SIGN TO TEXT AND SPEECH TRANSLATOR USING MACHINE LEARNING

A PROJECT REPORT

Submitted by

DANIYA MARIYA JIJU (TKR20CS025)
DINIYA P (TKR20CS030)
DIVYA A (TKR20CS031)
JISHNA BABU (TKR20CS086)

to

the A P J Abdul Kalam Technological University
in partial fulfillment of the requirements for the award of the Degree

of

Bachelor of Technology

In

Computer Science and Engineering

DEPT. OF COMPUTER SCIENCE & ENGINEERING

COLLEGE OF ENGINEERING TRIKARIPUR

MAY 2024

DECLARATION

We undersigned hereby declare that the project report ”SIGN TO TEXT AND SPEECH TRANSLATOR USING MACHINE LEARNING” , submitted for partial fulfillment of the requirements for the award of degree of Master of Technology of the APJ Abdul Kalam Technological University, Kerala is a bonafide work done by me under supervision of Mr. Anoop P V. This submission represents our ideas in my own words and where ideas or words of others have been included, We have adequately and accurately cited and referenced the original sources. We also declare that we have adhered to ethics of academic honesty and integrity and have not misrepresented or fabricated any data or idea or fact or source in my submission. We understand that any violation of the above will be a cause for disciplinary action by the institute and/or the University and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been obtained. This report has not been previously formed the basis for the award of any degree, diploma or similar title of any other University.

DANIYA MARIYA JIJU
Cheemeni DINIYA P
24/04/2024 DIVYA A
JISHNA BABU

DEPT. OF COMPUTER SCIENCE & ENGINEERING

COLLEGE OF ENGINEERING TRIKARIPUR

2023 - 2024

CERTIFICATE

This is to certify that the report entitled **SIGN TO TEXT AND SPEECH TRANSLATOR USING MACHINE LEARNING** submitted by **Daniya Mariya Jiju(TKR20CS025), Diniya P (TKR20CS030), Divya A(TKR20CS031), Jishna Babu(TKR20CS086)**, to the APJ Abdul Kalam Technological University in partial fulfillment of the B.Tech degree in Computer Science & Engineering is a bonafide record of the project work carried out by the under my/our guidance and supervision. This report in any form has not been submitted to any other University or Institute for any purpose.

Project Guide

External Examiner

Mr. Anoop P V
Assistant Professor
Dept. of Computer Science & Engineering
College of Engineering Trikaripur

Project Coordinator

Mr.Rafeekh A P
Assistant Professor
Dept. of Computer Science & Engineering
College of Engineering Trikaripur

Head of the Department

Dr. Naveena A K
Associate Professor
Dept. of Computer Science & Engineering
College of Engineering Trikaripur

ACKNOWLEDGEMENT

We take this opportunity to express our deep sense of gratitude and sincere thanks to all who helped me to complete the work successfully. Our first and foremost thanks goes to God Almighty who showered in immense blessings on my effort.

We wish to express our sincere thanks to **Dr.Mahesh V V, Principal College of Engineering Trikaripur**, for providing us with all the necessary facilities and support. We extend our genuine and heartfelt thanks to **Dr. Naveena A K, Associate Professor and Head, Department of Computer Science and Engineering** for having permitted us to undergo this project and for her valuable suggestions, help and support during the study, analysis, design and implementation of this project.

We are thankful and profoundly indebted to our Project guide **Mr. Anoop P V Assistant Professor in Department of Computer Science and Engineering** and Project Supervisor **Mr. Rafeekh A P, Assistant Professor in Department of Computer Science and Engineering** for their exemplary guidance, criticism, monitoring and constant encouragement throughout the course and for showing us the right way.

Also, we would like to express our sincere voice of gratitude to all the faculty members of CSE and IT departments who gave the support throughout the project. We also express our deepest thanks to our family members and friends for their moral support rendered to us to make this project a success.

DANIYA MARIYA JIJU
DINIYA P
DIVYA A
JISHNA BABU

VISION
(Institution)

To be a premier institution in education and research for moulding technically competent and socially committed professionals.

MISSION
(Institution)

(i) Promote interdisciplinary research and innovation so as to meet the current needs of industry and society.

(ii) Attract, nurture and retain the best faculty and technical manpower.

(iii) Provide state of art facility for quality technical education.

(iv) Develop personality and professional skills of the students through interaction with alumni academia and industry.

VISION
(Department)

To mould technically competent and socially committed professionals in the field of computer science.

MISSION
(Department)

(i) To provide a strong foundation in theoretical and practical aspects of computer science.

(ii) To impart technical skills necessary to generate quality professional according to industry needs.

(iii) To develop human resource with the ability to apply the knowledge for the benefit of the society.

ABSTRACT

It becomes very difficult to converse with deaf and mute people. Language is a barrier in between normal people and mute people. People with this disability use different modes to communicate with others, there are number of methods available for their communication one such common method of communication is sign languages via Mobile application and Desktop application. Developing sign language application for deaf people can be very important,as they will be able to communicate easily with even those who do not understand sign language. The reason for choosing a system based on vision relates to the fact that it provides a simpler and more intuitive way of communication between a human and a computer. In this report, 42 different gestures have been considered. Sign to text and speech translator provide a methodology that aims to simplify SLR through the utilization of Media Pipes open source framework and machine learning algorithms. In order to construct such an application designed to respond in real-time and aid a live conversation between a speaking and nonspeaking individual, it is necessary to allow for live video inputs to be made to the app which would then be translated to speech. To evaluate the capability of the framework, an American Sign Language (ASL), hand pose dataset was employed for training purposes. Through extensive experimentation, the proposed model achieved an impressive average accuracy of 99%. This demonstrates its efficiency in accurately recognizing and interpreting sign language gestures. One of the key advantages of the proposed methodology is the real time and accurate detection of sign language gestures. This is made possible by leveraging the CNN algorithm, which eliminates the need for wearable sensors. Consequently, this technology provides a more comfortable and convenient user experience. The lightweight and adaptable nature of the predictive model ensures compatibility with various smart devices, facilitating widespread accessibility. The combination of AI and SLR holds tremendous potential in breaking down communication barriers.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

ASL	American Sign Language
CNN	Convolution Neural Network
API	Application Programming Interface
AI	Artificial Intelligence
SLR	Sign Language Translator

CHAPTER 1

INTRODUCTION

The deaf-mute community faces significant communication challenges in their daily lives. Sign language is a visual language that uses a combination of hand gestures, facial expressions, and body language to convey meaning. It is the primary mode of communication for the deaf and hard-of-hearing community, and it is estimated that there are over 70 million deaf people worldwide. However, sign language can be challenging for individuals who are not familiar with it to communicate effectively with the deaf and hard-of-hearing. This communication gap can lead to social isolation and exclusion for the deaf and hard-of-hearing, who may struggle to access education, employment, and other aspects of daily life.

The advancement of computer vision and machine learning has opened up new possibilities for the development of SLR and translation systems. Such systems can provide a bridge between the deaf and hard-of-hearing community and the hearing world by enabling real-time translation of sign language gestures into text or speech. These systems can also improve accessibility and inclusivity for the deaf and hard-of-hearing community by facilitating better communication with the hearing world. The goal of this project is to develop a sign language translator that can recognize and translate ASL and hand poses into spoken and written English. The translator will use a camera to capture the user's sign language gestures or also provide uploading image as input via files, and then use computer vision and machine learning algorithms to recognize the gestures and translate them into English. The translated output will be provided as text or speech, making it easier for individuals who are not familiar with ASL to communicate effectively with the deaf and hard-of-hearing.

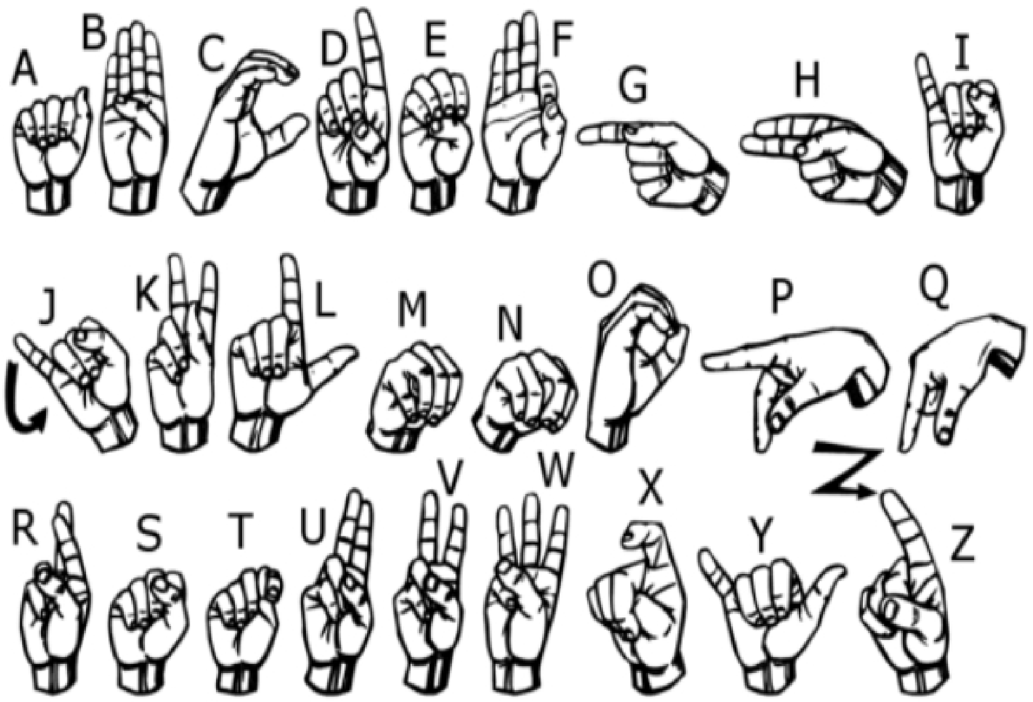


Figure 1.1: Finger Spelling American Sign Language

.

1.1 PROBLEM STATEMENT

The challenge lies in bridging these gaps to create a more inclusive and accessible environment for everyone. Develop a system that accurately translates sign language gestures into text or speech, facilitating effective communication between individuals who are deaf and mute and those who do not understand sign language. Sign language is a visual-gestural language used by the deaf and mute community for communication, but it poses a communication barrier with the general population. By creating a reliable and efficient sign language translation system, our aim is to bridge this communication gap and provide a means for inclusive and accessible communication. The primary goal is to develop a user-friendly Human-Computer Interface where the computer can understand human sign language. Sign language encompasses various languages worldwide, including ASL, French Sign Language, British Sign Language, Indian Sign Language, Japanese Sign Language, and more. Extensive work has been done on sign languages across the globe.

1.2 OBJECTIVES

The purpose of the project is to

- Develop a Reliable Sign Language Translation System:
  - To bridge the communication gap between deaf and mute individuals and those who do not understand sign language.
- Establish a User-Friendly Human-Computer Interface :
  - To make the computer understand human sign language.

1.3 SCOPE

The application has a very simple and easily navigable User Interface that suits to users.

- Accessibility for the Deaf and Hard of Hearing : Improving communication for individuals who are deaf or hard of hearing by translating sign language into text or speech.
- Educational Tools : Developing educational tools to assist in learning sign language, both for those who are hearing and for individuals with hearing impairments.
- Emergency Services : Enhancing emergency services by allowing individuals with hearing impairments to communicate effectively during critical situations.

CHAPTER 2

LITERATURE REVIEW

2.1 An Efficient Two-Stream Network for Isolated Sign Language Recognition Using Accumulative Video Motion

In this paper propose a hierarchical sign learning module that comprises three networks: dynamic motion network , accumulative motion network, and sign recognition network . Additionally, we propose a technique to extract key postures for handling the variations in the sign samples performed by different signers. The DMN stream uses these key postures to learn the spatio temporal information pertaining to the signs.Use CNN algorithm for higher accuracy, Robustness to Variability.

2.2 Recognition of isolated Indian Sign Language Gesture in Real Time

This paper demonstrates the statistical techniques for recognition of ISL gestures in real time which comprises both the hands. A video database was created by the authors and utilized which contained several videos for large number of signs. Direction histogram is the feature used for classification due to its appeal for illumination and orientation invariance. Two different approaches utilized for recognition were Euclidean distance and K nearest neighbor metrics.

2.3 Sign Language to Text-Speech Translator Using Machine Learning

Learn discriminative features from a limited dataset.The system used ASL dataset which is pre-processed based on threshold and intensity. This system recognizes sign language alphabet and by joining the letters it creates a sentence then it converts the text to speech. As the system is based on hand, hand gesture is used in sign language recognition system, for that the efficient hand tracking technique which is given by mediaPipe cross platform is used and it exactly detects the hand after that by using the ANN architecture the model has trained and which classifies the images. The system has achieved 74% accuracy and recognize almost all the letters.

2.4 Hand gesture recognition based on Karhumen-Lore transform

Proposed system hand gesture recognition using Karhunen-Lore (K-L) transform with this method they have also used CNN. For hands detection they used skin filtering, palm cropping to extract the palm area of hand and edge detection to extract the outline of palm. Then feature extraction of hand was carried out by using K-L transform method and image classification by using Euclidean distance. They tested for 10 different hand gestures with 96% of accuracy.

2.5 Sign Language Recognition System using CNN and Computer Vision

Proposed sign language recognition system using CNN and computer vision. The system used HSV color algorithm for hand gesture detection and they set the background black. Image pre-processing consists of gray scale conversion, dilation, mask operation and hand gesture was segmented. The CNN architecture was used for feature extraction in the first layer and then for image classification. This system was able to recognize 10 alphabets and it achieved 90% of accuracy.

2.6 Design and Implementation of CNN for sign language recognition

The Web Camera captures images of various gestures used as input, and this project shows the sign language recognition of 1-10 digits hand gestures, including OK and Salaam. Modules for preprocessing and feature extraction, model training, testing, and sign-to-text translation are included in the proposed system. Various CNN architectures and preprocessing techniques, such as grey scale and thresholding, were built and evaluated using our dataset to improve recognition accuracy. Our proposed system of convolutional neural networks obtains an average accuracy of 98.76 percent for real-time hand gesture identification on a dataset with nine hand movements and 500 photos for each gesture.

CHAPTER 3

METHODOLOGY

3.1 PROPOSED METHODOLOGY

The proposed sign language translator aims to provide a real-time and accurate solution for recognizing and interpreting sign language gestures. The system leverages the MediaPipe framework and machine learning techniques to enable seamless communication between sign language users and non-sign language users.

At the core of the proposed system is the hand detection module, which utilizes the mediaPipe hand detection model to identify and locate hands within the video frames captured by a webcam.Also arrange a mobile application for uploading gestures via files for translation. This step serves as the initial stage for gesture recognition and ensures that the subsequent analysis focuses on the relevant regions of interest. Once the hands are detected, the system employs machine learning algorithms, such as CNN, to recognize and classify the sign language gestures. The machine learning model is trained on a dataset of sign language gestures to achieve a high level of accuracy. The trained model then analyzes the hand gestures in real-time, extracting features and matching them to the learned patterns to determine the corresponding sign language representation.

The proposed system also includes a user-friendly interface developed using the Django framework. This web-based interface allows users to interact with the system, input sign language gestures through a webcam also generate a mobile application allow user to interact by uploading images via files, and observe the real-time interpretation of their gestures. The system's light weight nature and compatibility with smart devices make it easily accessible to a wide range of users.

3.2 FEASIBILITY ANALYSIS

Test of system according to work ability, impact on organization ability to meet user needs, and effective use of resource. Following studies are employed.

3.2.1 Technical Feasibility

1. Technology
2. Development risk
3. Resource availability

3.2.2 Economic Feasibility

1. **Cost of Development :** Estimate the costs associated with developing the CNN model, hardware, resources user interface, and integrating text-to-speech synthesis.
2. **Project Timeline :** Develop a realistic timeline for the development, testing, and deployment phases.

3.3 SOFTWARE REQUIREMENTS

1. Python (version 3.6 or higher)
2. TensorFlow (version 2.0 or higher)
3. Android studio
4. OpenCV (version 3.4 or higher)
5. MediaPipe (version 0.8 or higher)
6. SQLyog and Pycharm
7. Numpy
8. Keras
9. tkinter

3.3.1 Functional Requirements

1. Sign Language Recognition: The system should be able to recognize and interpret different sign language gestures and signs accurately.

2. Translation Output: The system should generate the corresponding translated text or spoken language output for the recognized sign language gestures.
3. Real-Time Translation: The system should provide real-time translation, enabling immediate communication between sign language users and non-sign language users.
4. Vocabulary Expansion: The system should have the capability to expand its vocabulary and recognize a wide range of signs, including common signs, specialized signs, and regional variations.
5. Training and Adaptability: The system should be trainable and adaptable to accommodate user-specific signing styles and preferences, allowing it to improve over time.
6. User Interface: The system should have a user-friendly interface that facilitates easy interaction between the user and the translation functionality

3.3.2 Non Functional Requirements

1. Accuracy and Precision: The system should strive for high accuracy and precision in recognizing and translating sign language gestures to ensure effective communication.
2. Speed and Responsiveness: The system should perform recognition and translation tasks efficiently and respond promptly to ensure real-time translation capabilities.
3. Scalability: The system should be scalable to handle a growing dataset and accommodate an increasing number of users accessing the translation service simultaneously.
4. Accessibility: The system should be designed to be accessible to individuals with different abilities, including those with visual or hearing impairments.
5. Privacy and Security: The system should ensure the privacy and security of user data, protecting sensitive information during the translation process.
6. Compatibility: The system should be compatible with various devices and platforms, allowing users to access the translation service from different devices such as smartphones, tablets, or computers.
7. Reliability and Robustness: The system should be reliable and robust, able to handle variations in lighting conditions, hand movements, and environmental factors that may affect gesture recognition.
8. Ethical Considerations: The system should prioritize ethical considerations, ensuring that the project respects cultural sensitivity, user consent, and appropriate use of the technology.

3.4 HARDWARE REQUIREMENTS

1. Computer or laptop with 8GB RAM or higher
2. 320GB and above HDD
3. Processor : Pentium Dual or intel Core 3 and above
4. Android phones with 4GB RAM or higher
5. Webcam with High resolution(optional)

CHAPTER 4

SYSTEM DESIGN

4.1 BLOCK DIAGRAM

A block diagram or an architecture diagram is a drawing illustration of a system whose major parts or components working flow are represented by blocks. These blocks are joined by lines to display the relationship between subsequent blocks.

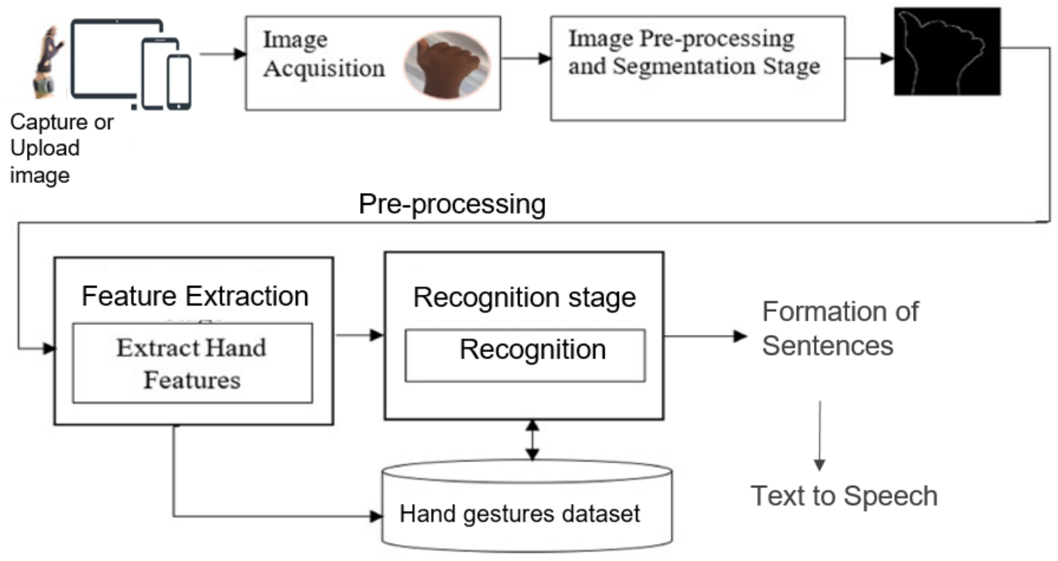


Figure 4.1: Block diagram

.

4.2 DATA FLOW DIAGRAM

A data flow diagram (DFD) maps out the flow of information for any process or system. It uses defined symbols like rectangles, circles and arrows, plus short text labels, to show data inputs, outputs, storage points and the routes between each destination. Data flowcharts can range from simple, even hand-drawn process overviews, to in-depth, multi-level DFDs that dig progressively deeper into how the data is handled.

4.2.1 Data Flow Diagram Level 0

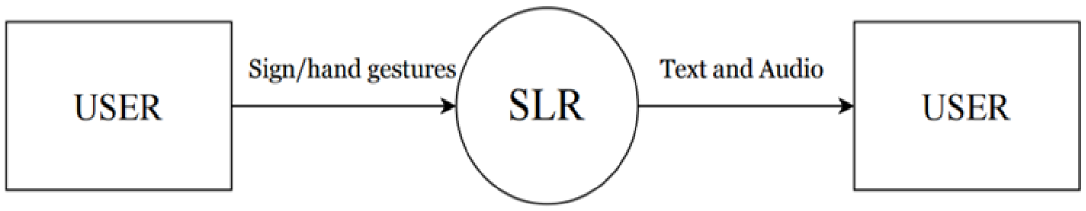


Figure 4.2: DFD Level 0

.

4.2.2 Data Flow Diagram Level 1

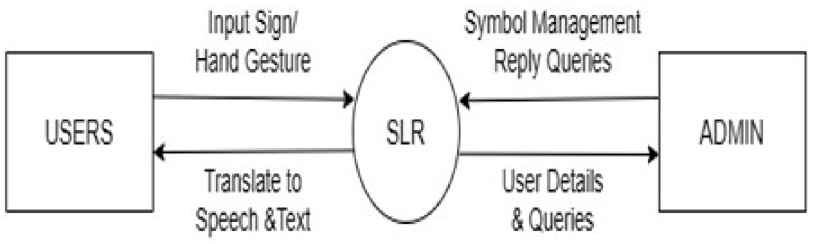


Figure 4.3: DFD Level 1

.

4.2.3 Data Flow Diagram Level 1.1 for Admin

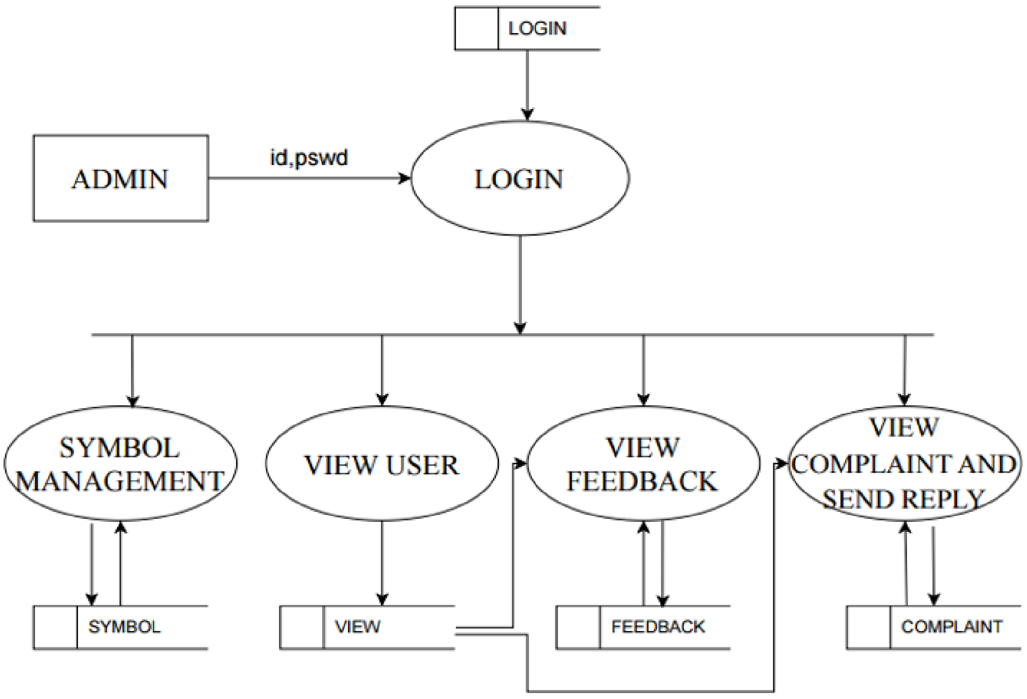


Figure 4.4: DFD Level 1.2 for Admin

.

4.2.4 Data Flow Diagram Level 1.2 for User

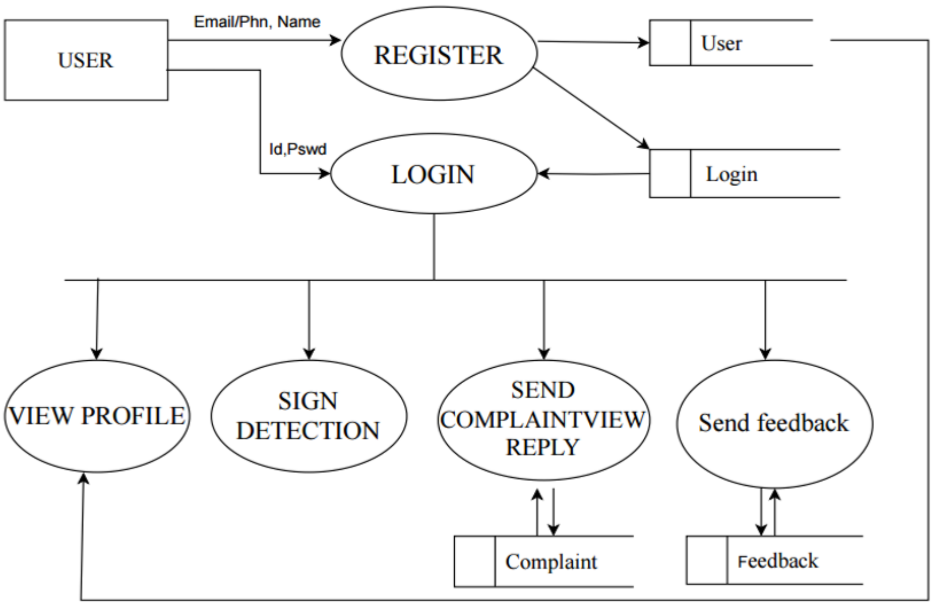


Figure 4.5: DFD Level 1.2 for User

.

4.3 USE CASE DIAGRAM

It is a graphical depiction of a users possible interactions with a system. A use cases diagram shows various use cases and different types of users the system has and will often be accompanied by other types of diagrams as well. The use cases are represented by either circle or ellipses. The actors are often shown as stick figures, a use case diagram can help provide a higher-level view of the system. It has been said before that use case diagrams are blueprints of our system.

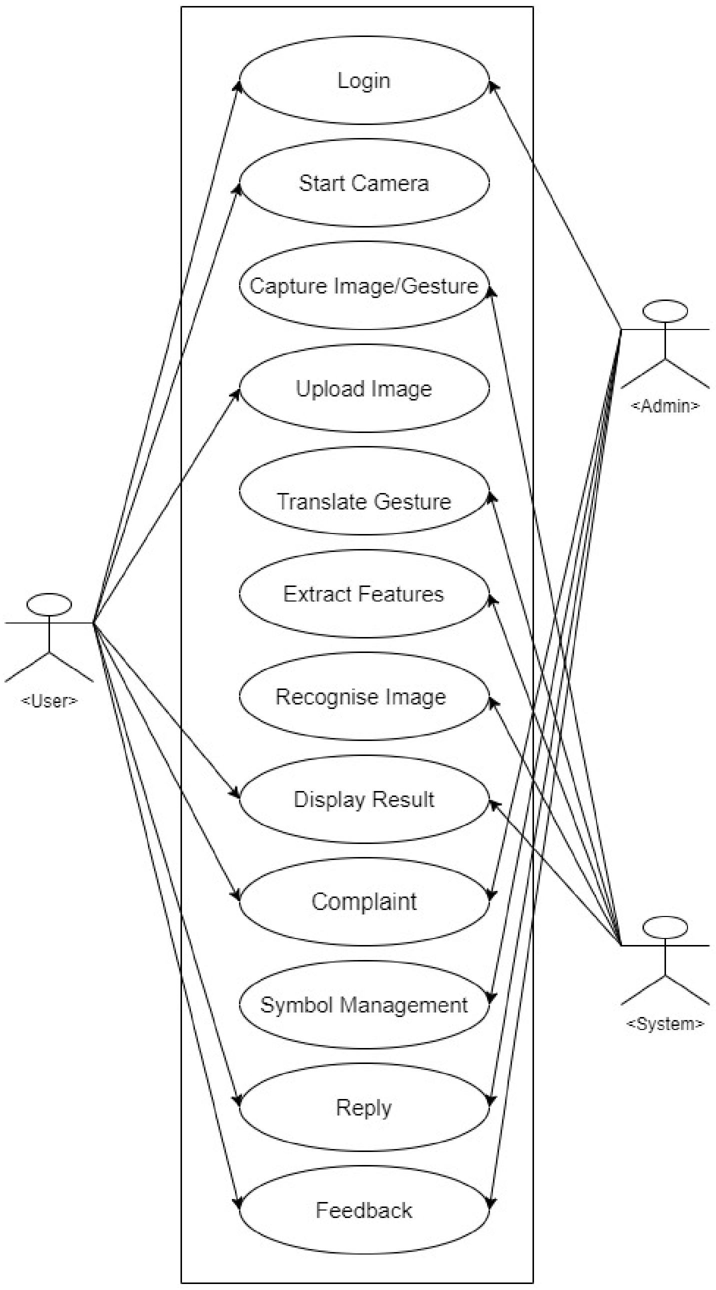


Figure 4.6: Usecase Diagram

.

4.4 ACTIVITY DIAGRAM

An activity diagram visually presents a series of actions or flow of control in a system similar to a flowchart or a data flow diagram. Activity diagrams are often used in business process modeling. They can also describe the steps in a use case diagram. Activities modeled can be sequential and concurrent. In both cases an activity diagram will have a beginning (an initial state) and an end (a final state).

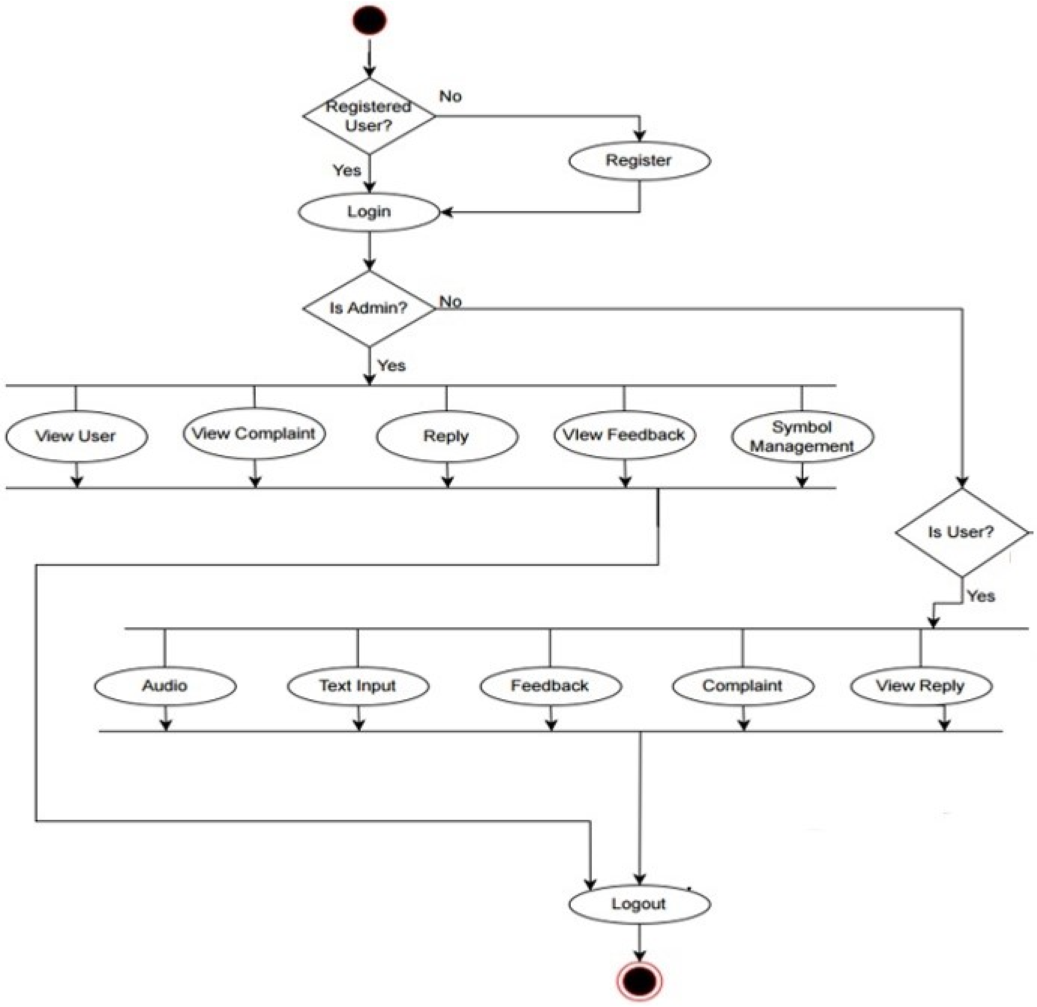


Figure 4.7: Activity Diagram

.

4.5 CONTROL FLOW DIAGRAM

This control flow diagram illustrates the step-by-step process for both training the model using a training dataset and performing real-time image/video processing to recognize and translate sign language gestures. It outlines the key stages from data collection and pre-processing to feature extraction, classification, and text translation. The diagram provides a comprehensive overview of how the system functions to achieve accurate sign language translation.

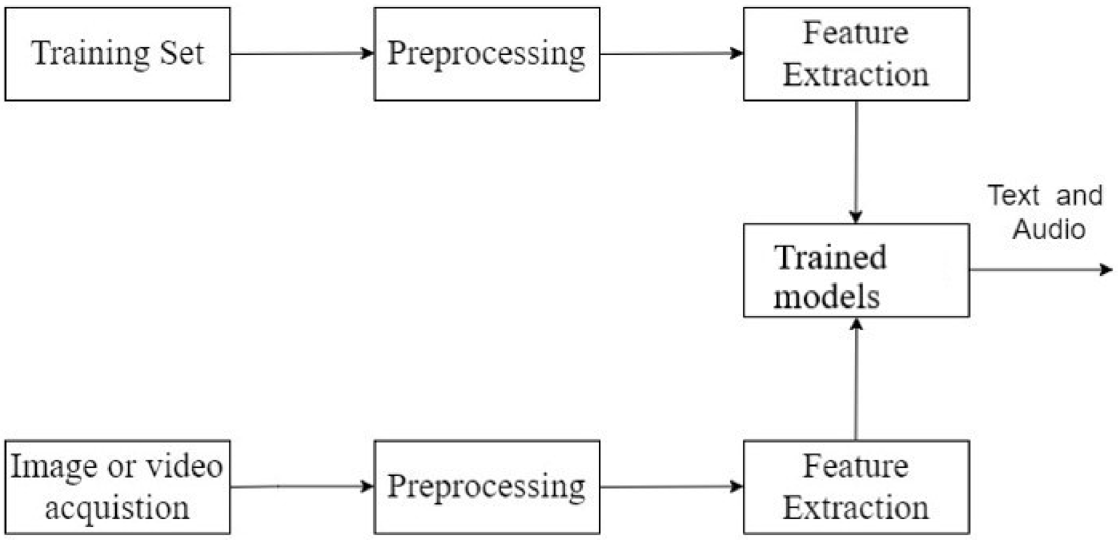


Figure 4.8: Control flow Diagram

.

4.6 MODULE DESCRIPTION

- **Module 1 : User**

  User can input symbols via video accumulation or files,then recognize as speech and text. Register complaints, send feedback, view reply, signs. Users can input words and see their corresponding sign as video.

- **Module 2 : Admin**

  Admin can manage whole system. Add, view and delete the symbols. Can view the complaints and feedback that submitted by users. Based on complaint admin can reply to users.

- **Module 3 : System**

  User Interface : Responsible for providing a user-friendly interface for users to interact with the sign language translation system. It includes features such as displaying instructions, capturing user input, and providing feedback to the user. The module handles the presentation layer of the system and ensures seamless communication between the user and the underlying modules.

Feature Extraction Description : The Feature Extraction module is responsible for extracting relevant features from the recognized sign language gestures. It analyzes the input data and identifies distinctive characteristics that are essential for translation. This module helps in reducing the complexity of the data and extracting meaningful information that can be used for further processing.

Text Translation Description : The Text Translation module is responsible for translating the recognized sign language gestures into textual form. It employs various natural language processing techniques to convert the extracted features into human-readable text. This module plays a vital role in bridging the gap between sign language and written/spoken language, enabling effective communication between users.

Gesture Recognition : The Gesture Recognition module is responsible for processing the captured sign language gestures and recognizing the intended signs. It utilizes computer vision techniques and machine learning algorithms to analyze the input data and identify the corresponding gestures.This module plays a crucial role in accurately interpreting the user's gestures and initiating the translation process.

CHAPTER 5

IMPLEMENTATION

5.1 CONVOLUTIONAL NEURAL NETWORK

As the technology is improving day by day, the improvement and advancement in the field of Computer Vision and Deep Learning enabled machines to see this world as humans do.That how CNN has been constructed. CNN has been widely applied on various fields such as medical image classification, natural language processing, recommendation system, video recognition etc. The whole idea of image classification is like this, taking an image as input and gives the output as a class. For example if the input images is of rat and cat then the result is the classes of rat and cat. A convolution tool separates and recognize images several features, this is called as feature extraction. A fully connected layer that uses the convolution process output to predict the images class based on the features that were separated from in prior stages. Convolutional neural network is made up of layers and they are categorized into three layers that are Convolution, Pooling and Fully Connected. While training a image it has to pass through these sequence of layers. We finally get output after passing through these layers. Other than these two layer there are two more important layer, one is activation and other is dropout layer.

Overall implementation of the model with detailed process flow, experiment setup,, final output are illustrated. In section 3 Data Acquisition,Data Pre-processing is discussed and , In this research Convolution Neural Network with extra layers and various preprocessing techniques like Image Augmentation, image size reduction, colour conversion are preformed. The proposed system is for Sign Language Recognition and Translation. That is it will recognize the hand gesture or the ASL Alphabet that is been shown and display the letter in text and also converts into audio. The input image is ASL Alphabet and hand poses, the CNN model is trained with ASL Alphabets and hand poses with special characters blank.

5.1.1 CNN Layers

CNN is composed of three layers: a convolutional layer, a pooling layer, and a fully connected layer. The first layer is the convolutional layer, while the final layer is the fully connected layer. The complexity of the CNN grows from the convolutional layer to the fully connected layer. The CNN is able to identify increasingly larger and more intricate aspects of an image until it successfully recognizes the complete thing as a result of the rising complexity.

- Convolution Layer : The convolutional layer, the central component of a CNN,is where most computations take place. The first convolutional layer may be folï¿¾lowed by a subsequent convolutional layer. A kernel or filter inside this layer moves over the images receptive fields during the convolution process to determine whether a feature is present. The kernel traverses the entire image over a number of iterations. A dot product between the input pixels and the filter is calculated at the end of each iteration. A feature map feature is the result of the dots being connected in a certain pattern. In this layer, the image is ultimately transformed into numerical values that the CNN can understand and extract pertinent patterns from.
- Pooling Layer : The pooling layer similarly to the convolutional layer sweeps a kernel or filter across the input image. Contrary to the convolutional layer, the pooling layer has fewer input parameters but also causes some information to be lost. Positively, this layer simplifies the CNN and increases its effectiveness.
- Fully Connected Layer : Based on the features extracted in the preceding layers, picture categorization in the CNN takes place in the FClayer. Fully connected in this context means that every activation unit or node of the subsequent layer is connected to every input or node from the preceding layer. The CNN does not have all of its layers fully connected because that would create an excessively dense network. It would cost a lot to compute, increase losses, and have an impact on output quality.

5.2 PROCESS FLOW

5.2.1 Data Acquisition and Pre-Processing

The dataset is taken from kaggale. It has 42 classes, 26 alphabets, 15 hand poses and space character. The images are pre-processed by Image Augmentation that has, shifting, rotating, flipping. Then image size reduction to 64 x 64 pixel.The color conversion of image from BGR(Blue, Green, Red) to RGB(Red, Green, Blue)

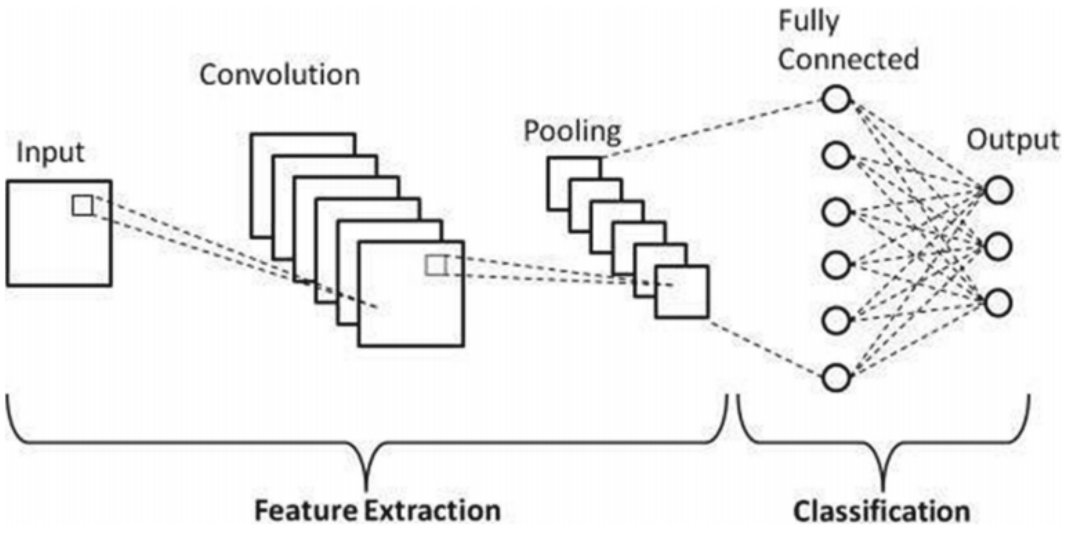


Figure 5.1: Basic CNN architecture

.

5.2.2 Convolution Neural Network model

The pre-processed images are fed into the CNN model. Pre-processed images are divided into train and test. The model has 2 Convolution layers with max pooling on both of the layer, batch normalisation and dropout layer to control overfitting, then 2 fully connected layer which contain dense. In the context of sign language recognition , Convolutional Neural Networks play a crucial role in automatically recognizing hand gestures and translating them into corresponding textual or spoken representations. Here's an overview of how CNNs are applied in SLR: In SLR, these convolutional layers help in detecting patterns and features within the hand gestures, such as the position of fingers, hand shape, and movement trajectories.

5.2.3 Pooling Layers

- Pooling layers are used to down sample the feature maps obtained from the convolutional layers. They help reduce the spatial dimensions of the feature maps while retaining the most important information. Common pooling operations include max pooling and average pooling, which extract the maximum or average value from each local region of the feature maps, respectively.

  Pooling layers in SLR CNNs help in capturing the invariant features of hand gestures, such as their spatial arrangement and orientation, regardless of slight variations in positioning or scaling.

- Fully Connected Layers: After several convolutional and pooling layers, the extracted features are typically flattened and passed to one or more fully connected layers. These fully connected layers act as classifiers, learning complex patterns in the extracted features and mapping them to output classes. In SLR, fully connected layers are responsible for translating the learned features of hand gestures into textual or spoken representations, such as letters, words, or sentences in sign language.

- Output Layer: The output layer of the CNN produces the final prediction or classification result. Depending on the SLR task, it may output probabilities for each class or directly generate the textual or spoken representation of the input gesture. During training, the output layer is optimized using appropriate loss functions to minimize the difference between predicted and ground truth outputs.

5.2.4 Training and Optimization

Train with labelled data. We convert our input images (RGB) into gray scale and apply gaussian blur to remove unnecessary noise. We apply adaptive threshold to extract our hand from the background and resize our images to 128 x 128. We feed the input images after pre-processing to our model for training and testing after applying all the operations mentioned above. The prediction layer estimates how likely the image will fall under one of the classes. So, the output is normalized between 0 and 1 and such that the sum of each value in each class sums to 1. We have achieved this using SoftMax function. The network learns to recognize patterns and features in the input images through forward and backward propagation, adjusting its parameters (weights and biases) to minimize the prediction error.

Adam optimizer techniques used to update the network parameters during training.

5.2.5 Predicting and categorizing the frame

After training the model, it is saved. In OpenCV, the trained model is called and performs the prediction. A full frame is designed to show the hand gesture and the translated letter comes below and in order to convert the text into audio(Google Text to Speech) is used and a small rectangle on top right corner is present to convert into audio.

5.3 TENSORFLOW

An end-to-end open-source platform for Machine Learning. It has a comprehensive, flexible ecosystem of tools, libraries and community resources that lets researchers push the state-of-the-art in Machine Learning and developers easily build and deploy Machine Learning powered applications.

TensorFlow offers multiple levels of abstraction so you can choose the right one for your needs. Build and train models by using the high-level Keras API, which makes getting started with TensorFlow and machine learning easy.

If you need more flexibility, eager execution allows for immediate iteration and intuitive debugging. For large ML training tasks, use the Distribution Strategy API for distributed training on different hardware configurations without changing the model definition.

5.4 KERAS

Keras is a high-level neural networks library written in python that works as a wrapper to TensorFlow. It is used in cases where we want to quickly build and test the neural network with minimal lines of code. It contains implementations of commonly used neural network elements like layers, objective, activation functions, optimizers, and tools to make working with images and text data easier.

5.5 HUNSPELL

A python library Hunspell suggest is used to suggest correct alternatives for each (incorrect) input word and we display a set of words matching the current word in which the user can select a word to append it to the current sentence. This helps in reducing mistakes committed in spellings and assists in predicting complex words.

5.6 OPENCV

OpenCV (Open-Source Computer Vision) is an open-source library of programming functions used for real-time computer-vision. It is mainly used for image processing, video capture and analysis for features like face and object recognition. It is written in C++ which is its primary interface, however bindings are available for Python, Java, MATLAB/OCTAVE.

At first the output of the prediction layer will be somewhat far from the actual value. To make it better we have trained the networks using labelled data. The cross-entropy is a performance measurement used in the classification. It is a continuous function which is positive at values which is not same as labelled value and is zero exactly when it is equal to the labelled value. Therefore, we optimized the cross-entropy by minimizing it as close to zero. To do this in our network layer we adjust the weights of our neural networks. TensorFlow has an inbuilt function to calculate the cross entropy.

As we have found out the cross-entropy function, we have optimized it using Gradient Descent in fact with the best gradient descent optimizer is called Adam Optimizer.

5.7 SAMPLE CODE

```
import numpy as np
import cv2
import os, sys
import time
import operator
import pyttsx3
from string import ascii uppercase
import tkinter as tk
from PIL import Image, ImageTk
from hunspell import Hunspell
from tensorflow.keras.models import model from json
os.environ[”THEANOFLAGS”] = ”device=cuda, assert nocpuop=True”
class Application:
def __init__(self):
self.hs = Hunspell(’en_US’)
self.vs = cv2.VideoCapture(0)
self.current_image = None
self.current_image2 = None self.json_file = open(r”C:path”, ”r”)
self.model_json = self.json_file.read()
self.json_file.close()
self.sent = ””
self.loaded_model = model_from_json(self.model_json)
self.loaded_model.load_weights(r”C:pathnew.h5”)
self.json_file_dru = open(r”C:pathbw dru.json”, ”r”)
self.model_json_dru = self.json_file_dru.read()
self.json_file_dru.close()
self.loaded_model_dru = model_from_json(self.model_json_dru)
self.loaded_model_dru.load_weights(r”C:path modelbw_dru.h5”)
self.json_file_tkdi = open(r”C:path bw_tkdi.json”, ”r”)
self.model_json_tkdi = self.json_file_tkdi.read()
self.json_file_tkdi.close()
self.loaded_model_tkdi = model_from_json(self.model_json_tkdi)
self.loaded_model_tkdi.load_weights(r"C:pathbw_tkdi.h5")
self.json_file_smn = open(r"C:pathbw_smn.json", "r")
self.model_json_smn = self.json_file_smn.read()
self.json_file_smn.close()
self.loaded_model_smn = model_from_json(self.model_json_smn)
self.loaded_model_smn.load_weights(r"path bw_smn.h5")
self.ct =
self.ct['blank'] = 0
self.blank_flag = 0
for i in ascii_uppercase:
self.ct[i] = 0
print("Loaded model from disk")
self.root = tk.Tk()
self.root.title("Sign Language To Text Conversion")
self.root.protocol('WM_DELETE_WINDOW', self.destructor)
self.root.geometry("900x600+20+20")
self.panel = tk.Label(self.root)
self.panel.place(x=60, y=10, width=580, height=580)
self.panel2 = tk.Label(self.root)
self.panel2.place(x=360, y=65, width=275, height=275)
self.T = tk.Label(self.root)
self.T.place(x=60, y=5)
self.T.config(text="Sign Language To Text Conversion", font=("Courier", 30, "bold"))
self.panel3 = tk.Label(self.root)
self.panel3.place(x=650, y=190)
self.T1 = tk.Label(self.root)
self.T1.place(x=650, y=150)
self.T1.config(text="Character :", font=("Courier", 15, "bold"))
self.panel4 = tk.Label(self.root)
self.panel4.place(x=150, y=540)
self.T2 = tk.Label(self.root)
self.T2.place(x=10, y=540)
self.T2.config(text=”Word :”, font=(”Courier”, 15, ”bold”))
self.panel5 = tk.Label(self.root)
self.panel5.place(x=150, y=565)
self.T3 = tk.Label(self.root)
self.T3.place(x=10, y=565)
self.T3.config(text=”Sentence :”, font=(”Courier”, 15, ”bold”))
self.T4 = tk.Label(self.root)
self.T4.place(x=250, y=690)
self.T4.config(text=”Suggestions :”, fg=”red”, font=(”Courier”, 15, ”bold”))
self.bt1 = tk.Button(self.root, command=self.action1, height=0, width=8)
self.bt1.place(x=700, y=500)
self.bt1.config(text=”Speak”, font=(”Courier”, 10, ”bold”))
self.bt2 = tk.Button(self.root, command=self.action2, height=0, width=8)
self.bt2.place(x=700, y=450)
self.bt2.config(text=”Clear”, font=(”Courier”, 10, ”bold”))
self.str = ””
self.word = ” ”
self.current_symbol = ”Empty”
self.photo = ”Empty”
self.video_loop()
def video_loop(self):
ok, frame = self.vs.read()
if ok:
cv2image = cv2.flip(frame, 1)
x1 = int(0.5 * frame.shape[1])
y1 = 10
x2 = frame.shape[1] - 10
```

```
y2 = int(0.5 * frame.shape[1])
cv2.rectangle(frame, (x1 - 1, y1 - 1), (x2 + 1, y2 + 1), (255, 0, 0), 1)
cv2image = cv2.cvtColor(cv2image, cv2.COLOR_BGR2RGBA)
self.current_image = Image.fromarray(cv2image)
imgtk = ImageTk.PhotoImage(image=self.current_image)
self.panel.imgtk = imgtk
self.panel.config(image=imgtk)
cv2image = cv2image[y1: y2, x1: x2]
gray = cv2.cvtColor(cv2image, cv2.COLOR_BGR2GRAY)
blur = cv2.GaussianBlur(gray, (5, 5), 2)
th3 = cv2.adaptiveThreshold(blur, 255, cv2.ADAPTIVE_THRESH_GAUSSIAN_C, cv2.THRESH_BINA
11, 2) ret, res = cv2.threshold(th3, 70, 255, cv2.THRESH_BINARY_INV + cv2.THRESH_OTSU)
self.predict(res)
self.current_image2 = Image.fromarray(res)
imgtk = ImageTk. PhotoImage(image=self.current_image2)
self.panel2.imgtk = imgtk
self.panel2.config(image=imgtk)
self.panel3.config(text=self.current_symbol, font=(”Courier”, 15))
self.panel4.config(text=self.word, font=(”Courier”, 15))
self.panel5.config(text=self.sent, font=(”Courier”, 15))
self.root.after(5, self.video_loop)
def predict(self, test_image):
test_image = cv2.resize(test_image, (128, 128))
result = self.loaded_model.predict(test_image.reshape(1, 128, 128, 1))
result_dru = self.loaded_model_dru.predict(test_image.reshape(1, 128, 128, 1))
result_tkdi =self.loaded_model_tkdi.predict(test_image.reshape(1, 128, 128, 1))
result_smn = self.loaded_model_smn.predict(test_image.reshape(1, 128, 128, 1))
prediction =
prediction[’blank’] = result[0][0]
inde = 1
for i in ascii_uppercase:
```

```
prediction[i] = result[0][inde]
inde += 1
prediction = sorted(prediction.items(), key=operator.itemgetter(1), reverse=True)
self.current_symbol = prediction[0][0]
if (self.current_symbol == ’D’ or self.current_symbol == ’R’ or self.current_symbol == ’U’):
prediction =
prediction[’D’] = result_dru[0][0]
prediction[’R’] = result_dru[0][1]
prediction[’U’] = result_dru[0][2]
prediction = sorted(prediction.items(), key=operator.itemgetter(1), reverse=True)
self.current_symbol = prediction[0][0]
if ( self.current_symbol == ’D’ or self.current_symbol == ’I’ or self.current_symbol == ’K’
or self.current_symbol == ’T’):
prediction =
prediction[’D’] = result_tkdi[0][0]
prediction[’I’] = result_tkdi[0][1]
prediction[’K’] = result_tkdi[0][2]
prediction[’T’] = result_tkdi[0][3]
prediction = sorted(prediction.items(), key=operator.itemgetter(1), reverse=True)
self.current_symbol = prediction[0][0]
if (self.current_symbol == ’M’ or self.current_symbol == ’N’ or self.current_symbol == ’S’):
prediction1 =
prediction1[’M’] = result_smn[0][0]
prediction1[’N’] = result_smn[0][1]
prediction1[’S’] = result_smn[0][2]
prediction1 = sorted(prediction1.items(), key=operator.itemgetter(1), reverse=True)
if (prediction1[0][0] == ’S’):
self.current_symbol = prediction1[0][0]
else:
self.current_symbol = prediction[0][0]
if (self.current_symbol == ’blank’):
```

```
if self.word != ””:
self.sent += ” ” + self.word
self.word = ””
for i in ascii_uppercase:
self.ct[i] = 0
self.ct[self.current_symbol] += 1
if (self.ct[self.current_symbol] ¿ 60):
for i in ascii_uppercase:
if i == self.current_symbol:7
continue
tmp = self.ct[self.current_symbol]  self.ct[i]
if tmp ¡ 0:
tmp *= -1
if tmp ¡= 20:
self.ct[’blank’] = 0
for i in ascii_uppercase:
self.ct[i] = 0
return
self.ct[’blank’] = 0
for i in ascii_uppercase:
self.ct[i] = 0
if self.current_symbol == ’blank’:
if self.blank_flag == 0:
self.blank_flag = 1
if len(self.str) ¿ 0:
self.str += ” ”
self.str += self.word
self.word = ””
else:
if (len(self.str) greather than 16):
self.str = ””
self.blank_flag = 0
self.word += self.current_symbol
def action1(self):
engine = pyttsx3.init()
engine.say(self.sent)
engine.runAndWait() def action2(self):
self.word = ””
def destructor(self):
print(”Closing Application...”)
self.root.destroy()
self.vs.release()
cv2.destroyAllWindows()
```

CHAPTER 6

TESTING

Testing is a process of creating a program with the explicit intention of finding error that is making the program fail. We had used many types of testing to check the functionality of our software.

6.1 Functional Testing

In Functional Testing, we evaluated the work of various functional parts of SLR.

6.2 Performance Testing

Details like network latency and how long SLR takes to return the output is tested.

6.3 Integration Testing

In Integration testing, we ensured that all the components of the application are integrated properly and performing the actions appropriately.

6.4 Module Testing

Each module is created as an impenetrable mechanism for performing a function. And modules are tested separately to checking working conditions. Each module is taken up individually and tested for correctness in coding and logic. Here, we had taken each module separately and checked its working.

6.5 Output Testing

The output generated or displayed by the system under consideration is tested. Through output testing, we verified whether we get the accurate output or not.

6.6 Interface Testing

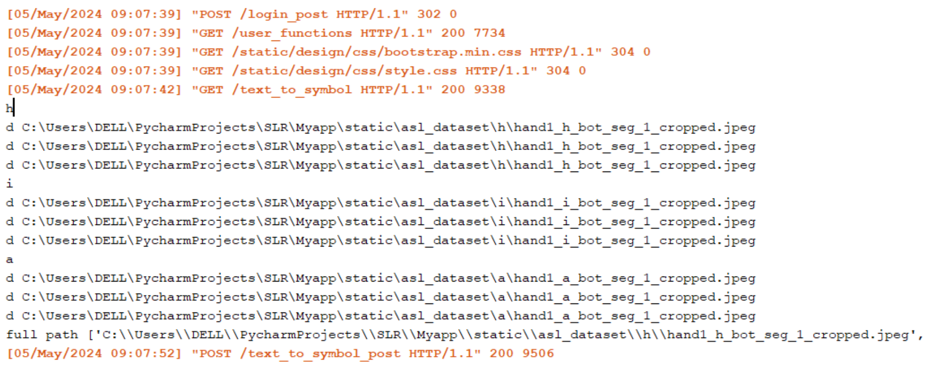
An interactive interface is the system that interacts between the system and external agents, such as human. And check whether the external agents are friendly with the interface.

6.7 Test cases

Case id	Description	Expected Result	Actual result	Status
1	Upload trained image as input	Proper Speech and text translation displayed	Proper Speech and text translation displayed	PASS
2	Upload a non-trained image as input	Output will displayed as per matching trained dataset	Output will displayed as per matching trained dataset or Bad request	PASS
3	Capturing images without background substration	Accuracy of recognition varies	Accuracy of recognition varies	PASS
4	Capturing images with background substration	Display correct output	Display correct output	PASS
5	Sending a empty complaint and feedback by user	Error message will be displayed with suggestions	Error message will be displayed with suggestions	PASS
6	Enter valid key for getting their sign	Result will displayed as video	Result will displayed as video	PASS
7	Login and sign_up with invalid credentials	Error message will displayed with suggestions	Error message will displayed with suggestions	PASS
8	Enter invalid key for getting their sign	Attribute Error	Attribute Error	PASS

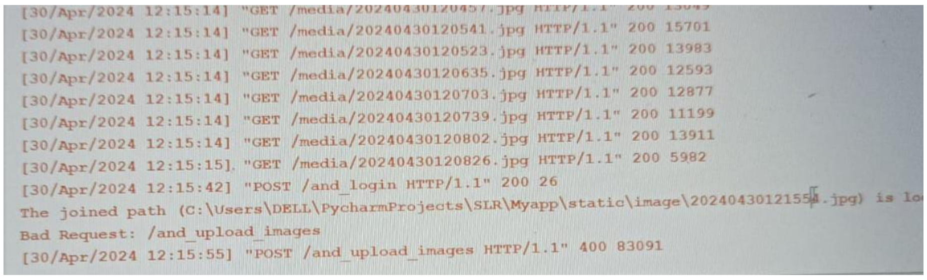
Table 6.1: Test cases and test results

6.8 Snapshots of Testcases



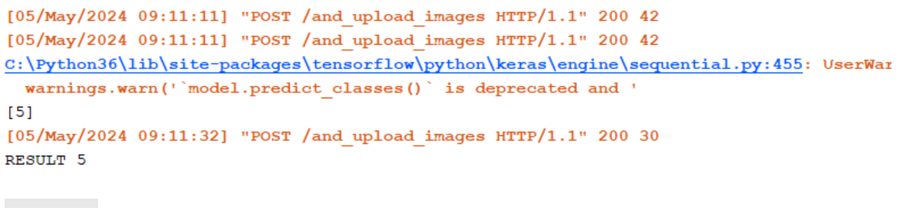
```
[05/May/2024 09:07:39] "POST /login_post HTTP/1.1" 302 0
[05/May/2024 09:07:39] "GET /user_functions HTTP/1.1" 200 7734
[05/May/2024 09:07:39] "GET /static/design/css/bootstrap.min.css HTTP/1.1" 304 0
[05/May/2024 09:07:39] "GET /static/design/css/style.css HTTP/1.1" 304 0
[05/May/2024 09:07:42] "GET /text_to_symbol HTTP/1.1" 200 9338
h
d C:\Users\DELL\PycharmProjects\SLR\Myapp\static\asl_dataset\h\hand1_h_bot_seg_1_cropped.jpeg
d C:\Users\DELL\PycharmProjects\SLR\Myapp\static\asl_dataset\h\hand1_h_bot_seg_1_cropped.jpeg
d C:\Users\DELL\PycharmProjects\SLR\Myapp\static\asl_dataset\h\hand1_h_bot_seg_1_cropped.jpeg
i
d C:\Users\DELL\PycharmProjects\SLR\Myapp\static\asl_dataset\i\hand1_i_bot_seg_1_cropped.jpeg
d C:\Users\DELL\PycharmProjects\SLR\Myapp\static\asl_dataset\i\hand1_i_bot_seg_1_cropped.jpeg
d C:\Users\DELL\PycharmProjects\SLR\Myapp\static\asl_dataset\i\hand1_i_bot_seg_1_cropped.jpeg
a
d C:\Users\DELL\PycharmProjects\SLR\Myapp\static\asl_dataset\a\hand1_a_bot_seg_1_cropped.jpeg
d C:\Users\DELL\PycharmProjects\SLR\Myapp\static\asl_dataset\a\hand1_a_bot_seg_1_cropped.jpeg
d C:\Users\DELL\PycharmProjects\SLR\Myapp\static\asl_dataset\a\hand1_a_bot_seg_1_cropped.jpeg
full path ['C:\\Users\\DELL\\PycharmProjects\\SLR\\Myapp\\static\\asl_dataset\\h\\hand1_h_bot_seg_1_cropped.jpeg',
[05/May/2024 09:07:52] "POST /text_to_symbol_post HTTP/1.1" 200 9506
```

(a) Valid inputs for signs



```
[30/Apr/2024 12:15:14] "GET /media/20240430120457.jpg HTTP/1.1" 200 15049
[30/Apr/2024 12:15:14] "GET /media/20240430120541.jpg HTTP/1.1" 200 15701
[30/Apr/2024 12:15:14] "GET /media/20240430120523.jpg HTTP/1.1" 200 13983
[30/Apr/2024 12:15:14] "GET /media/20240430120635.jpg HTTP/1.1" 200 12593
[30/Apr/2024 12:15:14] "GET /media/20240430120703.jpg HTTP/1.1" 200 12877
[30/Apr/2024 12:15:14] "GET /media/20240430120739.jpg HTTP/1.1" 200 11199
[30/Apr/2024 12:15:14] "GET /media/20240430120802.jpg HTTP/1.1" 200 13911
[30/Apr/2024 12:15:15] "GET /media/20240430120826.jpg HTTP/1.1" 200 5982
[30/Apr/2024 12:15:42] "POST /and_login HTTP/1.1" 200 26
The joined path (C:\Users\DELL\PycharmProjects\SLR\Myapp\static\image\20240430121554.jpg) is lo
Bad Request: /and_upload_images
[30/Apr/2024 12:15:55] "POST /and_upload_images HTTP/1.1" 400 83091
```

(b) invalid input



```
[05/May/2024 09:11:11] "POST /and_upload_images HTTP/1.1" 200 42
[05/May/2024 09:11:11] "POST /and_upload_images HTTP/1.1" 200 42
C:\Python36\lib\site-packages\tensorflow\python\keras\engine\sequential.py:455: UserWar
  warnings.warn('`model.predict_classes()` is deprecated and '
[5]
[05/May/2024 09:11:32] "POST /and_upload_images HTTP/1.1" 200 30
RESULT 5
```

(c) valid input

Figure 6.1: Snapshots of testcases

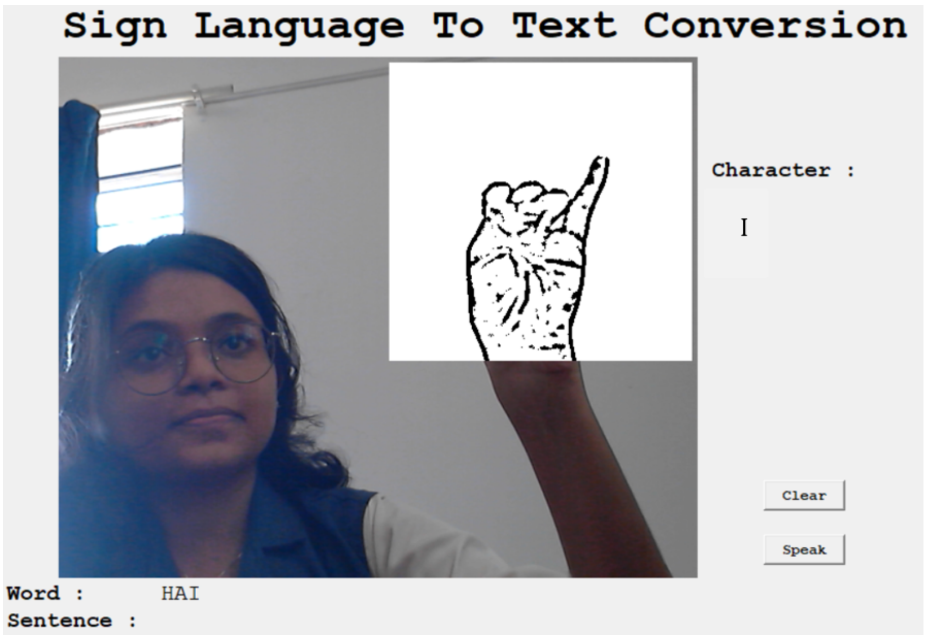
CHAPTER 7

RESULT AND FUTURE SCOPE

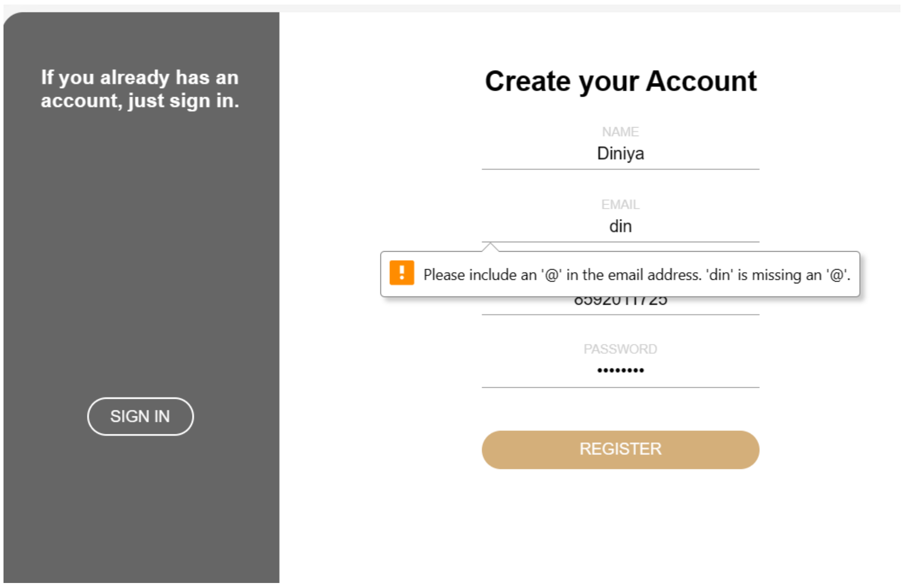
We achieve an accuracy of 99.0%, which is a better accuracy then most of the current research papers on American sign language.

Also used CNN for their recognition system. One thing should be noted that our model does not uses any background subtraction algorithm whiles some of the models present above do that. So, once we try to implement background subtraction in our project the accuracies may vary. On the other hand, most of the projects use Kinect devices but our main aim was to create a project which can be used with readily available resources.

7.1 EXPERIMENTAL OUTPUTS



(a) Sign Translator for web application



NAME
Diniya

EMAIL
din

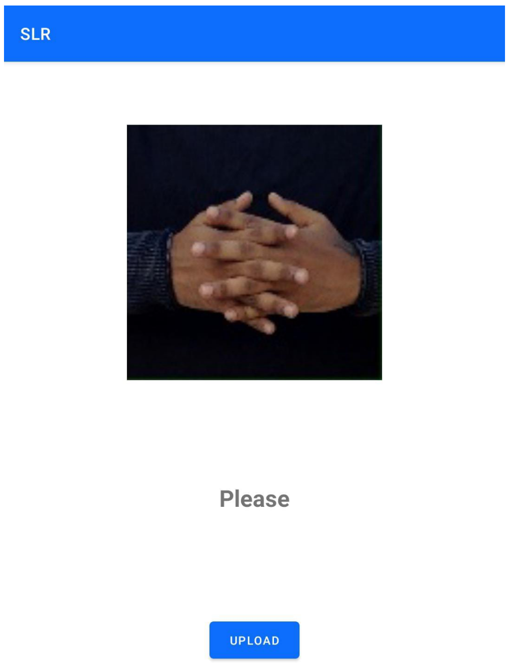
Please include an '@' in the email address. 'din' is missing an '@'.

8592011725

PASSWORD
••••••••

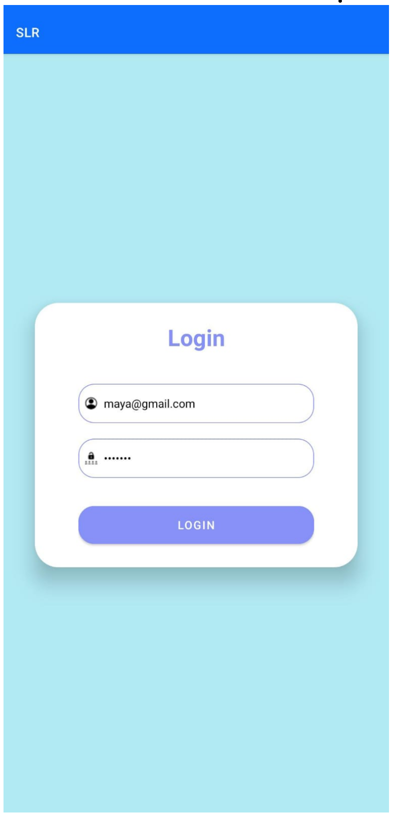
(b) Web application validation

Figure 7.1: Web Application snapshots



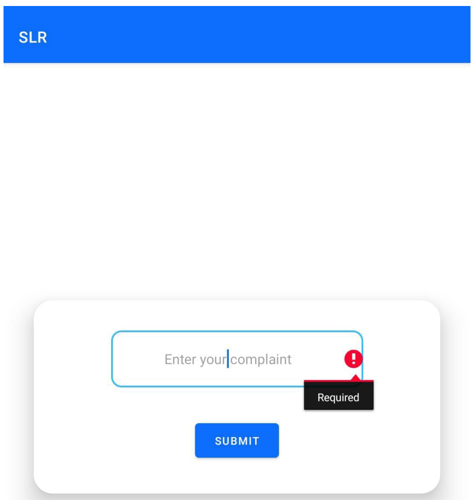
(a) Sign Translator for mobile app

.



(b) login interface

.

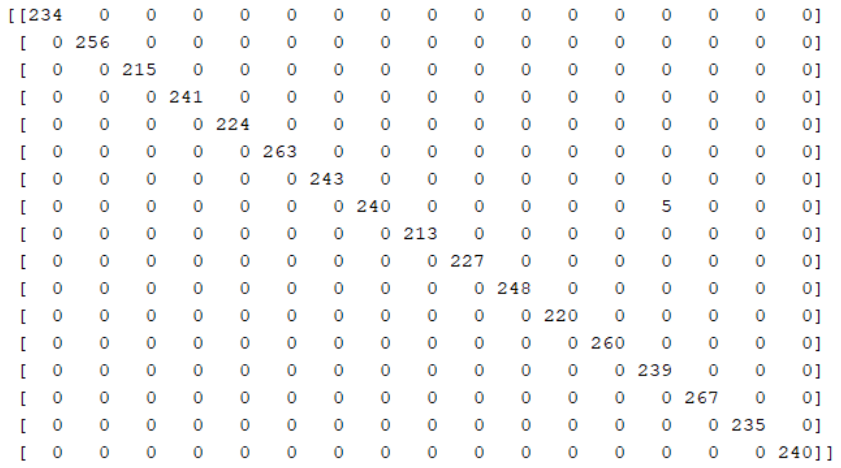


(c) Complaint panel

Figure 7.2: Mobile app snapshots

7.1.1 Confusion matrix

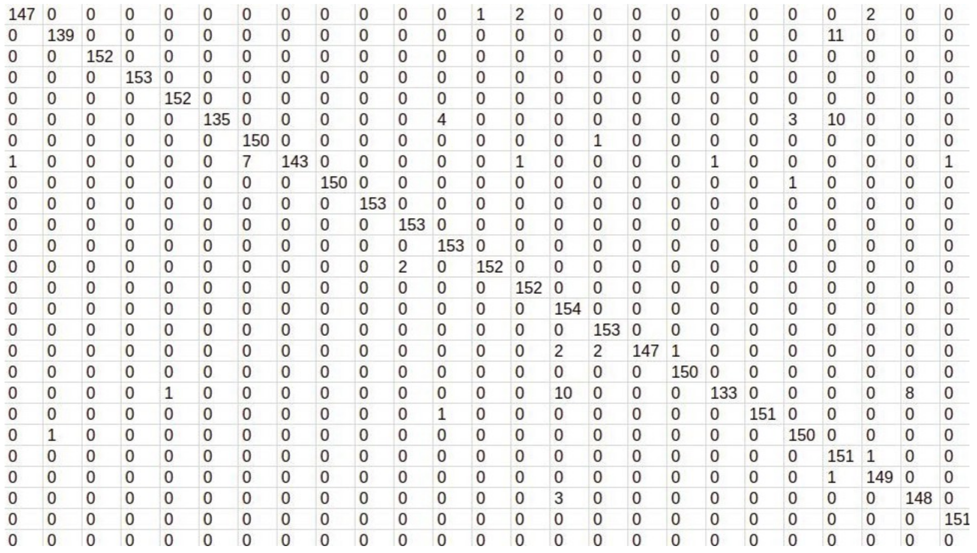
A confusion matrix for a sign-to-text translator is a table that is used to evaluate the performance of the translator by comparing the predicted text output with the actual text corresponding to the signs.



```
[[234   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0 256   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0 215   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0 241   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0 224   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0 263   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0 243   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0 240   0   0   0   0   0   5   0   0   0]
 [  0   0   0   0   0   0   0   0 213   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0 227   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0 248   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0 220   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0 260   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0 239   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0 267   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 235   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 240]]
```

Figure 7.3: Confusion matrix for word

.



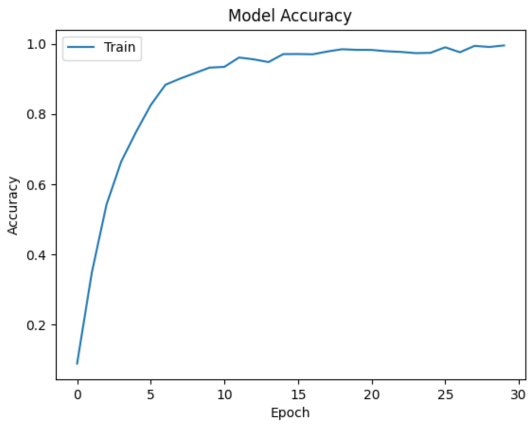
147	0	0	0	0	0	0	0	0	0	0	0	1	2	0	0	0	0	0	0	0	0	2	0	0
0	139	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	11	0	0	0
0	0	152	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	153	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	152	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	135	0	0	0	0	0	4	0	0	0	0	0	0	0	0	3	10	0	0	0
0	0	0	0	0	0	150	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	7	143	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	1
0	0	0	0	0	0	0	0	150	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	0	0	153	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	153	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	153	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	2	0	152	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	152	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	154	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	153	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	2	147	1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	150	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0	0	0	0	0	10	0	0	0	133	0	0	0	0	8	0
0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	151	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	150	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	151	1	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	149	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	148	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	151
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 7.4: Confusion matrix for ASL

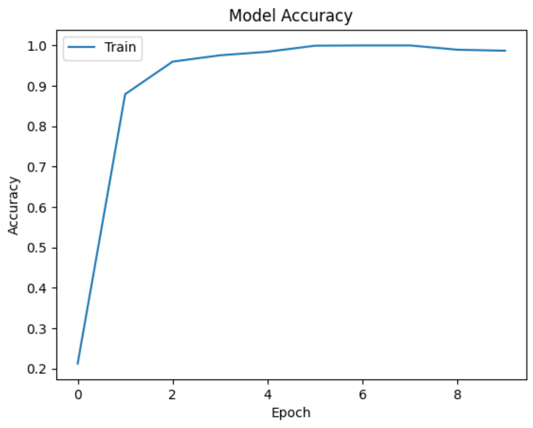
.

7.1.2 Accuracy curve

By visualizing the accuracy curve, you can gain insights into the behavior of the sign-to-text translator and make informed decisions to improve its performance.



(a) ASL accuracy



(b) Word accuracy.

Figure 7.5: Accuracy curve

7.2 FUTURE SCOPE

We are planning to achieve higher accuracy even in case of complex backgrounds by trying out various background subtraction algorithms. We are also thinking of improving the Pre Processing to predict gestures in low light conditions with a higher accuracy.

This project can be enhanced by being built as a web/mobile application for the users to conveniently access the project. Personalized hand gestures. Increasing dataset folders. Generate mobile application for ASL without higher negative impacts.

CHAPTER 8

CONCLUSION

In this project, we have successfully designed real-time vision-based ASL recognition system specifically for individuals with hearing and speech impairments. The system focuses on recognizing ASL alphabet gestures, pocket sign(hand pose) and achieves a commendable accuracy rate of 99.0% on our dataset. Throughout the project, we implemented a two-layer algorithmic approach to improve the prediction accuracy. This gives us the ability to detect almost all the symbols provided that they are shown properly, there is no noise in the background and lighting is adequate

The developed system demonstrates the ability to accurately detect and recognize a wide range of ASL alphabet symbols, assuming proper hand gestures, minimal background noise, and adequate lighting conditions. It provides a valuable tool for facilitating communication between individuals with hearing and speech impairments and those who do not understand sign language.

BIBLIOGRAPHY

[1] Rawan A Al Rashid Agha, Muhammed N Sefer, and Polla Fattah. A comprehensive study on sign languages recognition systems using (svm, knn, cnn and ann). In *Proceedings of the First International Conference on Data Science, E-learning and Information Systems*, pages 1–6, 2018.

[2] Mandeep Kaur Ahuja and Amardeep Singh. Hand gesture recognition using pca. *International Journal of Computer Science Engineering and Technology (IJCSET)*, 5(7):267–27, 2015.

[3] Bharath Kumar Akshatharani and N Manjanaik. Sign language to text-speech translator using machine learning. *International Journal of Emerging Trends in Engineering Research*, 9(7), 2021.

[4] Vivek Bheda and Dianna Radpour. Using deep convolutional networks for gesture recognition in american sign language. *arXiv preprint arXiv:1710.06836*, 2017.

[5] Vimal Gaur, Rinky Dwivedi, Pankhuri, and Shivam Dhar. Conversion of sign language into devanagari text using cnn. In *Advances in Computational Intelligence and Communication Technology: Proceedings of CICT 2021*, pages 85–94. Springer, 2022.

[6] Imran Khan, Sohail Rana, NADIA MUSTAQIM ANSARI, RIZWAN IQBAL, TALHA TARIQ, MAQSOOD UR REHMAN AWAN, and ADNAN WAQAR. design and implementation of cnn for sign language recognition. *Talha Tariq, Maqsood Ur, et al., design and implementation of cnn for sign language recognition*, 42:135–145, 2022.

[7] Hamzah Luqman. An efficient two-stream network for isolated sign language recognition using accumulative video motion. *IEEE Access*, 10:93785–93798, 2022.

[8] Romala Sri Lakshmi Murali, LD Ramayya, and V Anil Santosh. Sign language recognition system using convolutional neural network and computer vision. 2020.

[9] Anup Nandy, Jay Shankar Prasad, Soumik Mondal, Pavan Chakraborty, and Gora Chand Nandi. Recognition of isolated indian sign language gesture in real time. In *Information Processing and Management: International Conference on Recent*

Trends in Business Administration and Information Processing, BAIP 2010, Trivandrum, Kerala, India, March 26-27, 2010. Proceedings, pages 102–107. Springer, 2010.

[10] Qian Zhang, Dong Wang, Run Zhao, and Yinggang Yu. Myosign: enabling end-to-end sign language recognition with wearables. In *Proceedings of the 24th international conference on intelligent user interfaces*, pages 650–660, 2019.