A Project Report on

Indian Sign Language Interpreter

Submitted by

Akshi Kamde (Roll no. 17)

Mansi Khamkar (Roll no. 18)

Vighnesh Kalekar (Roll no. 19)

Arsalan Ahmed Qureshi (Roll no. 20)

in partial fulfillment for the award of the degree

BACHELOR OF ENGINEERING

in

Electronics and Telecommunication Engineering

Under the Guidance of

Ms. Pallavi Patil

St. Francis Institute of Technology, Mumbai

University of Mumbai

2022 - 2023

CERTIFICATE

This is to certify that Akshi Kamde, Mansi Khamkar, Vighnesh Kalekar and Arsalan Ahmed Qureshi are the bonafide students of St. Francis Institute of Technology, Mumbai. They have successfully carried out the project titled "Indian Sign Language Interpreter" in partial fulfilment of the requirement of B. E. Degree in Electronics and Telecommunication Engineering of Mumbai University during the academic year 2022-2023. The work has not been presented elsewhere for the award of any other degree or diploma prior to this.

_____

(Ms. Pallavi Patil)

_____ _____

(Dr. Kevin Noronha)
EXTC HOD

(Dr. Sincy George)
Principal

Project Report Approval for B.E.

This project entitled ***'Indian Sign Language Interpreter'*** by **Akshi Kamde, Mansi Khamkar, Vighnesh Kalekar and Arsalan Ahmed Qureshi** is approved for the degree of Bachelor of Engineering in Electronics and Telecommunication from University of Mumbai.

Examiners

1. - - - - - - - - - - - - - - - - - - -

2. - - - - - - - - - - - - - - - - - - -

Date:

Place:

ACKNOWLEDGEMENT

We are thankful to a number of individuals who have contributed towards our final year project and without their help; it would not have been possible. Firstly, we offer our sincere thanks to our project guide, Ms. Pallavi Patil for her constant and timely help and guidance throughout our preparation.

We are grateful to all project co-ordinators for their valuable inputs to our project. We are also grateful to the college authorities and the entire faculty for their support in providing us with the facilities required throughout this semester.

We are also highly grateful to Dr. Kevin Noronha , Head of Department (EXTC), Principal, Dr. Sincy George, and Director Bro. Shantilal Kujur for providing the facilities, a conducive environment and encouragement.

Signatures of all the students in the group

(Akshi Kamde)

(Mansi Khamkar)

(Vighnesh Kalekar)

(Arsalan Ahmed Qureshi)

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 this 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.

Signatures of all the students in the group

(Akshi Kamde)

(Mansi Khamkar)

(Vighnesh Kalekar)

(Arsalan Ahmed Qureshi)

Abstract

Sign language is a complete visual language with its own grammar, vocabulary, syntax, and other unique linguistic traits. It is primarily used by the Deaf and Hard of Hearing (DHH) community to communicate. However, since many people are unfamiliar with sign language, it can be extremely difficult for Deaf individuals to express their thoughts and emotions. Recognizing Indian Sign Language (ISL) requires specific techniques due to differences in vocabulary and grammar compared to other international sign languages. To address this, a system has been developed that takes in video inputs of gestures and uses a feature extraction method like MediaPipe and a deep learning model to recognize these actions. MediaPipe was used to extract keypoints of the hands and body determining their location, shape, and orientation. The LSTM model addressed the issue of frame dependency in sign movement. Due to the lack of video-based dataset for sign language, a dataset, which contains twelve vocabularies with 30 videos of each class was created and experiments conducted on this dataset revealed that the model achieved an accuracy of more than 91%.

Keywords: *Deep Learning, Tensorflow, Indian Sign Language Recognition, Action Recognition, MediaPipe Holistic*

Contents

List of Figures

List of Tables

List of Abbreviations

ANN	Artificial Neural Network
CNN	Convolution Neural Network
DHH	Deaf and Hard of Hearing
ISL	Indian Sign Language
LSTM	Long Short Term Memory
RNN	Recurrent Neural Network
SLR	Sign Language Recognition
WHO	World Health Organization

Chapter 1

Introduction

Sign Language is a complete language with its own grammar, syntax, vocabulary and other linguistic attributes. Sign Language is used as a medium for communication by visually impaired, deaf and dumb people which constitute a significant portion of our population. Sign language communication incorporates hand, body movement, arms movement, and facial expressions to communicate the feelings of a speaker. Due to the lack of a formal training, sign language is not widely understood. This work aims to bridge this gap of communication by developing a system that takes Indian sign language as input and generates meaningful sentences as output using a variety of feature extraction approaches and deep learning models.

This project is created by keeping the deaf and speech-impaired community in mind. The system receives input in video format. The output will be generated after a variety of processes, such as feature extraction, classification, and sentence formation. Thus, to develop a real-time interactive system that can help the hearing and speech impaired to communicate with normal people using Indian Sign Language and to develop a scalable project which can be extended to capture the whole vocabulary of Indian Sign Language through manual and non-manual signs is the main aim of this work.

1.1 Motivation

According to a report by the World Health Organization (WHO), more than 6 percent of people worldwide have hearing impairment. There are 466 million people with this disability as of March 2018, and by 2050, there are projected to be 900 million.[9] Text messaging, writing, using visual media, and finger spelling are just a few of the ways that people with hearing and speech impairments and those with normal hearing and speech are able to communicate. In India, the size of DHH community is 65 million to which there are only 250 number of certified sign language interpreters. In addition, no standard dataset is available. Indian Sign Language (ISL) recognition cannot entirely adopt the techniques used for other international sign languages due to variances in vocabulary and grammar.

All these constraints manifest the complexity of Indian sign language. Hence, there is a need of a system which recognizes the different signs and conveys proper context to the people who do not know sign language.

1.2 Problem Statement

The goal is to create a system that can recognize Indian sign language videos by utilizing feature extraction methods and sequence modeling techniques. The system will take in a dataset of Indian sign language videos as input and interpret which class each sign belongs to.

1.3 Methodology

Recognizing human gestures using machine learning is a complex process that involves multiple steps. Firstly, all necessary dependencies must be imported and installed, such as TensorFlow, NumPy, and OpenCV. These dependencies will be used for the key aspects of the project, including training and testing the model and processing data. Next, the Mediapipe Holistic model is used to extract keypoint values from a given dataset of videos or a live feed. The Holistic model is

an all-in-one solution that extracts multiple keypoints simultaneously, including hand, face, and body keypoints. This allows for a more comprehensive understanding of the gestures being performed, which is essential for accurate recognition. Once the keypoint values have been extracted, it's important to set up folders for collecting these values. This is done to prepare the data for training and testing the machine learning model. The keypoint values are collected from the dataset and stored in the appropriate folder. This folder is then used to preprocess the data, which involves normalizing the data and converting it to a format that can be used for training and testing the model. After preprocessing the data, labels and features are created for the training and testing datasets. These labels are used to identify the gestures being performed in the videos, while the features are the keypoint values that the model will use to recognize these gestures. An LSTM neural network is then built and trained with the preprocessed data. LSTM is a type of recurrent neural network that is effective in recognizing sequential data, which is essential for recognizing human gestures. Once the model is trained, the weights are saved for future use. Finally, the model is tested in real-time with a live video feed or a new dataset of videos to evaluate its accuracy and performance. The model is evaluated based on metrics such as precision, recall, and F1 score, which provide a comprehensive understanding of the model's accuracy.

1.4 Scope of Project

- The aim is to create a system that can recognize videos of Indian sign language by using techniques for feature extraction and sequence modeling. The system will receive a dataset of videos containing Indian sign language as input and will be capable of determining which class each sign pertains to.
- We can create a model for recognizing Indian sign language at both the word and sentence levels. Accomplishing this will necessitate the development of a system capable of detecting changes over time.
- We have the potential to design a comprehensive product that will assist individuals who are deaf or hard of hearing, thereby lessening the gap in

communication.

1.5 Organization of Project Report

This project report is organized as follows:

Chapter 2 presents the literature survey on the existing techniques

Chapter 3 provides a brief explanation of theoretical background

Chapter 4 is dedicated to the simulation and experimental results.

Chapter 5 presents the conclusions and future scope for this project.

Chapter 2

Literature Review

Recognition of sign language is a well-known scientific issue. While a single video corresponds to a single sign in regular or isolated SLR and numerous signs do so in continuous SLR, The more challenging challenge, continuous SLR, necessitates both segmentation and classification of the video. Even though numerous sign language identification techniques have previously been created and show promising results, the field of study remains difficult for researchers. The majority of the effort is put into recognising solitary sign languages. Due to its complexity, there is very little literature in the field of continuous sign language recognition.

2.1 Literature Review

1. Disha Gangadia,Varsha Chamaria, Vidhi Doshi, Jigyasa Gandhi, "Indian Sign Language Interpretation and Sentence Formation". The project was divided into two main components, namely Gesture Recognition and Sentence Generation, which in turn have many sub-components, which includes image preprocessing,removing light variations, reducing noise, motion detection,edge detection, segmentation into frames, extracting gesture labels through Convolutional Neural Network (CNN) based learning, converting the labels to tokens, fitting the tokens inside a grammar, using web results and corpus results that include similarity checks and grammar correction to create top 10 most meaningful and relevant sentences, converting text format to speech and showing the text and speech output

results to the user. The System Architecture consists of many stages. The preprocessing phase consists of methods that extract important discernible features from the image like Grayscaling, illumination normalization, noise removal, edge detection, corner detection, thresholding, etc. The system contains a data set of preprocessed hand gestures. These are used to evaluate the performance of the proposed method. Data set is a collection of around 10000 hand gesture images. For each gesture class, 100 instances are captured. Since Indian Sign language data set (ISL) is not available, we have created a data set of 100 Indian signs. While creating the data set, image for a particular gesture is stored in 4 different formats: 1) NoFilter 2) Features from accelerated segment test (FAST) 3) Canny Edge 4) Scale-Invariant Feature Transform (SIFT) The classification uses a CNN-hybrid model which is trained and validated using the database. Also, Data Augmentation is done on the training database images to include various transformations(geometric variance and illumination variance).The input videos from the user are converted into frames at real-time and passed to the trained model and the gesture with the highest probability is selected and passed on to the Natural Language Processing Stage. The Natural Language Processing phase forms sentences by adding relevant words and correcting grammar and returns a text output which is converted to audio as well for the end-user. NLP phase is also called the speech synthesis phase. The grammar is checked using Bidirectional Encoder Representations from Transformers (BERT) which is used on the data set of Lang-8 Corpus of Learner English . A multi-headed language model is generated that uses BERT as encoder and decoder Transformer for grammar correction (excluding spelling correction) with Replace and Range Heads options. This resulted in overall accuracy of around 93.4 percent.[1]

2. Kinjal Mistree,Devendra Thakor,Brijesh Bhatt,"Towards Indian Sign Language Sentence Recognition using INSIGNVID: Indian Sign Language Video Dataset". The process of formation of English sentences from continuous ISL gestures mainly involves ISL video recognition. Videos in

dataset were captured using 30 fps, with resolution of 1920 x 1088 pixels. Taking these characteristics, RGB frames are generated for each video of different length.Signer can be either right-handed or left-handed. On the batch of frames, horizontal flipping is performed in order to incorporate both left-handed and right- handed signs. Though horizontal flipping is one of the data augmentation techniques, this technique has been explicitly separated from the other set. Reason behind this is, if online data augmentation technique is used, the modifications in images will be applied randomly, so not every image will be changed every time. Signer can be either right-handed or left-handed. On the batch of frames, horizontal flipping is performed in order to incorporate both left-handed and right-handed signs. Though horizontal flipping is one of the data augmentation techniques, this technique has been explicitly separated from the other set. Reason behind this is, if online data augmentation technique is used, the modifications in images will be applied randomly, so not every image will be changed every time. Video frame generator is created that acts as video generator. They have taken 5 frames per second for each video. The inflated dataset after image augmentation technique is given as input to the Convolutional Neural Network (CNN). CNN architecture is selected for their work as it is best suited for capturing internal representation of features of visual world . Here, their CNN is initiated with MobileNetV2 model that was pretrained on ImageNet dataset. Pretrained model is chosen because image augmentation increases the size of the dataset which is originally very small but the data similarity is still very high. Also, MobileNetV2 is light-weight model that uses deep neural network that has proven best for mobile and embedded vision applications . Fine-tuning is performed on the MobileNetV2 model by experimentally changing the top layer configuration of the model to get the best classification result. Moreover, LSTM (Long-Short Term Memory) model has been added that needs one dimension. As MobileNetV2 model is used without top layers, one Time Distributed layer has been added as a top layer to have the one-dimension shape compatible with LSTM. Finally, dense layer is added to get the prediction of English sentence. Output of overall

process will be semantically equivalent English sentencecorresponding to ISL sentence. The performance of MobileNetV2 model is also compared with three popular pretrained models used for object recognition:ResNet50, VGG16 and MobileNet. On comaprison the Mobilenetve model had the best accuarcy with accuracy of 91 percent. [2]

3. Advaith Sridhar,Rohith Gandhi Ganesan,Pratyush Kumar,Mitesh Khapra,"INCLUDE: A Large Scale Dataset for Indian Sign Language Recognition". In this paper they have used the INCLUDE-50 dataset. Include-50 is a subset of INCLUDE dataset which consisted of 0.27 milllion frames with 4287 videos of 263 classes. The INCLUDE 50 consist of only 50 classes for rapid evaluation They need to map such high-dimensional data to one of 50 signs using just tens of example videos per sign. To enable such learning,they first extract lower-dimensional features using pre-trained models. OpenPose is a model for real-time multi-person pose estimation on single two-dimensional images. In addition, OpenPose estimates about 135 key-points in each image which detect different body parts such as faces, hands, and feet. Thus, given an image, OpenPose computes the key-points and the PAF.Thus, with a pre-trained OpenPose tool we can extract features in the form of the key-point vector, pose video, and the PAF video. Different combinations of these features are used in different methods. For instances, instead of sending the original video, they can send either the pose or PAF video or a concatenation of both to the deep learning pipelines.Based on the features extracted using the OpenPose network, we study two broad methods for SLR on the INCLUDE-50 dataset. In the first method we use only the key-points vector and an efficient XGBoost classifier. In the second method, we use combinations of the Pose and PAF videos with different encoder and decoder deep neural networks (DNNs).Using Key-Points Vector. they extract shoulder, arm, hand and finger key-points for each frame using the pre-trained OpenPose model. In some frames, OpenPose fails to generate key-points. They optionally add an imputation step using the Lucas-Kanade sparse optical flow method . This method fills in the missing key-points

assuming a continuity in the spatial movement of each key-point across frames. The feature vector is then flattened across frames and normalised. The feature vector has 96 points per frame, and videos are zero-padded to 200 frames. This results in a feature vector of size 19,200. This vector can then be input to any machine learning model for classification. For the encoder, they had use four well known CNN backbone networks: MobileNetV2,ResNet50V2, DenseNet201 and InceptionV3. Each of these networks are pre-trained on the ImageNet dataset . In each network, the weights of the pretrained networks are frozen, the last softmax layer (used for classifying into the ImageNet classes) is deleted, and the output of the penultimate layer is passed through an average-pool layer. The output of this average-pool layer is then the latent embedding for each input frame. The size of the embedding vector varies depending on the backbone CNN: It is 640 for MobileNetV2, 960 for DenseNet201, and 1024 for ResNet50v2 and Inceptionv3. The encoder generates a fixed size latent embedding for each frame. The decoder then processes these to generate a class label. After experimenting with different sequence models, we chose a Bidirectional LSTM (BiLSTM) as the decoder. The decoder model has a single Bidirectional LSTM layer with 128 units, the output of which is flattened and fed into a fully-connected layer with 128 units and a softmax layer for classification, with a dropout layer of value 0.4 in between. The model design is captured in Figure 2. Note that in this method the only trainable parameters are the weights in the decoder. Thus, while the end-to-end network is large, only a small fraction of the weights are fine-tuned for a task-set. If such a method achieves a good accuracy it can enable supporting SLR on multiple sign languages efficiently.dataset. The best performing model has an accuracy of 94.5 percent on the 50-class classification task. It uses the Pose video as the extracted feature, a MobileNetV2 backbone, and a BiLSTM decoder.[3]

4. S. Reshna, A. Sajeena, and M. Jayaraju, "Recognition of static hand gestures of Indian sign language using CNN" This paper proposes the classification

of ISL gestures using convolutional neural networks. Convolutional Neural Networks (CNN) comprises of a number of convolutional and subsampling layers.This project is implemented using tensor flow. The training images are taken from the data set of Robita Lab of Indian Institute of Information Technology, Allahabad. Five gestures of ISL - Above, Across, Afraid, All and Advanced is taken for classification Three convolutional layers are designed for the training of the dataset where the corresponding weights, biases are defined and implemented. The convolutional layer is defined and appended with the biases. Then the max-pooling is applied where a local maxima -is obtained and the output at each stage is given as input to the activation function. The matrix data is expressed as a row matrix. The fully connected layer takes input as x and produces wx4b. A sample test image for "above' gesture and the output obtained All of the patterns have some changes in the positions of the hand. All are classified with a probability of 0.99. This method provides for the recognition and relation of available datasets with maximum accuracy Any segmented images with dimensions identical to the trained images can be recognized under this method From the results obtained from the CNN implementation of the above datasets, an accuracy of 9 percent is summarized. This method provides for the recognition and relation of available datasets with maximum accuracy. Any segmented images with dimensions identical to the trained images can be recognized under this method This work shows that convolutional neural networks can be used to accurately recognize different signs of a sign language included in the training set. This generalization capacity of CNNs in spatio-temporal data can contribute to the research on automatic sign language recognition The input images Above, Across, Afraid, All and Advanced are classified using CNN. An accuracy of 95 percent is obtained. All of the patterns have some changes in the positions of the hand. All are classified with a probability of 0.99. [4]

5. R. Dhiman, "A deep learning approach for Indian sign language gestures classification with different backgrounds". In this paper, various Convolution Neural Network (CNN) parameters have been explored to propose a CNN model with the best accuracy. Similarly, three optimizers were explored i.e. Adam, Sgdm, RMSprop. First, the dataset of Indian alphabets was acquired. Based on the preliminary study and experience on the machine learning architectures, during the pre-processing step, the input image is resized to 128*128 size. To raise the number of images in the dataset and introduce the variability in the dataset, data augmentation is done. Here, augmentation is done by rotating the image by 20 degrees and translating the image both in the x and y direction by 50,50. So the original 100 images per class are augmented by a factor of three resulting in a total of 7800 images for ISL uniform and complex datasets each. The proposed model in this paper is focused primarily on Indian sign language datasets referred to as ISL datasets in the following sections. To estimate the effectiveness of the proposed model, it has been validated on publicly available NUS dataset-I and NUS dataset-II..It is publicly available which has been created by the National University of Singapore (NUS). NUS dataset-I consists of 10 classes of hand gestures with a uniform background . NUS Dataset-II also consists of 10 hand gestures but with complex backgrounds The five-layer CNN architecture proposed in this paper has been further analyzed for three optimizers: Sgdm (Stochastic gradient descent with momentum), RMSProp (Root mean square propagation), and Adam (Adaptive moment estimation). After implementing the proposed model, this section discusses the obtained results. For testing, 10 percent of the dataset is considered. The developed ISL recognition model is tested by varying many parameter values. The model is also tested by using different activation functions and three optimizers on simple and complex background images and the mixture of both as well. As dataset shuffling is effective in enumerate random nature to the neural network training process which prevents the network from biasing towards certain parameters. Training has been done with three different optimizers to analyse which one is the best for sign language

recognition.datasets. The model tested on NUS dataset-I, NUS dataset-II, and mixed dataset where the accuracy of 100 percent, 95.95 percent, and 97.22 percent respectively has been achieved. [5]

6. Palash Kamble, Dr Rathna G N, "Word Level Sentence Generation using Deep Learning for Indian Sign Language" Due to the complexity of CNN and being inefficient at run-time, so they proposed making sign language recognition without using any explicit CNN architecture. They used feature keypoints extracted per frame from a video using the mediapipe package and feed the extracted keypoints per frame to the sequential model to predict the associated word. They used LSTM based sequential model and encoder layer from transformer to establish sign language recognition tasks. Between these two models, the transformer encoder is giving better performance. The Indian Sign Language dataset they used is replicated from the freely available ISLRTC New Delhi YouTube page. The playlist found on the youtube channel consists of more than four thousand words and their associated gestures . They made 50 videos per 20 gestures. Each video has 20 frames of sequential data, i.e., sign language. Then with the help of the mediapipe package, They extracted keypoints associated with each gesture per frame. They extracted keypoints from the face, pose, left hand, and right hand for each frame. A total of 1662 keypoints are extracted per frame. That is, they get a tensor of shape 20x1662 per gesture (or per video). After that they then feeded this sequential data with a sequence length of 20 to the sequential model to recognize that particular video. They Used two different models, 1) LSTM-RNN Classifier 2) Transformer Encoder Classifier. LSTM-Classifier was first model they used. The keypoints data extracted from a video dataset having 20 frames have a tensor of shape (20x1662). So we have 20 such sequential input vectors (each vector having size 1662) to be fed to our LSTM-RNN architecture. The class which has highest probability is chosen as the predicted class. The transformer model is generally suitable for the sequence to sequence tasks. It has two components encoder and decoder. The encoder component takes in the input sequence and encodes it. The decoder component decodes the encoded sequence to its corresponding

output sequence. But since we are doing a recognition task, we only leveraged the encoder component of the transformer model. The accuracy of second model is much better than first model. The first model got accuracy of 54 percent and second model got accuracy of 70 percent. [6]

Table 2.1: Literature Review Table

Title, Author, Year	Dataset	Methodology	Accuracy
DishaGangadia, Varsha Chamaria,, Vidhi Doshi, Jigyasa Gandhi, "**Indian Sign Language Interpretation and Sentence Formation**", Dec 16-18, 2020	Data set is a collection of around 10000 hand gesture images. For each gesture class, 100 instances are captured	CNN-SIFT Hybrid Bert	93.4%
Kinjal Mistree, Devendra Thakor, Brijesh Bhatt, "**Towards Indian Sign Language Sentence Recognition using INSIGNVID: Indian Sign Language Video Dataset**", 2021	Generated by them (1289 videos of ISL sentences corresponding to 55 unique classes)	MobileNetV2 CNN	91%
Advaith Sridhar, Rohith Gandhi Ganesan, Pratyush Kumar, Mitesh Khapra, "**INCLUDE: A Large Scale Dataset for Indian Sign Language Recognition",** OCT 2020	It comprises of 4,287 videos across 263 classes out of which 50 classes were chosen. INCLUDE50 contains 958 videos and 60897 frames.	OpenPose MobileNetV2 BiLSTM XGBoos	94.5%
S. Reshna, A. Sajeena, and M. Jayaraju, **"Recognition of static hand gestures of Indian sign language using CNN"**, April 2020	Dataset of four images Advanced, Across, Afraid and Above are used	CNN	95%
R. Dhiman, **"A deep learning approach for Indian sign language gestures classification with different backgrounds",** Jan 2021	The data set consists of 50 videos per 20 gestures. Each video has 20 frames of sequential data	Deep learning based-five layer CNN model	95.95%
Palash Kamble, Dr Rathna G N, **"Word Level Sentence Generation using Deep Learning for Indian Sign Language",** April 2021	The data set consists of 50 videos per 20 gestures. Each video has 20 frames of sequential data	-LSTM-RNN Classifier -Transformer Encoder Classifier	70%

Chapter 3

Theoretical Background and Design Methodology

3.1 Input Data

Due to primary and secondary parameter issues, the majority of ISL methods have difficulty accurately recognizing gestures. To address this problem, our approach is divided into two parts. The first part involves feature extraction, which utilizes the MediaPipe framework to extract keypoints. The second part consists of the ISL recognition module, which analyzes sign movement and produces the sign label as output.

A dynamic dataset was created and analyzed during the study which comprises of 360 videos that were split into into training and testing sets in the ratio of 4:1. All videos in the dataset were captured indoors with ordinary lighting and a web camera. Each video has the same duration and frame count, recorded at a speed of 30 frames per second (FPS). However, in instances where different datasets with varying frame counts are employed, it is essential to standardize the total frame count for each clip to ensure that the models have a consistent number of frames per sign language clip.

No effort was made to control the signer's attire or signing technique during the videos' bright, natural lighting shoots. The background varies across videos, though all videos have been shot in a classroom setting with clutter such as desks, boards and cabinets.

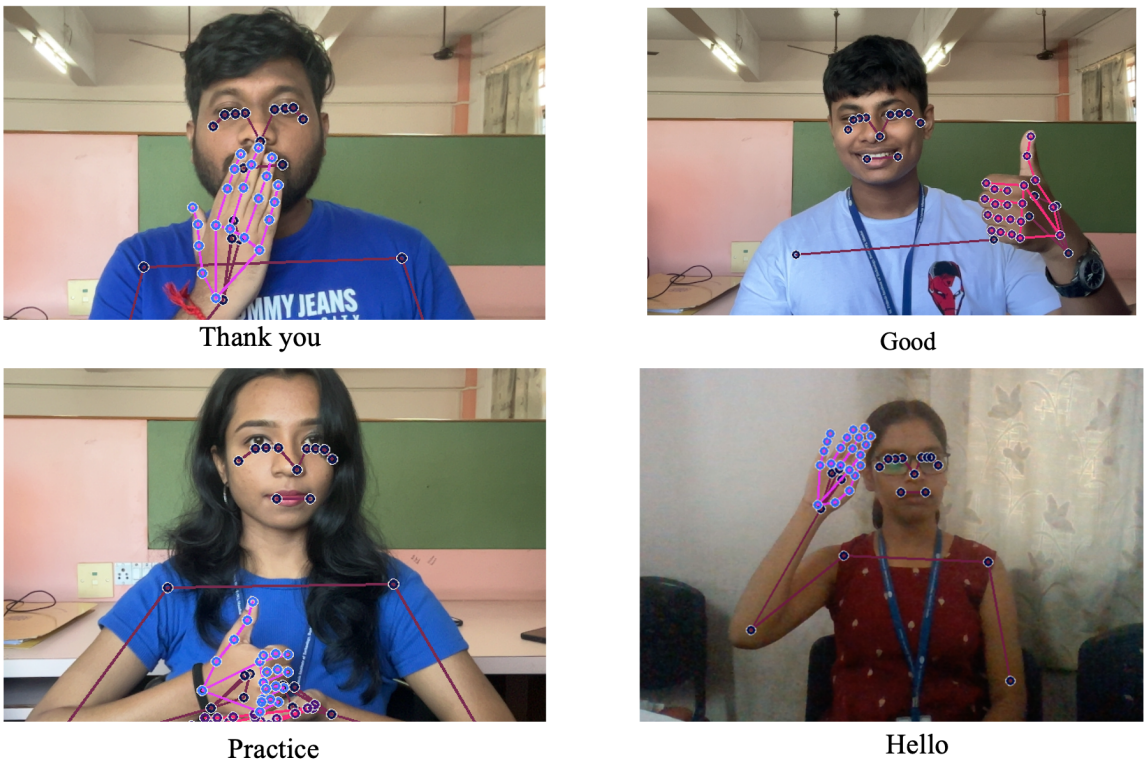


Figure 3.1: Snapshots from the dataset for various classes.

3.2 Feature Extraction using MediaPipe

The use of hands and poses is fundamental in sign language; however, continuous movement poses challenges for ISL recognition, such as difficulties in locating the hands, determining their shape and direction. To address these issues, MediaPipe[10] was employed. MediaPipe extracts keypoints for the X, Y, and Z dimensions of both hands and performs pose estimation for each frame, providing a solution to these problems. MediaPipe is a framework that processes perceptual data, such as images, videos, and audio, using machine learning for real-time hand tracking and gesture recognition.

MediaPipe Holistic pipeline is used to obtain landmarks from the face, hands, and body pose, which accurately detect sign gestures. The MediaPipe Holistic body pose model infers 33 3D landmarks from the input image or video, while the MediaPipe Holistic hands model infers 21 3D hand landmarks using a single-shot detector called Blaze Palm for palm detection and hand keypoint localization. The outputs of the MediaPipe Holistic hands model include 21 hand knuckle points, a hand flag showing the probability of hand presence, and binary classification of left

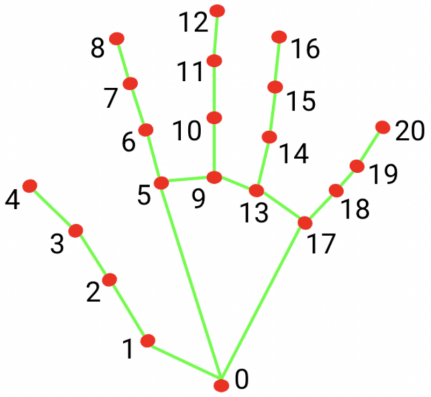
Characteristic	Dataset
Categories	15
Words	50
Videos	958
Avg Videos per Class	19.16
Avg Video Length	2.54s
Min Video Length	1.44s
Max Video Length	6.16s
Frame Rate	25fps
Resolution	1920x1080

Table 3.1: Key statistics of generated dataset

and right hand.

For each hand, MediaPipe extracts 21 keypoints as shown in Figure 3.2. The keypoints are calculated in the three-dimension space: X, Y, and Z for both hands. Thus,the number of extracted keypoints of hands is calculated as follows:

$$\text{Keypoints in hand} \times \text{Three dimensions} \times \text{No. of hands} = (21 \times 3 \times 2) = 126 \text{ keypoints.} \tag{3.1}$$



0. WRIST
1. THUMB_CMC
2. THUMB_MCP
3. THUMB_IP
4. THUMB_TIP
5. INDEX_FINGER_MCP
6. INDEX_FINGER_PIP
7. INDEX_FINGER_DIP
8. INDEX_FINGER_TIP
9. MIDDLE_FINGER_MCP
10. MIDDLE_FINGER_PIP
11. MIDDLE_FINGER_DIP
12. MIDDLE_FINGER_TIP
13. RING_FINGER_MCP
14. RING_FINGER_PIP
15. RING_FINGER_DIP
16. RING_FINGER_TIP
17. PINKY_MCP
18. PINKY_PIP
19. PINKY_DIP
20. PINKY_TIP

Figure 3.2: The order and labels for keypoints that exist in the hands of MediaPipe

For Pose Estimation MediaPipe Extracts 33 keypoints as shown in Figure 3.3. They are calculated in the three-dimension space: X, Y, and Z in addition to the visibility. The visibility is a value indicating if the point is visible or hidden (occluded by another body part) on a frame. Thus, the number of extracted keypoints from the

pose estimation is calculated as follows:

$$\text{Keypoints in pose} \times (\text{Three dimensions} + \text{Visibility}) = (33 \times (3+1)) = 132 \text{ keypoints.} \tag{3.2}$$

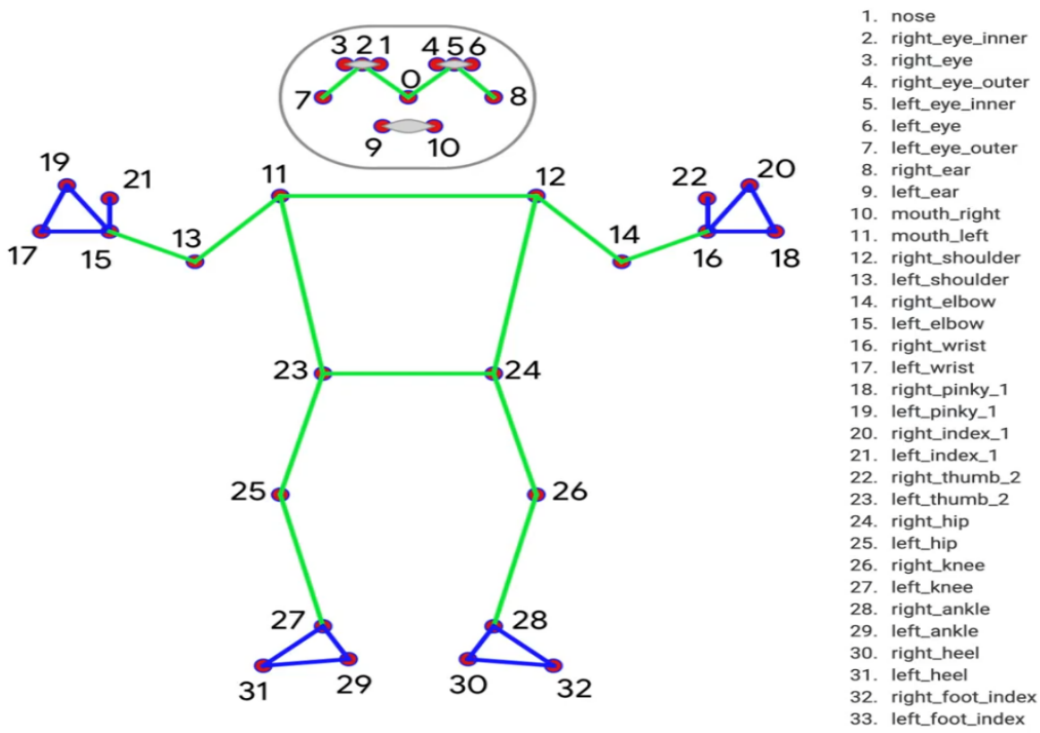


Figure 3.3: The order and labels for keypoints that exist in the pose of MediaPipe

Without including face keypoints, the total number of keypoints for each frame is calculated as follows: keypoints in hands + keypoints in pose = (126 + 132) = 258 keypoints.

3.3 LSTM Model

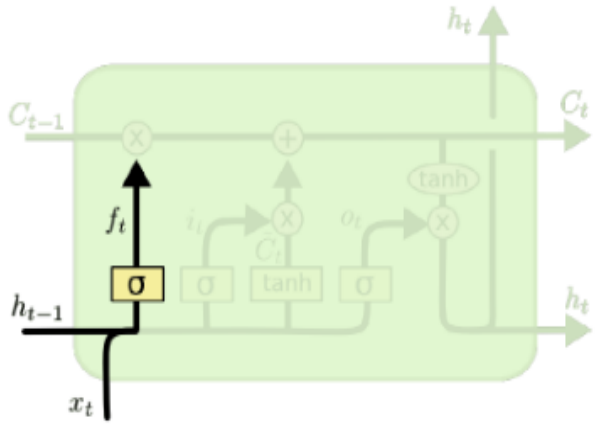
LSTM (Long Short-Term Memory)[11] is a type of recurrent neural network (RNN) that has been found to be effective for sequence modeling tasks such as recognizing sign language gestures. In sign language recognition, the input data consists of a sequence of video frames, and the LSTM model is trained to predict the sign language gesture being performed in each frame.

The LSTM [11] network has the ability to selectively forget and remember information from previous time steps in the sequence, which allows it to capture long-term dependencies in the data. This is particularly useful for sign language recognition, where signs may be performed over a series of frames and require context from previous frames to be accurately recognized.

In combination with other techniques such as feature extraction and pose estimation, LSTM networks have shown promising results in accurately recognizing sign language gestures. These models can be trained on large datasets of sign language videos and can be fine-tuned to recognize specific sign languages or dialects. Overall, LSTM networks are a powerful tool in the field of sign language recognition and have the potential to improve communication and accessibility for deaf and hard-of-hearing individuals.

In an LSTM network, the neural network cells are augmented with a memory cell that allows the network to store information for long periods of time. The memory cell is updated through a series of gates that control the flow of information into and out of the cell, allowing the network to selectively remember or forget information based on the current task.

LSTM networks have become popular in a wide range of applications, including natural language processing, speech recognition, and image captioning, due to their ability to model sequential data with long-term dependencies. They have been shown to be particularly effective in tasks that require modeling and prediction of temporal sequences, where the inputs are a series of data points that are ordered in time. Following are the 3 gates in an LSTM cell.

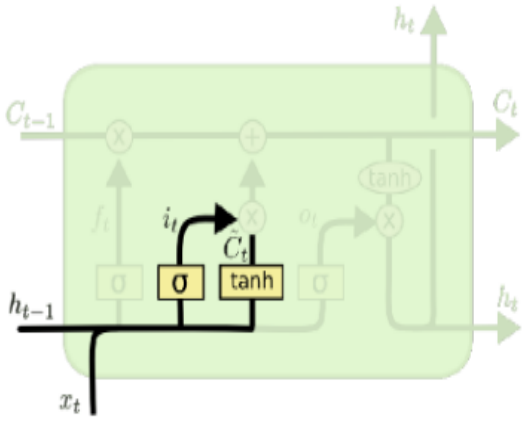


$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

Figure 3.4: Forget Gate.

Forget Gate:

The first step in our LSTM is to decide what information we're going to throw away from the cell state. This decision is made by a sigmoid layer called the "forget gate layer".



$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

Figure 3.5: Store Gate.

Store Gate:

The next step is to decide what new information we're going to store in the cell state. This has two parts. First, a sigmoid layer called the "input gate layer" decides which values we'll update. Next, a tanh layer creates a vector of new candidate values,

$$C \sim t \tag{3.3}$$

, that could be added to the state. In the next step, we'll combine these two to create an update to the state.

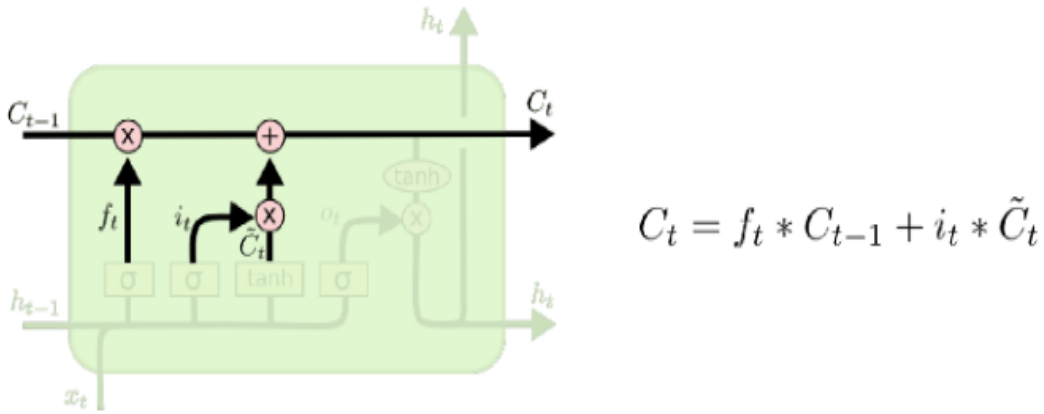


Figure 3.6: Update Gate.

Update Gate:

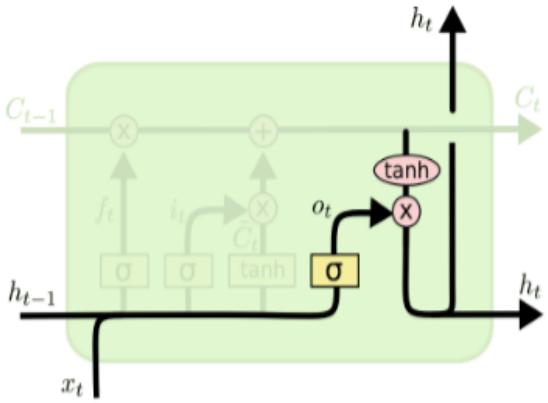
We multiply the old state by ft, forgetting the things we decided to forget earlier. Then we add,

$$it^*C \sim t \tag{3.4}$$

This is the new candidate values, scaled by how much we decided to update each state value.

Output Gate:

Finally, we need to decide what we're going to output. This output will be based on our cell state, but will be a filtered version. First, we run a sigmoid layer which decides what parts of the cell state we're going to output. Then, we put the cell state through tanh and multiply it by the output of the sigmoid gate, so that we only output the parts we decided to. Once the model is trained, the weights are saved for future use. Finally, the model is tested in real-time with a live video feed to evaluate its accuracy and performance.



$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

Figure 3.7: Output Gate.

3.4 Softmax Activation function

The softmax activation function is a type of activation function used in neural networks, which is primarily used for multi-class classification problems. It takes a vector of real numbers as input and returns a probability distribution over a set of possible outcomes.The output of the softmax function is a vector of the same dimension as the input vector, with each element representing the probability of the corresponding class. The probabilities of all the classes sum up to 1, making it suitable for multi-class classification problems.

The formula for the softmax function is as follows:

$$\text{softmax}(x_i) = \frac{\exp(x_i)}{\sum_{j=1}^{n} \exp(x_j)}$$

xi= input value for the ith classes and j iterates over all the classes.

The softmax function exponentiates each element of the input vector, making them positive and then normalizes them by dividing by the sum of the exponentiated values. This normalization ensures that the output of the softmax function is a probability distribution.The softmax function is used in the output layer of neural networks for multi-class classification problems. It is also used in language modeling, image classification, and other applications where multiple classes need to be predicted.

3.5 Software and Hardware Support

Software Support: Jupyter Notebook

Jupyter Notebook is a popular web-based interactive computing environment used for data analysis, data visualization, and machine learning. It allows users to write and execute code in a variety of programming languages, including Python, R, and Julia, and view the output directly in the notebook interface. Jupyter notebooks are composed of cells, which can contain code, text, equations, and visualizations, making them a useful tool for documenting and sharing code.

TensorFlow, on the other hand, is an open-source software library developed by Google for building and training machine learning models. It provides a wide range of tools and resources for developers, including a high-level API for building neural networks, pre-trained models for common tasks, and support for distributed computing. TensorFlow is widely used in industry and academia for a variety of applications, including computer vision, natural language processing, and robotics.

Jupyter Notebook and TensorFlow can be used together for building and training machine learning models. TensorFlow can be installed and used within a Jupyter notebook, allowing for the creation of interactive machine learning workflows. For example, users can write code to load data, preprocess it, build and train a model, and visualize the results all within a single notebook. This can make it easier to experiment with different models and hyperparameters, and to document and share the results.

Hardware Requirements:

For the proper functioning of the system, certain hardware requirements must be met. These requirements include a webcam, a PC or laptop with a minimum of 16GB RAM, and an internet connection. Additionally, an NVIDIA graphics card is recommended for optimal performance. These hardware components work together to provide the necessary resources for image and video processing, machine learning algorithms, and real-time feedback to the user.

Chapter 4

Simulation and Experimental Results

4.1 Mediapipe LSTM model Experimental results

To start the project, the newly created dataset was used to train a Mediapipe-LSTM model for sign language recognition. The Mediapipe framework was used to extract important features from the videos in the dataset, while the LSTM model was utilized for classification.

During the training of the neural network, the categorical accuracy metric is calculated at the end of each epoch to assess the network's performance in classifying the training data. As the number of epochs increases, the network is exposed to the training data more times, and its ability to classify the data is expected to improve.

The categorical accuracy metric measures the proportion of correctly classified samples in each category. As the neural network continues to learn from the training data, it should become better at distinguishing between the different sign language gestures, leading to an improvement in categorical accuracy.

Indeed, in this case, there was an improvement in categorical accuracy as the number of epochs increased, indicating that the neural network was learning from the data and improving its ability to classify the sign language gestures accurately.

```
In [56]: model = Sequential()
         model.add(LSTM(256, return_sequences=True, input_shape=(30,258)))
         model.add(LSTM(128))
         model.add(Dense(actions.shape[0], activation='softmax'))

In [57]: model.compile(optimizer='Adam', loss='categorical_crossentropy', metrics=['categorical_accuracy'])

In [ ]:

In [58]: model.fit(X_train, y_train, epochs=500, callbacks=[tb_callback])
         Epoch 71/500
         9/9 [==============================] - 2s 176ms/step - loss: 0.0828 - categorical_accuracy: 0.9669
         Epoch 72/500
         9/9 [==============================] - 2s 172ms/step - loss: 0.1604 - categorical_accuracy: 0.9351
         Epoch 73/500
         9/9 [==============================] - 2s 181ms/step - loss: 0.1628 - categorical_accuracy: 0.9518
         Epoch 74/500

         9/9 [==============================] - 2s 180ms/step - loss: 0.2586 - categorical_accuracy: 0.8946
         Epoch 75/500
         9/9 [==============================] - 2s 172ms/step - loss: 0.2446 - categorical_accuracy: 0.9162
         Epoch 76/500
         9/9 [==============================] - 2s 176ms/step - loss: 0.1055 - categorical_accuracy: 0.9796
         Epoch 77/500
         9/9 [==============================] - 2s 173ms/step - loss: 0.1032 - categorical_accuracy: 0.9708
         Epoch 78/500
         9/9 [==============================] - 2s 188ms/step - loss: 0.0841 - categorical_accuracy: 0.9774
         Epoch 79/500
         4/9 [============>.................] - ETA: 0s - loss: 0.0729 - categorical_accuracy: 0.9720
```

Figure 4.1: Training of LSTM model.

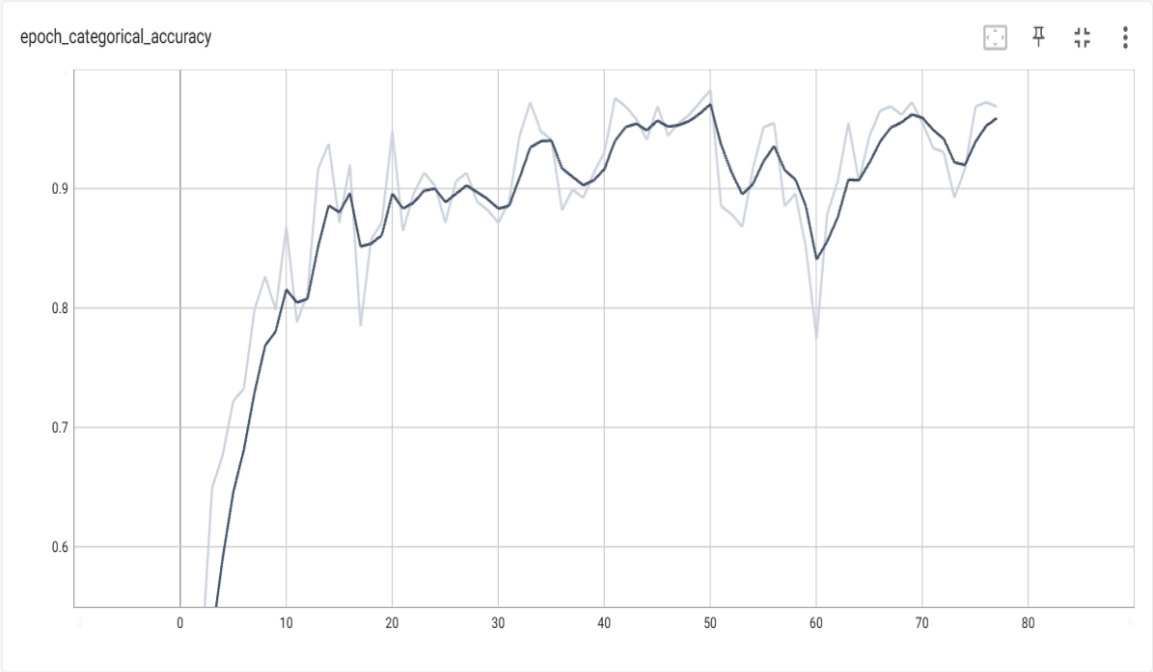


Figure 4.2: Epoch vs categorical accuracy.

The accuracy achieved by the sign language recognition model developed using the Mediapipe-LSTM approach was found to be satisfactory. Additionally, the probability distribution of the model between the 12 classes was accurate, indicating that the model was able to classify the sign language gestures correctly.

The table 4.1 presents an evaluation of the model using a confusion matrix. It

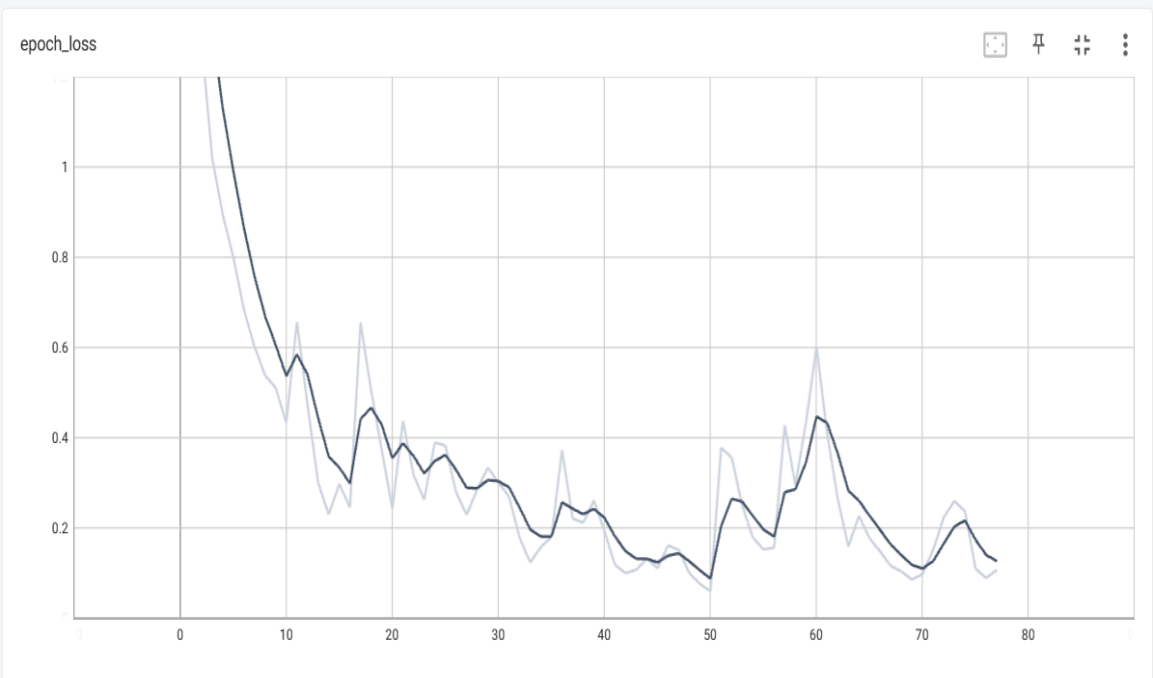


Figure 4.3: Epoch vs categorical loss.

Accuracy	0.916667
Precision	0.937169
Recall	0.916667
F1 Score	0.914106

Table 4.1: Evaluation using Confusion Matrix

contains four performance metrics: accuracy, precision, recall, and F1 score. The accuracy of the model is 0.916667, indicating the proportion of correctly classified instances out of the total number of instances. The precision, which is the proportion of true positives out of all positive predictions, is 0.937169. The recall, which is the proportion of true positives out of all actual positive instances, is also 0.916667. The F1 score, which is the harmonic mean of precision and recall, is 0.914106. Overall, the model appears to have good performance based on these metrics. Class with probability greater than 0.5 is displayed as the output on the screen. This is the prediction Logic used for the output.

Class with probability greater than 0.5 is displayed as the output on the screen. This is the prediction Logic used for the output.

This system is designed to recognize individual signs, the output is a sequence of recognized sign labels corresponding to the input video of sign language. For example given below(Figure 4.5) the real time input video is of someone signing

"HELLO", the output is the recognized sign labels for the individual signs, which is represented by text.

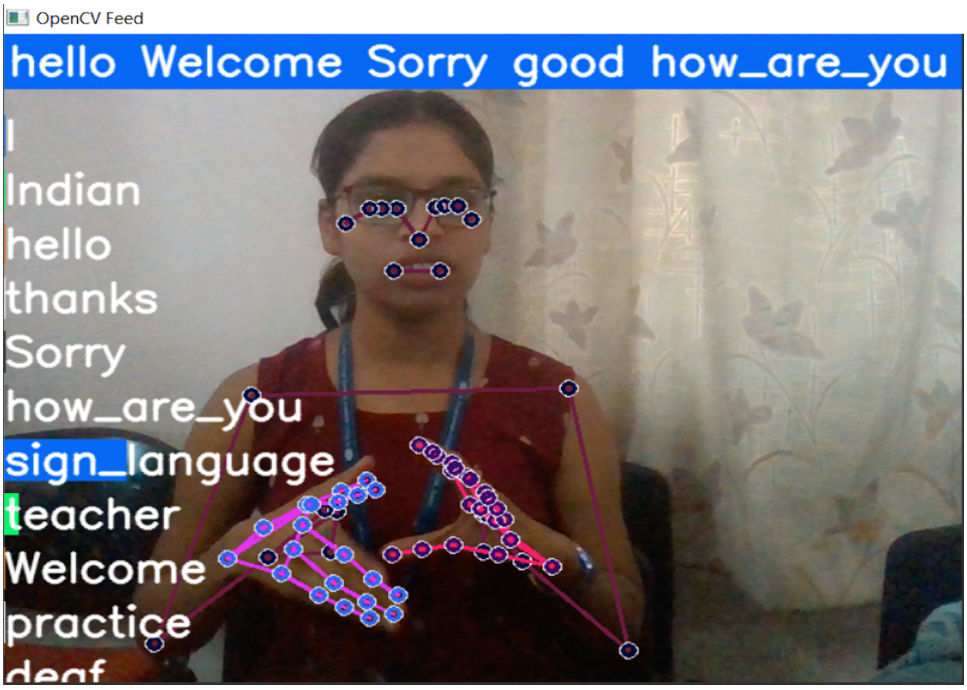


Figure 4.4: Sign language gesture

Figure 4.5: How are you gesture

Figure 4.6: Indian gesture

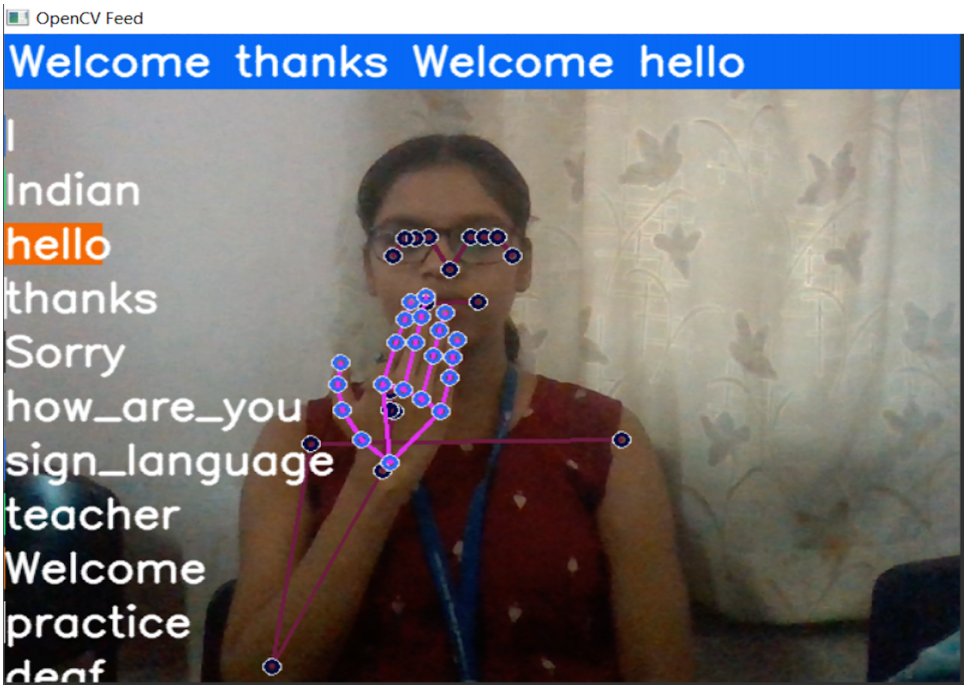


Figure 4.7: Hello gesture

Chapter 5

Conclusion and Future Scope

5.1 Conclusion

Sign language recognition is a challenging task due to the lack of sufficient training data. Researchers have attempted to address this issue by creating new sign language datasets. In this case, a new dataset was created that contains video recordings of 12 different sign language vocabularies. Each vocabulary has 30 videos, resulting in a total of 360 videos in the dataset.

Experiments were then conducted on this new dataset to evaluate the performance of a sign language recognition model. The model was able to achieve an accuracy rate of over 91%, indicating that it is effective in recognizing sign language gestures from videos. The accuracy rate achieved by the model is a promising result, suggesting that the use of video-based datasets can be a useful approach for training sign language recognition models.

By combining feature extraction and pose detection with the trained LSTM model, the accuracy of dynamic sign language recognition was improved. The LSTM model is a type of recurrent neural network that is well-suited for sequence modeling tasks such as recognizing sign language gestures. MediaPipe Holistic pose detection network was used to extract keypoints and estimate poses for each frame of the video. This data was then fed into the LSTM model to train it to recognize different signs.

To determine the appropriate number of neural units to use in the model, the researchers experimented with varying numbers and measured the resulting accuracy. They found that by adjusting the number of units, they were able to

achieve a level of accuracy that was suitable for their needs.

Overall, the study provides a promising approach to recognizing Indian Sign Language gestures and lays the groundwork for a larger, more comprehensive project that could capture the entire vocabulary of the language.

5.2 Future Work

A grammar correction model for sign language could help improve the accuracy and fluency of sign language communication. Sign language, like spoken language, has its own grammatical rules and structures that can be challenging for non-native signers or those learning sign language as a second language.

A grammar correction model for sign language would need to be trained on a large dataset of signed language samples, with annotations indicating where errors in grammar or syntax occur. The model could use this data to learn the grammatical rules and structures of sign language, and identify common errors that signers make.

Once trained, the model could be integrated into a sign language translation or communication system, where it would analyze the signer's input in real-time and provide suggestions for corrections or improvements in grammar and syntax. The model could use techniques such as natural language processing and machine learning to identify common grammatical errors and suggest appropriate corrections.

Overall, a grammar correction model for sign language could be a valuable tool for improving the accuracy and fluency of sign language communication, and could help bridge the communication gap between signers and non-signers.

5.3 Application

Sign language recognition models can be used in a variety of applications to improve accessibility for individuals who are deaf or hard of hearing. Here are a few examples:

1. Sign language translation: A sign language recognition model can be used to translate sign language into written or spoken language, allowing non-signers to communicate with signers.

2. Accessibility for deaf and hard of hearing individuals: Sign language

recognition can be used to provide accessibility for deaf and hard of hearing individuals in public spaces, such as train stations, airports, or hospitals, where they may need to communicate with non-signers.

3. Education: Sign language recognition models can be used to develop educational tools for sign language learners. For example, a model could be used to provide real-time feedback on sign language accuracy or to generate sign language videos for educational purposes.

4. Communication aids: Sign language recognition models can be used to develop communication aids for deaf or hard of hearing individuals, such as smartphones apps or wearable devices.

5. Human-robot interaction: Sign language recognition models can be used to improve human-robot interaction for deaf or hard of hearing individuals. For example, a sign language recognition model could be integrated into a robot to allow it to understand and respond to sign language commands.

Overall, sign language recognition models have the potential to improve accessibility and communication for individuals who are deaf or hard of hearing, and can be applied in a variety of contexts, from education to public spaces to human-robot interaction.

References

[1] Disha Gangadia,Varsha Chamaria, Vidhi Doshi, Jigyasa Gandhi,"Indian Sign Language Interpretation and Sentence Formation", *2020 IEEE Pune Section International Conference (PuneCon) Vishwakarma Institute of Technology, Pune India.*, Dec 16-18, 2020.

[2] Yiyuan Li, Antonios Anastasopoulos, Alan W Black, "Towards Minimal Supervision BERT-based Grammar Error Correction",*(IJACSA) International Journal of Advanced Computer Science and Applications*, Vol. 12, No. 8, 2021.

[3] Advaith Sridhar,Rohith Gandhi Ganesan,Pratyush Kumar,Mitesh Khapra,"INCLUDE: A Large Scale Dataset for Indian Sign Language Recognition",*Multimedia (MM'20)*,October 12-16, 2020.

[4] S. Reshna, A. Sajeena, and M. Jayaraju, "Recognition of static hand gestures of Indian sign language using CNN",*AIP Conference Proceedings 2222, 030012 (2020)*,16 April 2020.

[5] R. Dhiman, "A deep learning approach for Indian sign language gestures classification with different backgrounds",*((ICMAI))*,27 February 2021.

[6] Palash Kamble, Dr Rathna G N, "Word Level Sentence Generation using Deep Learning for Indian Sign Language",*Department of Electrical Engineering Indian Institute of Science, Bangalore, India*, 2021.

[7] Kinjal Mistree, Devendra Thakor, Brijesh Bhatt, "Towards Indian Sign Language Sentence Recognition using INSIGNVID: Indian Sign Language Video Dataset",*(IJACSA) International Journal of Advanced Computer Science and Applications*,Vol. 12, No. 8, 2021

[8] Sridhar, Advaith; Ganesan, Rohith Gandhi; Kumar, Pratyush; Khapra, Mitesh https://zenodo.org/record/4010759#.YvNcHbZByM9,*INCLUDE: A Large Scale Dataset for Indian Sign Language Recognition*, October 16, 2020

[9] Data by World Health Organization(WHO) https://www.who.int/news-room/fact-sheets/detail/deafness-and-hearing-loss,*Deafness and hearing loss*,April 1, 2021

[10] https://colah.github.io/posts/2015-08-Understanding-LSTMs/,*LSTM*

[11] https://developers.google.com/mediapipe/solutions/vision/hand_landmarker,*Mediapipe*

Figure 5.1: FLOWCHART

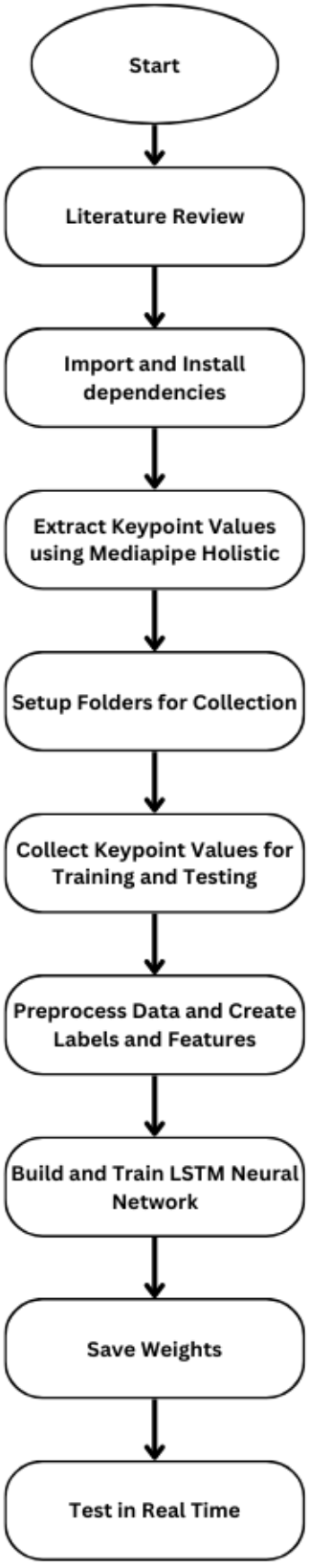


Figure 5.2: TIMELINE OF OUR PROJECT SEM VII

TIMELINE CHART FOR SEM VIII																
MONTH	JANUARY				FEBRUARY				MARCH				APRIL			
WEEK NO.	W1	W2	W3	W4	W1	W2	W3	W4	W1	W2	W3	W4	W1	W2	W3	W4
WORK TASKS																
1) Execution of project																
Installing all dependencies																
Data Collection																
Model Training (Mediapipe & LSTM)																
Model Testing (Mediapipe & LSTM)																
2) Output and Results																
Output with three classes																
Output with multiple classes																
3) Report																
Poster making																
Report Writing																
Black Book																