Table of Contents

LIST OF FIGURES

Chapter 1: Introduction

A proper means of communication remains vital for approximately 430 million people worldwide, about 5% percent of the world population [1] who have hearing difficulties. However, there are significant challenges due to the lack of written form, limited electronic resources, requiring an interpreter and over 300 sign languages globally [2]. To bridge this communication gap, modern technological advancements in hardware and software, particularly through machine learning and artificial intelligence, offer promising solutions.

Sign language translation (SLT) has become a crucial tool for bridging the communication gap between hearing and deaf individuals because there exists a significant gap in accessibility to information and services for this community because the majority of hearing people are not able to comprehend sign language [3]. Traditional methods of sign language interpretation, such as manual interpretation or text-based translation, are often slow and not universally available. This highlights the need for automated solutions that can recognize and translate sign language gestures in real-time.

Automated solutions leveraging computer vision and machine learning can significantly improve accessibility for deaf individuals, particularly in education, healthcare, and public services where they face significant challenges [4]. However, existing approaches often struggle with the dynamic nature of sign language expressions and may not accurately capture gestures in real-time as they often rely on static image or video-based approaches.

Addressing these gaps, the project of Action Recognition model for sign language translation attempts to bridge that gap and provide accessibility and inclusivity of deaf community. By leveraging computer vision and deep learning, this project aimed to demonstrate recognition and interpretation sign language gestures in real-time.

Chapter 2: Basic Concepts and Literature Review

2.1 Literature review

Previous Sign Language Recognition (SLR) systems were sensor based, which required specialized hardware that signers would use. The sensors were able to pick the sign and translate into a text readable format. However, the hardware was uncomfortable, restrictive and expensive for signers.

Today, new Sign Language Recognition systems are computer vision based. Computer vision techniques used in SLR allow signers to use their bare hands without wearing any specialized hardware, due to the use of cameras. Methods provided by computer vision are hand tracking, background detection and feature extraction.[5]

SLR systems also use Image processing techniques. These techniques involve a series of steps to convert visual information from sign language gestures into a form that can be understood by computers and translated into text. The steps are: Image acquisition, Preprocessing, Segmentation, Feature Extraction, Classification and Post processing.[6]

The downside to using image processing techniques is that the methods provided are complex with a requirement of computational power. This also increases the time consumption for training models. Computer vision techniques help in this area, as they need less computing power and have the ability to adapt to smart devices, which make the models robust and cost-effective.

2.2 Basic Concepts

The project is implemented using Media pipe Holistic, OpenCV, NumPy, and Scikit learn. Media Pipe is a cross-platform pipeline framework open-sourced by Google, to build custom machine learning solutions for live and streaming media [7]. With its optimized pose, face, and hand components that operate in real-time with little memory transfer between their inference backends, Media Pipe Holistic is a novel pipeline that also includes support for the three components being interchangeable based on quality/speed tradeoffs. [8].

NumPy is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays [9].

OpenCV (Open-Source Computer Vision Library) is an open-source computer vision and machine learning software library.[10].

Chapter 3: Problem Statement

This project aims to build a model that can recognize human gestures and translate it to words which ultimately address the communication challenges encountered by individuals who are deaf or hard of hearing, as well as those unfamiliar with sign language. Despite recent advancements in interactive software and applications that support sign language learning and translation, a significant gap remains in effective communication within the deaf and hard-of-hearing communities.

3.1 Project Planning

To execute the project development effectively, the following steps and requirements have been identified:

- Research and understand the intricacies of sign language, including the most commonly used signs and grammatical rules.
- A dataset of sign language videos with corresponding text annotations needs to be made for training the machine learning model.
- Define the software, model and hardware requirements necessary to develop and run the sign language translation system.
- Develop a project timeline with milestones for completing various phases of the project, such as data collection, model training, and system testing.

3.2 Project Analysis:

To make sure the model is feasible and clear, the gathered requirements and the formulated problem statement are carefully examined. The analysis involved is listed below.

- Verifying the completeness and accuracy of the sign language dataset. Ensuring that the proposed solution is inclusive and can be adapted to different sign languages if needed.

- Assessing the technical challenges and limitations that may arise during the development process.
- Evaluating the potential impact of the project on the target user group.

3.2 System Design

3.2.1 Design Constraints:

The system will be designed with the following constraints in mind:

- The software will be developed using Python programming languages and libraries/modules such as OpenCV, NumPy, Media pipe, matplotlib, Sklearn and machine learning frameworks like TensorFlow (LSTM).
- The hardware used will need to have sufficient computational power to process video data and perform real-time translation, including a high-quality 720p camera for capturing hand gestures/action for gathering feature points.
- The system should be compatible with various operating systems and devices to ensure accessibility for a wide range of users.

3.2.2 System Architecture:

The system architecture for the sign language translator project will consist of the following components:

1. Input Module: Captures live video feed of sign language gestures using laptop webcam.
2. Preprocessing Module: Processes the video data in individual frames to identify and extract relevant features such as hand movements and facial expressions.
3. Translation Module: Utilizes a trained machine learning model to interpret the extracted features and translate them into corresponding integer arrays.
4. Output Module: Displays the translated text in real time to the user by predicting the gestures.

Figure 1: Use Case diagram for the model

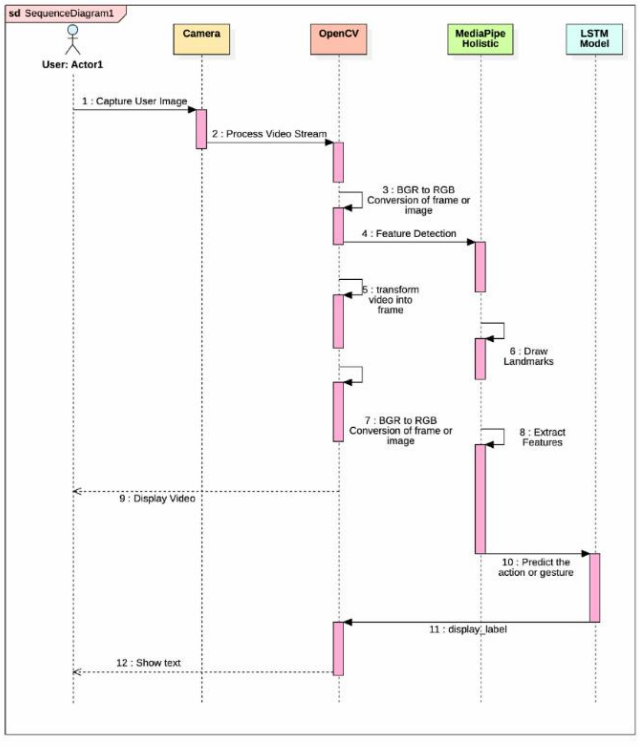

Figure 2: Sequence Diagram for the Model

Chapter 4: Implementation

4.1 Methodology

A strategic way of building the model was developed as follows.

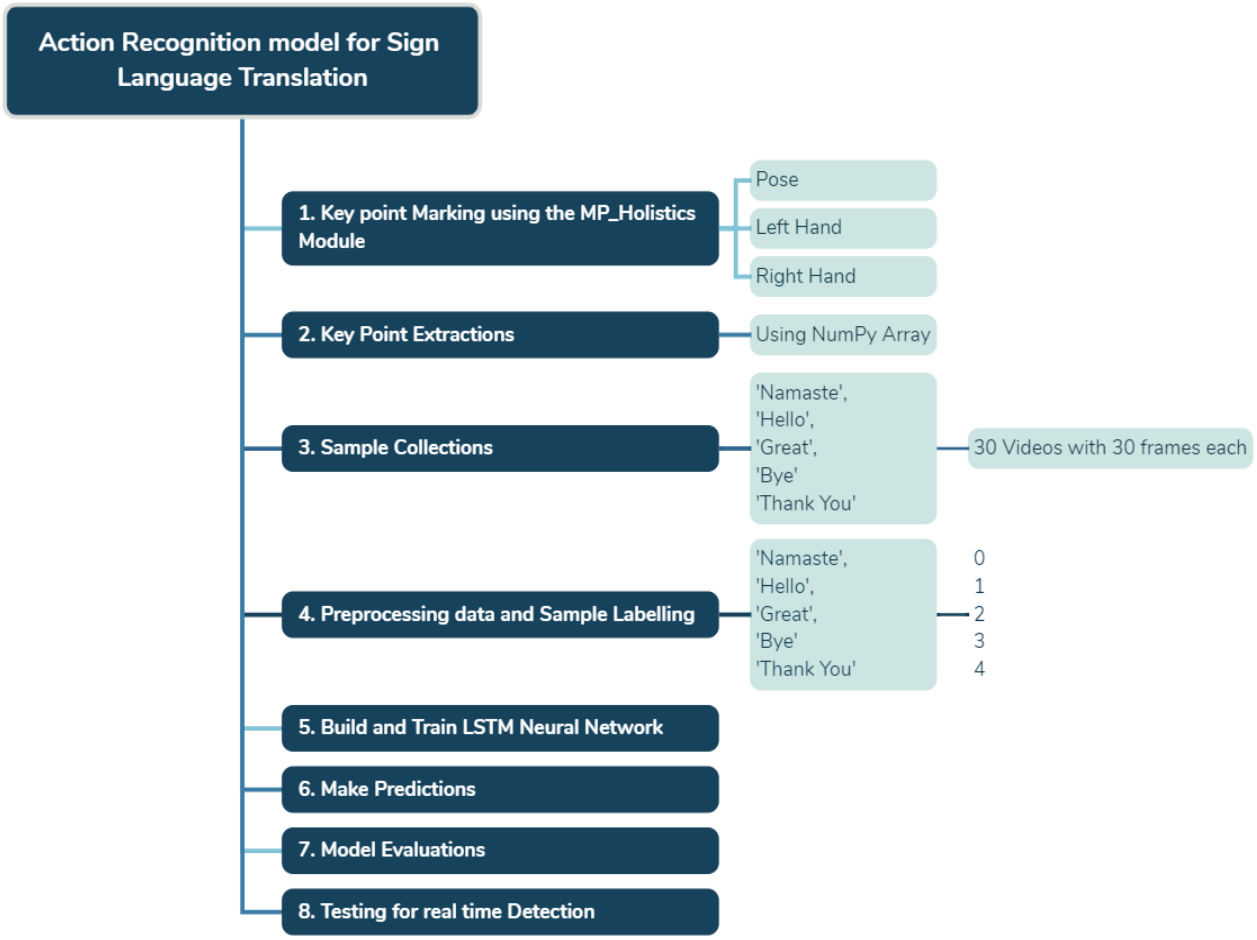

Figure 3: Methodology Diagram

4.1.1. Key point Marking using the MP_Holistics Module

Using the media pipe holistic module, the key land marks or a specific point on the body was marked for reference. The main landmarks included right hand, left hand and the posture. There were 33 key points for posture, 21 each for left hand and right hand.

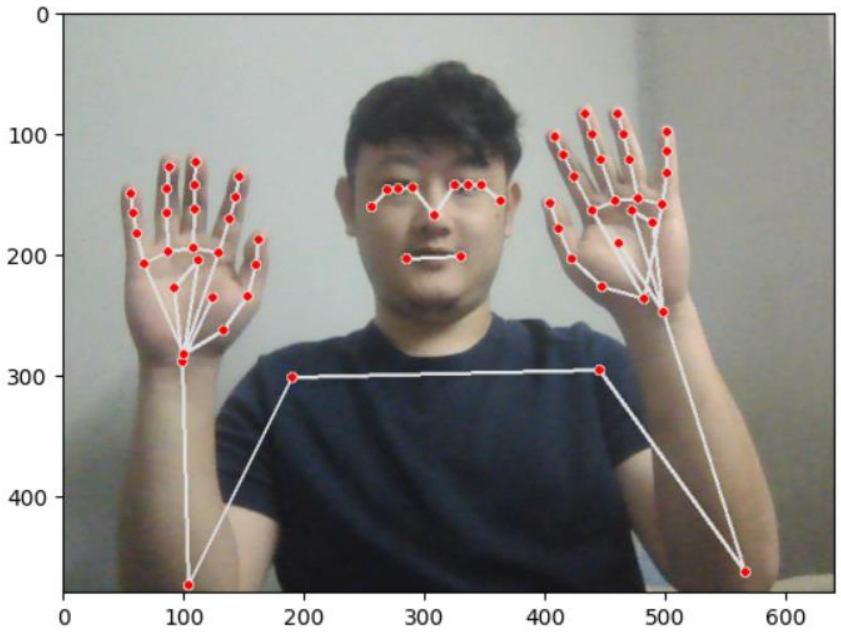

Figure 4: Key Point Detection using MP_holistics

4.1.2. Key Point Extractions

From the Key points that were marked in the previous part, the key points are then extracted as a NumPy array, which makes it easier and more efficient to work with. This is flattened into a one-dimensional array, where all the details of the land marks are concatenated into a single array. If incase during the sample collection, if the body part is not detected, an array of the same size with all zeros was appended in the end to avoid any error.

This essentially would break down all the extracted key point into arrays. For example, for a posture it consisted of 33 markers, whose position and orientation were provided by x, y, and z coordinate. The entire 33 markers coordinate are than converted into 1D array using NumPy for further operations.

4.1.3. Sample Collections

A folder for collecting the data was created, which had subfolder of all the actions that were to be detected and translated. The collection of words for this particular model are 'Namaste', 'Hello', 'Great', 'Bye' and 'Thank You'.

For each of the words, the action was captured using webcam, and every 30 frames were stored as a NumPy array. There were 30 videos for each word, with 30 frames per video, which essentially means that 1 video composed of 30 pictures in total.

4.1.4. Preprocessing data and Sample Labelling

The NumPy array which was stored in the individual sample folder was extracted and split into training and testing set. Where 5 % of the data was used for the testing purpose.

In order to work with the words, it was encoded using the 'to_categorical' class from tensor flow. With this, the words were assigned with an integer value for interpretation. The labels were 'Namaste': 0, 'Hello': 1, 'Great': 2, 'Bye': 3, 'Thank You': 4.

4.1.5. Build and Train LSTM Neural Network

With the Training and testing data split, the model is trained using the LSTM neural network. For this model, 3 LSTM layer and 2 Dense layer was used, subjected to an epoch of 500, 1000 and 2000 for the test.

4.1.6. Make Predictions

To Evaluate the model after training. The test data was passed as the input to the model for predictions. This was to ensure that the model was able to predict the correct value.

4.1.7. Model Evaluations

After the model has been built and trained on the training set. It was tested against the test set of the data. This to check how well the model is able to predict the class of the test data, and to check how well the model performs.

4.1.8. Testing for real time Detection

The model was subjected to real time video feed from the web cam using the OpenCV. This is to check if the model can read the action and the detect the correct word from the user.

4.1 Verification Plan

4.1.1 Test cases

Various test cases were adopted in order to train the model. The test case summary is as mentioned below.

Test Case	Videos	Frames	Epochs	Results	Inference
A	20	30	500	< 50%	Under performance
B	20	30	1000	≈98%	Well trained
C	30	60	1000	≈88%	Unstable Prediction
D	30	60	2000	≈88%	Unstable Prediction
E.1	30	30	1000	<50%	Underperformance
E.2	30	30	2000	99%	Well trained

4.2 Result Analysis

Initially the results were very erratic, where some of the words were predicted rightly but for most of the words, the model was losing a lot of information and not being able to predict correctly.

For Test Case A:

For the first test case, the model was highly unstable with huge data loss and very low prediction accuracy. The main reason for this could have been the smaller number of epochs and the smaller number of data points for training the model.

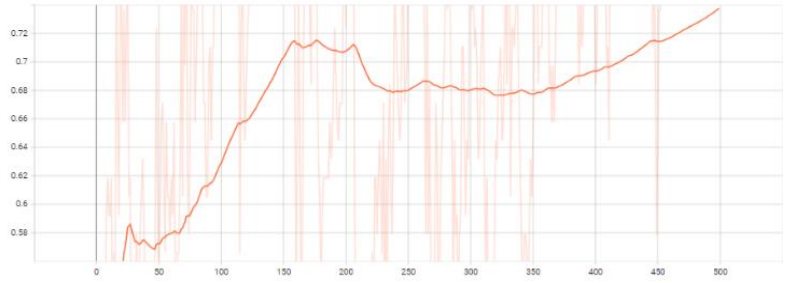

Figure 5: Epoch Categorical accuracy for Test Case A

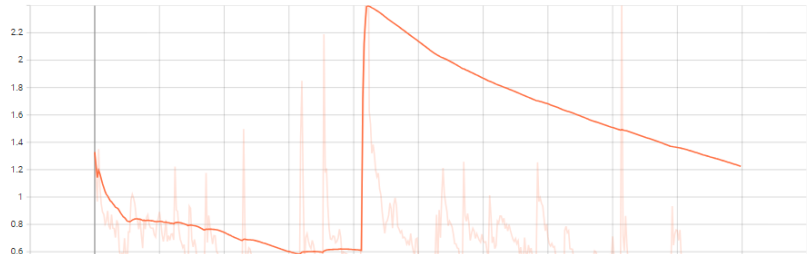

Figure 6: Epoch Loss for Test Case A

For Test Case B:

For this test case, the sample point and the samples were kept, but the epoch was increased to 1000. This time the result was near perfect and the model was performing good predictions.

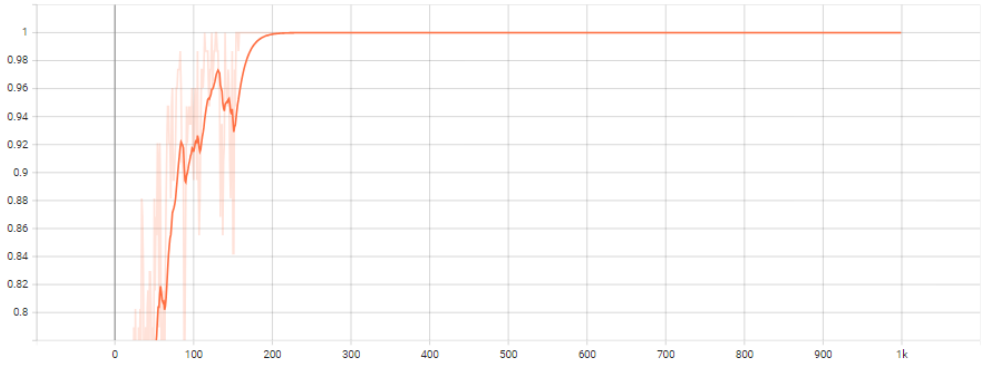

Figure 7: Epoch Categorical accuracy for Test Case B

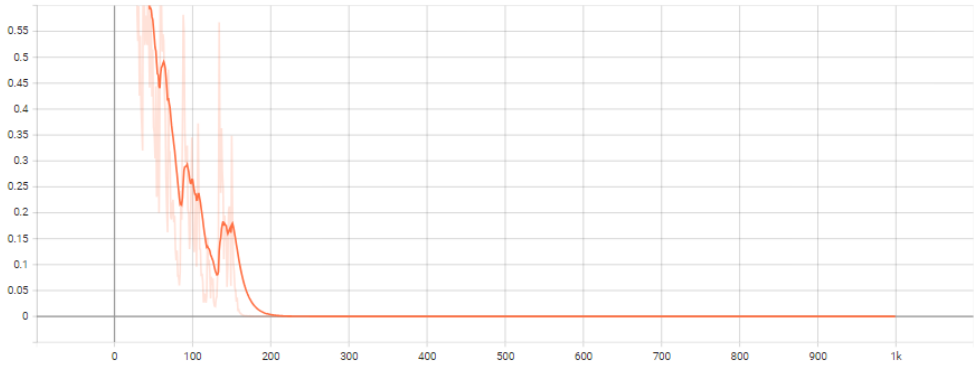

Figure 8: Epoch Loss for Test Case B

Test Case C:

For this case, the data points were increased with more video feed and more frames per video. Maintaining the same epoch of 1000, the model performed with an accuracy of 88% but the prediction was highly unstable.

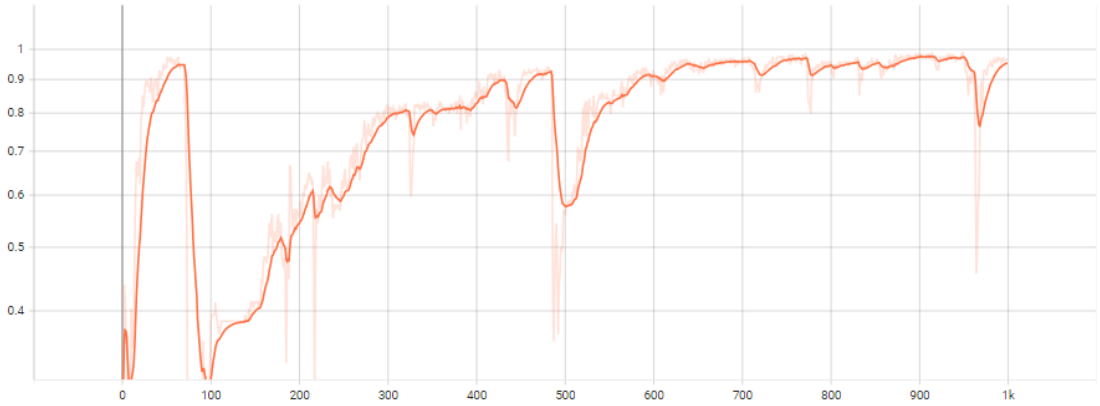

Figure 9: Epoch Categorical accuracy for Test Case C

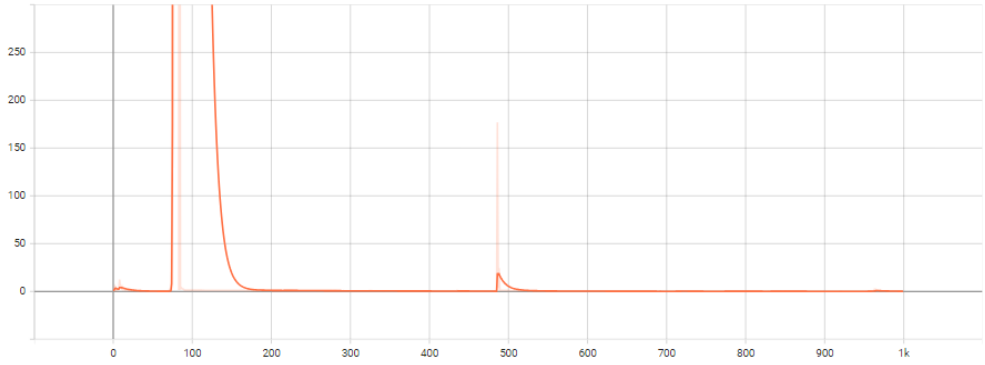

Figure 10: Epoch Loss for Test Case C

Test Case D

Data sample was kept same as Test Case c but the epoch was increased to 2000. This time the model was performing slightly better than the previous model, with similar prediction accuracy but with slightly more stable performance.

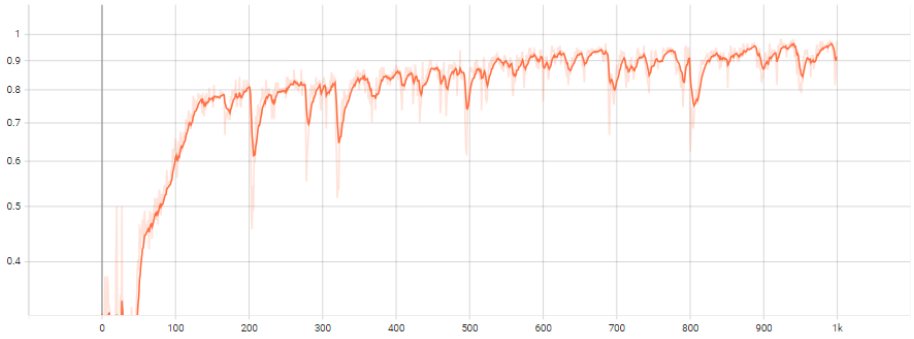

Figure 11: Epoch Categorical accuracy for Test Case D

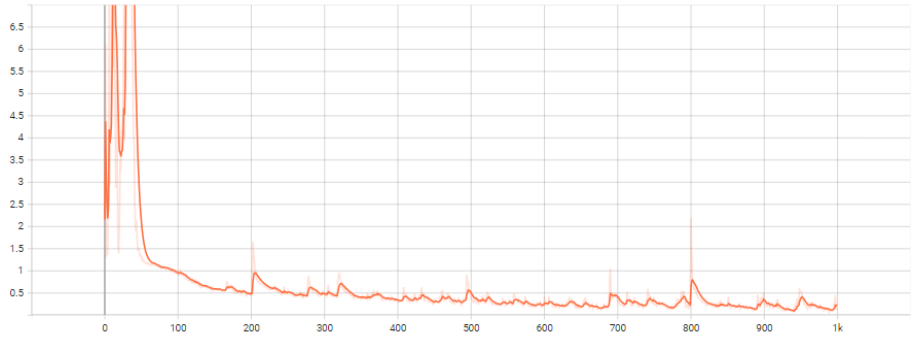

Figure 12: Epoch Loss for Test case D

Test case E.1 and E.2

In E.1 the sample size was 30 videos with 30 frames per video, and a epoch of 1000. This model underperformed in the accuracy test. However, after increasing the epoch to 2000 the model was predicting nearly perfectly with accuracy touching almost 1.

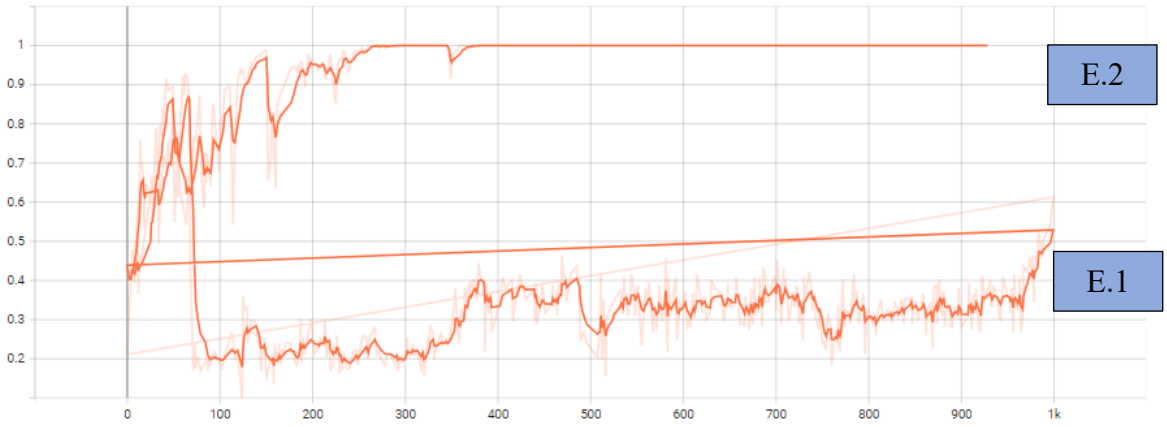

Figure 13: Epoch Categorical accuracy for Test Case E.1 and E.2

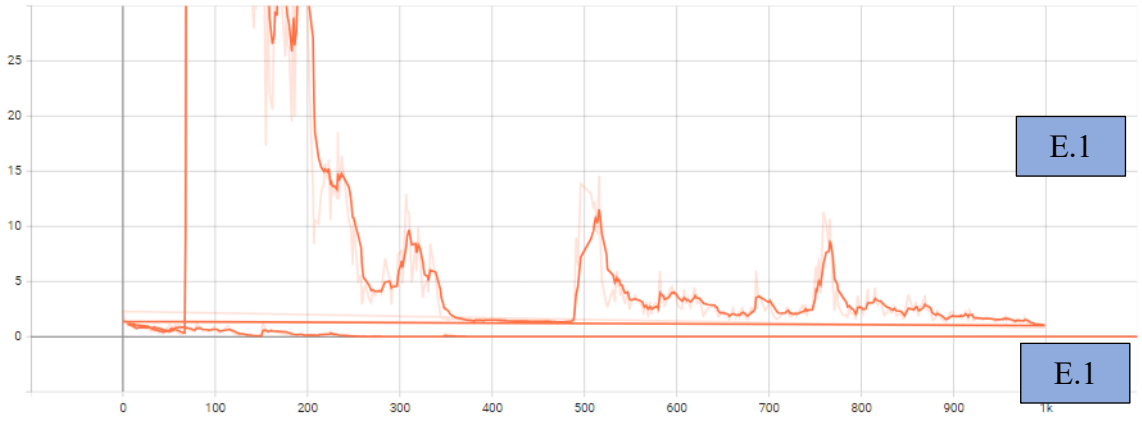

Figure 14: Epoch Loss for Test Case E.1 and E.2

Chapter 5: Standards Adopted

5.1 Model Used

Long Short-Term Memory (LSTM)

LSTM are effective for sequential data and can be used to recognize temporal generalization patterns in sign language, which is essential for understanding the dynamic gestures and the language prediction.

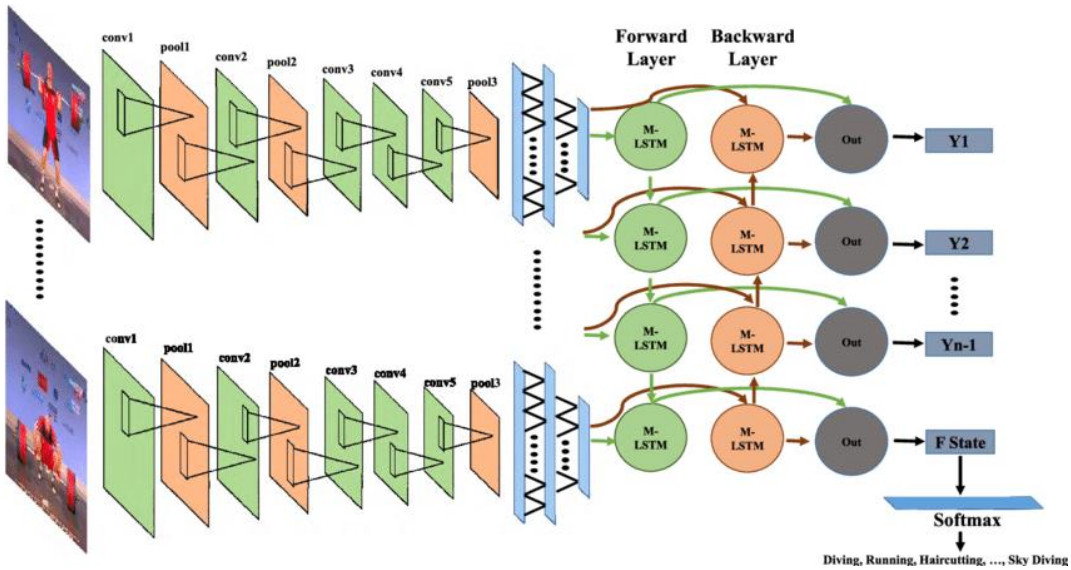

Figure 15: LSTM neural net (image source: Google Image)

5.2 Transfer Learning

It is kind of learning which is pre-trained models on large dataset can help in achieving higher accuracy without the need for a sign language-specific dataset. Transfer learning benefits in a very different way in this project.

5.3 Data Augmentation

For the increasing the diversity of the training set and improve the robustness of the model, data augmentation techniques which is rotation, scaling, and translation can be applied tothe sign language video, image and audio frames.

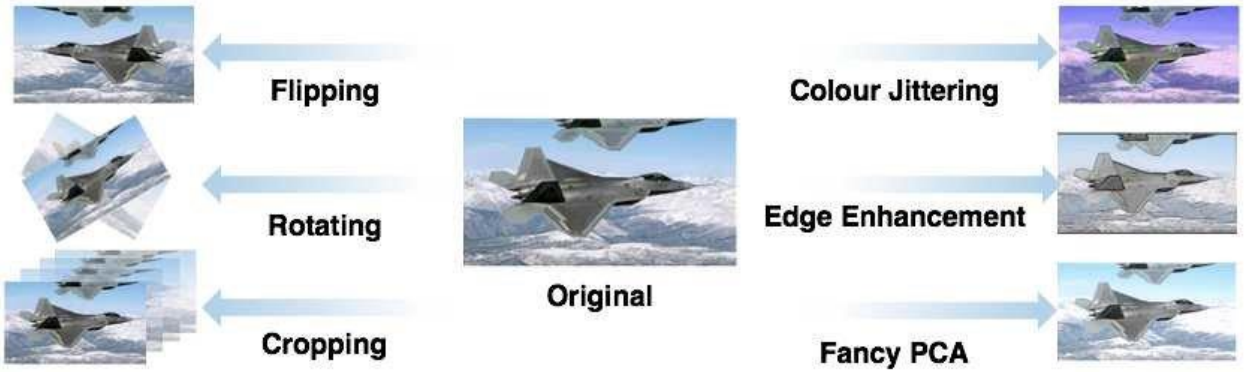

Figure 16: Data Augmentation (Source Google Image)

5.5 Coding Standards Adopted

- The policy of DRY was adopted for the coding part, trying to reduce repetition as much as possible.
- Developing proper functions that can be used again as and when required.
- Use of Camal casing naming convention for naming variables and functions
- Minimizing the number of lines using proper loop functions

Chapter 6: Conclusion and Future Scope

6.1 Conclusion

With an accuracy of almost 99%, the models were able to learn and interpret the words based on the action of the user, with near perfection. A major finding is that quality of the sample size can significantly affect the model predictions and lower the chances of getting a good result. Optimal size of sample as per this research was 30 videos with 30 frames per video with an epoch of 2000.

If the number of frames was increased, additional data points were also captured, which in most cases acted as a noise to the actual data, thus causing an underperformance to the model.

6.2 Future Scope

There is also the possibility of training the model with various other words in English and in the native language. The number of words can be limitless only limited by the number of actions one can associate with it.

The code can be further optimized to get facial expression input to integrate emotion detection as well, which could help provide a better context of the word or sentence being communicated. This could include, facial expression and lip-reading models for better context.

With proper optimization and further perfecting the algorithms to make it lighter and more portable, this model can be integrated with various mobile applications and be used as a translation tool to communicate with the regular people who can’t understand sign language.

Reference

[1]World Health Organization, "Deafness and hearing loss," *WHO*, Feb. 02, 2024. https://www.who.int/news-room/fact-sheets/detail/deafness-and-hearing-loss

[2]A. Sultan, W. Makram, M. Kayed, and A. A. Ali, "Sign language identification and recognition: A comparative study," *Open Computer Science*, vol. 12, no. 1, pp. 191–210, Jan. 2022, doi: https://doi.org/10.1515/comp-2022-0240.

[3]J. Zheng *et al.*, "An Improved Sign Language Translation Model with Explainable Adaptations for Processing Long Sign Sentences," *Computational Intelligence and Neuroscience*, vol. 2020, pp. 1–11, Oct. 2020, doi: https://doi.org/10.1155/2020/8816125.

[4]M. Masoumi, "AI-based solutions for Deaf people," *MIT SOLVE*, 2024. https://solve.mit.edu/challenges/learning-for-civic-action-challenge/solutions/73205#:~:text=The%20use%20of%20motion%20capture

[5]A. Tayade, "Real-time Vernacular Sign Language Recognition using MediaPipe and Machine Learning," *International Journal of Research Publication and Reviews*, vol. 2, no. 5, 2021, doi: https://doi.org/10.13140/RG.2.2.32364.03203.

[6]B. A. Divyashree and K. Manjushree, "Image Processing Techniques for Hand Gesture and Sign Recognition," 2008.

[7]G. Boesch, "MediaPipe: Google's Open Source Framework for ML solutions (2024 Guide)," *viso.ai*, Oct. 30, 2023. https://viso.ai/computer-vision/mediapipe/#:~:text=MediaPipe%20is%20an%20open%2Dsource

[8]I. Grishchenko and V. Bazarevsky, "MediaPipe Holistic — Simultaneous Face, Hand and Pose Prediction, on Device," *blog.research.google*, Dec. 10, 2020. https://blog.research.google/2020/12/mediapipe-holistic-simultaneous-face.html

[9]NumPy, "What is NumPy? — NumPy v1.19 Manual," *numpy.org*, 2022. https://numpy.org/doc/stable/user/whatisnumpy.html

[10]OpenCV, "About OpenCV," *OpenCV*, 2018. https://opencv.org/about/