SpeakEasy : A Tool for people with communication disabilities

A Mini-Project Report
Submitted For
Partial Fulfillment of the Requirements of
the Degree of Bachelor of Engineering

In

COMPUTER ENGINEERING

(Semester V)
By

Max Gonsalves 9541
Vedant Chawardol 9527
Saville D'silva 9536
Ivan Dsouza 9601

Under the guidance of

Dr. Vijay Shelake

DEPARTMENT OF COMPUTER ENGINEERING
Fr. Conceicao Rodrigues College of Engineering
Bandra (W), Mumbai - 400050
University of Mumbai

2023-2024

This work is dedicated to my family.

I am very thankful for their motivation and support.

CERTIFICATE

This is to certify that the mini-project entitled **"SpeakEasy : A Tool for people with communication disabilities"** is a bonafide work of " Max Gonsalves **(9541),** Vedant Chawardol **(9527),** Saville D'silva**(9536),** Ivan Dsouza **(9601)** " submitted to the University of Mumbai in partial fulfillment of the requirement for the award of the degree of **Bachelor of Engineering** in **Computer Engineering** (Semester- V).

(Name and Sign)

Guide/ Supervisor

Dr. Sujata Deshmukh

Dr. S. S. Rathod

Approval Sheet

Mini Project Report Approval for T.E. (Semester-V)

This mini-project report entitled SpeakEasy : A Tool for people with communication disabilities submitted by Max Gonsalves (9541), Vedant Chawardol (9527), Saville D'silva(9536), Ivan Dsouza (9601) is approved for the degree of Bachelor of Engineering in **Computer Engineering** (Semester-V).

Examiner 1.———————————

Examiner 2. ———————————

Date:

Place
:

Declaration

We declare that this written submission represents our ideas in our own words and where others' ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Date:

Max Gonsalves
(9541)

Vedant
Chawardol
(9527)

Saville
D'silva(9536)

Ivan Dsouza
(9601)

Abstract

This project aims to develop an AI-powered communication system that addresses the needs of individuals with communication disabilities. It will provide sign language to text, voice. The system will support multiple local Indian languages, promoting inclusivity at national level . By incorporating AI-based tools, it will empower individuals with communication disabilities by breaking the barrier of sign language and helping them express themselves to others without any restrain. Ultimately, this project seeks to create a more inclusive society where communication barriers are eliminated for individuals with disabilities.

Keywords:

Inclusive Communication, AI-Powered, Sign Language Accessibility Technology, Multilingual Support, Empowering Disabilities, Inclusivity Initiative, Assistive Technology.

Acknowledgments

We have great pleasure in presenting the report on “SpeakEasy : A Tool for people with communication disabilities”. I take this opportunity to express my sincere thanks towards the guide Dr. Vijay Shelake, C.R.C.E, Bandra (W), Mumbai, for providing the technical guidelines, and the suggestions regarding the line of this work. We enjoyed discussing the work progress with her during our visits to the department.

We thank Dr. Sujata Deshmukh, Head of Computer Engineering department, Dr. Surendra Rathod, Principal and the management of C.R.C.E., Mumbai for encouragement and providing necessary infrastructure for pursuing the project.

We also thank all non-teaching staff for their valuable support, to complete our project.

Finally, we would like to thank all our family and friends for their help and constant cooperation during our project period. We are much intended to our parents for their support and encouragement.

Date:

Max Gonsalves (9541)
Vedant Chawardol (9527)
Saville D'silva(9536)
Ivan Dsouza (9601)

TABLE OF CONTENT

LIST OF FIGURES

Chapter 1
INTRODUCTION

SpeakEasy, a practical and intuitive app designed to improve communication with people dealing with a variety of conversation types and challenges. In a world where the power of education is often underestimated, SpeakEasy is a beacon of support that promotes everyone's right to be heard, understood and respected. In fact, SpeakEasy represents a combination of technology and human design. This change enables efficient and powerful execution of Python in the powerful environment of Jupyter Notebook, using the full potential of carefully designed sequential models. SpeakEasy's machine learning components leverage the power of long-term memory (LSTM) networks and strategically place dense layers to create a complex backbone that enables seamless, intuitive communication for users with diverse needs. Integration with OpenCV is crucial as it is the building block that allows the collection of important videos and careful management of public information. This new approach, combined with the ease of use of Numpy libraries for data storage and management, forms the basis of a general and flexible communication system. In the constant pursuit of excellence, SpeakEasy actually relies on the power of the intuitive Sklearn package to offer tools such as confusion matrices and metrics to ensure the accuracy and reliability of their communications. At the same time, the integration of Google's text-to-speech system made it possible to have a smooth and interactive conversation by bypassing barriers and converting text into words. Be clear and concise. Building on the strong foundation of TensorFlow, SpeakEasy brings carefully designed learning models to bring user-friendly, flexible and effective communication to life. Additionally, Mediapipe Holistic's intelligent integration helps eliminate critical elements of analysis, helping to create a more intuitive and engaging user experience.

Chapter 2

OBJECTIVES OF THE PROJECT

- Develop innovative solutions to promote effective communication and participation in hearing and speech communities.
- Innovate technology to bring greater convenience and power to the deaf and hard of hearing.
- Instant translation and interpreting services are available for the hearing impaired and those with speech difficulties.
- Create a multi-channel, multi-lingual support system to close the communication gap and create a more inclusive environment for hearing and speech.

Chapter 3

SCOPE OF THE PROJECT

1. **Scope:** "SpeakEasy" addresses the challenge of communication disabilities, encompassing speech impairments, nonverbal communication difficulties, motor disabilities, cognitive impairments and temporary communication challenges.

2. **Relevance:** In the digital age, effective communication is vital. Addressing communication disabilities is relevant for inclusivity, technological advancements, improved quality of life, and social integration..

3. **Significance:** "SpeakEasy" is significant as it empowers individuals with communication disabilities, fosters inclusivity, improves well-being, showcases innovation, and strengthens connections, all contributing to a more equitable and connected world.

Chapter 4

REVIEW OF

LITERATURE

Sr. No	Title of the paper	Author	Year	Proposed Methodology
1	An Efficient Two-Stream Network for Isolated Sign Language Recognition Using Accumulative Video Motion	Hamzah Luqman	2022	Two-stream network for sign language recognition with accumulative video motion.
2	Real-Time Sign Language Detection using Human Pose Estimation	Amit Moryossef, Ioannis Tsochantaridis, Roee Aharoni, Sarah Ebling, Srini Narayanan	2020	Real-time sign language detection with human pose estimation
3	Real-time Conversion of Sign Language to Text and Speech	Kohsheen Tiku, Jayshree Maloo, Aishwarya Ramesh, Indra R	2020	Real-time conversion of sign language to text and speech
4	Machine Learning solutions with MediaPipe	Yadira Quiñonez,Carmen Lizarraga,Raquel Aguayo	2022	Implementation of Machine Learning solutions is addressed using the MediaPipe framework developed by Google
5	Sign Language Detection using LSTM Deep Learning Model and Media Pipe Holistic Approach	Mihir Deshpande; Vedant Gokhale; Adwait Gharpure; Aayush Gore; Harsh Yadav; Pankaj Kunekar; Aparna Mete Sawant	2023	Develop a deep learning model to recognize alphabet signs depicted in images and videos. We collect hand movement data using media pipe, which extracts coordinates of 21 points on the palm. These coordinates are then converted into a NumPy array and input into a long short-term memory (LSTM) model for sign detection

6	Data classification with deep learning using Tensorflow	Fatih Ertam; Galip Aydın	2017	Using Tensorflow, which is an open source artificial intelligence library developed by Google, we have studied and compared the effects of multiple activation functions on classification results
7	TensorFlow: A System for Large-Scale Machine Learning	Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng	2017	TensorFlow for large-scale machine learning
8	Conversion of Sign Language into Text	Mahesh Kumar N B	2018	Conversion of sign language into text
9	TensorFlow Data Validation: Data Analysis and Validation in Continuous ML Pipelines	Emily Caveness, Paul Suganthan G. C., Zhuo Peng, Neoklis Polyzotis, Sudip Roy, Martin Zinkevich	2019	Data analysis and validation in continuous ML pipelines using TensorFlow
10	Image text to speech conversion in the desired language by translating with Raspberry Pi	H Rithika; B. Nithya Santhoshi	2016	This paper is based on a prototype which helps user to hear the contents of the text images in the desired language. It involves extraction of text from the image and converting the text to translated speech in the user desired language.
11	Building a Light Weight Intelligible Text-to-Speech Voice Model for Indian Accent Telugu	Chevella Anil Kumar; Y. Chalapathi Rao; Ranjan K. Senapati; Vondala Naveen Kumar	2023	A dependable and effective approach is proposed for creating speech synthesis voices that are specialized for specific domains and can handle non-standard words

12	Text-to-Speech Recognition using Google API	Orlunwo Placida Orochi, Ledisi Giok Kabari	2021	The aim of the study is to use the Python programming language to introduce a text-to-speech model to see whether the messages written are read. Using Google API, text-to-speech conversion was successful.
13	Sign Language Detection using LSTM	Shreyas Mhatre; Sarang Joshi; Hrushikesh B. Kulkarni	2023	The proposed system uses a Long Short-term memory model to train the dataset. The LSTM model works similarly as compared to recurrent neural networks. LSTM neural networks are used for the classification of sign language actions.
14	Sign Language Recognition using LSTM and Media Pipe	G. Mallikarjuna Rao; Cheguri Sowmya; Dharavath Mamatha; P. A. Sujasri; Soma Anitha; Ramavath Alivela	2023	With the use of effective algorithms, top-notch data sets, and improved sensors, the system aims to enhance the performance of the current system in this area in terms of response time and accuracy.

Chapter 5
PROPOSED
SYSTEM

5.1 DRAWBACKS OF EXISTING SYSTEM

The biggest limitations of the existing system include lack of knowledge of the Indian Sign Language and lack of translation services in local languages. These limitations hinder its effectiveness in meeting multilingualism and communication needs and may limit its accessibility and inclusion to a wider audience. There is also an absence of a real time translation mobile application whereby the translation can be made more accessible to the general public.

5.2 PROBLEM STATEMENT

Our main goal is to create an artificial intelligence-supported communication system that meets the needs of people with poor communication skills. Our aim is to bridge the gap between text and speech with translation, especially for Indian languages (especially Marathi), and we are committed to promoting unity and easy access. By addressing these important issues, we aim to promote diversity and inclusion in society and improve communication.

Chapter 6

SYSTEM DESIGN

6.1 BLOCK DIAGRAM

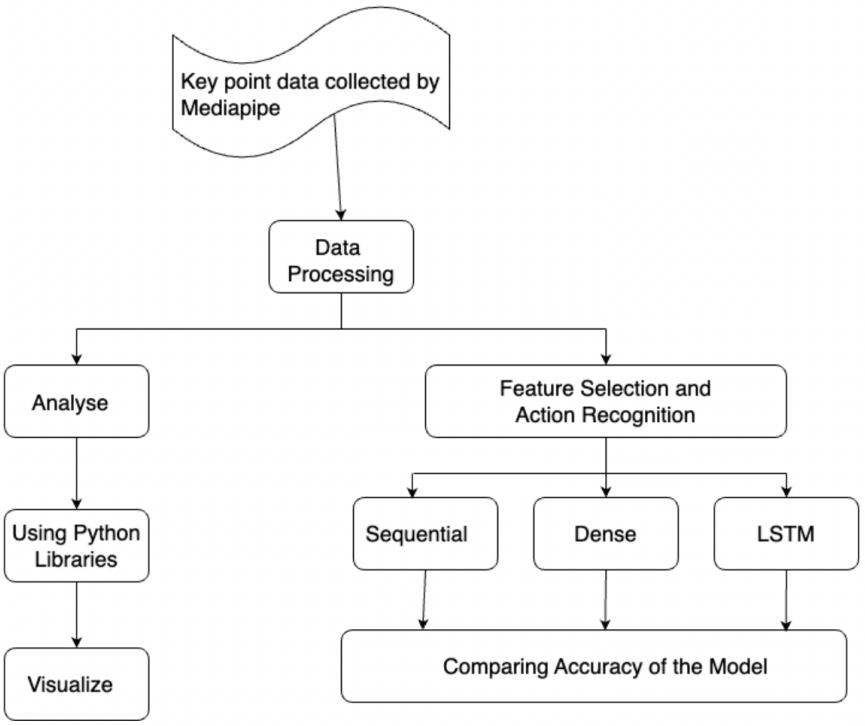


Fig. 1. Architecture Diagram

6.2 MODULE DESCRIPTION

1. **Computer Vision:**
    a. The code uses OpenCV *(Open Source Computer Vision Library)* to capture video frames from a camera *(webcam)*.
    b. It employs the MediaPipe library to perform real-time pose and gesture recognition using the Holistic model, which includes facial landmarks, body pose, and hand landmarks detection.

2. **Deep Learning:**
    a. Deep learning techniques are used for gesture recognition, specifically LSTM *(Long*

Short-Term Memory) neural networks.

b. A deep learning model is trained to recognize specific gestures. The code uses TensorFlow/Keras for building and training the neural network model.

3. **Data Collection:**
    a. The code collects training data by capturing sequences of keypoint data (pose, face, left hand, right hand) as the user performs gestures *(e.g., "hello," "thanks," "yes")*.
    b. This data is saved as NumPy arrays.

4. **Model Training:**
    a. The code trains a deep learning model using the collected keypoint data and corresponding gesture labels *(e.g., "hello", "thanks", "yes")*.
    b. The model architecture consists of LSTM layers and dense layers.
    c. Categorical cross-entropy is used as the loss function, and the Adam optimizer is used for training.
    d. Model training is supervised, and the model is saved after training.

5. **Gesture Recognition:**
    a. The code uses the trained model to recognize gestures in real-time. It continuously captures keypoint data, feeds it into the model, and makes predictions.
    b. A threshold is applied to determine when a gesture is recognized (e.g., when the prediction probability is above a certain threshold).
    c. When a recognized gesture is detected, the code responds with audio feedback by playing a corresponding audio file.

6. **Text-to-Speech (TTS):**
    a. The code uses the gTTS library to convert text into speech. It generates audio files for each gesture label in different languages (English and Marathi in this case).
    b. These audio files are stored in directories for later use.

7. **Miscellaneous:**
    a. The code integrates the Pygame library for audio playback.
    b. Tkinter is used to create a simple GUI window. This GUI window is used to display the recognized Marathi gestures in real-time.
    c. It employs NumPy for data manipulation and storage.

d. It also uses the matplotlib library for visualizing gesture recognition probabilities.

6.2.1 ALGORITHMS/ LIBRARIES AND TECHNIQUES

1. **OpenCV (cv2):**
    a. OpenCV, or Open Source Computer Vision Library, is used for computer vision tasks.
    b. It provides functions for capturing video from a webcam, image processing, and computer vision operations.
    c. In the code, it is used to capture video frames from the camera and display them.

2. **NumPy (numpy):**
    a. NumPy is a fundamental library for scientific computing with Python.
    b. It provides support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.
    c. In the code, NumPy is used for storing and manipulating data, particularly for storing keypoint information.

3. **Matplotlib (matplotlib.pyplot):**
    a. Matplotlib is a data visualization library used for creating static, animated, or interactive visualizations in Python.
    b. In the code, it is used for visualizing gesture recognition probabilities using the imshow function from the pyplot module.

4. **MediaPipe (mediapipe):**
    a. MediaPipe is an open-source library developed by Google for building real-time applications related to computer vision and machine learning.
    b. The code uses the MediaPipe library to perform real-time pose and gesture recognition using the Holistic model. It detects facial landmarks, body pose, and hand landmarks.

5. **TensorFlow and Keras:**
    a. TensorFlow is an open-source machine learning library developed by Google.
    b. Keras is a high-level neural networks API running on top of TensorFlow.
    c. In the code, TensorFlow and Keras are used for building and training the deep learning model for gesture recognition.

6. **gTTS (gTTS):**
    a. The gTTS (Google Text-to-Speech) library is used for converting text to speech.
    b. It can generate audio files from text input and supports multiple languages.
    c. In the code, gTTS is used to create audio files for Marathi and English translations of the recognized gestures.

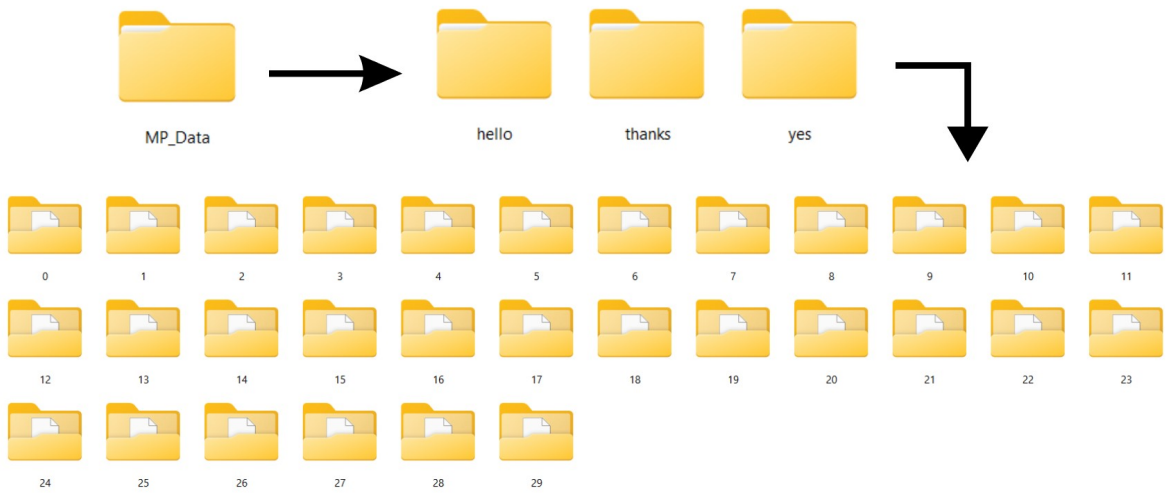
7. **Pygame:**
    a. Pygame is a library for creating video games and multimedia applications in Python.
    b. It is used in the code for audio playback.
    c. Specifically, Pygame is used to play audio files as feedback when recognized gestures are detected.

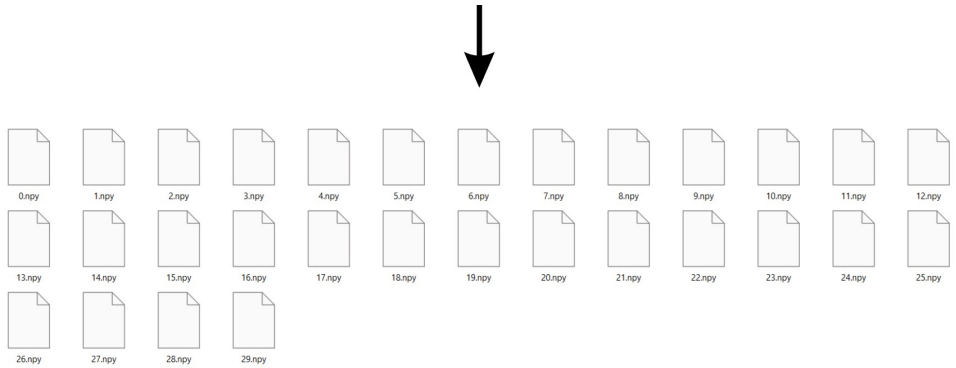
8. **Tkinter:**
    a. Tkinter is a standard Python library for building graphical user interfaces (GUIs).
    b. In the code, Tkinter is used to create a simple GUI window.
    c. This GUI window is used to display the recognized Marathi gestures in real-time.
    d. The Marathi word recognized is shown as text in the Tkinter window, allowing the user to see the recognized gestures.

6.2.2 DATASET

- **Custom Gesture Dataset:**
  - The dataset is not pre-collected or pre-structured; it is dynamically created during runtime.
  - It is a collection of keypoint data captured in real-time using the MediaPipe Holistic model.
  - The keypoint data includes information about the pose, face, and hand landmarks of a user while performing specific gestures *(e.g., "hello," "thanks," "yes")*.

- The dataset is generated while users perform gestures in front of a camera, and the corresponding keypoint information is recorded and stored for training the gesture recognition model.
  - It is created using keypoint data captured from video frames in real-time. The dataset consists of sequences of keypoint data representing various gestures. The length of the dataset and the number of frames used can vary based on the specific recording duration, the number of gestures captured, and other factors.
  - Here's a general overview:
    - **Number of Frames:** The number of frames used to create the dataset depends on the duration of the recording. For example, if the code captures 30 frames per second and records gestures for 10 seconds, you'd have approximately 300 frames in the dataset. The exact number of frames is determined by the recording session.
    - **Numpy Array Length:** Each frame captures keypoint data, which is represented as a numpy array. The length of each numpy array can vary depending on the number of keypoints detected and the specific data recorded (e.g., pose, face, and hand landmarks). Typically, each keypoint's data is stored in a flattened numpy array. Therefore, the length of each numpy array is related to the total number of keypoint coordinates recorded for the pose, face, and hands.





6.2.3 UML DIAGRAM

1. State Diagram

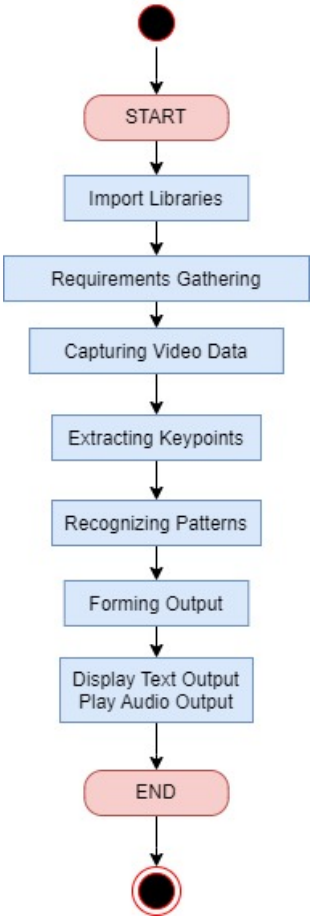


Fig. 2. State Diagram

6.2.4 SOFTWARE AND HARDWARE USED

6.2.4.1 Software Requirements

- Anaconda (or)
- Google colaboratory (or)
- Jupyter Notebook
- Python (3.x recommended)
- Libraries: OpenCV, NumPy, Matplotlib, MediaPipe, TensorFlow, Keras, gTTS, Pygame, Tkinter, and standard Python libraries.
- MediaPipe Dependencies: Ensure that you have MediaPipe's dependencies and models installed, which may require a separate installation step.

6.2.4.2 Hardware Requirements

- Operating System - Mac / Windows / Linux
- Processor - Intel i3 / i7 / i5
- Hard Disk - 250 GB and above
- RAM - 4 GB and above
- Webcam or camera.
- Audio output (speakers or headphones).

Chapter 7

IMPLEMENTATION

1. Import and Install Dependencies

In [2]:
```
!pip install tensorflow==2.14.0
!pip install opencv-python
!pip install mediapipe
!pip install scikit-learn
!pip install matplotlib
```

...

In [2]:
```
import cv2
import numpy as np
import os
from matplotlib import pyplot as plt
import time
import mediapipe as mp
```

2. Keypoints using MediaPipe Holistic

In [3]:
```
mp_holistic = mp.solutions.holistic #holistic model
mp_drawing = mp.solutions.drawing_utils #drawing utilities
```

In [4]:
```
def mediapipe_detection(image , model):
    image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB) #Conversion of color formats
    image.flags.writeable = False
    results = model.process(image)
    image.flags.writeable = True
    image = cv2.cvtColor(image, cv2.COLOR_RGB2BGR)
    return image, results
```

In [5]:
```
def draw_landmarks(image, results):
    mp_drawing.draw_landmarks(image, results.face_landmarks, mp_holistic.FACEMESH_TESSELATION)
    mp_drawing.draw_landmarks(image, results.pose_landmarks, mp_holistic.POSE_CONNECTIONS)
    mp_drawing.draw_landmarks(image, results.left_hand_landmarks, mp_holistic.HAND_CONNECTIONS)
    mp_drawing.draw_landmarks(image, results.right_hand_landmarks, mp_holistic.HAND_CONNECTIONS)
```

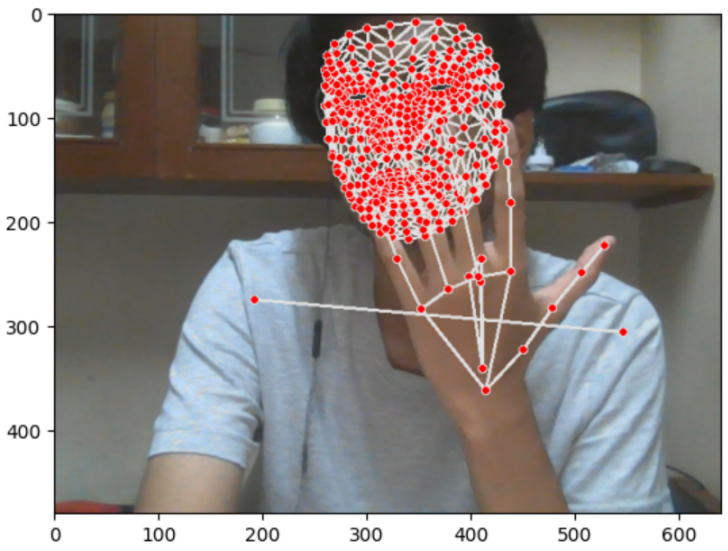
In [6]:
```
def draw_styled_landmarks(image, results):
    # Draw face connections
    mp_drawing.draw_landmarks(image, results.face_landmarks, mp_holistic.FACEMESH_TESSELATION,
                              mp_drawing.DrawingSpec(color=(80,110,10), thickness=1, circle_radius=1),
                              mp_drawing.DrawingSpec(color=(80,256,121), thickness=1, circle_radius=1)
                              )
    # Draw pose connections
    mp_drawing.draw_landmarks(image, results.pose_landmarks, mp_holistic.POSE_CONNECTIONS,
                              mp_drawing.DrawingSpec(color=(80,22,10), thickness=2, circle_radius=4),
                              mp_drawing.DrawingSpec(color=(80,44,121), thickness=2, circle_radius=2)
                              )
    # Draw left hand connections
    mp_drawing.draw_landmarks(image, results.left_hand_landmarks, mp_holistic.HAND_CONNECTIONS,
                              mp_drawing.DrawingSpec(color=(121,22,76), thickness=2, circle_radius=4),
                              mp_drawing.DrawingSpec(color=(121,44,250), thickness=2, circle_radius=2)
                              )
    # Draw right hand connections
    mp_drawing.draw_landmarks(image, results.right_hand_landmarks, mp_holistic.HAND_CONNECTIONS,
                              mp_drawing.DrawingSpec(color=(245,117,66), thickness=2, circle_radius=4),
                              mp_drawing.DrawingSpec(color=(245,66,230), thickness=2, circle_radius=2)
                              )
```

In [11]:
```
cap = cv2.VideoCapture(0)
# Set mediapipe model
with mp_holistic.Holistic(min_detection_confidence=0.5, min_tracking_confidence=0.5) as holistic:
    while cap.isOpened():

        # Read feed
        ret, frame = cap.read()

        # Make detections
        image, results = mediapipe_detection(frame, holistic)
        print(results)

        # Draw landmarks
        draw_styled_landmarks(image, results)

        # Show to screen
        cv2.imshow('OpenCV Feed', image)

        # Break gracefully
        if cv2.waitKey(10) & 0xFF == ord('q'):
            break
    cap.release()
    cv2.destroyAllWindows()
```

In [12]:
```
draw_landmarks(frame, results)
```

In [13]:
```
plt.imshow(cv2.cvtColor(frame, cv2.COLOR_BGR2RGB))
```

Out[13]: <matplotlib.image.AxesImage at 0x27868c26710>



3. Extract Keypoint Values

In [13]:
```
len(results.left_hand_landmarks.landmark)
len(results.left_hand_landmarks.landmark)
```

...

In [9]:
```
pose = []
for res in results.pose_landmarks.landmark:
    test = np.array([res.x, res.y, res.z, res.visibility])
    pose.append(test)
```

...

In [14]:
```
pose = np.array([[res.x, res.y, res.z, res.visibility] for res in results.pose_landmarks.landmark]).flatten() if results.pose
face = np.array([[res.x, res.y, res.z] for res in results.face_landmarks.landmark]).flatten() if results.face_landmarks else
lh = np.array([[res.x, res.y, res.z] for res in results.left_hand_landmarks.landmark]).flatten() if results.left_hand_landmar
rh = np.array([[res.x, res.y, res.z] for res in results.right_hand_landmarks.landmark]).flatten() if results.right_hand_landm
```

In [15]:
```
def extract_keypoints(results):
    pose = np.array([[res.x, res.y, res.z, res.visibility] for res in results.pose_landmarks.landmark]).flatten() if results.
    face = np.array([[res.x, res.y, res.z] for res in results.face_landmarks.landmark]).flatten() if results.face_landmarks e
    lh = np.array([[res.x, res.y, res.z] for res in results.left_hand_landmarks.landmark]).flatten() if results.left_hand_lan
    rh = np.array([[res.x, res.y, res.z] for res in results.right_hand_landmarks.landmark]).flatten() if results.right_hand_l
    return np.concatenate([pose, face, lh, rh])
```

In [16]:
```
result_test = extract_keypoints(results)
```

In [17]:
```
result_test
```

Out[17]:
```
array([ 0.37766296,  0.3877742 , -0.99037838, ...,  0.        ,
        0.        ,  0.        ])
```

4. Setup Folders for Collection

In [8]:
```
DATA_PATH = os.path.join('MP_Data')

actions = np.array(['yes','thanks','hello'])

no_sequences = 30

sequence_length = 30
```

In [90]:
```
for action in actions:
    for sequence in range(no_sequences):
        try:
            os.makedirs(os.path.join(DATA_PATH, action, str(sequence)))
        except:
            pass
```

5. Collect Keypoint Values for Training and Testing

In [107]:
```
cap = cv2.VideoCapture(0)
# Set mediapipe model
with mp_holistic.Holistic(min_detection_confidence=0.5, min_tracking_confidence=0.5) as holistic:

    # NEW LOOP
    # Loop through actions
    for action in actions:
        # Loop through sequences aka videos
        for sequence in range(no_sequences):
            # Loop through video length aka sequence length
            for frame_num in range(sequence_length):

                # Read feed
                ret, frame = cap.read()

                # Make detections
                image, results = mediapipe_detection(frame, holistic)
#                 print(results)

                # Draw landmarks
                draw_styled_landmarks(image, results)

                # NEW Apply wait logic
                if frame_num == 0:
                    cv2.putText(image, 'STARTING COLLECTION', (120,200),
                                cv2.FONT_HERSHEY_SIMPLEX, 1, (0,255, 0), 4, cv2.LINE_AA)
                    cv2.putText(image, 'Collecting frames for {} Video Number {}'.format(action, sequence), (15,12),
                                cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 255), 1, cv2.LINE_AA)
                    # Show to screen
                    cv2.imshow('OpenCV Feed', image)
                    cv2.waitKey(2000)
                else:
                    cv2.putText(image, 'Collecting frames for {} Video Number {}'.format(action, sequence), (15,12),
                                cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 255), 1, cv2.LINE_AA)
                    # Show to screen
                    cv2.imshow('OpenCV Feed', image)

                # NEW Export keypoints
                keypoints = exract_keypoints(results)
                npy_path = os.path.join(DATA_PATH, action, str(sequence), str(frame_num))
                np.save(npy_path, keypoints)

                # Break gracefully
                if cv2.waitKey(10) & 0xFF == ord('q'):
                    break

    cap.release()
    cv2.destroyAllWindows()
```

In [84]:
```
cap.release()
cv2.destroyAllWindows()
```

6. Preprocess Data and Create Labels and Features

In [9]:
```
from sklearn.model_selection import train_test_split
from tensorflow.keras.utils import to_categorical
```

In [10]:
```
label_map = {label:num for num, label in enumerate(actions)}
```

In [11]:
```
label_map
```

Out[11]:
```
{'yes': 0, 'thanks': 1, 'hello': 2}
```

In [12]:
```
sequences, labels = [], []
for action in actions:
    for sequence in range(no_sequences):
        window = []
        for frame_num in range(sequence_length):
            res = np.load(os.path.join(DATA_PATH, action, str(sequence), "{}.npy".format(frame_num)))
            window.append(res)
        sequences.append(window)
        labels.append(label_map[action])
```

In [13]:
```
x = np.array(sequences)
```

In [14]:
```
y = to_categorical(labels).astype(int)
```

In [15]:
```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.05)
```

7. Build and Train LSTM Neural Network

In [28]:
```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense
from tensorflow.keras.callbacks import TensorBoard
```

In [29]:
```
log_dir = os.path.join('Logs')
tb_callback = TensorBoard(log_dir=log_dir)
```

In [30]:
```
model = Sequential()
model.add(LSTM(64, return_sequences=True, activation='relu', input_shape=(30,1662)))
model.add(LSTM(128, return_sequences=True, activation='relu'))
model.add(LSTM(64, return_sequences=False, activation='relu'))
model.add(Dense(64, activation='relu'))
model.add(Dense(32, activation='relu'))
model.add(Dense(actions.shape[0], activation='softmax'))
```

In [31]:
```
res = [.7, 0.2, 0.1]
```

In [33]:
```
model.compile(optimizer='Adam', loss='categorical_crossentropy', metrics=['categorical_accuracy'])
```

In [34]:
```
model.fit(X_train, y_train, epochs=2000, callbacks=[tb_callback])
```

```
Epoch 1992/2000
3/3 [==============================] - 0s 82ms/step - loss: 0.4252 - categorical_accuracy: 0.7294
Epoch 1993/2000
3/3 [==============================] - 0s 86ms/step - loss: 0.5320 - categorical_accuracy: 0.7176
Epoch 1994/2000
3/3 [==============================] - 0s 88ms/step - loss: 0.5699 - categorical_accuracy: 0.7059
Epoch 1995/2000
3/3 [==============================] - 0s 87ms/step - loss: 0.3599 - categorical_accuracy: 0.8824
Epoch 1996/2000
3/3 [==============================] - 0s 79ms/step - loss: 0.3699 - categorical_accuracy: 0.8353
Epoch 1997/2000
3/3 [==============================] - 0s 77ms/step - loss: 0.3701 - categorical_accuracy: 0.9059
Epoch 1998/2000
3/3 [==============================] - 0s 88ms/step - loss: 0.3684 - categorical_accuracy: 0.8471
Epoch 1999/2000
3/3 [==============================] - 0s 78ms/step - loss: 0.3067 - categorical_accuracy: 0.9059
Epoch 2000/2000
3/3 [==============================] - 0s 75ms/step - loss: 0.2686 - categorical_accuracy: 0.9294
```

Out[34]: <keras.src.callbacks.History at 0x1d3626dfad0>

In [35]:
```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 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

=================================================================
Total params: 596675 (2.28 MB)
Trainable params: 596675 (2.28 MB)
```

8. Make Predictions

In [36]:
```
res = model.predict(X_test)
```

```
1/1 [==============================] - 1s 669ms/step
```

In [37]:
```
actions[np.argmax(res[4])]
```

Out[37]: 'hello'

In [38]:
```
actions[np.argmax(y_test[4])]
```

Out[38]: 'hello'

In [39]:
```
actions[np.argmax(res[1])]
```

Out[39]: 'yes'

In [40]:
```
actions[np.argmax(y_test[1])]
```

Out[40]: 'yes'

9. Save Weights

In [43]:
```
model.save('action.h5')
```

```
C:\Users\Max Gonsalves\AppData\Roaming\Python\Python311\site-packages\keras\src\engine\training.py:3079: UserWarning: You are saving your model as an HDF5 file via `model.save()`. This file format is considered legacy. We recommend using instead the native Keras format, e.g. `model.save('my_model.keras')`.
  saving_api.save_model(
```

In [67]:
```
del model
```

In [44]:
```
model.load_weights('action.h5')
```

10. Evaluation using Confusion Matrix and Accuracy

In [45]:
```
from sklearn.metrics import multilabel_confusion_matrix, accuracy_score
```

In [46]:
```
yhat = model.predict(X_test)
```

```
1/1 [==============================] - 0s 43ms/step
```

In [47]:
```
ytrue = np.argmax(y_test, axis=1).tolist()
yhat = np.argmax(yhat, axis=1).tolist()
```

In [48]:
```
multilabel_confusion_matrix(ytrue, yhat)
```

Out[48]:
```
array([[[3, 0],
        [0, 2]],

       [[4, 0],
        [0, 1]],

       [[3, 0],
        [0, 2]]], dtype=int64)
```

In [49]:
```
accuracy_score(ytrue, yhat)
```

Out[49]:
```
1.0
```

11. Test in Real Time

In [52]:
```
sequence = []
sentence = []
threshold = 0.8

cap = cv2.VideoCapture(0)
# Set mediapipe model
with mp_holistic.Holistic(min_detection_confidence=0.5, min_tracking_confidence=0.5) as holistic:
    while cap.isOpened():

        ret, frame = cap.read()

        image, results = mediapipe_detection(frame, holistic)
        print(results)

        draw_styled_landmarks(image, results)

        keypoints = extract_keypoints(results)

        sequence.append(keypoints)
        sequence = sequence[-30:]

        if len(sequence) == 30:
            res = model.predict(np.expand_dims(sequence, axis=0))[0]
            print(actions[np.argmax(res)])
```

```
            if res[np.argmax(res)] > threshold:
                if len(sentence) > 0:
                    if actions[np.argmax(res)] != sentence[-1]:
                        sentence.append(actions[np.argmax(res)])
                else:
                    sentence.append(actions[np.argmax(res)])

            if len(sentence) > 5:
                sentence = sentence[-5:]



        cv2.rectangle(image, (0,0), (640, 40), (245, 117, 16), -1)
        cv2.putText(image, ' '.join(sentence), (3,30),
                       cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 255, 255), 2, cv2.LINE_AA)

        cv2.imshow('OpenCV Feed', image)

        if cv2.waitKey(10) & 0xFF == ord('q'):
            break
    cap.release()
    cv2.destroyAllWindows()
```

In [73]:
```
from gtts import gTTS
import os
from IPython.display import Audio
```

In [109]:
```
def marathi_text_to_speech(text, output_file):
    tts = gTTS(text, lang='mr')
    tts.save(output_file)
```

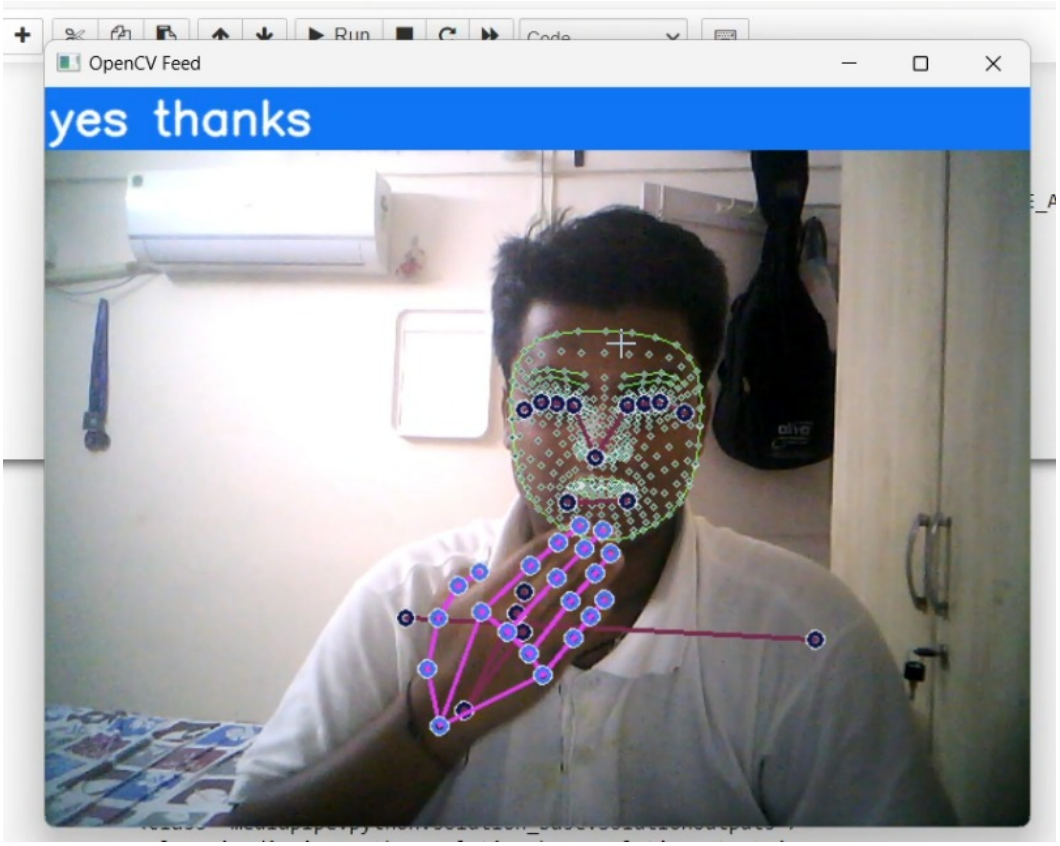
In [76]:
```
def play_audio(x):
    marathi_text = x
    output_file = "marathi_output.mp3"
    marathi_text_to_speech(marathi_text, output_file)

    output_file = "marathi_output.mp3"
    Audio(output_file, autoplay=True)
```

Chapter 8

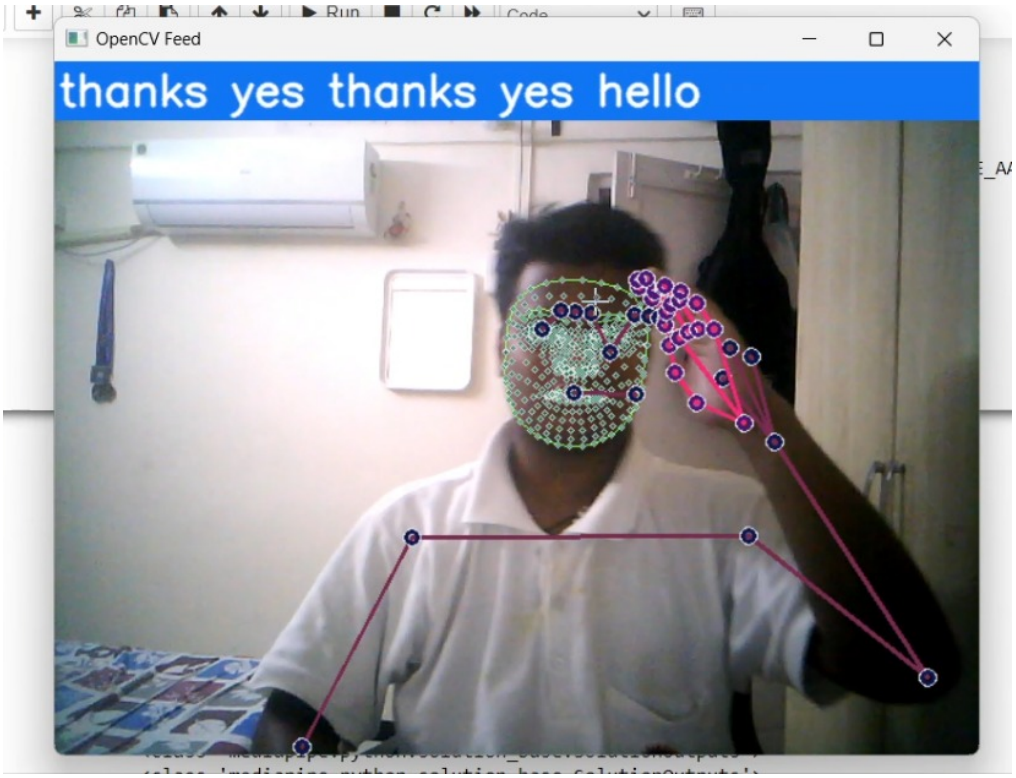
RESULTS

1. Detection of Actions:

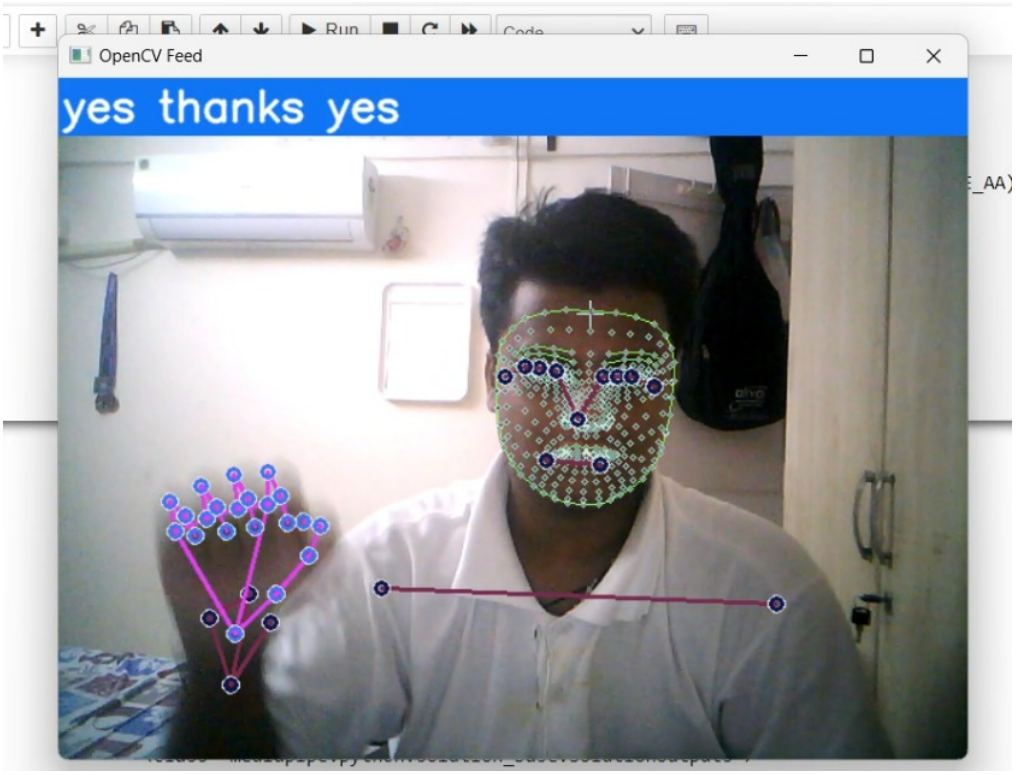
1.1 Thanks



1.2 Hello

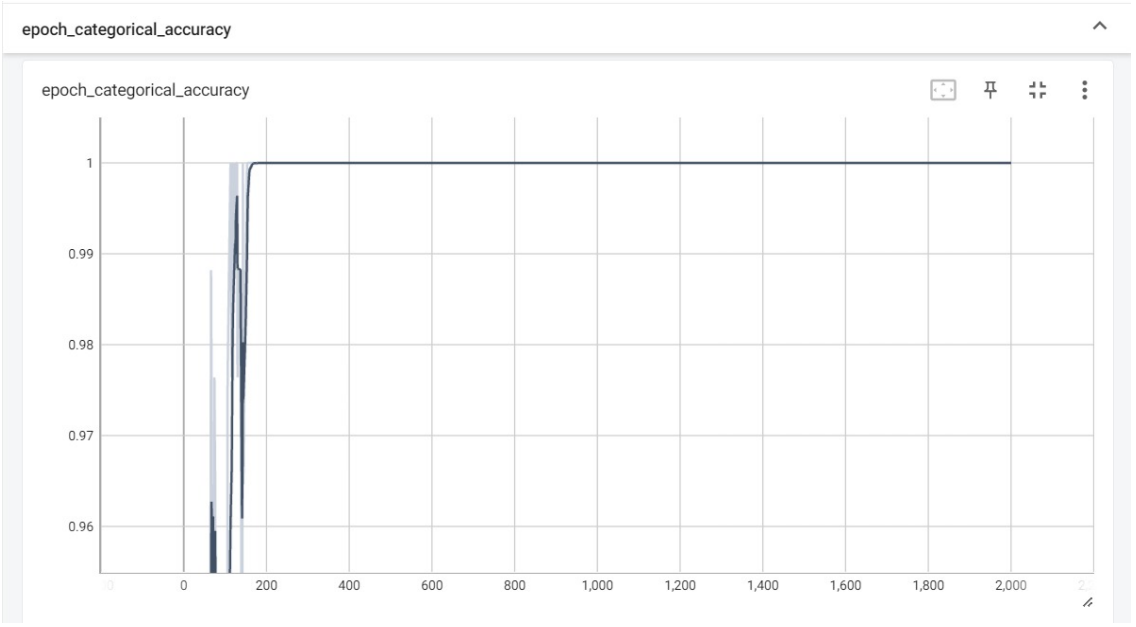


1.3 Yes

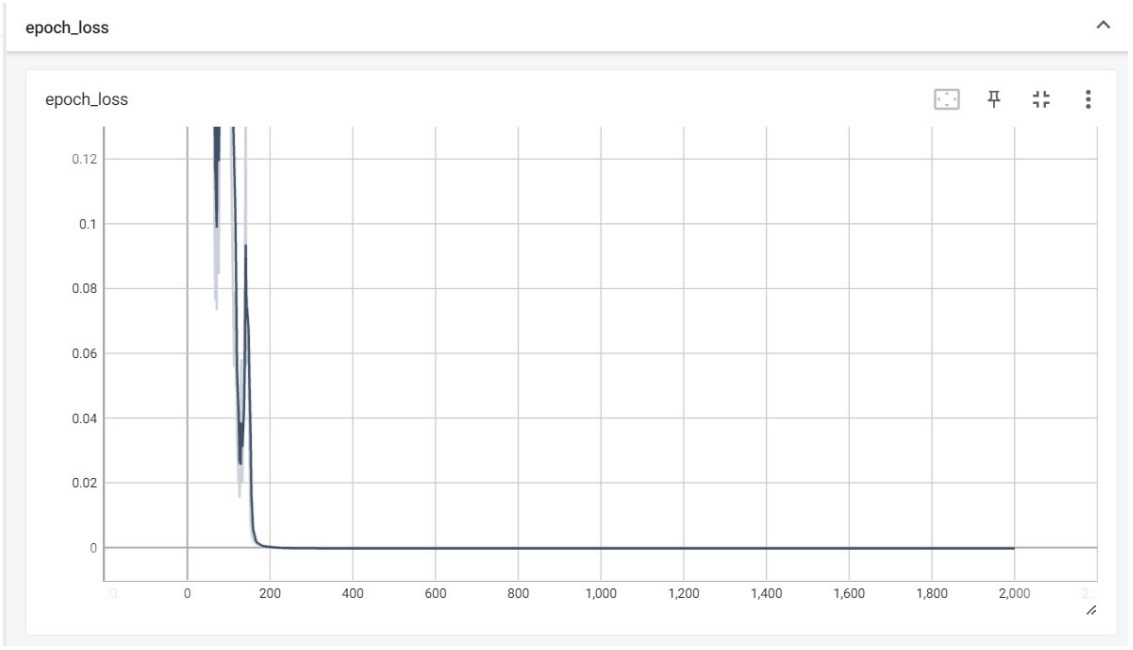


2. Graphs

2.1 Epoch Categorical Accuracy



2.2 Epoch Loss



REFERENCES

[1] K. Tiku, J. Maloo, A. Ramesh and I. R., "Real-time Conversion of Sign Language to Text and Speech," 2020 Second International Conference on Inventive Research in Computing Applications (ICIRCA), Coimbatore, India, 2020, pp. 346-351, doi: 10.1109/ICIRCA48905.2020.9182877.

[2] Jimmy Jiménez-Salas and Mario Chacón-Rivas, "A Systematic Mapping of Computer Vision-Based Sign Language Recognition", 2022 International Conference on Inclusive Technologies and Education (CONTIE), pp. 1-11, 2022

[3] Mihir Deshpande, Vedant Gokhale, Adwait Gharpure, Aayush Gore, Harsh Yadav, Pankaj Kunekar and Aparna Mete Sawant, “Sign Language Detection using LSTM Deep Learning Model and Media Pipe Holistic Approach”, 2023 International Conference on Artificial Intelligence and Smart Communication(AISC),GreaterNoida,India,2023,pp.345-356, doi: 10.1109/AISC56616.2023.10085375

[4] Hamzah Luqman, “An Efficient Two-Stream Network for Isolated Sign Language Recognition UsingAccumulativeVideoMotion”,IEEEAccess(Volume:10),2022,pp.735-785,doi:10.1109/ACCESS.2022.3204110

[5] M. Al-Qurishi, T. Khalid and R. Souissi, "Deep Learning for Sign Language Recognition: Current Techniques Benchmarks and Open Issues", IEEE Access, vol. 9, pp. 126917-126951, 2021.

[6] Nishi Intwala, Arkav Banerjee, Meenakshi and Nikhil Gala, "Indian Sign Language converter using Convolutional Neural Networks", 2019 5th International Conference for Convergence in Technology (I2CT), Mar 29-31, 2019.

[7] R Rumana, Reddygari Sandhya Rani and Mrs. R. Prema, "Real-time Vernacular Sign Language Recognition using MediaPipe and Machine Learning", International Journal of Engineering Research & Technology (IJERT), vol. 10, October 2021.

[8]

[9] W. Safat, S. Asghar and S. A. Gillani, "Empirical Analysis for Crime Prediction and Forecasting Using Machine Learning and Deep Learning Techniques," in IEEE Access, vol. 9, pp. 70080-70094, 2021, doi: 10.1109/ACCESS.2021.3078117.

[10] S. Kim, P. Joshi, P. S. Kalsi and P. Taheri, "Crime Analysis Through Machine Learning," 2018 IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON), Vancouver, BC, Canada, 2018, pp. 415-420, doi: 10.1109/IEMCON.2018.8614828.

[11] S. Yadav, M. Timbadia, A. Yadav, R. Vishwakarma and N. Yadav, "Crime pattern detection, analysis & prediction," 2017 International conference of Electronics, Communication and Aerospace Technology (ICECA), Coimbatore, India, 2017, pp. 225-230, doi: 10.1109/ICECA.2017.8203676.

[12] A. Joshi, A. S. Sabitha and T. Choudhury, "Crime Analysis Using K-Means Clustering," 2017 3rd International Conference on Computational Intelligence and Networks (CINE), Odisha, India, 2017, pp. 33-39, doi: 10.1109/CINE.2017.23.

[13] M. Hassan and M. Z. Rahman, "Crime news analysis: Location and story detection," 2017 20th International Conference of Computer and Information Technology (ICCIT), Dhaka, Bangladesh, 2017, pp. 1-6, doi: 10.1109/ICCITECHN.2017.8281798.

[14] C. Chauhan and S. Sehgal, "A review: Crime analysis using data mining techniques and algorithms," 2017 International Conference on Computing, Communication and Automation (ICCCA), Greater Noida, India, 2017, pp. 21-25, doi: 10.1109/CCAA.2017.8229823.

[15] S. G. Krishnendu, P. P. Lakshmi and L. Nitha, "Crime Analysis and Prediction using Optimized K-Means Algorithm," 2020 Fourth International Conference on Computing Methodologies and Communication (ICCMC), Erode, India, 2020, pp. 915-918, doi: 10.1109/ICCMC48092.2020.ICCMC-000169.

[16] A. alqahtani, A. Garima and A. Alaiad, "Crime Analysis in Chicago City," 2019 10th International Conference on Information and Communication Systems (ICICS), Irbid, Jordan, 2019, pp. 166-172, doi: 10.1109/IACS.2019.8809142.

[17] A. Mary Shermila, A. B. Bellarmine and N. Santiago, "Crime Data Analysis and Prediction of Perpetrator Identity Using Machine Learning Approach," 2018 2nd International Conference on Trends in Electronics and Informatics (ICOEI), Tirunelveli, India, 2018, pp. 107-114, doi: 10.1109/ICOEI.2018.8553904.

[18]https://data.cityofchicago.org/Public-Safety/Crimes-2021/dwme-t96c

APPENDIX-I

Plagiarism Report