Sign Language Recognition

Ai Platforms

SUPERVISED BY: DR. LAMIA ALY

FACULTY OF COMPUTERS AND DATA SCIENCES,
ALEXANDRIA UNIVERSITY.
MAY 2024

Sign Language Recognition

Team Members

Member Name	***ID***	***Department***
Alia Medhat	20221440735	AI
Aya Abdelmoniem	20221462478	AI
Malak Mahmoud Aref	20221445867	AI
Nouran Mohamed	20221321932	AI

Table of Contents

1. Introduction

Deaf individuals face significant communication barriers in their daily lives, hindering their ability to interact effectively with others. This project aims to address this challenge by developing a real-time Arabic sign language recognition system using Python to create a solution that enhances communication accessibility for the deaf community. The proposed solution is a sign language translation that can translate between sign language and spoken language in real time. Will use the device's camera to capture the user's hand gestures, and use artificial intelligence to interpret them as text or speech. Conversely, will also use the device's speaker or earphones to output speech or text as sign language animations on the screen. In today's digital age, communication is essential for social interaction, education, and employment. However, for the deaf community, traditional communication methods such as speech and text may not always be accessible. "Hands Speak" aims to address this challenge and to facilitate seamless communication between deaf individuals and the hearing community through sign language interpretation.

2. Dataset Overview

The dataset used in our project is the "ARASL Database 54K Final" available on Kaggle. This dataset contains hand gesture images representing the Arabic sign language alphabet. Here's an overview of the dataset:

- **Dataset Source:** ARASL Database 54K Final
- **Total Images:** 54,000
- **Format:** Image files (JPG or PNG) organized into folders representing different letters of the Arabic alphabet.
- **Labeling:** Each image is labeled according to the corresponding Arabic letter it represents.
- **Usage:** The dataset is utilized for training machine learning models to recognize and translate hand gestures into Arabic letters.

3. Data Preprocessing

3.1. Data Cleaning and Processing:
The image_processed() function applies several preprocessing steps to the input image. It converts the image to RGB format, flips it horizontally for consistency, and uses MediaPipe's Hand module to detect hand landmarks. These landmarks are then extracted and cleaned to remove unnecessary information, resulting in a list of landmark coordinates.

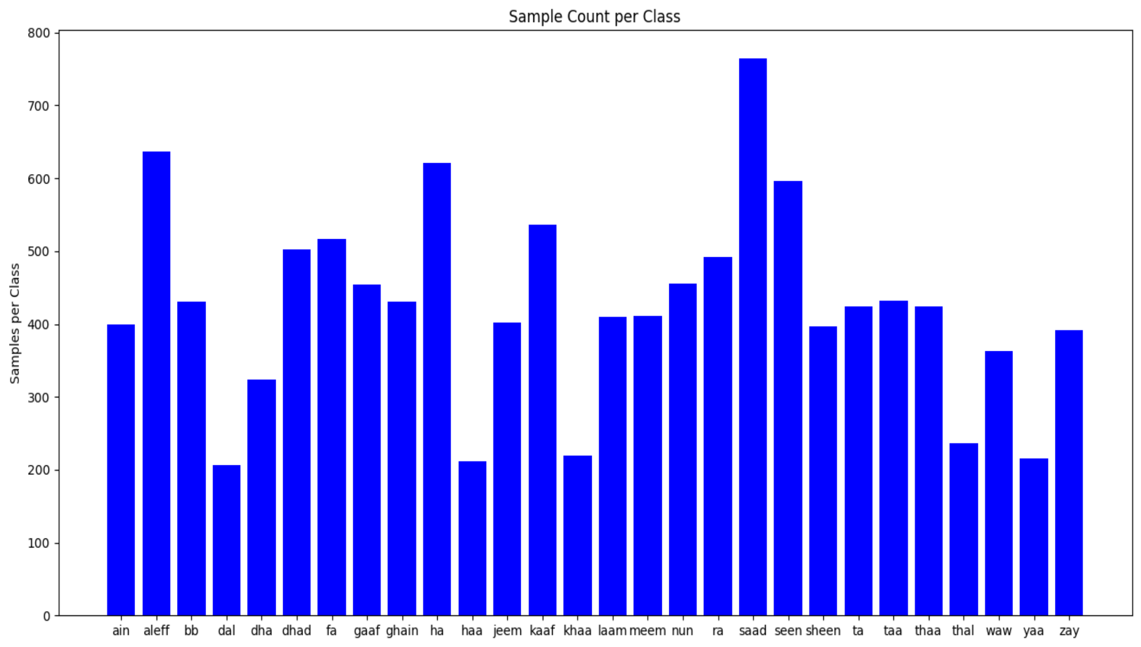
3.2. Dataset Creation:
The make_csv() function iterates through the dataset folders and images, calling image_processed() to extract hand landmarks. It writes these landmarks along with their corresponding labels into a CSV file named

datasettrial.csv. This step ensures that the data is in a suitable format for training machine learning models.

3.3. Error Handling: The code includes robust error handling to manage cases where hand landmarks cannot be detected or processed. If an error occurs during image processing, placeholder values are written to maintain the structure of the dataset.

4. Gesture Recognition

4.1. Data Loading and Visualization: After loading the dataset from datasettrial.csv, null values are removed to ensure data integrity. The distribution of samples per class is visualized using a bar plot, providing insights into the dataset's composition and potential class imbalances.



4.2. Model Training and Evaluation:

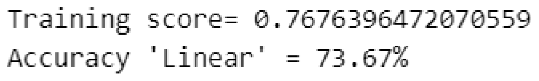
The dataset is split into training and testing sets, with SVM models trained using different kernels. Training scores, accuracy metrics, confusion matrices, and classification reports are generated to evaluate model performance. This thorough evaluation helps identify the best-performing model for gesture recognition.

SVM is a powerful supervised learning algorithm used for classification tasks. SVM can efficiently perform linear and nonlinear classification by finding the optimal hyperplane that best separates the classes in feature space. The choice of kernel in SVM determines the decision boundary shape and thus affects the model's performance.

4.2.1. Linear Kernel:

- The linear kernel is the simplest form of SVM kernel.
- It assumes that the data is linearly separable in the input space.
- The decision boundary is a straight line or hyperplane.

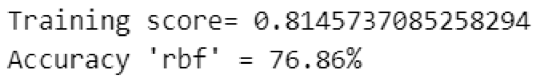
- Linear kernels are computationally efficient and work well when the data is linearly separable.
- However, they may not perform well when the data is not linearly separable.



```
Training score= 0.7676396472070559
Accuracy 'Linear' = 73.67%
```

4.2.2. Radial Basis Function (RBF) Kernel:

- The RBF kernel is a popular choice for SVMs as it can handle non-linear decision boundaries.
- It is suitable for cases where the classes are not linearly separable in the input space.
- The decision boundary is non-linear and can adapt to complex patterns in the data.
- The RBF kernel introduces a parameter, gamma (γ), which controls the influence of each training example.
- It is more computationally intensive compared to the linear kernel, especially with large datasets.



```
Training score= 0.8145737085258294
Accuracy 'rbf' = 76.86%
```

4.3. Model Persistence: The trained SVM model with the highest accuracy (rbf kernel) is saved using joblib for future use. This ensures that the model can be easily deployed without the need for retraining, saving time and resources.

5. Real-Time Translation

5.1. Hand Landmark Detection: The code utilizes MediaPipe to detect hand landmarks in real-time webcam input. This process involves converting the image to RGB format, detecting landmarks, and drawing them on the image for visualization.

5.2. Gesture Prediction and Translation: Detected landmarks are cleaned and used to predict hand gestures using the trained SVM model. These predicted gestures are then translated into Arabic letters and displayed on the webcam feed in real-time. Additionally, user input is displayed in Arabic, allowing for seamless communication between the user and the system.

5.3. Text-to-Speech Conversion: The final Arabic text is converted to speech using gTTS, providing an additional layer of accessibility for users. This feature enables deaf individuals not only to see the translated text but also to hear it, enhancing the overall user experience.

6. Future Directions

6.1. Mobile Application Development: In the future, the HANDS speak project could be expanded into a mobile application, allowing users to access the gesture translation functionality on their smartphones or tablets. This application could leverage the device's camera for real-time gesture detection and provide a user-friendly interface for communication.

6.2. Enhanced Accessibility and Features: The mobile application could include additional features such as a larger vocabulary of gestures, customizable settings for gesture recognition sensitivity, and integration with other communication tools used by the deaf community. This would further enhance accessibility and usability for users with different communication needs.

6.3. Community Engagement and Feedback: Engaging with the deaf community for feedback and suggestions would be essential for refining the application's features and ensuring that it meets the needs of its users. User feedback could inform future updates and improvements to the application's functionality and user interface.