FINAL REVIEW:

DSN2098 PROJECT EXHIBITION-I

VIT BHOPAL
www.vitbhopal.ac.in

PROJECT TITLE:

SIGN LANGUAGE DETECTION USING SCIKIT-LEARN AND PYTHON

23BAI10458 AARUSHI SINGH
23BAI10345 PRATYAKSHA SINGH
23BAI10437 PRIYAM
23BAI10278 JATIN KUMAR VERMA
23BAI10469 DIGVIJAY KHATRI

SUPERVISOR - DR. RAGHAVENDRA MISHRA
REVIEWER1 - DR. GOPAL SINGH TANDEL
REVIEWER2 - DR. G. PRABU KANNA

OVERVIEW

- In our increasingly digital world, accessibility and inclusivity are paramount.
- The deaf and mute community faces unique communication challenges.
- Utilizing a machine learning classifier for computer vision, we can develop a robust system capable of recognizing and interpreting sign language gestures.
- Gesture recognition technology combined with real-time video analysis enables the interpretation of sign language into text or speech.
- This technology opens doors for improved communication and accessibility for individuals with hearing impairments.

PROBLEM STATEMENT

"Communication barriers remain a significant challenge for the deaf and mute community, limiting their ability to engage effectively with the wider world. The lack of accessible, real-time sign language recognition technology exacerbates these difficulties, hindering seamless interaction. Current solutions often fall short in terms of accuracy, affordability, and support for a diverse range of signs. This project seeks to bridge this gap by developing a robust and cost-effective Sign Language Detection System (SLDS) using advanced computer vision and machine learning techniques. By facilitating real-time and accurate recognition of sign language, this system aims to promote inclusive communication and empower the deaf and mute community."

INTRODUCTION

The deaf and mute community faces unique communication challenges, which our Sign Language Detection System (SLDS) aims to address. SLDS leverages computer vision and machine learning to enable real-time recognition of sign language gestures. It aims to empower this community by accurately recognizing and classifying sign language gestures. Our goal is to bridge communication gaps, promote inclusivity, and create a user-friendly, accessible technology solution that enhances the lives of those who rely on sign language.

REQUIREMENTS

HARDWARE

Laptop with intel i5 processor or Nvidia Graphics card 1650 and RAM 8GB or any other equivalent variant of laptop or more. Laptop with webcam is required or external webcam.

SOFTWARE

SOFTWARE REQURIMENTS
Python, VScode or any platform to run python programs. Libraries required: - OpenCV, media pipes, Pickle, scikit-learn.

LIBRARIES USED

PICKLE

The pickle library is a built-in Python module that allows for the serialization and deserialization of Python objects. Serialization is the process of converting a Python object into a stream of bytes that can be stored in a file or transmitted over a network, while deserialization is the reverse process, where a stream of bytes is converted back into a Python object. This module is versatile and can be used to serialize a wide variety of Python objects, including basic types like integers, floats, strings, lists, dictionaries, and tuples. Additionally, it supports user-defined classes and instances of these classes, making it a powerful tool for saving and loading complex objects, such as machine learning models or datasets, to and from disk.

LIBRARIES USED

OpenCV

OpenCV (Open Source Computer Vision Library) is a powerful and widely-used library in Python that provides an extensive range of functions for computer vision and image processing. It includes tools for various tasks such as image processing (e.g., image filtering, morphological operations, and feature extraction), video analysis (including motion tracking, object detection, and video stabilization), and camera calibration (which involves estimating the intrinsic and extrinsic parameters of a camera). OpenCV also supports machine learning algorithms designed for computer vision tasks like face recognition, object recognition, and scene understanding. This makes OpenCV an essential tool for developing applications that involve real-time image and video processing, as well as implementing advanced computer vision solutions.

LIBRARIES USED

MEDIAPIPE

Mediapipe is an open-source framework developed by Google that facilitates the creation of cross-platform, customizable machine learning (ML) solutions for live and streaming media. It supports a wide range of tasks including face detection and tracking, hand detection and tracking, pose estimation, object detection and tracking, image segmentation, style transfer, and video stabilization. Mediapipe is designed for real-time applications and is commonly used in interactive computer vision systems, providing high performance and accuracy in live media analysis.

LIBRARIES USED

NUMPY

Numpy is a fundamental package for numerical computing in Python. It provides support for large, multi-dimensional arrays and matrices of data, along with a collection of mathematical functions to operate on these arrays. Numpy enables element-wise operations and broadcasting, making it an essential tool for scientific computing. It offers a wide range of mathematical functions, such as linear algebra, statistical analysis, and Fourier transforms, among others.

LIBRARIES USED

SCIKIT-LEARN

Scikit-learn is a powerful and widely-used machine learning library in Python that offers a wide variety of classification, regression, and clustering algorithms. It includes popular algorithms like support-vector machines, random forests, gradient boosting, and k-means clustering. Scikit-learn is well-regarded for its simplicity, efficiency, and ease of use. It also allows for easy extension by adding new algorithms and functions, making it a versatile tool for both beginners and advanced machine learning practitioners.

LIBRARIES USED

MATPLOTLIB

Matplotlib is a plotting library for Python that enables users to create static, animated, and interactive visualizations. It supports a wide variety of plot types including line plots, bar plots, scatter plots, histograms, pie charts, box plots, and 3D plots. Matplotlib is highly customizable, allowing users to fine-tune the appearance of visualizations, and is commonly used in data analysis, research, and scientific computing for presenting data in an intuitive, graphical format.

ALGORITHM USED

HOW IT WORKS?

Steps Involved in Random Forest Algorithm:

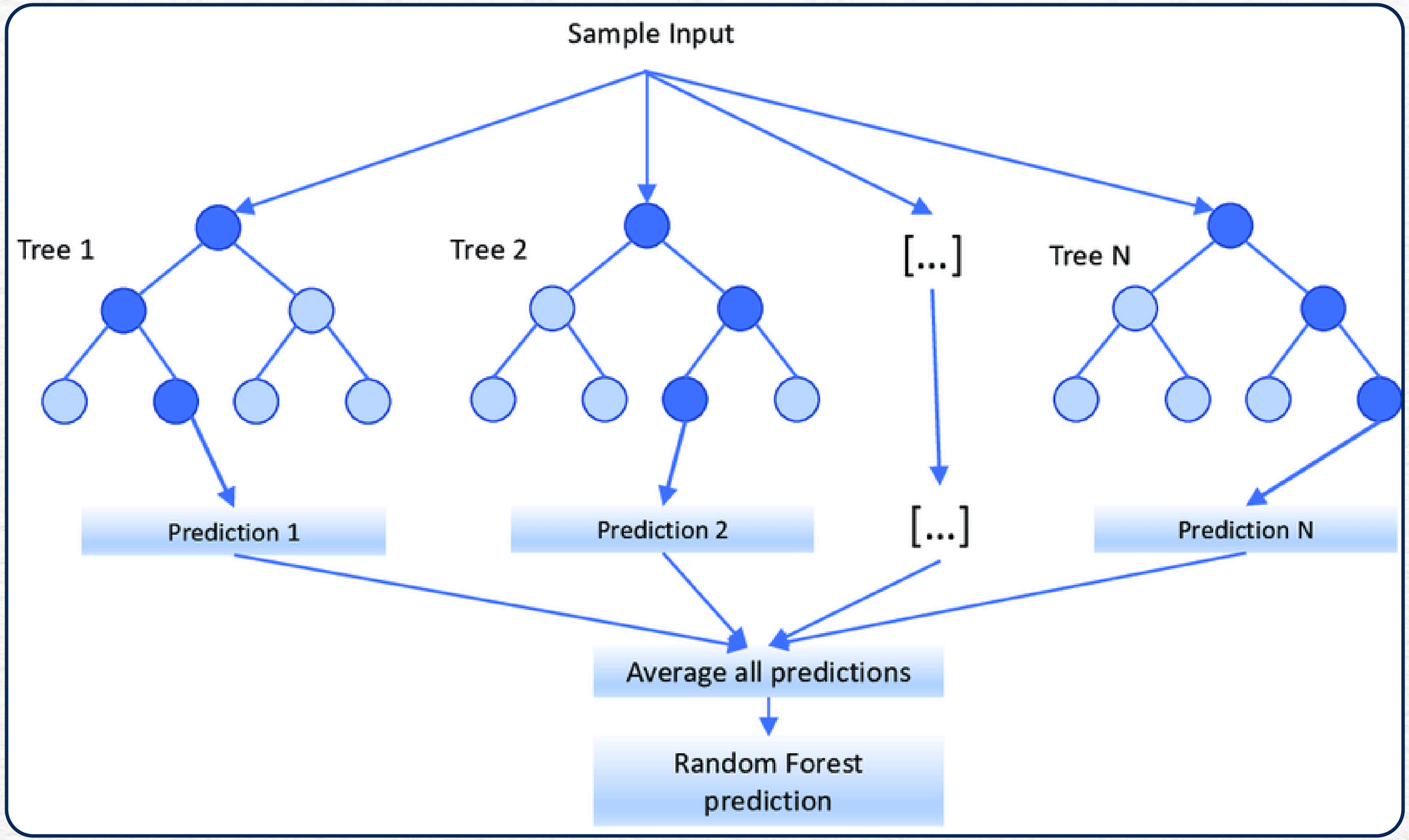
Step 1: In the Random forest model, a subset of data points and a subset of features is selected for constructing each decision tree. Simply put, n random records and m features are taken from the data set having k number of records.

Step 2: Individual decision trees are constructed for each sample.

Step 3: Each decision tree will generate an output. Step 4: Final output is considered based on Majority Voting or Averaging for Classification and regression, respectively.

WHY WE USE IT?

We are using a random forest classifier as it is less prone to overfitting, offers high accuracy, better handling of imbalanced data, immune to dimensionality curse etc.


Sample Input
Tree 1
Tree 2
[...]
Tree N
Prediction 1
Prediction 2
[...]
Prediction N
Average all predictions
Random Forest prediction

PROCESS FLOW

DATA COLLECTION

- Collect a dataset of sign language gestures using a webcam.
- Ensure the dataset covers various real-life scenarios, capturing a wide variety of hand gestures and movements.

DATA PREPROCESSING

- Label each gesture with its corresponding sign language element.
- Split the dataset into training and testing sets to ensure proper model evaluation.

FEATURE EXTRACTION

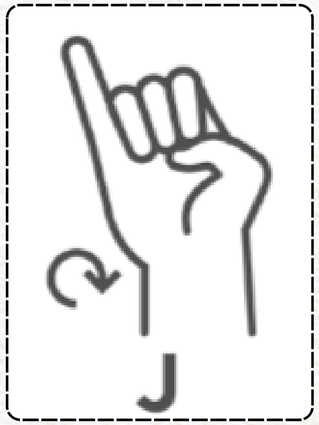
- Extract key features from the hand gestures such as hand shape, position, and movement.
- Utilize techniques like edge detection, contour analysis, and keypoint extraction to represent these features in a form suitable for machine learning models.

TRAIN MODEL

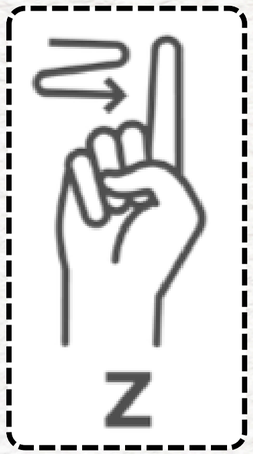
- Choose a machine learning algorithm. For sign language recognition, classifiers like Support Vector Machines (SVM) or Random Forests can be effective.
- Train the chosen model on the preprocessed training dataset.

TEST MODEL

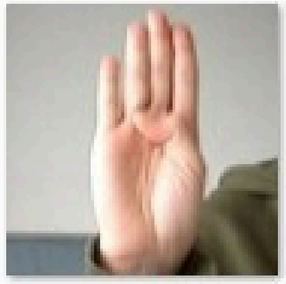
- Fine-tune hyperparameters to improve the model's performance.
- Implement real-time sign language detection using the trained model. This involves capturing video frames from the webcam, extracting features, and feeding them to the model for prediction.

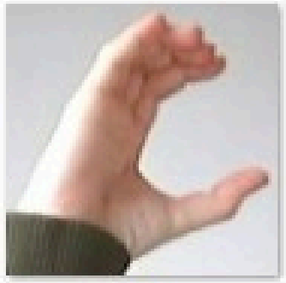


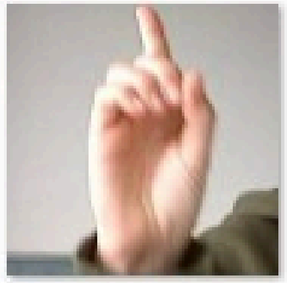
ASL CHART



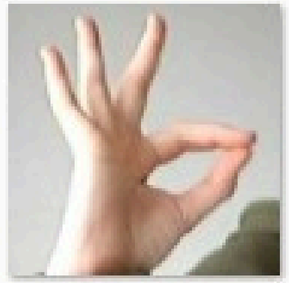

A

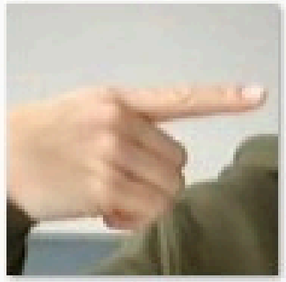

B

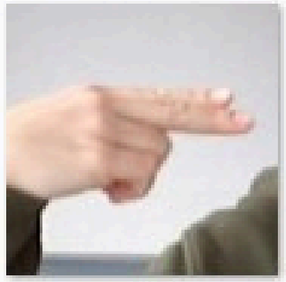

C

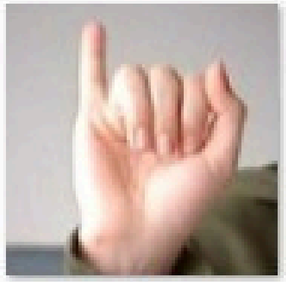

D

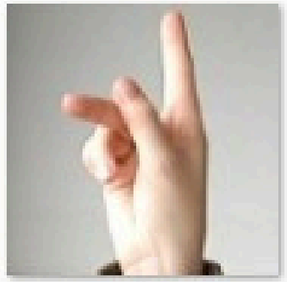

E

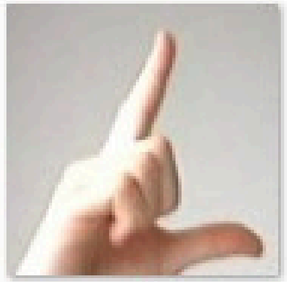

F


G


H


I

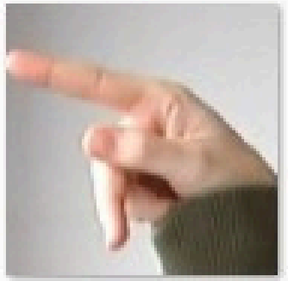

K

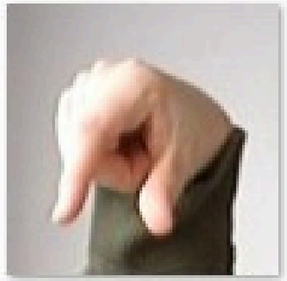

L

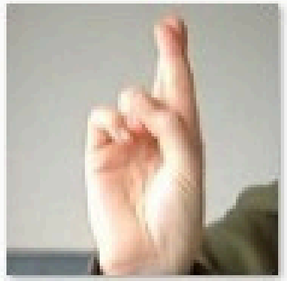

M

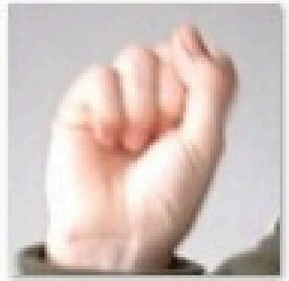

N

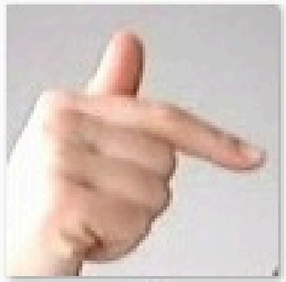

O

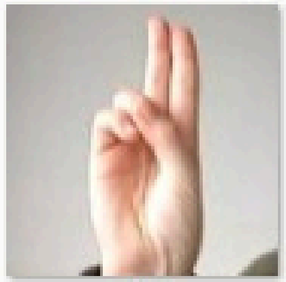

P

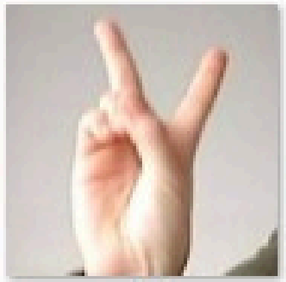

Q


R

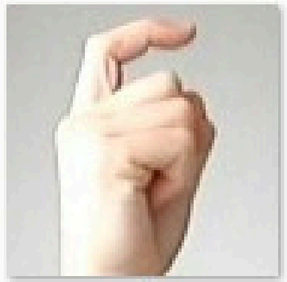

S

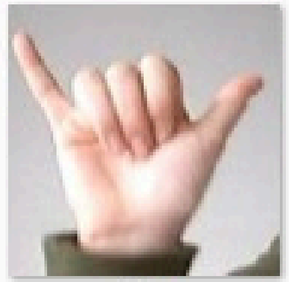

T


U


V


W


X


Y

MODEL ANALYSIS AND RESULT

Dataset: The model was trained and tested on a carefully curated dataset of hand gesture images, ensuring a diverse representation of each alphabet. This comprehensive dataset enabled the model to learn intricate patterns and variations effectively.

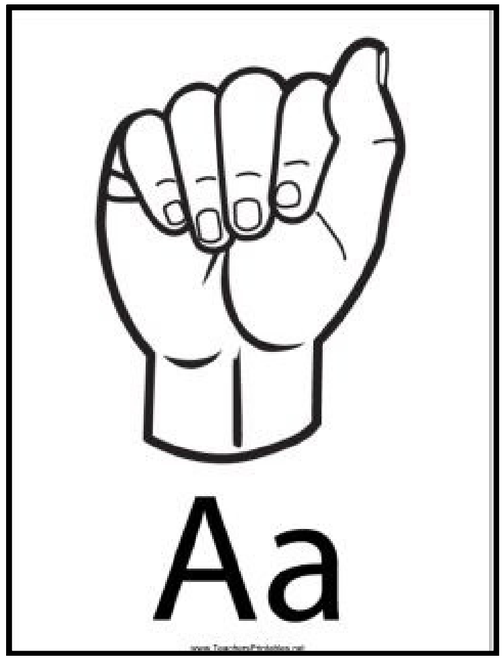
Observations: Accuracy: Achieving 100% accuracy highlights the robustness and precision of the Random Forest Classifier used in this project.

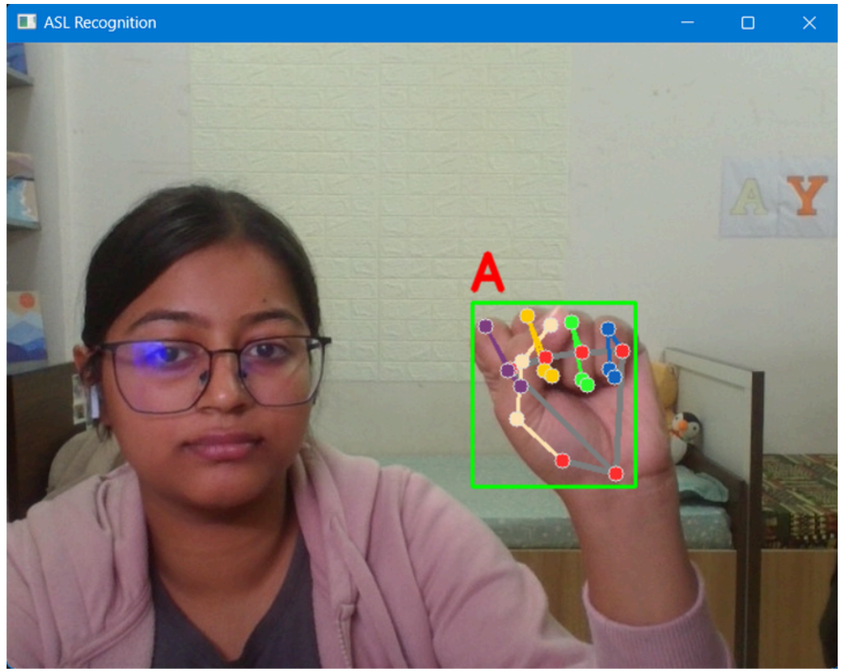
Model Reliability: The flawless accuracy reflects that the model was able to identify every hand gesture correctly without any misclassification.

Data Size and Quality: The dataset's size and quality played a pivotal role in attaining this exceptional accuracy.

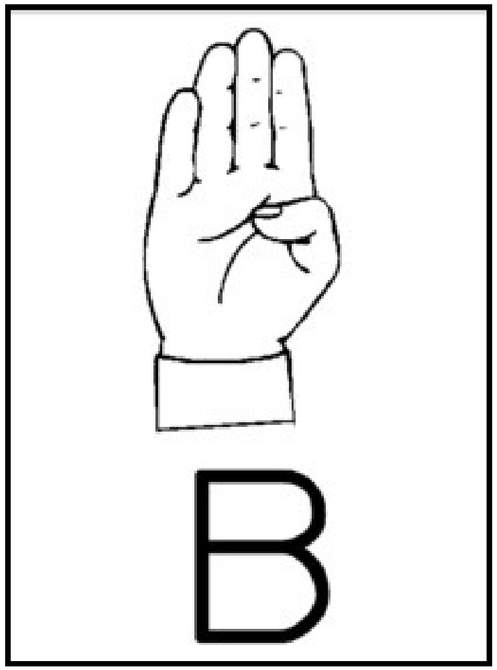
Trial and Error: Various iterations were performed to fine-tune the model, including adjusting dataset splits, hyperparameters, and preprocessing steps, ensuring optimal performance

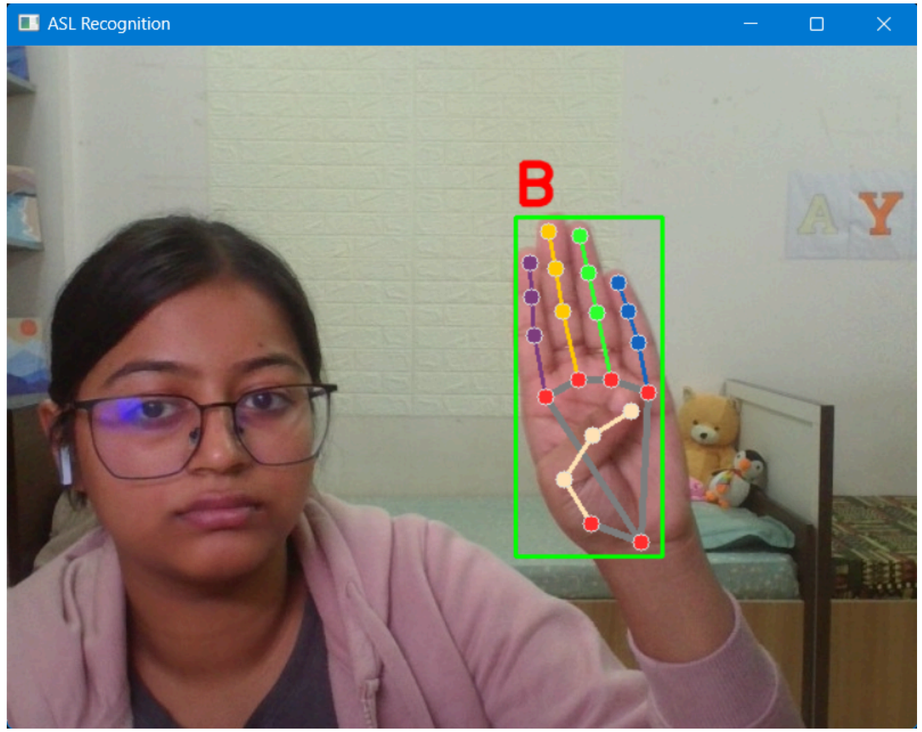
REAL TIME DETECTION



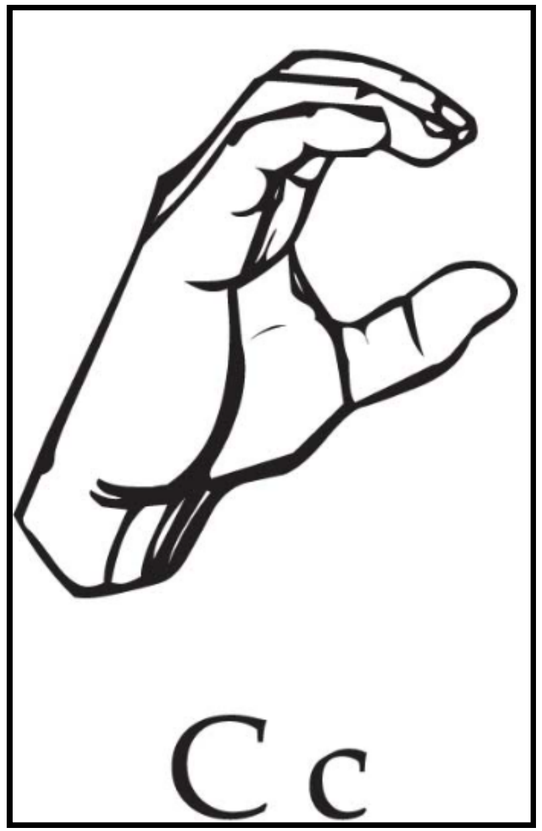


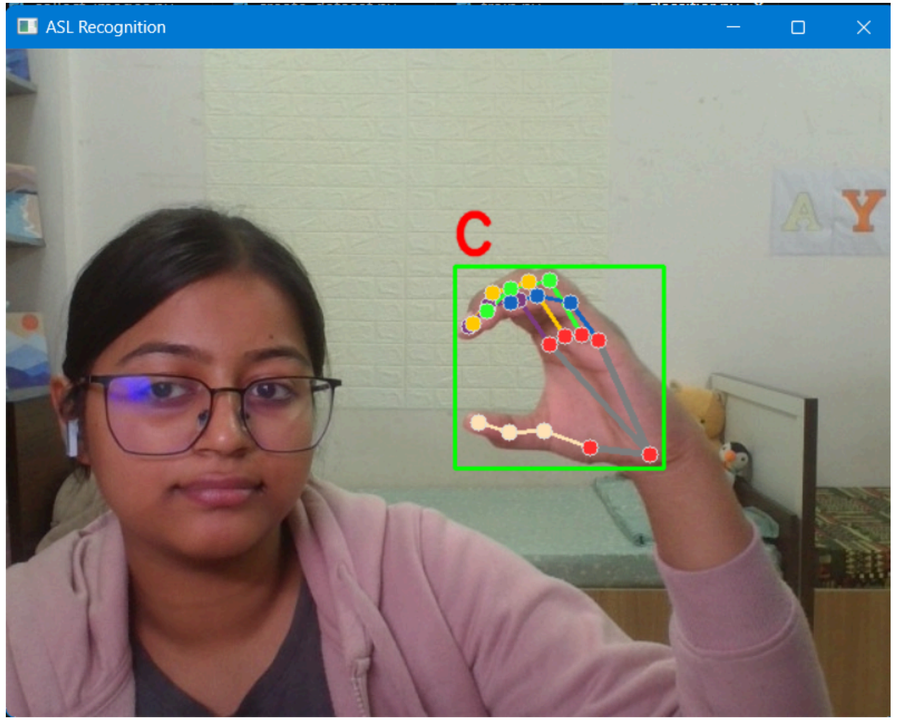
REAL TIME DETECTION





REAL TIME DETECTION





FURTHER ENHANCEMENTS

- **Multilingual Support:** Recognize signs in different sign languages (e.g., BSL, ISL).
- **Multiple Gesture Detection:** Allow recognition of more complex gestures and sentences.
- **Mobile Application:** Develop a mobile app version of the project for easier access and usability.

CONCLUSION

The Sign Language Detection System (SLDS) represents a significant step toward enhancing inclusivity and accessibility for the deaf and mute community. By harnessing the power of computer vision and machine learning, this project has successfully developed a system capable of real-time sign language recognition. Through rigorous data collection, model training, and testing, we have achieved accurate and efficient recognition of a diverse sign language vocabulary.

The potential for this technology is immense. It has the capacity to bridge communication gaps, making sign language more accessible and practical in a wide range of applications, from educational settings to everyday interactions. While we have made significant strides in this project, the process doesn't end here. We are committed to continual improvement, guided by user feedback and future collaborations with sign language experts.

THANK YOU!