National University of Computer and Emerging Sciences

Applied Artificial Intelligence
Project Report

Course Instructor: Ms. Shahela Saif

Project Name: SyncSign

Project Introduction:

LipSyncSign is a dynamic real time sign language. The application is carefully crafted to cater the needs of the hearing impaired people. With **Computer Vision** and **Long Short Term Memory** model of deep learning, it is a fascinating and complex project. Its goal is to accurately identify and interpret the signs or symbols. LSTM model is a specialized type of recurrent neural network (RNN) which excels in processing sequential data by capturing long term dependencies and patterns. Therefore, these models are well suited for tasks including computer vision and real time detection.

Project Scope:

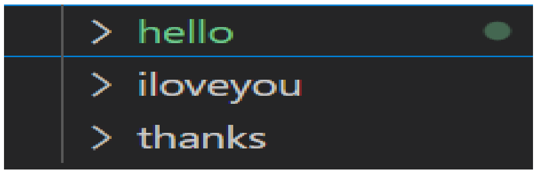
- **Project Definition:**

  LipSyncSign includes real time sign language interpretation within a dynamic environment. By employing computer vision and LSTM models, it can accurately identify different signs in varying environments like changing lighting and sign orientations.

  - Objectives:
    1. Ensure real time processing capabilities for practicality.
    2. Help hearing impaired in interpretation and communication.
    3. Understand working of LSTM deep learning models.
- **Data Collection:**

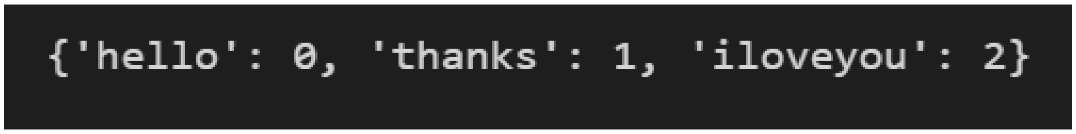
  In the data collection phase, we utilized OpenCV to capture real-time video feed for sign recognition. Each sign was annotated and recorded in sequences of 30 frames, with each sequence having a length of 30 frames. The annotated signs included "hello," "thanks," and "iloveyou". Data for each sign was stored separately in .npy files within individual folders, ensuring organized and easily accessible datasets for training and testing purposes. This helped in data collection to provide a space for LSTM model implementation for real-time sign detection and classification.

❖ Actions Folder



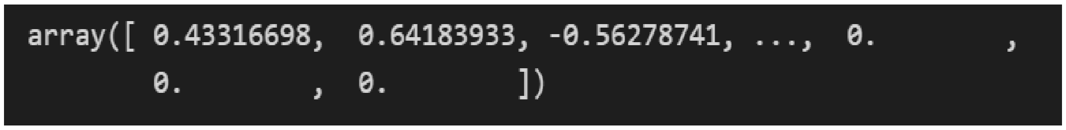
```
> hello
> iloveyou
> thanks
```

❖ Actions



```
{'hello': 0, 'thanks': 1, 'iloveyou': 2}
```

❖ Numpy arrays

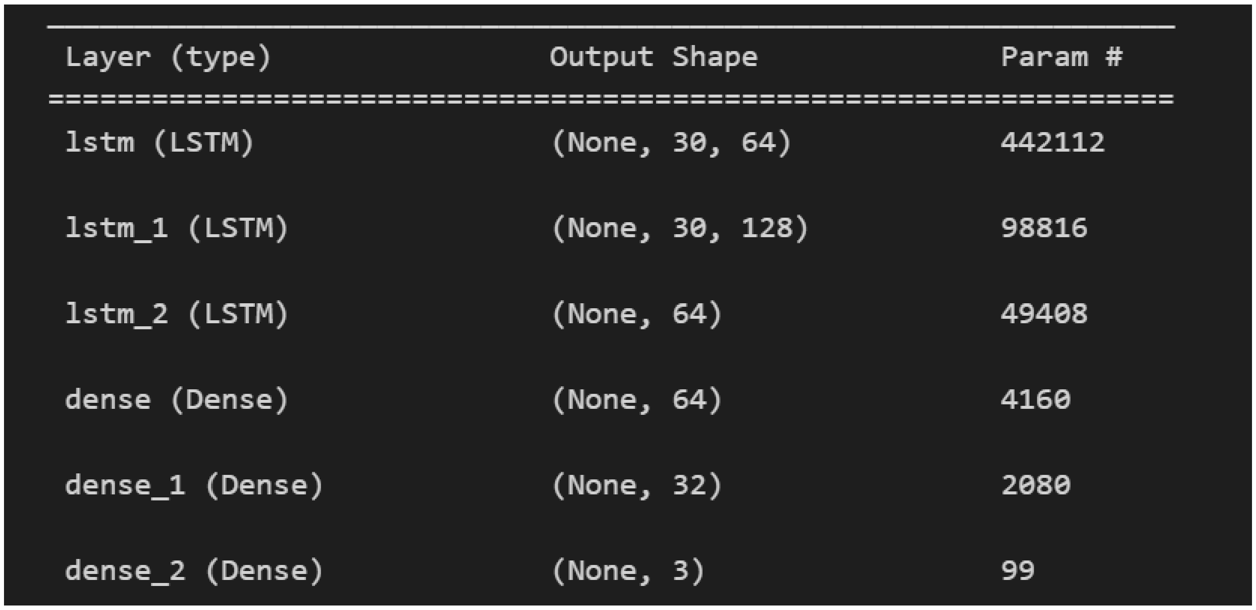


```
array([ 0.43316698,  0.64183933, -0.56278741, ...,  0.        ,
        0.        ,  0.        ])
```

- **Model Type and Architecture:**

  Through employing TensorFlow's Keras API, it is able to perform the real-time detection and classification of signs in dynamic environments. The model architecture comprises multiple **LSTM (Long Short-Term Memory)** layers for sequential processing, followed by dense layers for classification. Through implementing multiple layers of LSTM like:

- An input layer with LSTM units configured for 64 neurons, set to return sequences and utilizing the RELU activation function. This layer accepts input sequences of 30 frames, each containing 1662 features.
- A subsequent LSTM layer with 128 neurons, also configured to return sequences and utilizing RELU activation.
- A final LSTM layer with 64 neurons, configured to return only the final output sequence, employing RELU activation.
- Dense layers for further processing, including a 64-neuron layer followed by a 32-neuron layer, both utilizing RELU activation.
- An output layer employing the softmax activation function, with the number of neurons corresponding to the total number of sign classifications.

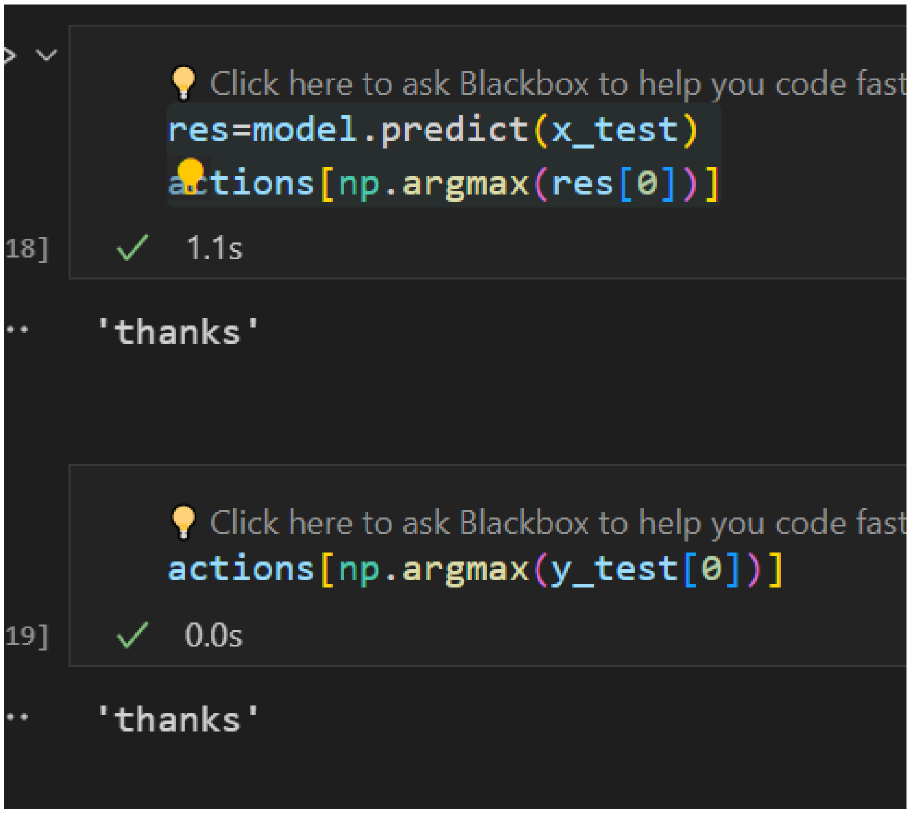


```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 30, 64)            442112

 lstm_1 (LSTM)               (None, 30, 128)           98816

 lstm_2 (LSTM)               (None, 64)                49408

 dense (Dense)               (None, 64)                4160

 dense_1 (Dense)             (None, 32)                2080

 dense_2 (Dense)             (None, 3)                 99
```

Additionally, the model is compiled using the Adam optimizer and categorical cross entropy loss function, with categorical accuracy as the evaluation metric.

- **Training and Optimization:**

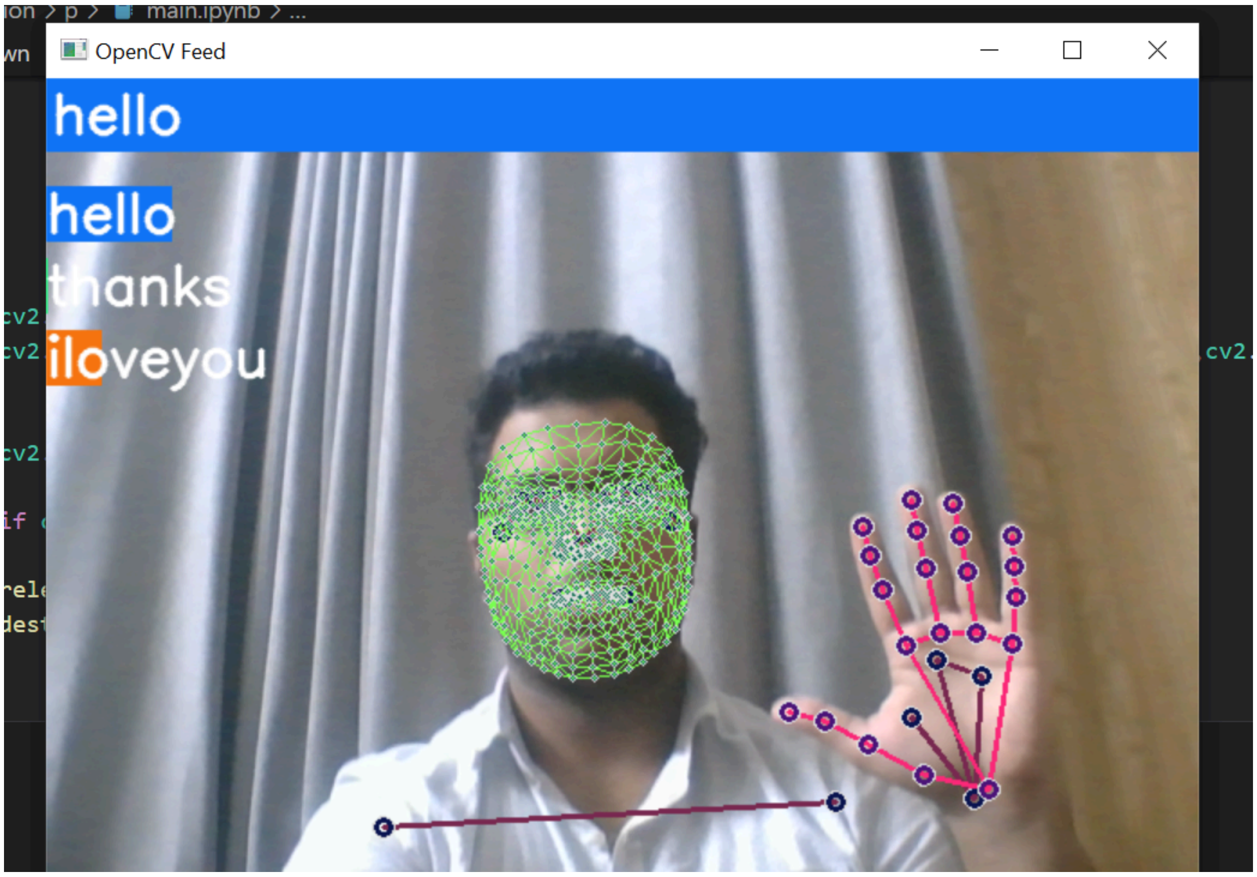
  To continuously improve the model's performance and efficiency, a separate python script can be executed to train and optimize the model whenever new data becomes available. Through TensorFlow's Keras API, the model is trained for 1000 epochs on the training dataset. To ensure the model's generalizability and its performance, testing and training data has a 95:5 partition. The data is prepared for training by loading sequences of frames corresponding to each sign action. These sequences were then converted into numpy arrays, annotated with their respective categorical labels, thus creating the input-output pairs for model training. This approach to training ensures the model's proficiency in real-time sign detection and classification. Additionally, it ensures adaptability to evolving datasets, therefore better results and optimization.



```
Click here to ask Blackbox to help you code fast
res=model.predict(x_test)
actions[np.argmax(res[0])]
18]   ✓  1.1s
..   'thanks'

Click here to ask Blackbox to help you code fast
actions[np.argmax(y_test[0])]
19]   ✓  0.0s
..   'thanks'
```

- **Results:**

  The model's performance is evaluated using the accuracy score metric. It yielded an accuracy of 0.8 out of 1. This metric indicates the model's efficacy in accurately identifying sign actions. To give an enhanced interpretation of the model's performance, a functionality was introduced to enable the display of the probability distribution for each sign action. Using OpenCV, different color rectangles representing the real time probabilities of different sign actions, overlaid on output frames. Additionally, it displays the sign actions in a sequence on the output frame for better interpretation and visual appeal.


OpenCV Feed
hello
hello
thanks
iloveyou