Sign Language Detection

Student Name : Karan Rana
University Roll :2018865
Section :C

Under the mentorship of :Mr
Arnav Kotiyal
(Assistant professor)

Introduction

Problem statement

Sign lagnuage detection

- Sign language is a vital means of communication for the deaf and hard-of-hearing community, offering a rich and expressive way to convey thoughts, emotions, and information.
- However, the widespread adoption and understanding of sign language among the hearing population remain limited, creating communication barriers that can lead to social isolation and misunderstandings.

- To bridge this gap, our sign language detection project aims to leverage cutting-edge machine learning and computer vision technologies to develop an intuitive and accurate sign language recognition system.
- This system is designed to translate sign language gestures into written or spoken language in real-time, facilitating smoother and more effective communication between sign language users and those unfamiliar with it.

By integrating advanced neural networks and image processing techniques, our project seeks to create a robust platform that can recognize a wide range of sign language gestures with high accuracy. This initiative not only enhances accessibility and inclusivity for the deaf and hard-of-hearing community but also fosters greater awareness and understanding of sign language among the general public.

Our sign language detection project represents a significant step towards a more inclusive society where communication barriers are minimized, and everyone can participate fully in social, educational, and professional interactions.

Methodology

1. Data Collection
2. Data Processing
3. Model Training
4. Real-Time Sign Language Detection

Data Collection

Setup and Initialization:

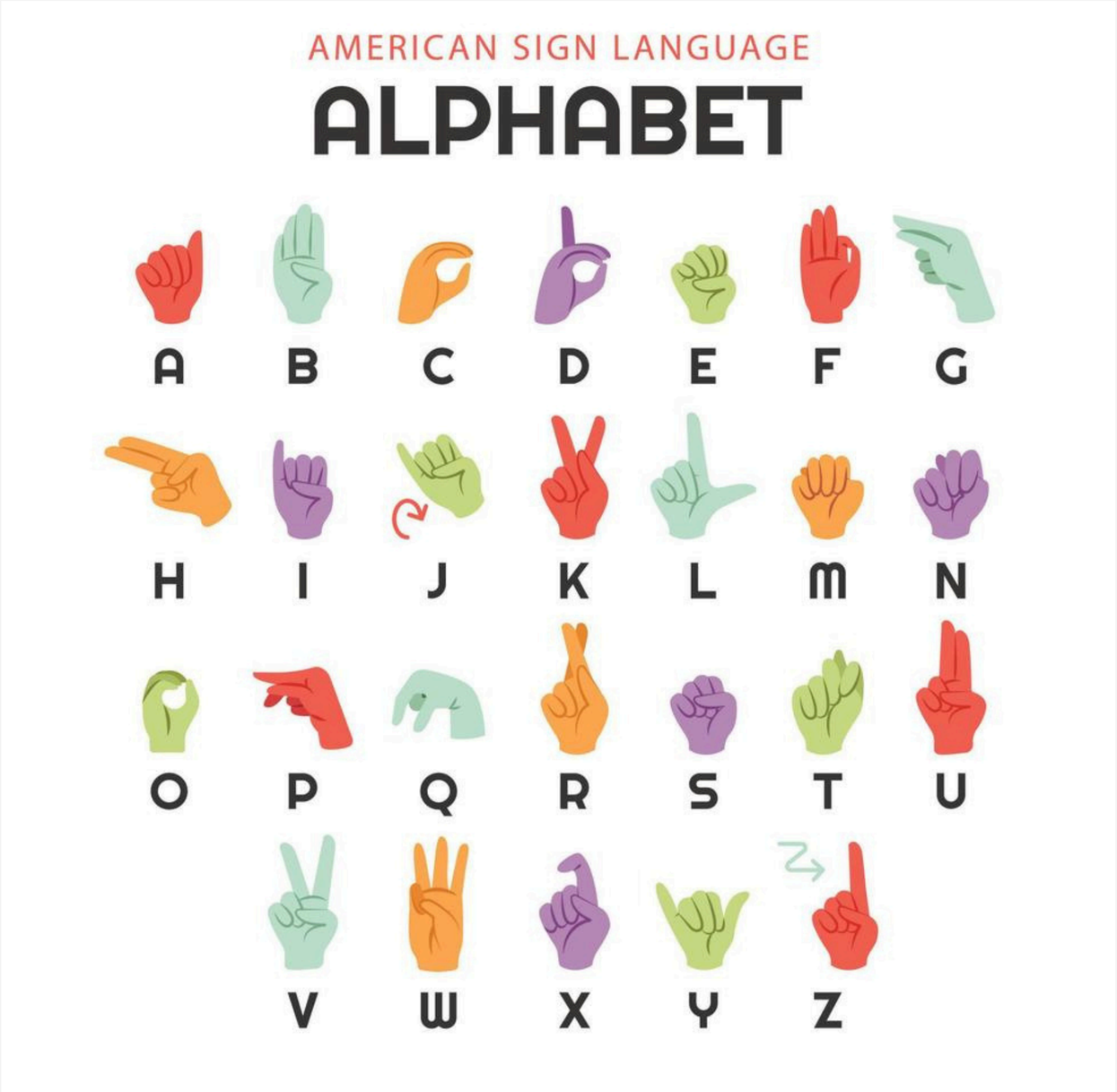
- The script begins by setting up the data directory and initializing the number of classes and dataset size.
- A video capture object (cap) is created to access the webcam.

Creating Directories:

- For each class (representing a different sign), a directory is created to store the captured images.

Capturing Images:

- The webcam captures images for each class.
- An initial screen prompt instructs the user to press 'Q' to start capturing images.
- Once 'Q' is pressed, 100 images are captured for each class and stored in the respective directory.



Data Processing

Loading Libraries:

- Libraries such as mediapipe for hand landmarks detection, cv2 for image processing, and pickle for data serialization are imported.
- Hand Landmark Detection:
- The mediapipe library is used to detect hand landmarks in the captured images.
- For each image, the hand landmarks are processed, and the coordinates are extracted and normalized.

Data Preparation:

- The extracted landmarks are stored in lists (data and labels), where each entry corresponds to the coordinates of the hand landmarks and their respective class labels.

Saving Data:

- The data and labels are serialized and saved into a pickle file (data.pickle) for later use.

Model Training

Loading Data:

The serialized data is loaded from the pickle file.

Data Padding

The data samples are padded to ensure they all have the same length for the machine learning model.

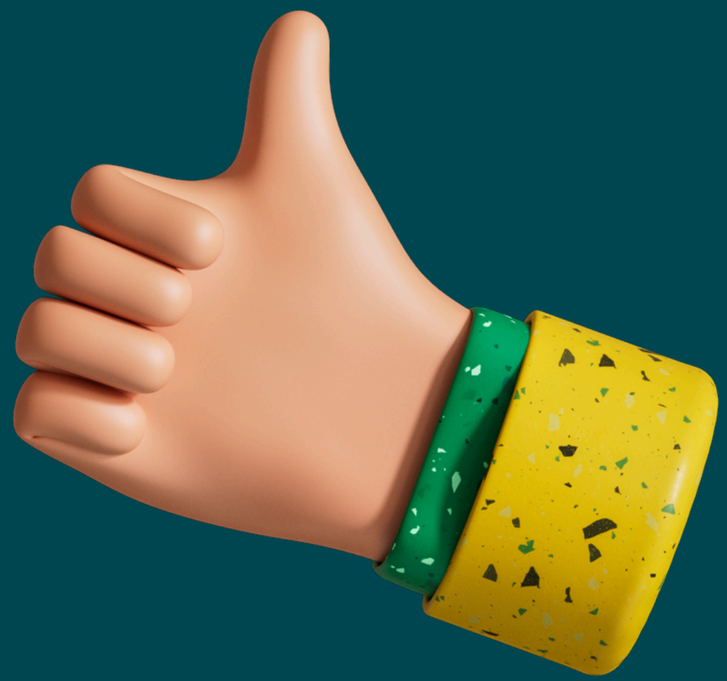
Splitting Data

- The data is split into training and testing sets using train_test_split from sklearn.

Training the Model

- A RandomForestClassifier from sklearn is used to train the model on the training data.

Real-Time Sign Language Detection



Loading the Model:

- The trained model is loaded from the pickle file.

Capturing Real-Time Video:

- The webcam captures real-time video frames.

Hand Landmark Detection in Real-Time:

- The mediapipe library detects hand landmarks in each video frame.

Prediction:

- The detected hand landmarks are processed and normalized.
- The processed landmarks are fed into the trained model to predict the corresponding sign language character.

Displaying Results:

- The predicted character is displayed on the video frame along with a bounding box around the detected hand.

Technologies Used

OpenCV:

- For capturing video from the webcam and processing images.
- Used for drawing rectangles and text on the video frames.

Mediapipe:

- For detecting hand landmarks and extracting their coordinates.
- Provides a robust solution for real-time hand tracking and gesture recognition.

Scikit-learn:

- For training and evaluating the RandomForestClassifier.
- Provides tools for data preprocessing, model training, and evaluation.

Python Pickle:

- For serializing and deserializing data and models.
- Used to save the processed data and trained model for later use.

Thanks for giving your precious time