Sign-Language-to-Text-Conversion

Deep Learning Project for conversion of American Sign Language to Text format.

Documentation Folder

Project report

Source Code Folder

Model Folder Contains the Json and h5 files of trained CNN model

Pics folder Contains Sign Language of All Character

5 Files to Execute the Project

Importance of Each file

Collect-data.py file:. I have decided to create my own dataset. Using this file Own Dataset for each Character will be created.

Image_processing.py & preprocessing.py files: To improve the accuracy of the project I converted the RGB Images to Black & white Image. After Conversion I applied Gaussian Blur Filter which focuses on Boundary of the Sign Language.

train.py file: File contains the code of model training

app.py File: Contains the code of Front End which is created using TKinter.

Steps to Execute:

I have uploaded dataset. Execute Collect data file first and Create our own Dataset for each character.

Execute Image processing and Preprocessing files. So the Black & White Images of your dataset will be created.

Execute train.py file so json & h5 files for your dataset will be created

Execute the App.py file so the frontend will be loaded & conversion of each sign language will be displayed

How frontend Works

Once the Frontend is loaded,

Show the Sign language of character you wanted to translate.

For the 50 frames predicted text conversion will be stored in backend and most predicted character will be displayed in front of character.

When all the characters will be predicted don't show any sign language in screen. When Model finds blank screen it predicts word is completed.

After Predicting all characters it will be shifted in front of word and if blank screen continues it will be shifted in front of sentence.

Installation Requirements (With Versions)

Installation Requirements (With Versions)

1. Python Version

- **Python: 3.10.12**

---

2. Required Python Libraries

Library Name	Version	Purpose
tensorflow	**2.11.0**	**For CNN model (Keras-based)**
opencv-python	**4.8.0.76**	**For webcam capture and image processing**
mediapipe	**0.10.7**	**For real-time hand tracking**
numpy	**1.23.5**	**For numerical operations**
pillow	**9.5.0**	**For image handling in Tkinter**
scikit-learn	**1.2.2**	**For confusion matrix generation**
matplotlib	**3.7.1**	**To visualize confusion matrix**
tkinter	**built-in**	**GUI for the application (comes with Python)**
json	**built-in**	**To load model structure & class mappings**

h5py	**3.8.0**	**To load .h5 trained model file**

---

3. IDE / Code Editor

- **Visual Studio Code (recommended for development and debugging)**

---

4. Additional Recommendations

- **Pandas *(Optional)*: If working with CSV data or datasets.**
- **typing-extensions: Sometimes required for compatibility with TensorFlow.**
- **CUDA/cuDNN: If using GPU, install compatible versions matching TensorFlow.**

---

5. File Requirements

Ensure the following model files are present in your project:

- **model.json – CNN model structure**
- **model.h5 – Trained weights**
- **class_labels.json – Label mapping for predictions**

---

6. Folder Structure Suggestion

EchoVerse(Black)/

│

├── Source Code/

│ ├── app.py

│ ├── model.json

│ ├── model.h5

│ └── class_labels.json

│

├── test_images/

├── README.md (optional)

└── requirements.txt (optional)

Image of Project