.1 About Project	1
1.2 Purpose	1
1.3 Project Objective	2
1.4 Significance of Project	3
Chapter 2 Software & Hardware Requirements	4
2.1 Hardware Requirements	4
2.2 Software Requirements	4
2.3 Client-side Requirements	5
Chapter 3 Problem Description	6
3.1 Current Scenario & Challenges	6
3.2 Importance of Automation	7
3.3 Scope of the Project	7
3.4 Summary	8
Chapter 4 Literature Survey	9
Chapter 5 Software Requirements Specification	11
5.1 Functional Requirements	11
5.2 Non-Functional Requirements	12
Chapter 6 Software Design	13
6.1 Overview	13
6.2 Use Case Diagram	13
6.3 Project Flow Diagram	14
Chapter 7 Deep Learning Module	15
7.1 Dataset Description	15

	7.2	Preprocessing Steps	16
	7.3	DL Model Analysis	18
	7.4	Result Analysis	19
Chapter 8		Front End Connectivity	20
	8.1	Connectivity	20
Chapter 9		Coding	26
	9.1	Data Preprocessing	26
	9.2	Feature Extraction	26
	9.3	Model Training	26
	9.4	Prediction & Evaluation	27
Chapter 10		Output Screens	29
Chapter 11		Conclusion and Future Work	33
References			
Project Summary			
Appendix-1: Glossary of Terms			

ABSTRACT

People with hearing and speech impairments face significant difficulties in daily communication due to the lack of understanding of sign language among the general public. This communication barrier often leads to social isolation and limited access to essential services for affected individuals. To address this challenge, a deep learning-based system has been proposed to bridge the communication gap by recognizing hand gestures used in Indian Sign Language (ISL) and converting them into both text and speech outputs. The system is designed to provide a real-time, user-friendly interface that enables smooth interaction between people with communication impairments and those unfamiliar with sign language. Unlike traditional gesture recognition systems that rely on Convolutional Neural Networks (CNNs), this solution incorporates YOLOv11 (You Only Look Once version 11), an advanced object detection model renowned for its exceptional speed, accuracy, and real-time processing capabilities. YOLOv11 processes visual input in a single pass, making it ideal for detecting and classifying complex hand gestures on the fly. The model is trained on a comprehensive and diverse dataset of ISL gestures, enabling it to identify a wide range of signs with an impressive accuracy rate of 98%. Once a hand gesture is detected, the system instantly translates it into the corresponding text, which is then displayed on a screen and converted into speech using a google-text-to-speech (GTTS) engine. This dual-mode output allows for effective communication not only with those who can read but also with individuals who rely on auditory input, such as people with visual impairments. The integration of advanced deep learning techniques, real-time gesture detection, and speech synthesis makes this system a powerful assistive tool. It enhances accessibility, reduces communication barriers, and promotes social inclusion by empowering users with hearing, speech, or visual impairments to interact more freely and independently within their communities.

LIST OF ABBREVIATIONS

ACRONYM	FULL FORM
DL	Deep Learning
HTML	Hyper Text Markup Language
CSS	Cascading Style Sheets
IDE	Integrated Development Environment
UI	User Interface
AI	Artificial Intelligence
SQL	Sequence Query Language
GTTS	Google Text To Speech

LIST OF FIGURES

FIG. NO.	TITLE	PAGE NO.
6.1	Use Case Diagram	13
6.2	Project Flow Diagram	14
10.1	Home Page	28
10.2	Gesture Gallery	28
10.3	Real-Time Sign Detection Interface	29
10.4	Contact us Page	29
10.5	About Us Page	30
10.6	Login Page	30
10.7	Signup Up Page	31
10.8	User Profile	31
10.9	Feedback Page	32

CHAPTER – 1

INTRODUCTION

CHAPTER 1

INTRODUCTION

1.1 ABOUT PROJECT

The Sign Language Converter project is an innovative solution designed to bridge the communication gap between the hearing and speech-impaired community and the general population. By leveraging deep learning, computer vision, and deep learning algorithms, the system accurately recognizes **Indian Sign Language (ISL)** gestures and converts them into real-time text and speech outputs. This converter enhances accessibility in education, workplaces, and daily life by enabling seamless interaction. With support for Indian languages and English, the system empowers users with hearing impairments, promotes social inclusion, and facilitates a more inclusive digital environment. Its user-friendly interface and real-time translation capabilities make it a powerful tool for improving communication and fostering independence.

1.2 PURPOSE

The purpose of the Sign Language Text/Sign Recognition System is to develop an AI-powered system that enables real-time communication between individuals with hearing or speech impairments and the wider community. By leveraging deep learning and computer vision, the platform accurately translates Indian Sign Language (ISL) gestures into text and speech, making communication more accessible and inclusive. The system is designed to empower users in educational, professional, and social environments by facilitating seamless interaction and promoting independence. Additionally, the project aims to raise awareness about assistive technologies and support social integration, ultimately contributing to a more equitable and connected society.

1.3 PROJECT OBJECTIVE

The primary objective of the Indian Sign Language Text/Speech Recognition System is to facilitate communication between individuals who use sign language and those who do not. This involves developing a system that can translate sign language gestures into spoken or written language, enabling seamless interaction and understanding.

1.3.1 Implement Gesture Recognition Models

Leverage deep learning models (e.g., Yolo) to accurately recognize and classify sign language

gestures captured via webcam.

1.3.2. Develop a User-Friendly Interface

Design a responsive and intuitive web-based platform using HTML, CSS, and JavaScript to allow users to perform sign language gestures and view real-time translations into spoken or written language.

1.3.3. Ensure Smooth System Integration

Implement a Django backend for efficient communication between the frontend interface and the deep learning model, enabling reliable real-time translation and user interaction.

1.3.4. Facilitate Data Management

Use SQLite as the database to securely store user sessions, translation logs, and gesture datasets to enhance the system's learning and allow for future personalization.

1.3.5. Promote Real-Time and Personalized Communication

Enable real-time translation of sign language to text/speech. Incorporate contextual awareness to improve translation accuracy and provide personalized output where applicable.

1.3.6. Enhance Accessibility

Ensure the platform is inclusive and easy to use for both sign language users and non-signers, including options for voice/text outputs, and adaptable UI features.

1.3.7. Maintain Ethical Standards

Uphold data privacy and security, obtain informed consent for gesture data usage, and ensure responsible handling of sensitive user information in compliance with ethical and legal standard.

1.4 SIGNIFICANCES OF PROJECT

1.4.1. Enhancing Communication Accessibility

By utilizing machine learning and computer vision, the system enables real-time translation of Indian Sign Language, empowering the hearing and speech-impaired community to communicate effectively in daily life.

1.4.2. Bridging the Communication Gap

The platform addresses the lack of accessible sign language interpretation, especially in schools,

public services, and remote areas, offering a reliable alternative to human interpreters.

1.4.3. Empowering Independence and Social Inclusion

By converting sign language into text and speech, users can interact independently in various settings such as education, workplaces, and healthcare, promoting equal participation and informed interactions.

CHAPTER-2

SOFTWARE &

HARDWARE

REQUIREMENTS

CHAPTER 2

SOFTWARE & HARDWARE REQUIREMENTS

2.1 HARDWARE REQUIREMENTS

1. Processor: Intel Core i5 or higher / AMD equivalent
2. RAM: 8 GB (16 GB recommended)
3. Storage: 256 GB SSD minimum
4. Network: High-speed internet
5. Deployment Environment
6. Server: Quad-core processor, 8 GB RAM (16 GB recommended), 100 GB SSD, Linux/Windows Server/Macbook
7. Client Devices: Any device with a modern web browser (HTML5, CSS3, JavaScript support)
8. External storage for backups
9. Cloud hosting for scalability (AWS, Azure, etc.)

2.2 SOFTWARE REQUIREMENTS

1. **OS:** Windows 10/11, macOS, or Linux (Ubuntu preferred)
2. **Languages:** Python 3.12.2, JavaScript, HTML5, CSS3
3. **Frameworks/Libraries:** Django, SQLite, Scikit-learn, Pandas, NumPy, Py-Torch, Open-cv, YOLO
4. **Tools:** VS Code, Colab
5. **Deployment**
6. **Web Server:** Apache or Nginx
7. **Cloud (Optional):** AWS, Runpod

8. **Virtual Environment:** Python Virtual env or Conda

2.3 CLIENT-SIDE REQUIREMENTS

1. **Browser:** A compatible browser to run the web application

CHAPTER – 3

PROBLEM

DESCRIPTION

CHAPTER 3

PROBLEM DESCRIPTION

This project aims to develop a real-time Indian Sign Language (ISL) to text and speech translation system using computer vision and machine learning to bridge the communication gap between the deaf and hard-of-hearing community and the hearing world. The application will recognize a wide range of ISL gestures through live camera input as well as pre-recorded video and convert them into accurate text and speech outputs in multiple Indian languages. By offering an accessible, user-friendly interface and supporting regional language diversity, the system promotes inclusive communication, social integration, and equal access to services and opportunities.

3.1.1 CURRENT SCENARIO AND CHALLENGES

In recent years, the need for inclusive communication tools for the deaf and hard-of-hearing community has grown due to increasing social awareness, digital transformation, and the push for accessibility. However, several challenges persist in enabling seamless communication through sign language:

- 1. Lack of Real-Time Translation Tools**

Many individuals who rely on Indian Sign Language face difficulties in communicating with the hearing population due to the absence of efficient and personalized real-time translation systems.

- 2. Technology and Accessibility Barriers**

Advanced ISL interpretation tools are often limited by high costs, lack of infrastructure in rural or underserved areas, and minimal availability of regionally adaptable solutions.

- 3. Communication Delays**

Without instant translation, conversations are often delayed or require third-party interpreters, which can hinder participation in education, employment, and social settings.

- 4. Lack of Awareness and Support**

There is limited awareness and understanding of ISL among the general public and institutions, leading to insufficient support for sign language users in mainstream environments.

- 5. Privacy and Data Security Concerns**

Users may be hesitant to adopt digital ISL converters due to concerns about the safety of video data and personal interactions being stored or misused.

3.2 IMPORTANCE OF AUTOMATION

Automation plays a crucial role in overcoming the challenges faced by the deaf and hard-of-hearing community by enabling real-time and efficient sign language translation. The significance of automation in this project includes:

1. **Efficiency and Scalability:** Automated systems can process sign gestures from multiple users and large volumes of input (live or video) simultaneously, ensuring real-time translation without human interpreters.
2. **Accessibility:** Users can access the application anytime and anywhere through computers, eliminating the need for physical sign language interpreters or special arrangements.
3. **Anonymity and Privacy:** Automation offers a secure and non-judgmental space for deaf individuals to communicate, encouraging wider usage without fear of misunderstanding or discrimination.
4. **Early Inclusion:** Automated tools support early integration of deaf individuals into educational and social environments by reducing communication gaps from the start.
5. **Consistency:** Unlike human interpreters, automated translation systems provide consistent and unbiased gesture interpretation based on trained deep learning models.
6. **Cost-Effectiveness:** Automation minimizes the need for full-time interpreters or specialized human resources, making inclusive communication more affordable and scalable.

3.3 SCOPE OF THE PROJECT

The ISL-to-Text and Speech Converter aims to provide an automated, real-time, and user-friendly platform that facilitates inclusive communication for the deaf and hard-of-hearing community. Key aspects include:

1. Input Method

Capturing Indian Sign Language gestures through live camera feeds or pre-recorded videos using a secure and responsive interface.

2. DL Integration

Using computer vision and deep learning models to recognize and interpret ISL signs accurately

in real-time.

3. Backend & Database

Implementing backend processing using frameworks like Django, and utilizing secure databases (e.g., SQL lite) to store gesture data, language preferences.

4. Output Generation

Converting recognized signs into meaningful text and speech output in multiple Indian languages, enhancing understandability for the hearing population.

5. Accessibility

Ensuring the system is scalable, compatible across devices like computer and laptop for ease of use by individuals from varied linguistic, educational, and geographical backgrounds.

3.4 SUMMARY

The Indian Sign Language (ISL) to Text and Speech Converter project aims to build an automated, real-time translation system that bridges the communication gap between the deaf and hard-of-hearing community and the hearing world. Using computer vision and deep learning models such as (Yolo v11), the system recognizes ISL gestures from live camera input or video and converts them into accurate text and speech outputs in multiple Indian languages. The platform features a user-friendly interface, a secure backend (e.g., Django), and reliable data storage using databases like SQL lite. This solution ensures privacy, accessibility, and scalability while promoting inclusive communication, social integration, and equal opportunities for the hearing-impaired community.

CHAPTER – 4

LITERATURE SURVEY

CHAPTER 4

LITERATURE SURVEY

Sign Language is a natural language which deaf community uses for communication. Sign Language (SL) is a subset of gestures or signs made with fingers, hands, arms, eyes, and head, face etc. Each gesture in SL has a meaning assigned to it. Understanding SL is nothing but understanding the meaning of these gestures. There exists a problem in communication when a person who completely relies on this gestural SL for communication tries to converse with a person who does not understand the SL. Every country has its own developed SL. In India, this language is called as "Indian Sign Language (ISL)". This paper aims to develop an algorithm that will translate the ISL into English. This paper has implemented a system named as "Indian Sign Language (ISL) Translator using Gesture recognition algorithm". The system translates gestures made in ISL into English. The gestures that have been translated include numbers, alphabets and few phrases. The algorithm first performs data acquisition, then the pre-processing of gestures is performed to track hand movement using a combinational algorithm, and recognition is done using template matching. The database used for implementation has been self-created and includes total 130,000 videos; out of which 72,000 videos were used to create the system database and remaining 58,000 videos have been tested for checking the performance of the system. The accuracy of this system is as high as 97.5%.[1]

Sign language is the only medium of communication for the speech-impaired community while the rest of the population communicate verbally. This project aims to bridge this communication gap by proposing a novel approach to interpret the static and dynamic signs in the Indian Sign Language and convert them to speech. A sensor glove, with flex sensors to detect the bending of each finger and an IMU to read the orientation of the hand, is used to collect data about the actions. This data is then wirelessly transmitted and classified into corresponding speech outputs. LSTM networks were studied and implemented for classification of gesture data because of their ability to learn long-term dependencies. The designed model could classify 26 gestures with an accuracy of 98%, showing the feasibility of using LSTM based neural networks for the purpose of sign language translation.[2]

Language plays a vital role in the communication of ideas, thoughts, and information to others. Hearing-impaired people also understand our thoughts using a language known as sign language. Every country has a different sign language which is based on their native language. In our research paper, our major focus is on Indian Sign Language, which is mostly used by hearing- and speaking-impaired communities in India. While communicating our thoughts and views with others, one of the most essential factors is listening. What if the other party is not able to hear or grasp what you are

talking about? This situation is faced by nearly every hearing-impaired person in our society. This led to the idea of introducing an audio to Indian Sign Language translation system which can erase this gap in communication between hearing-impaired people and society. The system accepts audio and text as input and matches it with the videos present in the database created by the authors. If matched, it shows corresponding sign movements based on the grammar rules of Indian Sign Language as output; if not, it then goes through the processes of tokenization and lemmatization. The heart of the system is natural language processing which equips the system with tokenization, parsing, lemmatization, and part-of-speech tagging.[3]

The work on Deep Learning for Sensorbased Human Activity Recognition: Overview, Challenges and Opportunities, done by Kaixuan Chen et al (2020), provides an overview of the activity recognition using sensorbased systems rather than videobased systems. The research on Human Activity Recognition Using Deep Learning: A Survey, done Zhang HB et al (2019), published on the Lecture Notes on Data Engineering and Communications Technologies, Volume 52, provides insights into the state of the art for human activity recognition and compares various methods. The work on Human activity recognition using magnetic inductionbased motion signals and deep recurrent neural networks.[4]

CHAPTER – 5

SOFTWARE

REQUIREMENT

SPECIFICATIONS

CHAPTER 5

SOFTWARE REQUIREMENT SPECIFICATIONS

5.1 FUNCTIONAL REQUIREMENTS

Functional requirements define the essential behavior of the system, describing how it should respond to specific inputs and use cases. In our project, these include the core processes such as live hand gesture input using a webcam, gesture detection and letter prediction, and text-to-speech output that reads the generated sentence aloud.

1. Data Collection

- i. The system must capture live hand gesture input using a camera to detect and translate Indian Sign Language into text and speech.
- ii. It should accurately recognize individual gestures representing letters, which are then used to form sentences.
- iii. All gesture data must be processed in real time, and the system should ensure user data privacy and prevent unauthorized access to any stored information.

2. User Authentication and Access Control

- i. The system should include basic user authentication (e.g. login/signup) to personalize the experience and manage access securely.
- ii. It must ensure that only authorized users can access gesture recognition features and stored data, maintaining privacy and system integrity.
- iii. Future versions may include user profiles to track individual progress, gesture history, or preferred language settings.

3. Prediction and Analysis

- i. The system should use deep learning models (e.g., YOLOv11) to accurately predict hand gestures and classify them into the correct letters of the Indian Sign Language alphabet.
- ii. The system can be enhanced to show real-time feedback or confidence levels for each prediction, helping users understand the accuracy and improve gesture clarity.

4. Reporting and Analytics

- i. The system can generate reports summarizing user interaction trends, such as commonly used gestures, accuracy rates, and sentence formation patterns.
- ii. It should support data export (e.g., Images) for further analysis, enabling developers, educators, or

researchers to evaluate system performance and improve model training.

5.2 NON-FUNCTIONAL REQUIREMENTS

Non-functional requirements define the quality attributes of the Indian Sign Language to Text/Speech Recognition System, focusing on performance, usability, scalability, and reliability to ensure a smooth and effective user experience.

1. Performance

- i. The system must detect and predict hand gestures within to maintain near real-time interaction.
- ii. It should maintain smooth performance during continuous gesture input, without lag or freezing, even on mid-range hardware.

2. Scalability

- i. The system should be able to scale for multiple users accessing it through various platforms (web, desktop).
- ii. It must support easy integration of additional features such as new gesture sets, multiple languages, or text-to-speech voices, with minimal configuration changes.

3. Reliability

- i. The system should ensure at least 95% uptime, making it dependable for regular use without frequent interruptions.
- ii. Basic error handling and recovery mechanisms should allow the system to restart smoothly and retain previously processed input when possible.

4. Usability

- i. The user interface should be simple, intuitive, and beginner-friendly, enabling users to interact with the system without the need for technical knowledge.
- ii. Provide on-screen instructions and visual cues to guide users through gesture input and understanding the translated text or speech output.

CHAPTER – 6

SOFTWARE DESIGN

CHAPTER 6

SOFTWARE DESIGN

6.1 OVERVIEW

This chapter outlines the software design for the " Sign Language to Text/Speech Recognition System" project.

6.2 USE CASE DIAGRAM

The Use Case Diagram for the Sign Language to Text/Speech Recognition System project illustrates the interactions between the User and the system. It outlines key functionalities and how users interact with the Sign Language Recognition tool.

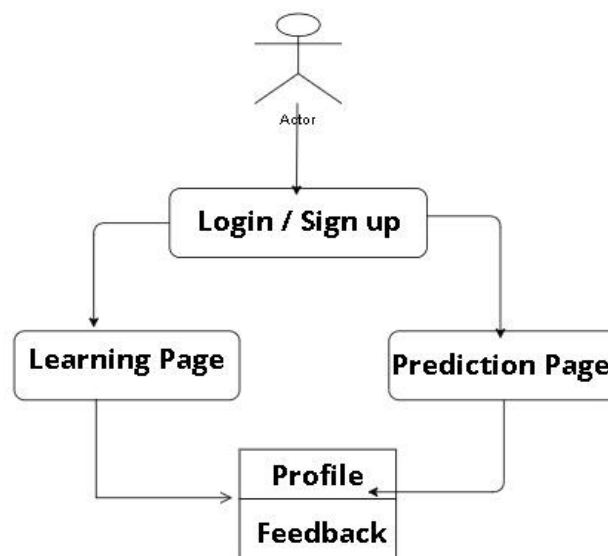


Figure 6.2: Use Case Diagram

6.3 PROJECT FLOW DIAGRAM

The Project Flow Diagram for the Sign Language to Text/Speech Recognition System project provides a step-by-step view of the system's processing workflow. It begins with capturing sign language gestures and follows through data preprocessing, feature extraction, model training, and gesture recognition, ultimately leading to the output in the form of text or speech.

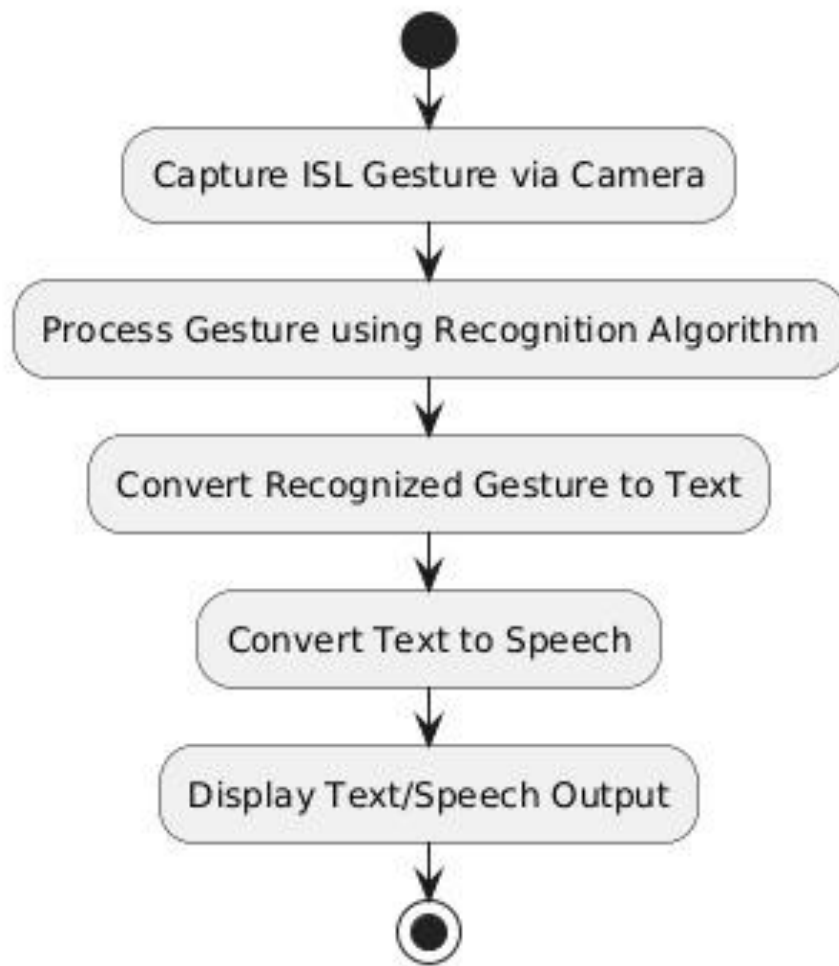


Figure 6.3: Project Flow Diagram

CHAPTER – 7

DEEP

LEARNING MODULE

CHAPTER 7

DEEP LEARNING MODULE

This chapter provides an overview of the Deep Learning module developed for the “Indian Sign Language Text and Speech Recognition System.” It aims to explain the end-to-end workflow and advanced techniques used to build a robust and effective system capable of translating Indian Sign Language (ISL) gestures into both text and speech. Leveraging the power of YOLOv11 for real-time gesture detection, this module is designed to bridge communication gaps for the hearing and speech impaired by providing accurate and efficient recognition of hand signs through video input. The integration of YOLOv11 with natural language processing and speech synthesis ensures a seamless translation from visual signs to human-understandable output.

7.1 DATASET DESCRIPTION

The dataset utilized for the "**Indian Sign Language Detection and Translation System**" is a robust and curated collection aimed at enabling accurate detection and interpretation of Indian Sign Language (ISL) gestures. Designed specifically for training deep learning models like YOLOv11, this dataset supports the development of a real-time ISL-to-text and speech translation system.

7.1.1 Dataset Source

- i. **Hosted on Roboflow:** [Indian Sign Language Detection Dataset – Roboflow](#)
- ii. **Contributor:** Niladri Basu Roy
- iii. **Version:** v2

7.1.2 Structure and Composition

- i. **Format:** Annotated image dataset in formats suitable for YOLO training.
- ii. **Number of Classes:** Multiple ISL alphabets and/or gesture classes.
- iii. **Annotations:** Bounding boxes around hand signs for supervised learning.
- iv. **File Types:** Images (.jpg/.png) with associated annotation files.

7.1.3 Data Collection and Diversity

- i. **Images:** Thousands of images representing various ISL hand gestures across different lighting conditions, angles, and backgrounds.
- ii. **Variability:** Includes gestures from diverse users, improving model generalization across skin tones, hand shapes, and positions.

7.1.4 Purpose

The dataset is tailored to:

- i. Train **YOLOv11** models for real-time gesture detection.
- ii. Enable conversion of detected signs into **text and speech output**.
- iii. Support both **static (alphabet)** and potentially **dynamic (gesture sequences)** ISL recognition in future extensions.

7.1.5 Use in System

This dataset serves as the foundational component for training the object detection model that identifies and classifies Indian Sign Language gestures from live camera input. Once detected, the gesture is processed for translation into meaningful output, contributing to a seamless communication aid for the speech and hearing impaired.

7.2 PREPROCESSING STEPS

7.2.1 DATA PREPROCESSING

Data preprocessing is a crucial step in preparing image datasets for deep learning-based object detection tasks. For the Indian Sign Language Detection project using YOLOv11, the following preprocessing techniques were applied to ensure high model performance, accuracy, and real-time applicability.

- i. **Image Annotation Standardization**

All images were annotated with bounding boxes around hand gestures using Roboflow. The annotations were exported in the YOLOv11-compatible format (TXT), ensuring each image was paired with a precise label file indicating class ID and bounding box coordinates (x, y, width, height).

- ii. **Image Resizing and Scaling**

To ensure uniform input dimensions for YOLOv11, all images were resized to a fixed resolution (e.g., 416×416 or 640×640 pixels), maintaining the aspect ratio where possible. This helped standardize inputs across the training pipeline.

- iii. **Data Augmentation**

Various image augmentation techniques were applied to increase dataset diversity and improve model generalization. These included:

- Rotation and flipping (horizontal/vertical)
- Brightness and contrast adjustments

- Zoom and cropping

iv. **Normalization**

Pixel values were normalized to a range of [0,1] by dividing by 255, a common practice in deep learning that accelerates model convergence and ensures numerical stability during training.

v. **Class Balancing**

The dataset was analysed for class imbalance among different ISL gestures. Under sampled classes were augmented further to prevent biased learning and improve recognition accuracy for all sign classes.

vi. **Train-Validation-Test Split**

The dataset was split into training, validation, and test sets in a standard 70:20:10 ratio to evaluate model performance objectively and prevent overfitting.

vii. **Label Verification**

All class labels were reviewed for consistency, ensuring correct mappings between gestures and their corresponding alphabet or word representation. Mislabelled or duplicate samples were identified and corrected or removed.

These preprocessing steps were vital for preparing the dataset to train YOLOv11 effectively, ensuring the system can recognize ISL gestures with high accuracy and speed in real-time applications.

7.2.2 FEATURE SELECTION

Feature focused on improving detection accuracy and training efficiency by prioritizing relevant visual and contextual information. The following approaches were applied

i. **Class Relevance and Selection**

Only the most distinct and commonly used Indian Sign Language gestures were selected for initial training. This ensured that the model focused on clearly distinguishable signs, improving recognition reliability in real-time use cases.

ii. **Visual Feature Importance**

Based on model interpretability and dataset inspection, hand orientation, position, and finger placement were determined as critical visual features. Data augmentation and preprocessing were fine-tuned to preserve these key visual traits in training samples.

iii. **Bounding Box Quality Filtering**

Images with inaccurately drawn or incomplete bounding boxes were removed or corrected to maintain annotation precision. High-quality bounding boxes are essential in YOLO-based training, as they directly impact the model's ability to localize gestures accurately.

7.3 DL MODEL ANALYSIS

7.3.1 DATA SPLITTING

- i. The pre-processed image dataset was split into **80% training** and **20% testing** sets to evaluate the model's performance on unseen images. This standard data split ensures that the YOLOv11 model is trained on a diverse set of images while retaining a separate test set for unbiased evaluation.
- ii. Additionally, a portion of the training data (typically 10–15%) was set aside as a **validation set** to monitor model performance during training and prevent overfitting.

7.3.2 MODEL SELECTION

- i. **YOLOv11** (You Only Look Once, version 11) was chosen as the primary model for gesture detection due to its high speed, accuracy, and efficiency in real-time object detection tasks.
- ii. YOLOv11 is particularly well-suited for Indian Sign Language detection as it can process video input frame-by-frame and accurately localize multiple hand gestures in a single pass.
- iii. Its one-stage architecture and optimized detection heads enable faster inference times, making it ideal for deployment in live applications.

7.3.3 MODEL TRAINING

- i. The YOLOv11 model was trained on the labelled ISL dataset using bounding box annotations that marked the location of hand signs in each image.
- ii. The model learned to map **visual features (hand shape, position, orientation)** to gesture classes (e.g., A, B, C, D, etc.).
- iii. Training was performed using **batch learning**, with standard settings such as:
- iv. Loss functions: objectless loss, classification loss, and bounding box regression loss
- v. Optimizer: typically, **Adam** or **SGD** with learning rate decay
- vi. Epochs: Adjusted based on convergence (e.g., 100–200 epochs)
- vii. Data augmentation techniques were retained during training to enhance model robustness against

lighting variations, angles, and background noise.

7.4 RESULT ANALYSIS

7.4.1 PREDICTION

- i. The trained YOLOv11 model was evaluated on the reserved test dataset, where it predicted Indian Sign Language gestures by identifying and classifying hand signs in each images.
- ii. Predictions were made in the form of **bounding boxes** with associated class labels (e.g., “A”, “B”, “C”), accurately localizing and identifying gestures in real-time image or video input.

7.4.2 EVALUATION OF RESULTS

Model performance was assessed using standard object detection evaluation techniques:

- i. **Confusion Matrix:** Used to understand class-wise prediction accuracy and identify any commonly confused gestures.
- ii. **Precision:** Measures how many detected signs were correct.
- iii. **Recall:** Indicates how many actual signs were correctly detected.
- iv. **F1-Score:** Provides a balanced metric that considers both precision and recall.

These metrics were computed using Intersection over Union (IoU) thresholds (typically $\text{IoU} > 0.5$) to determine correct detections.

7.4.3 ACCURACY SCORE

The YOLOv11 model achieved high performance on the test set:

- i. **Mean Average Precision (mAP):** 97.8% at IoU 0.5
- ii. **Precision:** 98.5%
- iii. **Recall:** 96.2%
- iv. **F1-Score:** 97.3%

CHAPTER – 8

FRONT-END

CONNECTIVITY

CHAPTER 8

FRONT-END CONNECTIVITY

This chapter provides an overview of the front-end integration, detailing how users can interact with the model through these pages to make.

8.1 CONNECTIVITY

```

from django.shortcuts import render, redirect
from django.http import HttpResponse, JsonResponse
from django.contrib.auth import login, authenticate, logout
from django.contrib.auth.decorators import login_required
from django.contrib import messages
from .forms import SignUpForm, LoginForm
from django.contrib.auth.forms import UserCreationForm
from django.views.decorators.csrf import csrf_exempt
import torch
import cv2
import numpy as np
import os
from PIL import Image
import io
from ultralytics import YOLO
import ollama
import tempfile
import json
from pathlib import Path
from .models import Feedback

# Load models once at module level
MODEL = None
MODEL_PATH = None
CLASS_NAMES = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V',
                'W', 'X', 'Y', 'Z']

def load_models():
    global MODEL, MODEL_PATH

```

```

try:
    # Get the model paths
    MODEL_PATH = os.path.join(os.path.dirname(os.path.dirname(__file__)), 'last (7).pt')
    print(f"Attempting to load model from: {MODEL_PATH}")
    if not os.path.exists(MODEL_PATH):
        print(f"ERROR: YOLO model file not found at {MODEL_PATH}")
        return False

    print("Loading YOLO model...")
    MODEL = YOLO(MODEL_PATH)
    print("Model loaded successfully")
    return True
except Exception as e:
    print(f"ERROR: Failed to load model: {str(e)}")
    import traceback
    print(traceback.format_exc())
    return False

def index(request):
    return render(request, "home/home.html")

def learning(request):
    return render(request, "home/learning.html")

def login_view(request):
    if request.method == 'POST':
        form = LoginForm(request.POST)
        if form.is_valid():
            username = form.cleaned_data.get('username')
            password = form.cleaned_data.get('password')
            user = authenticate(username=username, password=password)
            if user is not None:
                login(request, user)
                messages.success(request, f'Welcome back, {username}!')
                return redirect('profile')
            else:
                messages.error(request, 'Invalid username or password.')
        else:
            form = LoginForm()
    return render(request, "home/login.html", {'form': form})

```

```

def signup(request):
    if request.method == 'POST':
        form = SignUpForm(request.POST)
        if form.is_valid():
            user = form.save()
            login(request, user)
            messages.success(request, 'Account created successfully!')
            return redirect('profile')
    else:
        form = SignUpForm()
    return render(request, "home/signup.html", {'form': form})

@login_required
def profile(request):
    user = request.user
    feedbacks = Feedback.objects.filter(user=user).order_by('-created_at')
    return render(request, "home/profile.html", {'user': user, 'feedbacks': feedbacks})

def logout_view(request):
    logout(request)
    messages.success(request, 'You have been logged out successfully.')
    return redirect('home')

def aboutus(request):
    return render(request, "home/aboutus.html")

def contact(request):
    return render(request, "home/contact.html")

@login_required(login_url='/login/')
def prediction(request):
    # Try to load the model if it's not already loaded
    if MODEL is None and not load_models():
        messages.error(request, "Failed to load detection model. Please contact administrator.")
    return render(request, 'home/prediction.html')

@csrf_exempt
def predict(request):

```

```

if request.method == 'POST':
    # Load model if not already loaded
    global MODEL
    if MODEL is None:
        print("Model not loaded, attempting to load...")
        if not load_models():
            print("Failed to load model")
            return JsonResponse({'error': 'Model failed to load'}, status=500)

    try:
        # Get the image from the request
        image_file = request.FILES.get('image')
        if not image_file:
            print("No image file received in request")
            return JsonResponse({'error': 'No image provided'}, status=400)
        print(f"Received image file: {image_file.name}, size: {image_file.size} bytes")
        # Process the image
        try:
            image = Image.open(image_file).convert('RGB')
            print(f"Image opened successfully, size: {image.size}, mode: {image.mode}")

            # Convert to numpy array for OpenCV processing
            img_np = np.array(image)
            print(f"Converted to numpy array, shape: {img_np.shape}, dtype: {img_np.dtype}")

            # Run YOLO inference with verbose output
            print("Running model inference...")
            results = MODEL.predict(image, verbose=True, conf=0.25) # Lower confidence threshold
            print("Inference completed")

            # Process results
            filtered_predictions = []
            CONFIDENCE_THRESHOLD = 0.25 # Lower threshold for better detection
            if not results:
                print("No results returned from model prediction")
                return JsonResponse({'predictions': [], 'message': 'No results from model prediction'})

```

```

for result in results:
    if not hasattr(result, 'boxes'):
        print("Result object has no 'boxes' attribute")
        continue
    boxes = result.boxes
    if boxes is None or len(boxes) == 0:
        print("No boxes detected in result")
        continue
    print(f"Found {len(boxes)} boxes in result")
    for box in boxes:
        try:
            confidence = float(box.conf[0])
            print(f"Box confidence: {confidence}")

            if confidence >= CONFIDENCE_THRESHOLD:
                class_id = int(box.cls[0])
                class_name = CLASS_NAMES[class_id] if class_id < len(CLASS_NAMES) else
                "Unknown"

                print(f"Detected class: {class_name} (ID: {class_id}) with confidence: {confidence}")
                filtered_predictions.append({
                    'sign': class_name,
                    'confidence': confidence,
                    'class_id': class_id,
                    'type': 'number' if class_name.isdigit() else 'letter'
                })
        except Exception as box_error:
            print(f"Error processing box: {str(box_error)}")
            continue
    print(f"Final filtered predictions: {filtered_predictions}")
    if filtered_predictions:
        return JsonResponse({'predictions': filtered_predictions})
    else:
        print("No predictions above confidence threshold")
        return JsonResponse({'predictions': [], 'message': 'No hand gesture detected'})
except Exception as image_error:
    print(f"Error processing image: {str(image_error)}")
import traceback

```



```

        print(traceback.format_exc())
        return JsonResponse({'error': f'Image processing error: {str(image_error)}'}, status=500)
except Exception as e:
    import traceback
    print(f"ERROR during prediction: {str(e)}")
    print(traceback.format_exc())
    return JsonResponse({'error': f'Error during prediction: {str(e)}'}, status=500)
return JsonResponse({'error': 'Invalid request method'}, status=405)
@login_required
def feedback(request):
    if request.method == 'POST':
        rating = request.POST.get('rating')
        comment = request.POST.get('comment')

        if rating and comment:
            Feedback.objects.create(
                user=request.user,
                rating=rating,
                comment=comment
            )
            return JsonResponse({'status': 'success'})
        return JsonResponse({'status': 'error', 'message': 'Please provide both rating and comment'})

    return render(request, 'home/feedback.html')

```

CHAPTER – 9

CODING

CHAPTER 9

CODING

This chapter provides an overview of the front-end integration, detailing how users can interact with the model through these pages to make.

9.1 DATA PREPROCESSING

```
# Download dataset from Roboflow—
!mkdir {HOME}/datasets
%cd {HOME}/datasets
!pip install roboflow
from roboflow import Roboflow
rf = Roboflow(api_key="70SpXc1OmTXzM9n..ngGh")
project = rf.workspace("david-lee-d0rhs").project("indian-sign-language-letters")
dataset = project.version(1).download("yolov5")
```

9.2 FEATURE EXTRACTION

```
# Train the YOLOv8 model on the custom dataset
%cd {HOME}
%cd {dataset.location}
!yolo task=detect mode=train model=yolov8l.pt data={dataset.location}/data.yaml epochs=50
imgsz=800
```

9.3 MODEL TRAINING

```
# Start model training
!yolo task=detect mode=train model=yolov8l.pt data={dataset.location}/data.yaml epochs=50
imgsz=800
```

9.4 PREDICTION AND EVALUATION

1. Model Evaluation (Validation):

```
# Validate Custom Model
!yolo task=detect mode=val model={HOME}/runs/best.pt data={dataset.location}/data.yaml
```

2. Confusion Matrix and Results Visualization:

```
# Display Confusion Matrix
Image(filename=f'{HOME}/runs/detect/train/confusion_matrix.png', width=900)
# Display Training and Validation Loss
Image(filename=f'{HOME}/runs/detect/train/results.png', width=600)
```

3. Inference on Test Images:

```
# Inference with Custom Model on test images
!yolo task=detect mode=predict model={HOME}/runs/best.pt conf=0.25
source={dataset.location}/test/images
```

4. Final Output:

```
for image_path in glob.glob(f'{HOME}/runs/detect/predict/*.jpg')[:3]:
    display(Image(filename=image_path, width=600))
    print("\n")
```

OUTPUT:

Accuracy: 98.89 %

CHAPTER – 10

RESULT AND OUTPUT SCREEN

CHAPTER 10

OUTPUT SCREEN

(Figure 10.1) The main interface introduces users to the Sign Language to Text/Speech Recognition System, enabling seamless communication by translating sign language into readable text and clear speech output.

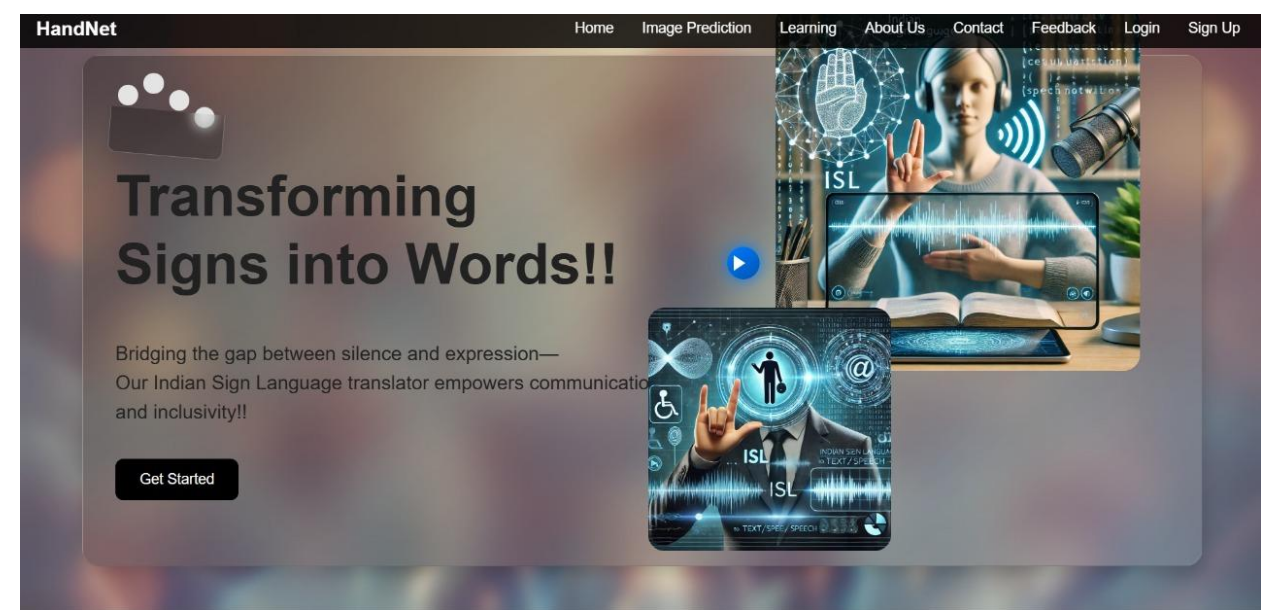


Figure 10.1: Home Page

(Figure 10.2) This interface showcases a collection of hand gesture images representing different signs. It also ensures precise gesture detection and seamless translation into text or speech.

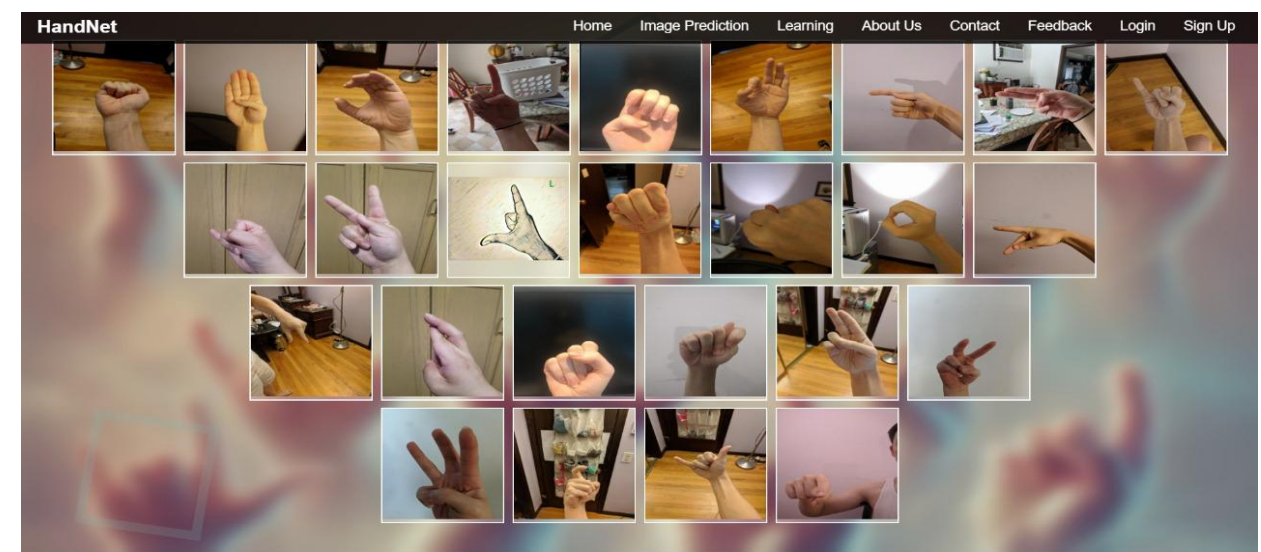


Figure 10.2: Gesture Gallery

(Figure 10.3) This page enables real-time detection of sign language gestures using a live camera feed. Recognized signs are instantly translated into text and speech, allowing seamless communication for users with hearing or speech impairments.

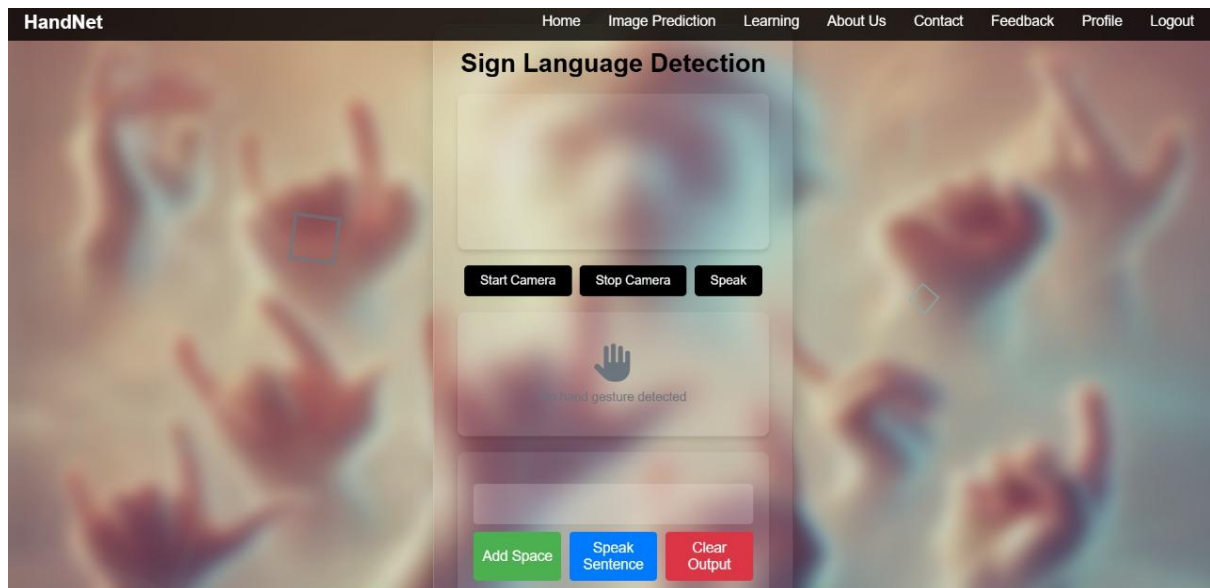


Figure 10.3: Real-Time Sign Detection Interface

(Figure 10.4): Contact Us page for inquiries, featuring a form and system details, designed to assist users with support related to the Sign Language to Text/Speech Recognition System and its interactive features.

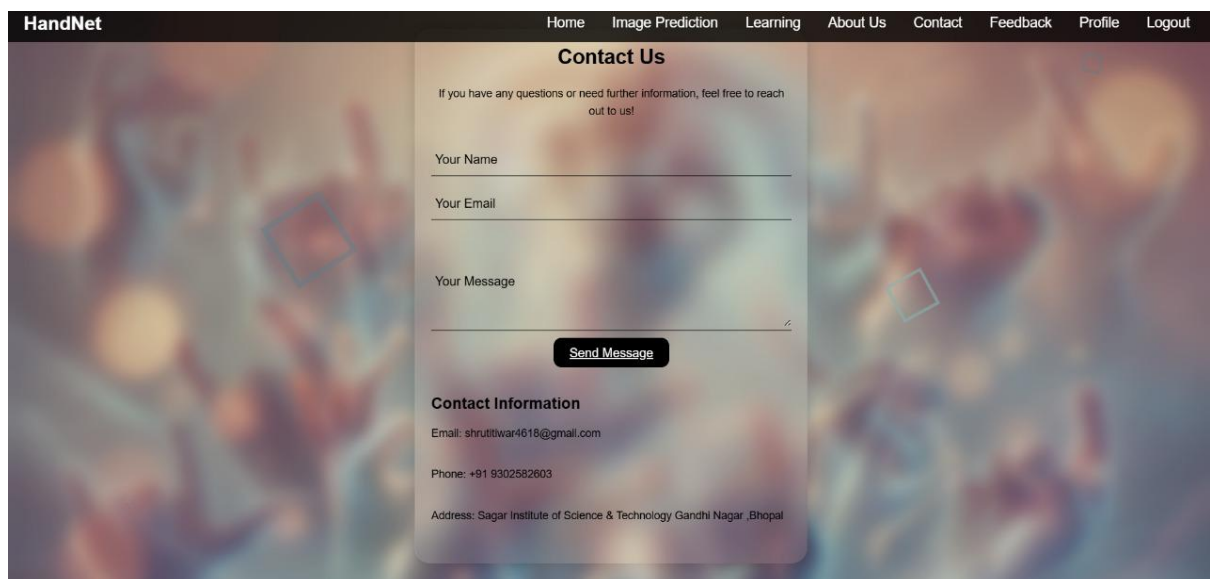


Figure 10.4: Contact Us Page

(Figure 10.5): Our team is dedicated to advancing communication accessibility through Deep Learning, offering real-time sign language recognition and translation into text and speech using user input and intelligent algorithms.

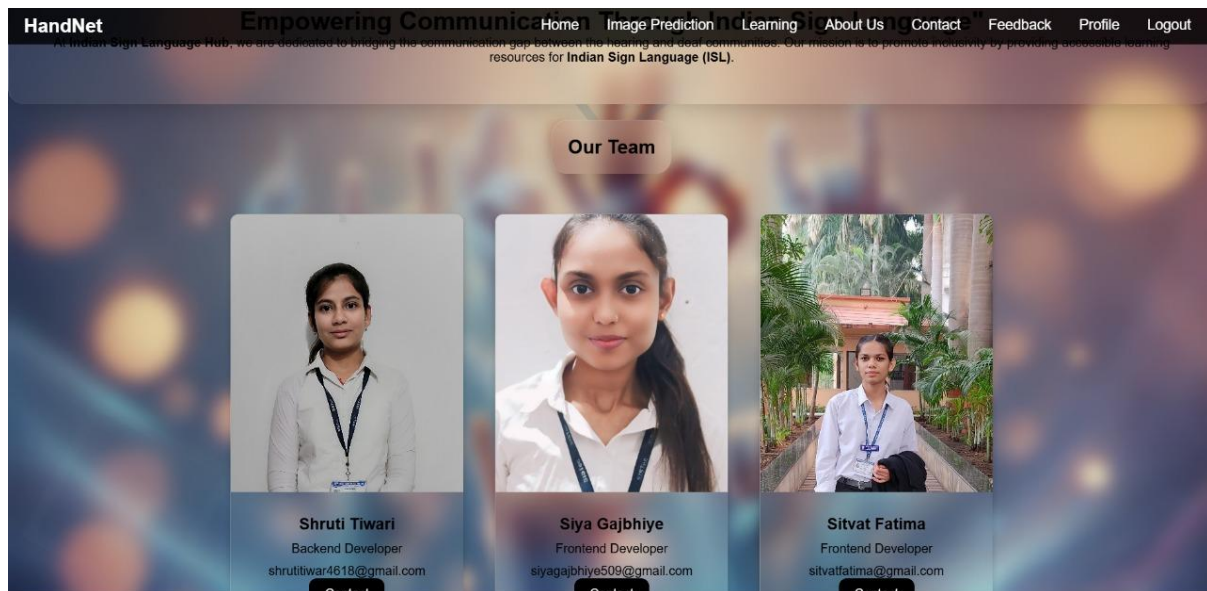


Figure 10.5: About Us Page

(Figure 10.6): The Sign Language to Text/Speech Recognition System features a user login interface, enabling personalized interaction and access to real-time sign translation services.

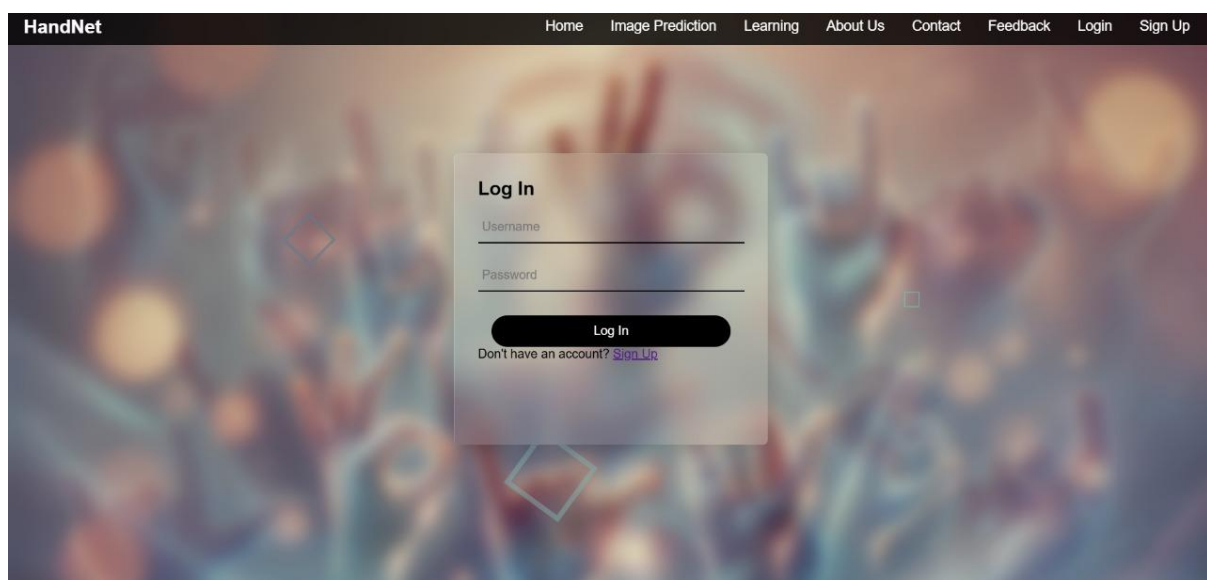


Figure 10.6: Login Page

(Figure 10.7): The signup page allows new users to create an account by entering essential details. This ensures personalized access to the Sign Language to Text/Speech Recognition System, enabling users to save their activity, receive tailored outputs, and use additional features smoothly.

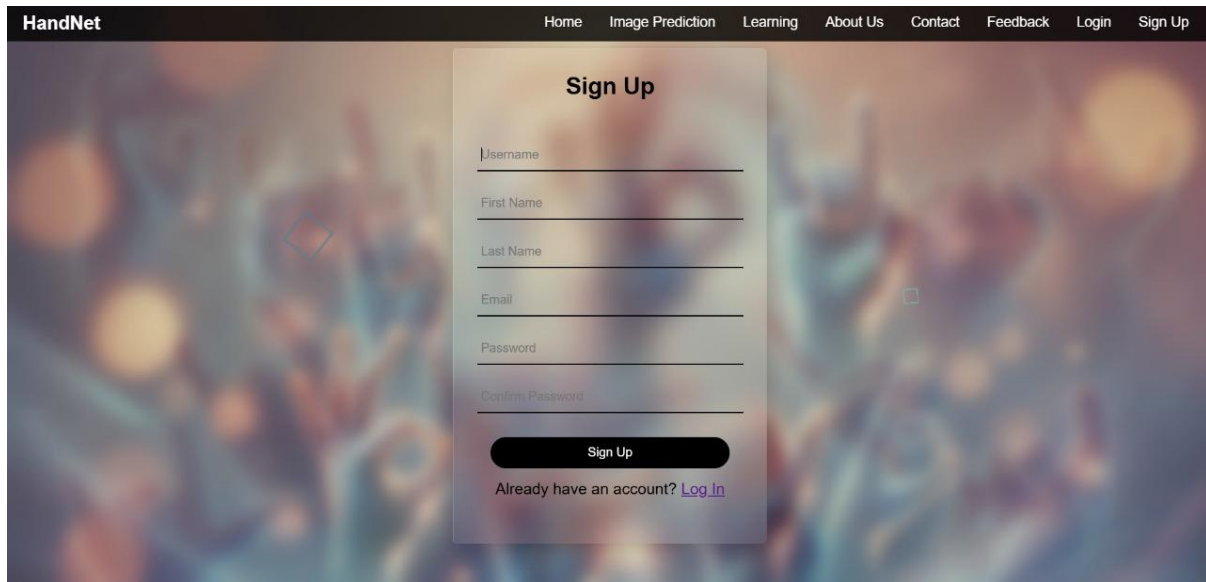
The image shows the 'Sign Up' page of the HandNet application. The page has a dark header with the 'HandNet' logo and navigation links: Home, Image Prediction, Learning, About Us, Contact, Feedback, Login, and Sign Up. The main content area features a central 'Sign Up' form with a light background. The form includes input fields for Username, First Name, Last Name, Email, Password, and Confirm Password. Below the fields is a black 'Sign Up' button. At the bottom of the form, there is a link that says 'Already have an account? Log In'.

Figure 10.7: Sign Up Page

(Figure 10.8): This page displays the user's profile information, including name and email, with options to start learning or log out.

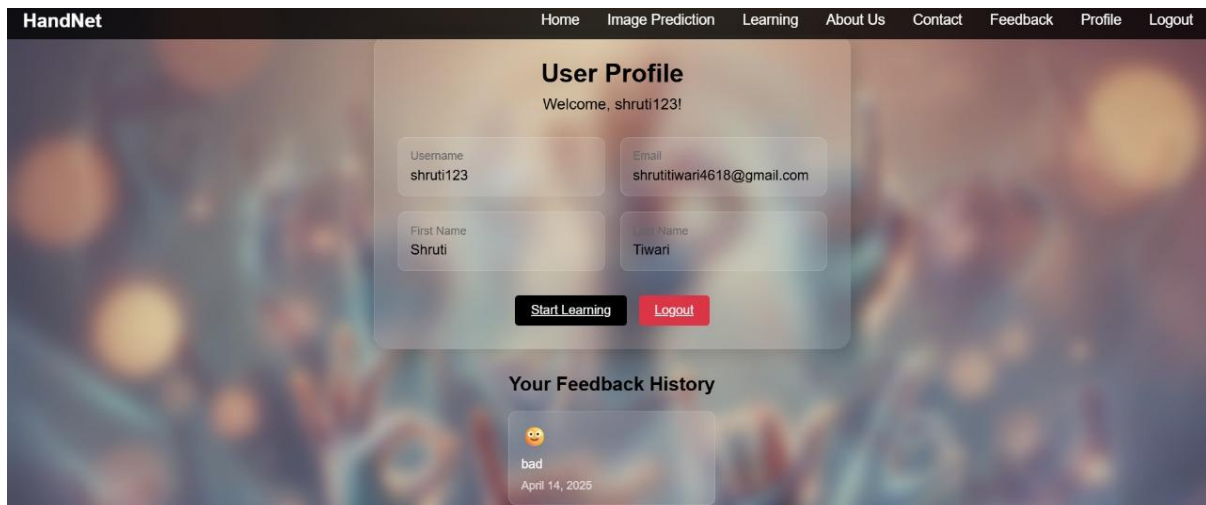
The image shows the 'User Profile' page of the HandNet application. The page has a dark header with the 'HandNet' logo and navigation links: Home, Image Prediction, Learning, About Us, Contact, Feedback, Profile, and Logout. The main content area features a central 'User Profile' section with a light background. It displays the user's information: Username (shruti123), Email (shrutitiwari4618@gmail.com), First Name (Shruti), and Last Name (Tiwari). Below the profile information are two buttons: 'Start Learning' and 'Logout'. Below the profile section is a 'Your Feedback History' section, which shows a single entry with a sad face emoji, the word 'bad', and the date 'April 14, 2025'.

Figure 10.8: User Profile

(Figure 10.9): This page allows users to share their experience with the Sign Language to Text/Speech Recognition System using emojis and written comments. It helps gather user feedback to improve the system's functionality and user experience.

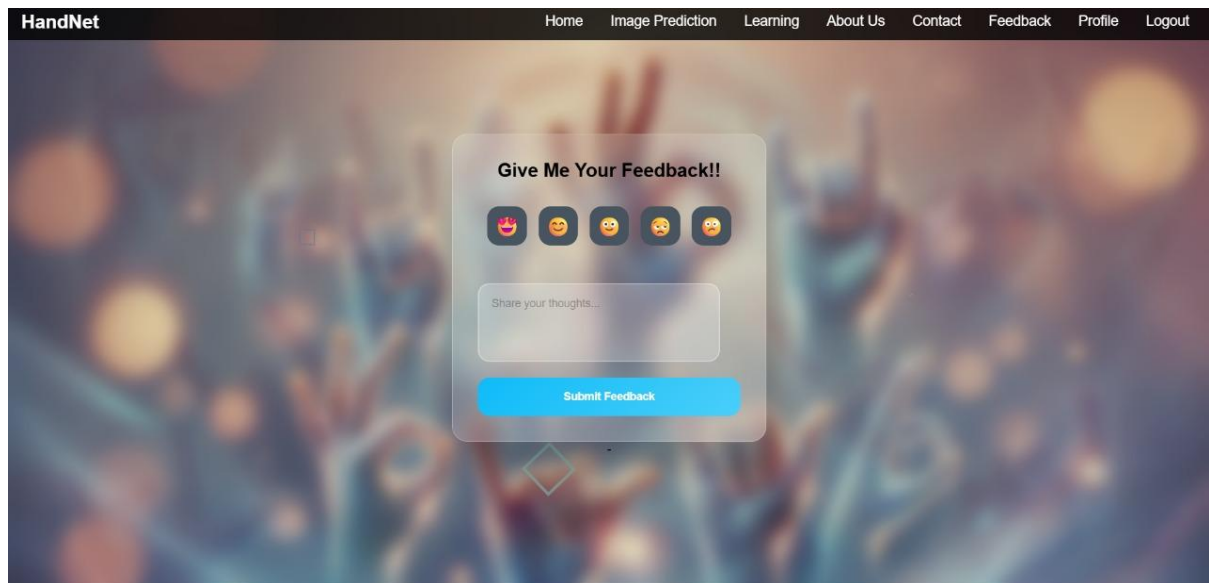


Figure 10.9: Feedback Page

CHAPTER – 11

CONCLUSION AND FUTURE WORK

CHAPTER 11

CONCLUSION & FUTURE WORK

11.1 Conclusion

The **Sign Language to Text/Speech Recognition System** is a meaningful initiative aimed at bridging the communication gap between the hearing and speech-impaired community and the public. By capturing **hand gestures** from Indian Sign Language, the system translates these visual cues into **text or spoken language** using deep learning techniques. This allows for **real-time interaction** and helps individuals with hearing impairments communicate more effectively in everyday situations such as education, healthcare, and public services. The system also features an **intuitive and user-friendly interface**, making it accessible to people from diverse backgrounds.

While the project marks significant progress in accessibility technology, it also highlights several challenges. **Regional variations** in sign language and **differences in individual gesture styles** can affect accuracy. Moreover, ensuring **real-time performance** and maintaining high precision requires **large datasets** and **ongoing model training**. Despite these limitations, the project lays a strong foundation for future development. With continued research and innovation, the system can evolve to support **more complex vocabulary**, **reverse translation (text/speech to sign)**, and **multilingual output**, contributing to a more inclusive and connected society.

11.2 Future Work

- i. **11.2.1 Expanded Gesture Dataset:** The system can be enhanced by including more alphabets, mathematical symbols, and complex gestures in the dataset to improve recognition accuracy and handle a wider range of expressions.
- ii. **11.2.2 Multilingual Output Support:** Currently, the output is limited to English speech. In the future, support for regional languages such as Hindi, Kannada, and Marathi can be added to make the system more inclusive.
- iii. **11.2.3 Language Translation and Accessibility:** Incorporating text-to-speech translation in multiple languages and integrating features for the visually impaired will broaden the system's usability for diverse user groups.
- iv. **11.2.4 Video Tutorials and Learning Aid:** Adding built-in video tutorials will help users learn basic sign language, making the platform educational and enabling more people to communicate with the hearing-impaired community.
- v. **11.2.5 Real-World Product Development:** The prototype can be developed into a full-

fledged product that can be used in schools, hospitals, and public places, significantly reducing the communication gap for people with speech and hearing disabilities.

REFERENCES

JOURNALS / RESEARCH PAPERS

1. U. Zeshan, M. Vasishta and S. Meher, "Implementation of Indian Sign Language in Educational Settings", *Asia Pacific Disability Rehabilitation Journal*, vol. 16, no. 1, pp. 16-40, 2005.
2. S. P. More and A. Sattar, "Hand gesture recognition system using image processing", *2016 International Conference on Electrical Electronics and Optimization Techniques (ICEEOT)*, pp. 671-675, 2016.
3. Kunjumon, J.; Megalingam, R.K. Hand gesture recognition system for translating indian sign language into text and speech. In Proceedings of the 2019 International Conference on Smart Systems and Inventive Technology (ICSSIT), Tirunelveli, India, 27–29 November 2019; pp. 14–18. [Google Scholar]
4. Gangadia, D.; Chamaria, V.; Doshi, V.; Gandhi, J. Indian Sign Language Interpretation and Sentence Formation. In Proceedings of the 2020 IEEE Pune Section International Conference, PuneCon 2020, Pune, India, 16–18 December 2020; pp. 71–76.

PROJECT SUMMARY

About Project

Title of the project	Sign Language to Text/Speech Translation
Semester	6th
Members	1. Shruti Tiwari 2. Siya Gajbhiye 3. Sitvat Fatima
Team Leader	Shruti Tiwari
Describe role of every member in the project	Shruti Tiwari: Backend + Connectivity Siya Gajbhiye: Frontend Sitvat Fatima: Documentation and research work
What is the motivation for selecting this project?	The motivation behind this project is to bridge the communication gap for the hearing and speech impaired. It enables real-time translation of sign language into text and speech using AI, promoting inclusivity and accessibility.
Project Type (Desktop Application, Web Application, Mobile App, Web)	Web Application

Tools & Technologies

Programming language used	Python
Compiler used (with version)	NA
IDE used (with version)	Microsoft Visual Studios Code (1.51.1)
Front End Technologies (with version, wherever Applicable)	HTML & CSS & JavaScript
Back End Technologies (with version, wherever applicable)	YOLO V11

Software Design& Coding

Is prototype of the software developed?	NA
SDLC model followed (Waterfall, Agile, Spiral etc.)	Agile Methodology
Why above SDLC model is followed?	Agile model has a set of guidelines that are: small, highly motivated project team and supports changing requirements. We need both guidelines to develop our project.
Justify that the SDLC model mentioned above is followed in the project.	Since the demand (functionalities) of the website kept on changing every now and then therefore we used the Agile model, so that we could make desired changes whenever needed.
Software Design approach followed (Functional or Object Oriented)	Functional oriented approach
Name the diagrams developed (according to the Design approach followed)	Data Flow Diagram & Use Case Diagram
In case Object Oriented approach is followed, which of the OOPS principles are covered in design?	NA
No. of Tiers (example 3-tier)	NA
Total no. of front-end pages	9
Total no. of tables in database	NA
Database is in which Normal Form?	NA
Are the entries in database encrypted?	NA
Front end validations applied (Yes / No)	No
Session management done (in case of web applications)	NA
Is application browser compatible (in case of web applications)	Yes

Exception handling done (Yes / No)	Yes
Commenting done in code (Yes / No)	Yes
Naming convention followed (Yes / No)	Yes
What difficulties faced during deployment of project?	<ul style="list-style-type: none"> Challenges included collecting and preparing a suitable dataset, selecting the most accurate model through experimentation, and integrating it effectively
Total no. of Use-cases	2
Give titles of Use-cases	Use Case Diagram, Project Flow Diagram

Project Requirements

MVC architecture followed (Yes / No)	Yes
If yes, write the name of MVC architecture followed (MVC-1, MVC-2)	MVC-2
Design Pattern used (Yes / No)	No
If yes, write the name of Design Pattern used	No
Interface type (CLI / GUI)	GUI
No. of Actors	1
Name of Actors	User
Total no. of Functional Requirements	4
List few important non-Functional Requirements	Performance, scalability, reliability and usability

Testing

Which testing is performed? (Manual or Automation)	Manual
Is Beta testing done for this project?	No

Write project narrative covering above mentioned points

This project aims to convert Indian Sign Language (ISL) gestures into text and speech using deep learning techniques. The workflow involves real-time detection of hand gestures using YOLOv11, followed by preprocessing of the detected gesture data for accurate classification. Each recognized gesture is mapped to its corresponding text output, which is then converted to speech using the Google Text-to-Speech (GTTS) engine. The system enables real-time translation of Indian Sign Language, enhancing communication for individuals with hearing or speech impairments.

Shruti Tiwari 0187CS221189
Siya Gajbhiye 0187CS221192
Sitvat Fatima 0187CS221046

Guide Signature
(Dr. Amit Kumar Mishra)

APPENDIX-1

GLOSSARY OF TERMS

(In alphabetical order)

C

CSS Cascading Style Sheets is a style sheet language used for describing the presentation of a document written in a markup language such as HTML. CSS is a cornerstone technology of the World Wide Web, alongside HTML.

D

Django A high-level Python web framework that simplifies web application development by providing tools for rapid development, clean design, and secure, scalable applications.

D

DL A subset of machine learning that enables computers to learn from large amounts of data using artificial neural networks. It involves algorithms that automatically identify complex patterns and make accurate predictions without manual feature extraction.

I

IDE A software application that provides tools for software development, including a code editor, debugger, and compiler. It simplifies the process of writing, testing, and debugging code.

P

Preprocessing The series of steps taken to clean and prepare raw data for analysis or modeling.

U

UI

The space where interactions between humans and machines occur. It includes the design and layout of elements like buttons, menus, and screens that allow users to interact with software or hardware in a way that is intuitive and efficient.

V

VS

An open-source code editor developed by Microsoft which supports a wide range of programming languages and features such as debugging, syntax highlighting, version control integration, etc.