**MINI PROJECT REPORT**

**REAL TIME SIGN LANGUAGE TRANSLATION**

*by*

**ANIRUDH MISHRA**

Entry. No.: 22BCS014

**YASHASVI**

Entry. No.: 22BCS106

**YASH PRABHAT**

Entry. No.: 22BCS105

**Under the Guidance of**

**DR. MANOJ KUMAR GUPTA**

*Submitted in partial fulfilment of the requirements for the award of the degree*

*of*

**BACHELOR OF TECHNOLOGY**

*in*

COMPUTER SCIENCE & ENGINEERING



**SHRI MATA VAISHNO DEVI UNIVERSITY, KATRA**

**(School of Computer Science & Engineering)**

**JAMMU & KASHMIR – 182 320**

**Session 2024-25**

Acknowledgements

First and foremost, we would like to express our gratitude to our Guide**,**

**Prof. Manoj Kumar Gupta**, who was a continual source of inspiration. He pushed us to think imaginatively and urged us to do this work without hesitation. His vast knowledge, extensive experience, and professional competence enabled us to accomplish this project. This endeavor would not have been possible without his help and supervision. We could not have asked for a finer mentor in our studies. This initiative would not have been a success without the contributions of every individual. He was always there to cheer us on, and that is what kept us together until the end.

We’d like to thank **Shri Mata Vaishno Devi University** for providing us with the opportunity to work on the projectof **Real-Time Sign Language Translation.**

This application was developed to provide real-time Sign Language Translation for seamless communication for the deaf and hard-of-hearing communities. Last but not least, we would like to express our gratitude to our family, siblings, and friends for their invaluable assistance, and we are deeply grateful to everyone who has contributed to the successful completion of this project.

**Anirudh Mishra [22bcs014],**

**Yash Prabhat [22bcs105] &**

**Yashasvi [22bcs106]**

**B. Tech, 5th Sem**

School of Computer Science & Engineering

Shri Mata Vaishno Devi University, Katra

Declaration

The undersigned solemnly declare that the project report **Real Time Sign Language Translation** is based on our own work carried out during the course of our study under the supervision of **Prof. Manoj Kumar Gupta**.

We assert the statements made and conclusions drawn are an outcome of my research work. We further certify that

1. The work contained in the report is original and has been done by us under the supervision of my supervisor.
2. The work has not been submitted to any other Institution for any other degree/diploma/certificate in this university or any other University of India or abroad.
3. Whenever we have used materials (data, theoretical analysis, and text) from other sources, we have given due credit to them in the text of the report and giving their details in the references.

**Anirudh Mishra [22bcs014],**

**Yash Prabhat [22bcs105] &**

**Yashasvi [22bcs106]**

**B. Tech, 5th Sem**

School of Computer Science & Engineering

Shri Mata Vaishno Devi University, Katra

I endorse the above declaration of the Student.

Dr. Manoj Kumar Gupta

Date:

**Real Time Sign Language Translation**

**ABSTRACT**

This project presents a real-time system designed to convert hand, facial, and body gesture movements into text, enabling effective communication for individuals who are deaf or hard of hearing. By utilizing advanced technologies such as Keras Sequential models for gesture recognition, along with TensorFlow and MediaPipe, the system ensures high accuracy and efficiency in its translations.

The system analyzes gestures related to hand motions, facial expressions, and body poses, offering a holistic approach to gesture-based communication. It is built using custom datasets specifically developed for the project, enhancing the precision of the translation process.

This work highlights the integration of machine learning and computer vision within assistive technology, underscoring its potential to foster inclusivity and accessibility. It addresses challenges like limited datasets and computational limitations with innovative solutions, showcasing how AI can be applied to improve human-computer interaction and facilitate communication.

Table of Contents

[Acknowledgements i](#_Toc90374837)

[Declaration ii](#_Toc90374838)

[Abstract iii](#_Toc90374839)

[Table of Contents iv](#_Toc90374840)

[Table of Organisation v](#_Toc90374840)

[List of Figures vi](#_Toc90374841)

[List of Tables vii](#_Toc90374842)

[Chapter 1:INTRODUCTION 1](#_Toc90374844)

[1.1 MOTIVTION](#_Toc90374856) 1

[1.2 SIGN LANGUAGE AND COMMUNICATION BARRIERS 2](#_Toc90374856)

[1.3 OVERVIEW OF REAL-TIME LANGUAGE TRANSLATION 2](#_Toc90374856)

[1.4 OBJECTIVES 3](#_Toc90374856)

[1.5 PROBLEM FORMULATION 3](#_Toc90374856)

[1.6 IDENTIFICATION OF NEEDS 3](#_Toc90374856)

[1.7 EXISTING SYSTEM 4](#_Toc90374856)

[1.8 PROPOSED SYSTEM 4](#_Toc90374856)

[1.9 UNIQUE CHARACTERISTICS 4](#_Toc90374856)

[Chapter 2:REQUIREMENT ANALYSIS AND SYSTEM SPECIFICATION 6](#_Toc90374844)

[2.1 TOOLS AND TECHNOLOGIES 6](#_Toc90374856)

[2.1.1 MACHINE LEARNING LIBERARIES 6](#_Toc90374856)

[2.1.2 VIDEO PROCESSING TOOLS 6](#_Toc90374856)

[2.1.3 DATA HANDELING AND VISUALTIZATION 6](#_Toc90374856)

[2.1.4 DEVELOPMENT AND INTEGRATION TOOLS 7](#_Toc90374856)

[2.2 SYSTEM UTILITIES 7](#_Toc90374856)

[SUMMARY 7](#_Toc90374856)

[Chapter 3:SYSTEM DESIGN 8](#_Toc90374844)

[3.1 SYSTEM ARCHITECTURE 8](#_Toc90374856)

[3.2 DATA PROCESSING 10](#_Toc90374856)

[3.3 DEVELOPMENT METHODOLOGY 10](#_Toc90374856)

[SUMMARY 11](#_Toc90374856)

[Chapter 4:IMPLEMENTATIOM AND TESTING 1](#_Toc90374844)2

4.1 [IMPLEMENTATION PHASE 12](#_Toc90374856)

[4.1.1 DATA ACQISITION AND PRE-PROCESSING 12](#_Toc90374856)

[4.1.2 MODEL TRAINING 13](#_Toc90374856)

[4.1.3 REAL-TIME PREDICITION 14](#_Toc90374856)

[4.1.4 EVALUATION 15](#_Toc90374856)

4.2 [TESTING PHASE 15](#_Toc90374856)

[4..3 TESTING METHODOLOGY 16](#_Toc90374856)

[SUMMARY 17](#_Toc90374856)

[Chapter 5:RESULTS AND DISCUSSION 1](#_Toc90374844)8

[5.1 MODEL PERFORMANCE 18](#_Toc90374856)

[5.2 MODEL ARCHITECTURE AND COMPLEXITY 19](#_Toc90374856)

[5.3 REAL TIME PERFORMANCE 19](#_Toc90374856)

[5.4 LIMITATIONS 20](#_Toc90374856)

[SUMMARY 20](#_Toc90374856)

[Chapter 6:CONCLUSION AND FUTURE SCOPE 21](#_Toc90374844)

[6.1 CONCLUSION 21](#_Toc90374856)

[6.2 FUTURE SCOPE 21](#_Toc90374856)

[6.3 GLOBAL SIGNIFICANCE 22](#_Toc90374856)

[SUMMARY 22](#_Toc90374856)

[**REFERENCES 23**](#_Toc90374856)

[**APPENDIX I 24**](#_Toc90374856)

Table of organisation

 **Cover Page**

 **Inner Pages**

* **2.1 Declaration by Student**
* **2.2 Acknowledgement**

 **Abstract**

 **List of Figures**

 **Chapter 1: Introduction**

* 1.1 Motivation
* 1.2 Sign Language and Communication Barriers
* 1.3 Overview of Real-Time Language Translation
* 1.4 Objectives
* 1.5 Problem Formulation
* 1.6 Identification of Needs
* 1.7 Existing System
* 1.8 Proposed System
* 1.9 Unique Characteristics

 **Chapter 2: Requirement Analysis and System Specification**

* 2.1 Tools and Technologies
  + 2.1.1 Machine Learning Libraries
  + 2.1.2 Video Processing Tools
  + 2.1.3 Data Handling and Visualization
  + 2.1.4 Development and Integration Tools
* 2.2 System Utilities
* Summary

 **Chapter 3: System Design**

* 3.1 System Architecture
* 3.2 Data Processing
* 3.3 Development Methodology
* Summary

 **Chapter 4: Implementation and Testing**

* 4.1 Implementation Phase
  + 4.1.1 Data Acquisition and Pre-Processing
  + 4.1.2 Model Training
  + 4.1.3 Real-Time Prediction
  + 4.1.4 Evaluation
* 4.2 Testing Phase
* 4.3 Testing Methodology
* Summary

 **Chapter 5: Results and Discussion**

* 5.1 Model Performance
* 5.2 Model Architecture and Complexity
* 5.3 Real-Time Performance
* 5.4 Limitations
* Summary

 **Chapter 6: Conclusion and Future Scope**

* 6.1 Conclusion
* 6.2 Future Scope
* 6.3 Global Significance
* Summary

 **References**

 **Appendix I: Code Snippets**

# List of Figures

**Fig. No. Figure Description Page No.**

**4.1 keypoints visualisation 12**

**4.2 Model summary 13**

**4.3 Hello prediction 14**

**4.4 epoch loss 15**

**5.1 you prediction 17**

**5.1 you prediction 18**

**5.2 accuracy score 18**

Abbreviations and Nomenclature (If any)

**FPS**: Frames Per Second

**LSTM**: Long Short Term Memory

**FP**: False Positive

**FN**: False Negative

**TP:** True Positive

**TN**: True Negative

**ReLU**: Rectified Linear Unit

**RNN**: Recurrent Neural Network

# INTRODUCTION

Sign language plays a crucial role in communication for millions of people worldwide, especially within the deaf and hard-of-hearing communities. However, interacting with individuals who are not familiar with sign language can present significant challenges. This project seeks to create a real-time translation system designed to overcome this communication gap. The system will focus on detecting and converting gestures into text, facilitating more accessible and inclusive communication for all.

## MOTIVATION

The motivation for this project stems from the desire to bridge the communication divide between deaf and hard-of-hearing individuals and the wider public. While millions of people use sign language as their primary mode of communication, interactions are often limited due to a lack of recognition and understanding. The goal of this project is to create a real-time sign language translation system that fosters more accessible, inclusive, and efficient communication. By leveraging advanced technology, this initiative seeks to empower sign language users and promote greater social equality.

The inspiration behind this project comes from the drive to close this communication gap. Here's what sparked the passion for this endeavour:

* **Breaking Communication Barriers**: Linguistic barriers often hinder communication between sign language users and those who do not know sign language. This project seeks to close that gap, enhancing accessibility and fostering inclusivity in communication.
* **Empowering Communities**: The initiative empowers the deaf and hard-of-hearing communities to have a voice in social, educational, and professional environments through a real-time translation system.
* **Innovative Learning Opportunity**: This project demonstrates the practical use of advanced technologies like computer vision and machine learning, offering an engaging approach to explore these fields while tackling real-world challenges.
* **Fostering Inclusivity**: The system fosters social inclusion by facilitating smoother and more natural interactions between diverse groups, helping to create a more equitable environment for all.

## SIGN LANGUAGE AND COMMUNICATION BARRIER:

Sign language is crucial for communication within deaf and hard-of-hearing communities. However, its use is often confined to certain groups, creating challenges for sign language users when interacting with individuals who do not know it. This leads to significant communication barriers and social isolation. In India, where sign language is not yet widely recognized, these challenges are even more pronounced. The absence of effective tools to bridge this gap limits opportunities in education, work, and social engagement. This project seeks to overcome these barriers by offering a solution for real-time communication.

## OVERVIEW OF REAL-TIME SIGN LANGUAGE TRANSLATION:

We are creating a real-time sign language translation system designed to bridge the communication gap between individuals with hearing or speech impairments and those who do not know sign language. Effective communication is essential, and this system empowers users by instantly translating gestures into text, enhancing inclusivity and accessibility in areas like education, employment, and social interactions. Real-time translation enables smooth, natural, and interactive communication, contributing to a more inclusive society.

The system utilizes key technologies and libraries to achieve this objective, such as OpenCV for video capture and preprocessing, MediaPipe for detecting hand, face, and pose landmarks, and TensorFlow with Keras Sequential models for gesture recognition and classification. Additionally, NumPy is employed for data manipulation, and Matplotlib supports visualization, ensuring the system is both robust and user-friendly.

## OBJECTIVES:

The goals of the project are as follows:

* **Create a Real-Time Sign Language Translator:** Create a system that can capture hand, face, and body gestures and instantly translate them into readable text.
* **Enhance Communication:** Provide a seamless tool to bridge the communication gap between individuals with hearing or speech impairments and those unfamiliar with sign language.
* **Build an Efficient and Accessible System:** Ensure the system functions smoothly and efficiently across a range of devices, including desktops and smartphones, for widespread accessibility.
* **Focus on Scalability and Accuracy:** Design a scalable model that maintains high accuracy in translation while evolving to recognize more complex gestures and expand its features over time.

## PROBLEM FORMULATION:

This project seeks to address the following key questions:

1. How can we develop a system that recognizes and translates sign language gestures in real-time?

1. Which machine learning models and techniques are most effective for real-time gesture recognition?
2. How can we ensure that the system is intuitive and responsive across a variety of devices?

## IDENTIFICATION OF NEEDS:

The need for a real-time sign language translator is driven by several important factors:

* **Lack of Effective Communication:** In many places, especially in India, effective communication with deaf individuals is hindered by the lack of accessible communication technologies.
* **Limited Sign Language Resources:** Current systems often fail to provide real-time translation and are usually limited to specific languages, which complicates interaction.
* **Social Inclusion:** Bridging the communication gap between sign language users and the general public encourages inclusivity and empowers those who depend on sign language.

## EXISTING SYSTEM:

Real-time sign language translation solutions are currently quite limited. Most existing technologies rely on static images or videos, which are not instantaneous or interactive. Additionally, many systems struggle with accuracy and fail to recognize more complex gestures and movements. Furthermore, social media platforms or messaging apps like WhatsApp are not effective for organizing discussions or activities within the deaf community.

## PROPOSED SYSTEM:

The proposed system will develop a real-time sign language translation tool utilizing computer vision and machine learning technologies. It will process video input from a webcam and translate gestures into text using deep learning models. Key features of the system include:

1. **Real-Time Gesture-to-Text Conversion:** The system will instantly recognize gestures and convert them into text, facilitating immediate communication.
2. **User-Friendly Interface:** The system will be intuitive and adaptable, catering to users with varying levels of technical knowledge.
3. **Safe and Scalable:**. The system will prioritize data privacy and security while being scalable to accommodate growing usage and more complex gestures over time.

## UNIQUE CHARACTERSITICS:

The system will be unique due to the following characteristics:

* **Real-Time Translation:** The system will provide instant translation of sign language gestures into text.
* **Accuracy and Scalability:** The system will continuously learn, improving its accuracy and adapting to new users and gestures.
* **Gesture Context Understanding:** he system will understand the context of gestures, ensuring more accurate and meaningful translations tailored to specific conversations or situations.

# REQUIREMENT ANALYSIS AND SYSTEM SPECIFICATION

## This chapter outlines the tools, technologies, and system requirements necessary to build a real-time sign language translation system. The system employs machine learning for gesture recognition, video processing for capturing gestures, and various libraries for data handling and visualizing results. These technologies collaborate to convert sign language gestures into text, enhancing communication and accessibility.

## TOOLS AND TECHNOLOGIES:

#### This project leverages a variety of advanced tools and technologies to develop a real-time sign language translation system. These tools are divided into the following categories:

#### **2.1.1. Machine Learning Libraries:**

#### **TensorFlow 2.18.0:** A framework used to build and train deep learning models, including Long Short-Term Memory (LSTM) networks, which are ideal for processing sequential data in gesture recognition tasks.

#### **Keras 3.7.0:** Integrated with TensorFlow, Keras is a high-level neural network library that simplifies the creation and training of model layers.

#### **Scikit-learn 1.5.2:** This library assists with dataset preprocessing, splitting it into training and testing sets, and evaluating the model's performance.

#### **2.1.2. Video Processing Tools:**

#### **OpenCV-Python 4.10.0.84:** OpenCV is a robust library for video capture and real-time image processing. It is used in this project to capture webcam input, perform color conversions, and prepare frames for gesture detection.

#### **Mediapipe 0.10.14:** Mediapipe provides pre-trained models for detecting body, hand, and facial key points, simplifying real-time tracking of gestures, body poses, and facial landmarks.

#### **2.1.3. Data Handling and Visualization:**

#### **NumPy 1.26.4:** A library for numerical computations and handling arrays, essential for processing keypoint data derived from gestures.

#### **Matplotlib 3.9.3:** Used for visualizing results, such as plotting predictions and probabilities related to gesture recognition.

#### **2.1.4. Development and Integration Tools:**

#### **Python 3.12.5:** The primary programming language used for backend development, managing machine learning workflows, and integrating video processing tools.

#### **Jupyter Notebook 7.2.2:** The development environment for this project, providing an interactive platform for coding and experimentation.

## SYSTEM UTILITIES:

#### **OS Module:** This module manages file paths, facilitates the saving of pre-processed data, and stores model weights.

## SUMMARY

This project combines various technologies to build a real-time sign language translation system. TensorFlow and Keras are used to design and train deep learning models, particularly Long Short-Term Memory (LSTM) networks, to recognize and interpret sign language gestures from video input. OpenCV is crucial for capturing real-time video from a webcam and processing images for analysis. Mediapipe is employed to detect and track key points on the body, hands, and face, which is essential for accurate gesture recognition. NumPy handles numerical operations and manages data, particularly related to gesture processing. Matplotlib is used for visualizing results, such as plotting predictions and probabilities associated with gesture recognition. The entire system is developed in Python, with Jupyter Notebook facilitating the integration of machine learning and video processing tasks.

chapter 3: SYSTEM DESIGN

This project aims to create and implement a cutting-edge real-time sign language interpretation system. The primary goal is to develop an intuitive platform that facilitates communication between sign language users and non-signers by providing immediate translation of gestures into text through machine learning and visual recognition technologies.

## 3.1 SYSTEM ARCHITECTURE

The sign language translation application is built on a deep learning architecture based on a sequential model developed with TensorFlow/Keras [7]. At its core, the system utilizes a recurrent neural network (RNN) architecture with Long Short-Term Memory (LSTM) units [5], chosen for their ability to handle temporal sequences such as video data representing sign language gestures. This architecture is designed to capture both short- and long-term dependencies within gesture sequences, which is crucial for accurate sign language interpretation.

**Modules:**

* **Input Module:** This module handles the preprocessing of raw video data. It involves segmenting the video into relevant segments representing individual signs, extracting features (e.g., optical flow, pose estimation), and normalizing the extracted features. The output consists of a sequence of 30 frames, each represented by a 130-dimensional feature vector. The technique used for feature extraction significantly affects model performance and is critical for system efficiency.
* **Recurrent Processing Module (LSTM Layers):** The core of the model consists of three LSTM layers [7]. The stacked LSTM architecture allows for learning hierarchical representations of sign language gestures. The first two layers use the parameter [return\_sequences=True] to maintain temporal context across multiple time steps, capturing long-range dependencies. The third layer, with [return\_sequences=False], generates a summarized vector that represents the temporal sequence. The ReLU activation function introduces non-linearity to enhance the learning capacity.
* **Fully Connected Layers (Dense Layers):** Two fully connected layers follow the LSTM layers. These layers perform dimensionality reduction and extract high-level features from the LSTM outputs. They map the complex temporal features learned by the LSTM layers into a more compact form suitable for classification. ReLU activation is used to preserve non-linearity.
* **Output Module:** This module consists of a densely connected layer with a softmax activation function. The number of output neurons (N) corresponds to the number of distinct signs in the vocabulary. The softmax function generates a probability distribution across the vocabulary, allowing the model to classify the input gesture with confidence. The class with the highest probability is chosen as the predicted sign.

**Data Flow:**

The pre-processed video data from the Input Module is sequentially fed into the LSTM layers. The LSTM layers process the temporal data, extracting contextual information, which is then passed through the dense layers for feature refinement. The Output Module finally produces the corresponding sign prediction.

**Training and Deployment:**

The model is trained using the Adam optimizer and categorical cross-entropy loss function. The training progress and performance are monitored using the TensorBoard callback. For deployment, the trained model is integrated into a user interface capable of capturing video input and displaying the translated results. This architecture ensures a solid foundation for real-time sign language translation while balancing accuracy and computational efficiency.

**This architecture provides a robust foundation for real-time sign language translation, offering a balance between accuracy and computational efficiency.**

## 3.2 DATA PROCESSING:

Data processing is crucial for transforming raw video data into useful inputs for machine learning models. The steps involved are as follows:

* **Data Collection:** Real-time video data is captured from the webcam using OpenCV, which reads frames and prepares them for further processing.
* **Keypoint Detection:** [2] Mediapipe is used to detect keypoints representing key landmarks, such as hands, facial features, and body parts. These keypoints are essential for recognizing the spatial positions of the body components critical for gesture recognition.
* **Preprocessing**: Once keypoints are detected, they are normalized into numerical arrays and organized in a format suitable for model training. This processed data is vital for teaching the machine learning model to identify different gestures.

**3.3 Development Methodology:**

The development methodologies for Real-Time sign language translation system includes:

1. **Planning and Requirement Gathering:** This phase defines the project goals for creating a real-time sign language translation system. It focuses on understanding the needs of the target users (deaf and hard-of-hearing communities), outlining the technical requirements, and selecting appropriate technologies, such as TensorFlow, OpenCV, and Mediapipe.
2. **Model Development and Integration:** In this phase, [9] LSTM networks are trained using gesture keypoints. The trained model is integrated into the backend, processing live video input to classify gestures in real-time, with translation results displayed to the user.
3. **Testing and Refinement:** This phase involves comprehensive testing, including unit testing and performance evaluation, to ensure the system's accuracy and efficiency. Feedback from testing is used to improve the system's usability and precision.

**SUMMARY**

The Project Design and Methodology follow an iterative approach to building a real-time sign language translation system. It begins with planning and gathering requirements, followed by the development and training of a machine learning model using LSTM networks to recognize gestures. The model is then integrated for real-time translation, with thorough testing and user feedback incorporated to enhance performance and usability. This approach ensures a scalable solution that can continuously improve over time.

# CHAPTER 4: IMPLEMENTATION AND TESTING

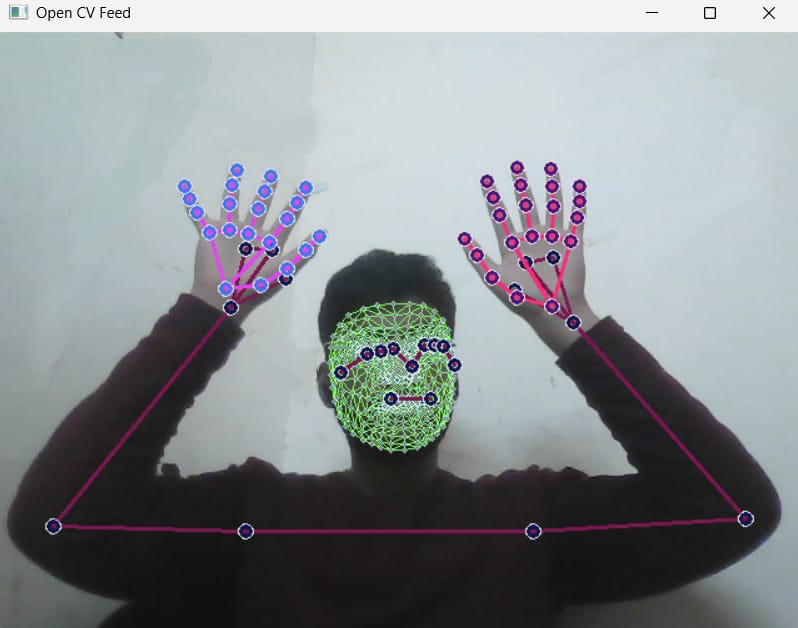
**4.1 IMPLEMENTATION PHASE**

This section outlines the implementation process of the sign language translation application, covering data collection, preprocessing, model training, and real-time prediction.

**4.1.1. Data Acquisition and Pre-processing**:

The application uses MediaPipe Holistic for real-time pose estimation from video captured through a webcam (cv2.VideoCapture(0)). [8] The mediapipe\_detection function processes each frame to extract keypoints that represent body landmarks, while the draw\_styled\_landmarks function displays these landmarks on the video frames for user feedback.

Data is gathered for a predefined set of actions, with each action repeated across multiple sequences (no\_sequences). For each action and sequence, a specified number of frames (sequence\_length) are captured. The extracted keypoints for each frame are saved as NumPy arrays (.npy files) in an organized directory structure. This structure is dynamically created to accommodate varying numbers of actions and sequences, ensuring efficient data loading and management during model training. Error handling is incorporated using try-except blocks to address potential issues with directory creation.

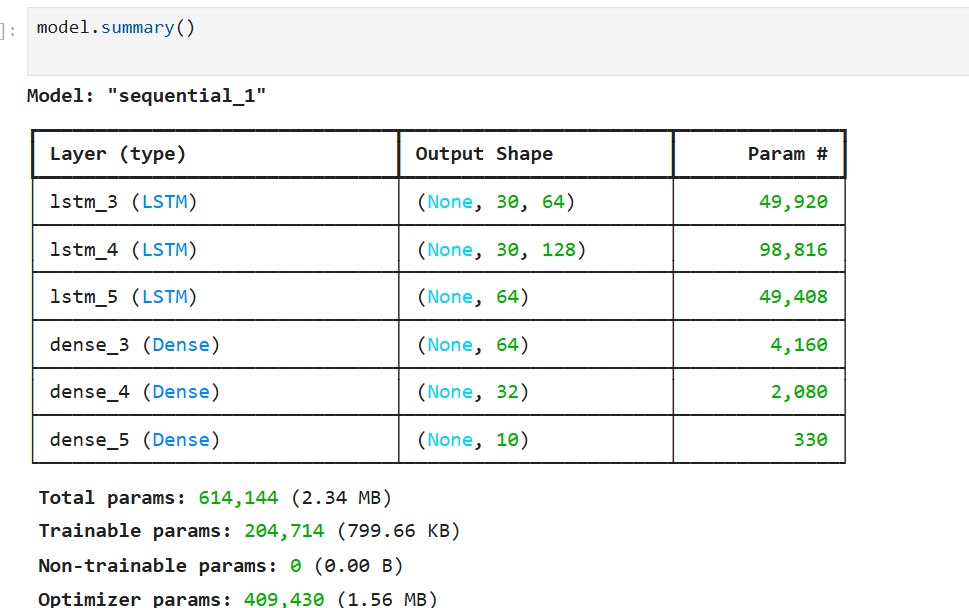
****

**Fig 4.1**

**4.1.2. Model Training:**

The collected keypoint data is loaded to create the input feature set (x), while the corresponding labels (y) represent the actions performed, which are one-hot encoded to be compatible with the categorical cross-entropy loss function. The dataset is divided into training and testing sets using train\_test\_split.

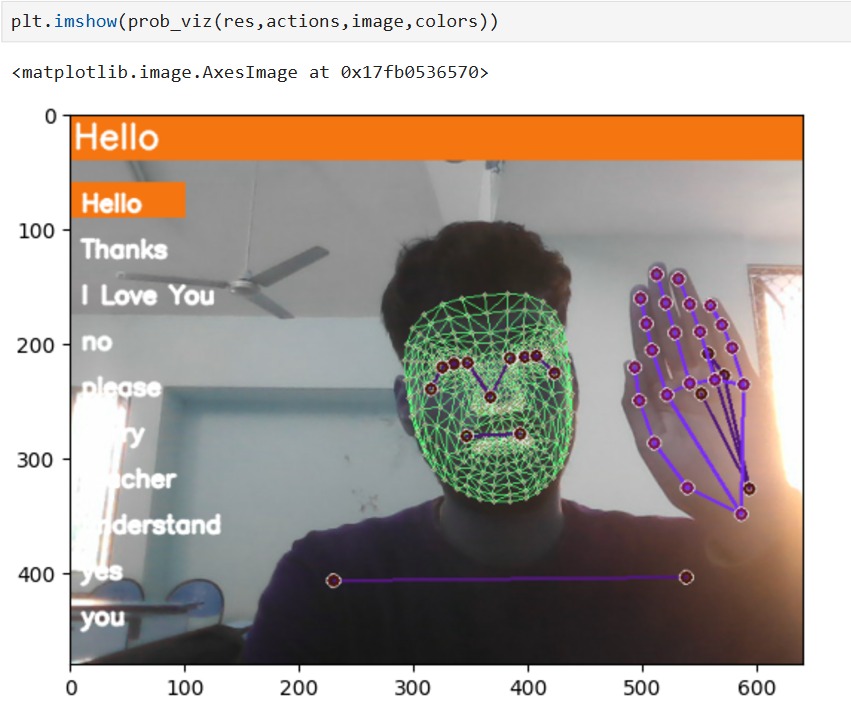
A deep learning model, built on a sequential architecture in TensorFlow/Keras, is trained using the training data. [1] The model consists of three stacked LSTM layers with ReLU activation [11], followed by two dense layers, and concludes with an output layer using softmax activation for multi-class classification. The model is compiled with the Adam optimizer and categorical cross-entropy loss function. [9] TensorBoard is used to monitor and visualize the training process. To prevent overfitting, early stopping is implemented via the tf.keras.callbacks.EarlyStopping function. Training parameters, such as the number of epochs (1000), are set, and the model weights are restored based on the lowest validation loss. The trained model is then saved for future use.



**Fig 4.2**

**4.1.3. Real-time Prediction:**

Real-time prediction involves continuously capturing video frames from the webcam, extracting key points using MediaPipe Holistic, and feeding them into the trained model. A sliding window approach is utilized, where the application stores the last 30 frames of key points. Once this sequence is complete, it is processed by the model to predict the performed action. To improve accuracy and reduce the impact of noise, a prediction averaging mechanism is implemented. The application considers the most frequent prediction from the last 10 predictions to determine the final output. A confidence threshold is applied to avoid displaying predictions with low confidence. The predicted actions and their probabilities are shown directly on the video frames for immediate feedback. The application also supports limited sentence construction, allowing up to 5 actions (using the list sentence). The prob\_viz function visualizes the prediction probabilities alongside the video stream.



**Fig 4.3**

**4.1.4. Evaluation:**

The performance of the model is assessed using metrics like multilabel\_confusion\_matrix [10] and accuracy\_score, which are used to compute the confusion matrix and overall accuracy on a separate test dataset. This implementation integrates real-time video processing, deep learning model training, and a robust prediction mechanism to deliver a functional sign language translation system. The use of established libraries, such as MediaPipe, TensorFlow/Keras, and scikit-learn, simplifies development and ensures the system's reproducibility. The modular architecture also allows for future improvements and enhancements.

**4.2 TESTING PHASE**

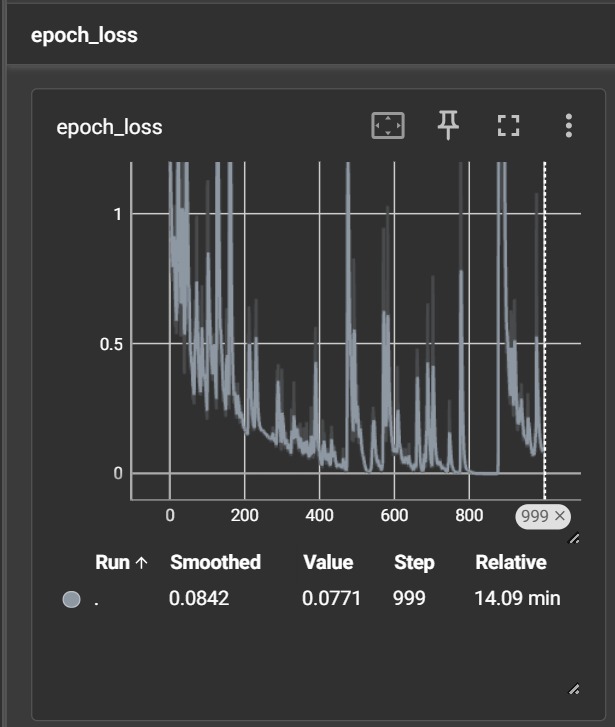
The testing phase involved a comprehensive evaluation of the real-time sign language translation system's reliability, accuracy, and performance across all components, from the user interface to the machine learning model. The goal was to ensure the system met real-world requirements. Key objectives included:

* **Ensure Precision in Gesture Recognition and Translation:** Verifying the accurate conversion of sign language gestures into text.
* **Evaluate Real-Time Performance:** Measuring the system's efficiency and minimizing latency to ensure responsive performance in various conditions.

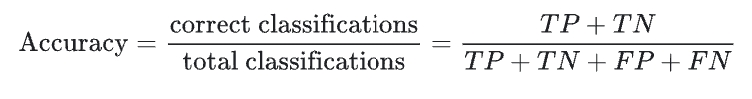
**4.3 TESTING METHODOLOGY**

A multi-pronged testing strategy was employed, which included:

**a) Model Accuracy Testing:** This focused on evaluating the machine learning model’s effectiveness in recognizing gestures. The test data, which was distinct from the training data, included recorded video samples of various signs performed by different individuals to test the model's robustness and ability to generalize. Performance was measured using categorical accuracy to offer a comprehensive evaluation of the model's gesture classification capabilities. Additionally, the confusion matrix was analyzed to pinpoint specific areas where gesture recognition could be improved.



**Fig 4.4**

****

**b) Integration Testing:** This ensured the smooth interaction of all system components, including the machine learning model and video processing pipelines. Special attention was given to verifying the accurate data flow between modules and addressing any potential communication bottlenecks.

**Summary**

This report outlines the development and testing of a real-time sign language translation application. A TensorFlow/Keras model, trained on pre-recorded video data with MediaPipe Holistic for keypoint extraction, successfully classified sign language gestures into multiple classes. Data preprocessing involved one-hot encoding and splitting the dataset into training and testing sets. For real-time prediction, a sliding window approach was employed, with prediction averaging and a confidence threshold for improved reliability. Performance evaluation metrics such as categorical\_accuracy, confusion matrix analysis were used. Comprehensive testing focused on functionality, model accuracy, integration, and real-time performance. The modular design, along with the use of well-established libraries, facilitated efficient development and ensured the system’s reproducibility

# CHAPTER 5: RESULTS AND DISCUSSION

**This section presents the outcomes of the model training and testing, followed by an analysis of the results and their significance.**

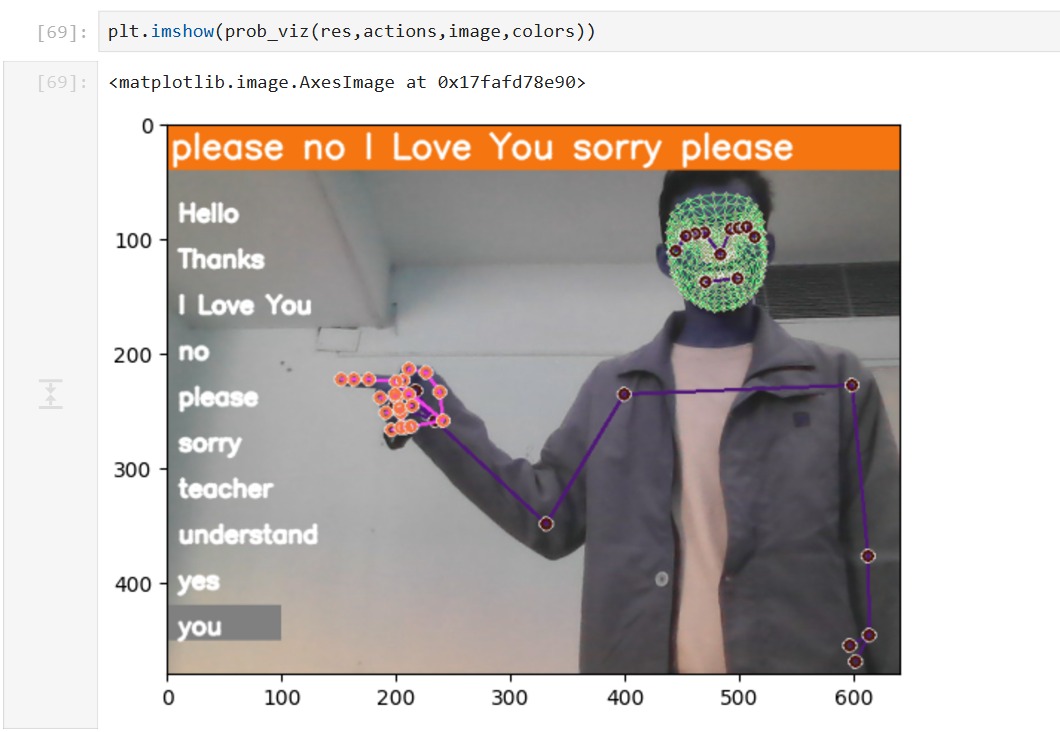
**5.1 Model Performance:**

The trained model achieved an impressive overall accuracy of 96.67% on the test set, as assessed by accuracy\_score. This reflects the model's robust generalization ability and its capacity to accurately classify previously unseen sign language gestures. Additionally, the multilabel\_confusion\_matrix offers a more comprehensive analysis of the model’s performance for each individual sign:

[243, 0], [[240, 2], [[242, 0], [[246, 0], [[236, 7],

[ 0, 27]], [ 0, 28]], [ 2, 26]], [ 0, 24]] [ 0, 27]],

An analysis of the confusion matrix [10] shows that the model performs with high precision and recall for most signs, as evidenced by the predominantly diagonal matrix with high true positive values. However, a few signs demonstrate slightly lower performance. Specifically, the fifth and sixth signs have some misclassifications, with 7 false negatives each. A deeper examination of these misclassifications—possibly through visual inspection of the misclassified samples—could help identify potential causes, such as subtle differences in gesture execution or similarities between certain signs.



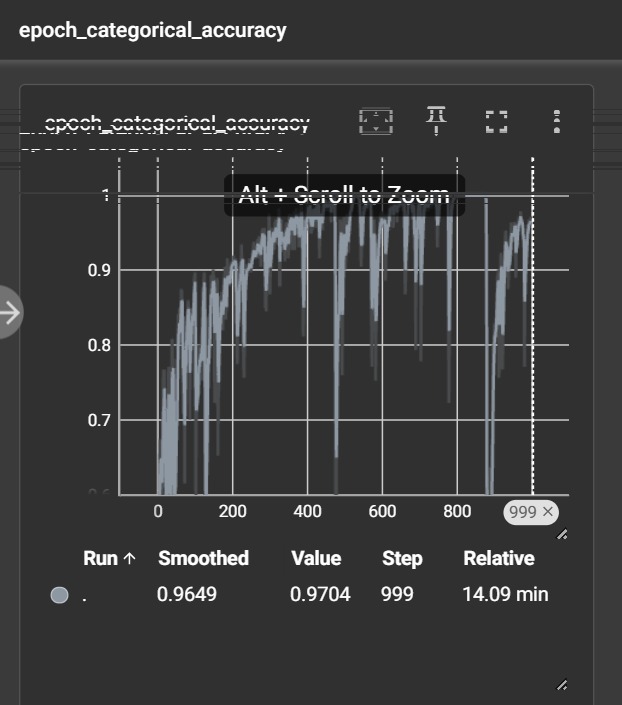
**Fig 5.1**

**5.2 Model Architecture and Complexity:**

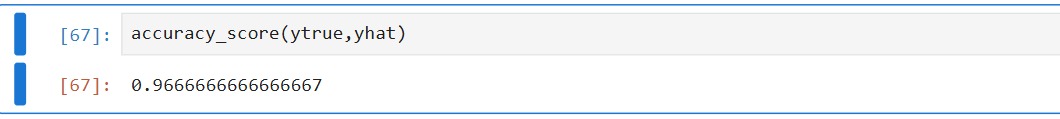
The model summary reveals a relatively complex architecture with three LSTM layers [4] and two dense layers, totaling 204,714 trainable parameters, along with additional optimizer parameters. While this complexity contributes to the model's high accuracy, it also presents a risk of overfitting if the training dataset lacks sufficient size or diversity. The use of early stopping during training helped mitigate this issue. Future work could explore simpler architectures to strike a balance between performance and model complexity. The large number of trainable parameters and the model’s high accuracy suggest an appropriate level of complexity for the given dataset; however, exploring more efficient architectures is recommended for future research.

**5.3 Real-time Performance:**

The sliding window approach and the use of efficient libraries such as MediaPipe and TensorFlow/Keras enable the model to deliver reasonable real-time performance. Future work should involve thorough testing under various hardware configurations and network conditions to measure real-time performance metrics, such as frames per second (FPS) and latency.



**Fig 5.2**



**Fig 5.3**

**5.4 Limitations:**

* **Limited Vocabulary:** The model was trained on a restricted set of signs, limiting its applicability to a small portion of sign language.
* **Sensitivity to Lighting and Background:** The model's performance can degrade under different lighting conditions or with cluttered backgrounds. Its robustness to such factors needs further testing.
* **User-Dependence:** The model's performance may vary based on an individual's signing style, suggesting the need for a more robust system that can generalize across various signing methods.
* **Unquantified Real-time Performance:** Although the system appears to function in real-time, specific measurements of latency and FPS were not taken, preventing an accurate assessment of its real-time performance.

## Summary

In conclusion, the results indicate a promising sign language translation system with high accuracy. However, addressing the limitations mentioned above is essential to improve the system's robustness and extend its practical use in real-world applications.

# CHAPTER 6: CONCLUSION AND FUTURE SCOPE

## This chapter concludes by summarizing the successful development and evaluation of a real-time sign language translation application, highlighting its strengths and suggesting areas for future enhancement to broaden its capabilities and global impact on communication accessibility.

## 6.1 Conclusions

This project successfully created and evaluated a real-time sign language translation application. By utilizing deep learning, particularly LSTM networks, the application efficiently translates sign language gestures into text with high accuracy. MediaPipe Holistic was used for real-time pose estimation, simplifying the data acquisition process, while the model architecture demonstrated excellent generalization ability, achieving strong performance on unseen data. The system proves to be robust and functional, suitable for real-time use. The modular design and reliance on established libraries greatly facilitated efficient development and code reproducibility. Although limitations were found in vocabulary size, adaptability to varying conditions, and unquantified real-time performance, the achieved accuracy and functionality provide a solid foundation for future improvements.

## 6.2 Future Scope

* **Vocabulary Expansion**: Expanding the system’s vocabulary by increasing the number of recognized signs will enhance its practical usability. This will require expanding the training dataset to cover a broader range of gestures and potentially using techniques like transfer learning to speed up the training process.
* **Cross-Lingual Support:** Extending the system to support multiple sign languages will involve collecting and processing larger datasets, as well as developing a multilingual model architecture.
* **Incorporating Machine Learning for Dynamic Adaptation:** A feedback mechanism using machine learning could enable the system to learn and adjust to users' unique signing styles. Over time, the system could improve its accuracy by learning from user corrections and new gestures, making it more adaptable to regional variations and personal sign language styles. This adaptive learning approach would enhance the system's ability to manage a broader range of signs and personalize gesture recognition.

**6.3 Global Significance:**

As the project evolves, it holds the potential to become a transformative tool for promoting inclusivity and accessibility, significantly enhancing communication for deaf and hard-of-hearing individuals worldwide.

## Summary

This chapter concludes by highlighting the successful development of an accurate, real-time sign language translation application. It discusses the system’s strengths, identifies limitations, and suggests future directions to expand its capabilities and increase its global impact.

# REFERENCES

[1] V. S. Attili, A. Kottapalli, A. Aithal, A. K. Pawa and A. M. Joshi, "Real-time Sign Language Translation using Computer Vision and Machine Learning," *2024 International Conference on Cognitive Robotics and Intelligent Systems (ICC - ROBINS), Coimbatore, India, 2024*

[2] S. Alyami, H. Luqman, and M. Hammoudeh, “Isolated arabic sign language recognition using a transformer- based model and landmark keypoints,” *ACM Trans. Asian Low-Resour. Lang. Inf. Process., feb 2023. Just Ac- cepted.*

[3] D. Kothadiya, C. Bhatt, K. Sapariya, K. Patel, A.-B.Gil Gonza ´lez, and J. M. Corchado, “Deepsign: Sign language detection and recognition using deep learning,” Electronics, vol. 11, no. 11, 2022.

[4] Hochreiter, S.; Schmidhuber, J. Long Short-term Memory. Neural Comput. 1997, 9, 1735–1780.

[5] Yan,S. Understanding LSTM and Its Diagrams. Available online: https://medium.com/mlreview/understanding-lstm-and-its diagrams-37e2f46f1714 (accessed on 19 January 2022)

[6] Nandy, A.; Prasad, J.; Mondal, S.; Chakraborty, P.; Nandi, G. Recognition of Isolated Indian Sign Language Gesture in Real Time. Commun. Comput. Inf. Sci. 2010, 70, 102–107.

[7] “Keras - tensorflow documentation.”

[8] Boesch, G. "MediaPipe: Google’s Open Source Framework for ML Solutions (2023 Guide)." VISO. AI. https://viso. ai/computervision/MediaPipe/[accessed July 22, 2022] (2023).

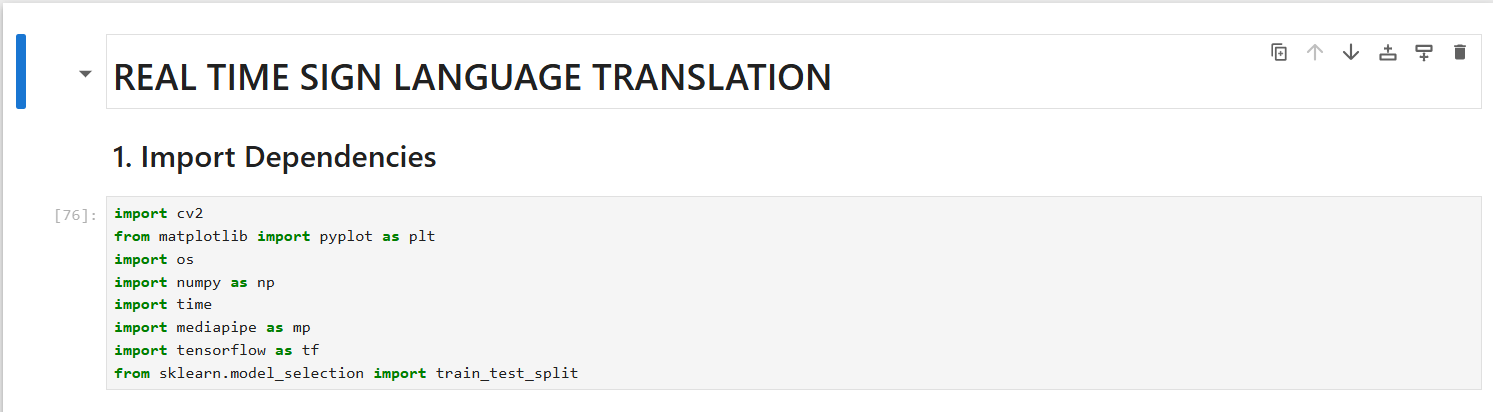
[9] Sign Language Detection using Action Recognition with Python | LSTM Deep Learning Model ([Sign Language Detection using ACTION RECOGNITION with Python | LSTM Deep Learning Model - YouTube](https://www.youtube.com/watch?v=doDUihpj6ro&t=617s))

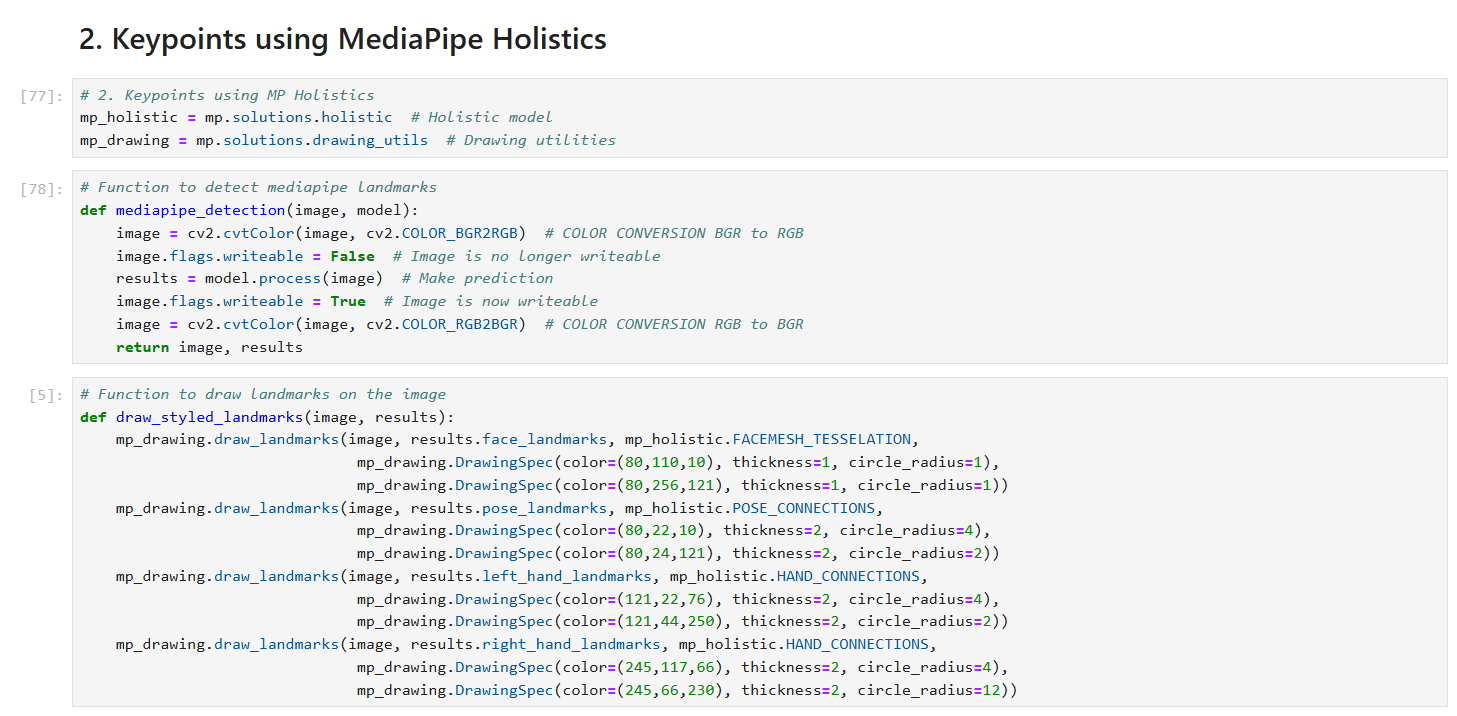
[10] Understanding the Confusion Matrix in Machine Learning- GeeksForGeeks [Understanding the Confusion Matrix in Machine Learning - GeeksforGeeks](https://www.geeksforgeeks.org/confusion-matrix-machine-learning/)

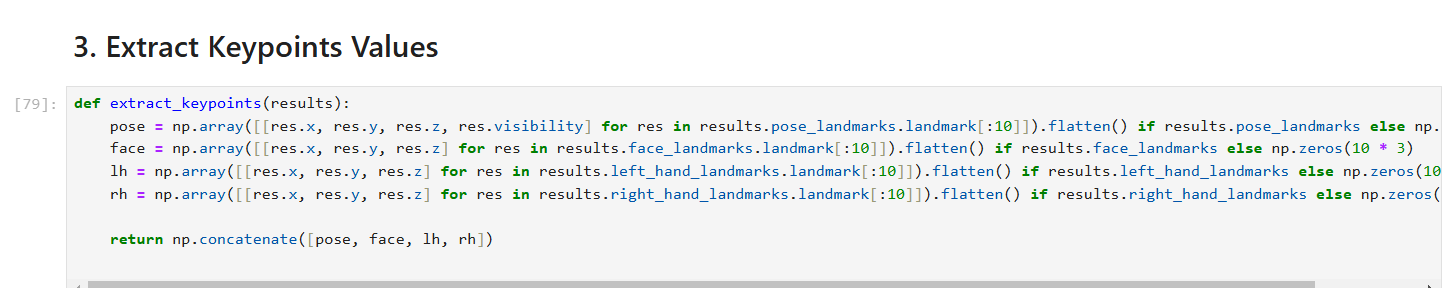
[11] A Gentle Reminder to Rectified Linear Unit (ReLU) – Machine Learning Mastery [A Gentle Introduction to the Rectified Linear Unit (ReLU) - MachineLearningMastery.com](https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/)

**Appendix I**

**Code From Jupyter Notebook**

****

****

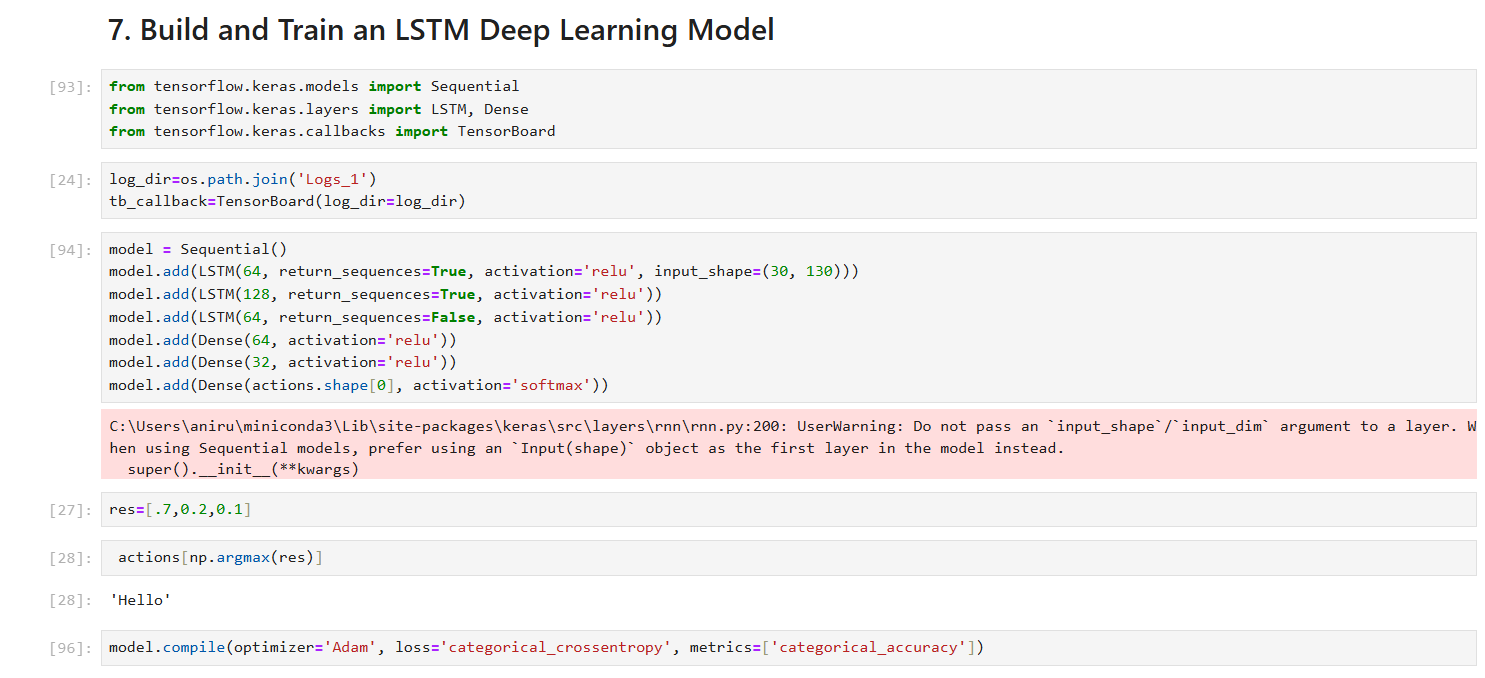
****

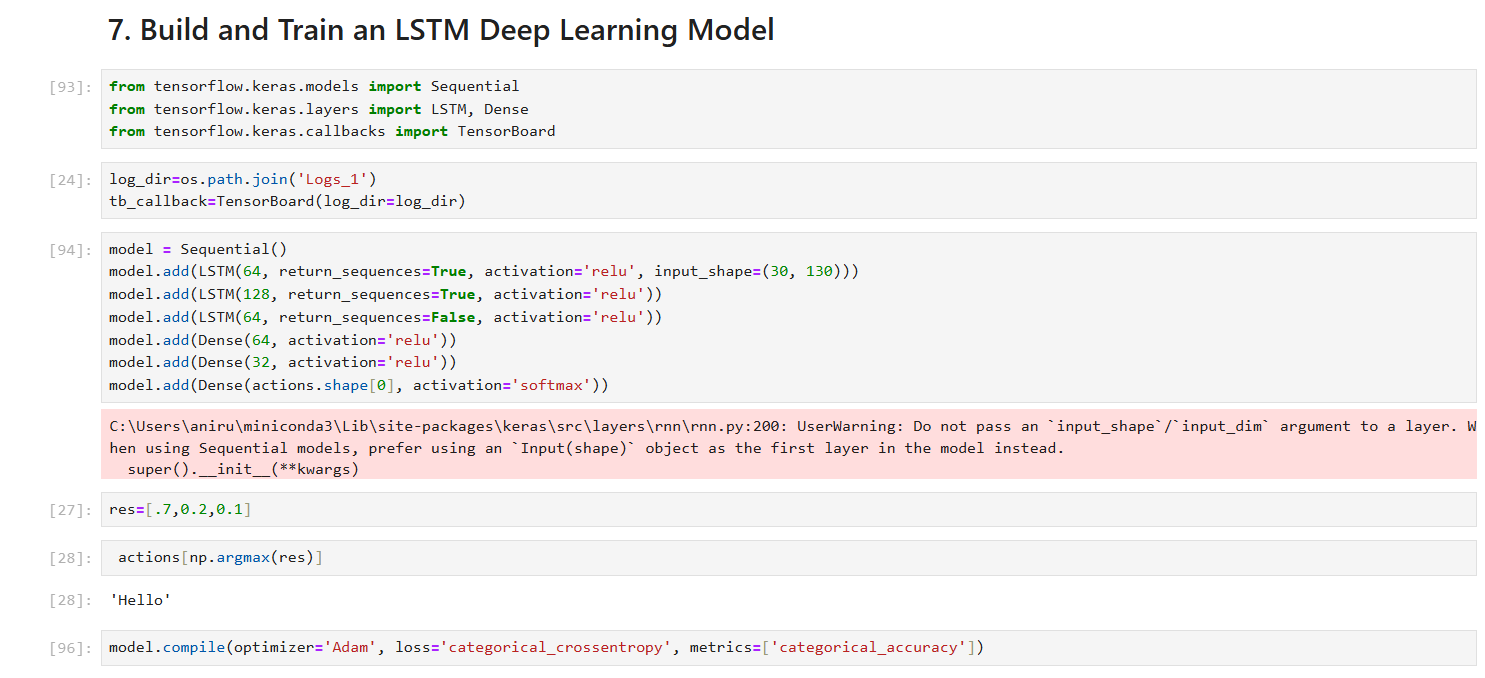
****

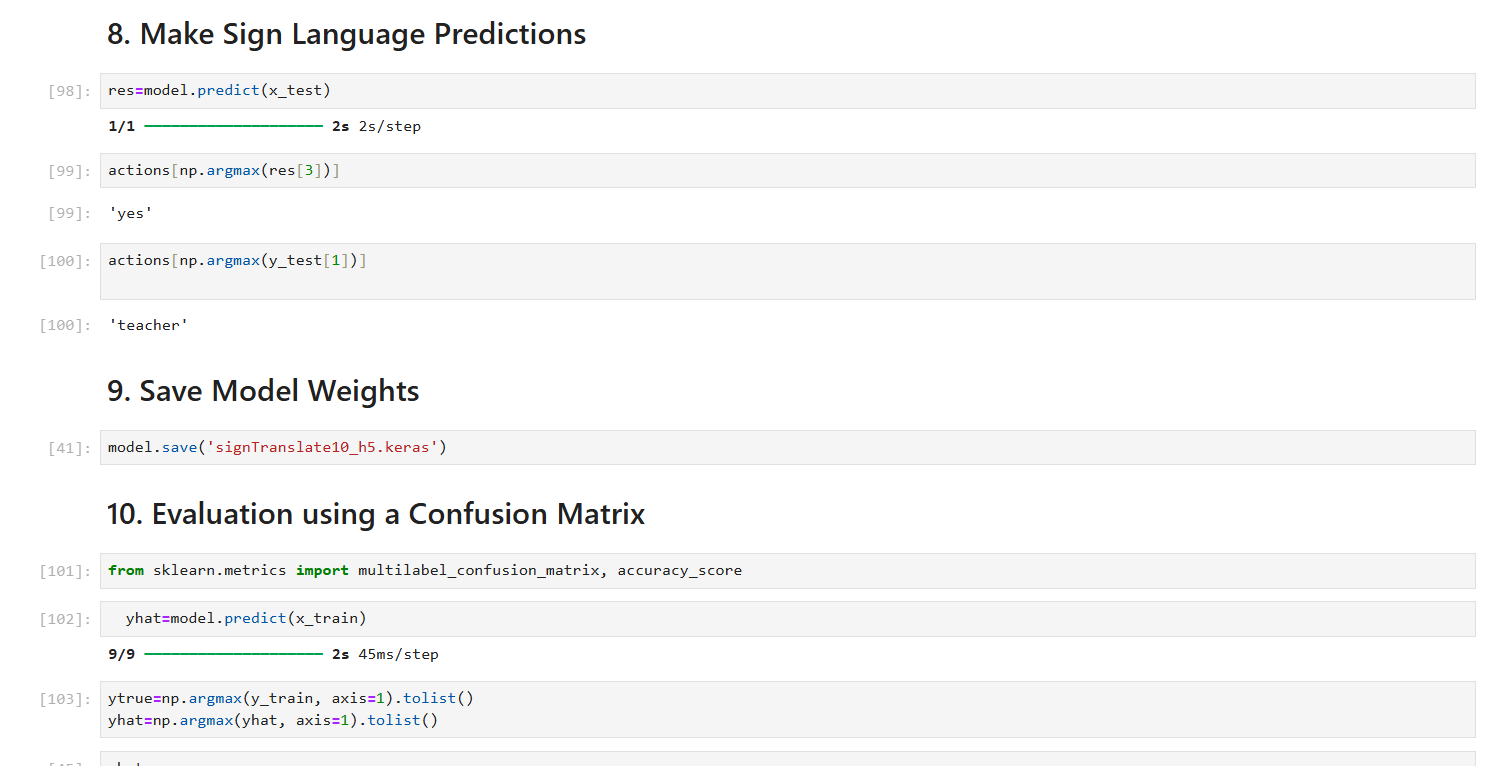
****

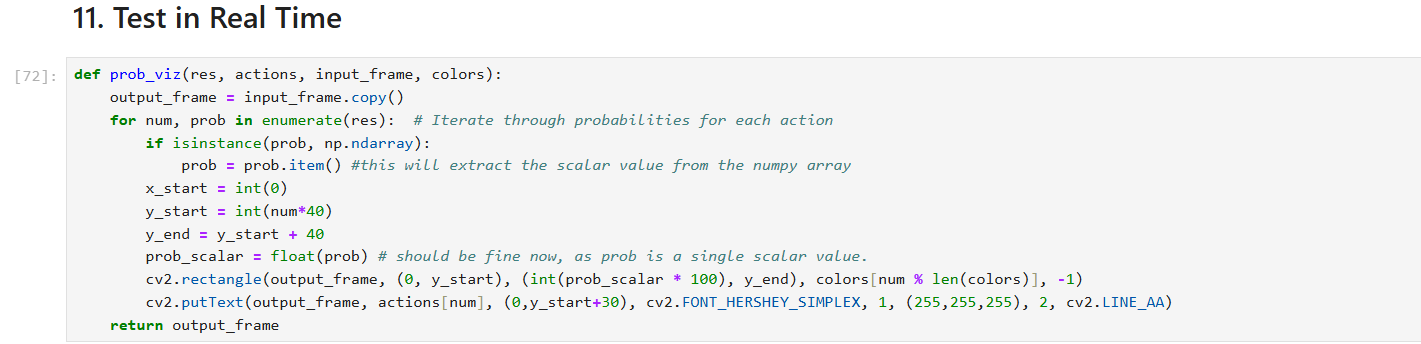
****

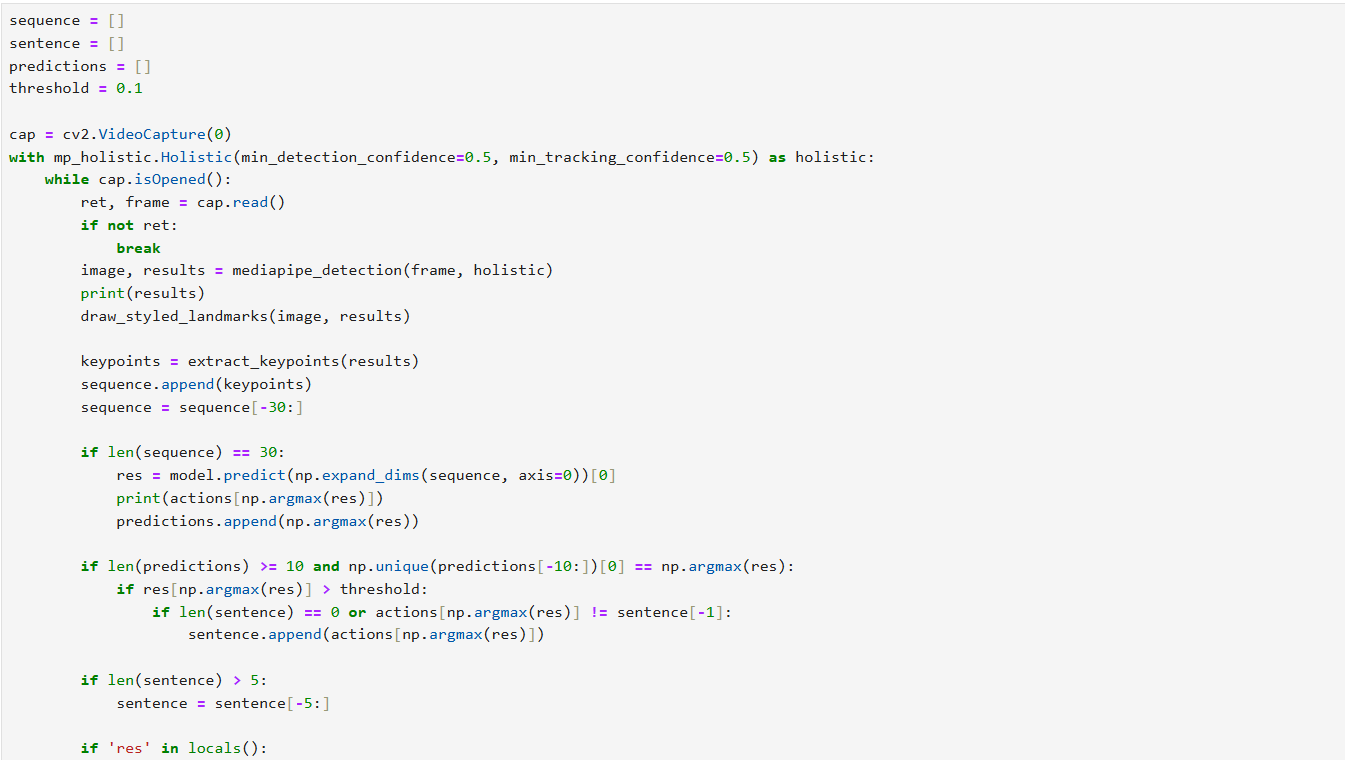
****

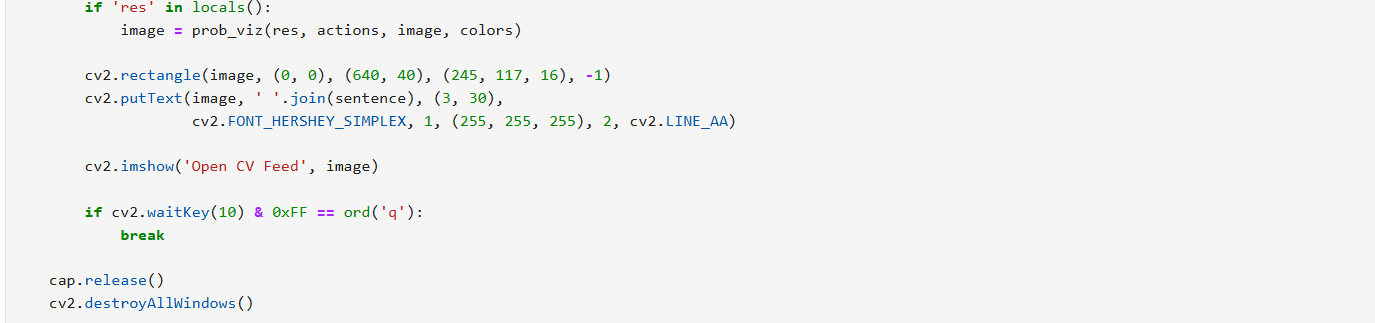
****

****

****







PLAGIARISM REPORT

**To be issued and certified by your Guide**