SIGN LANGUAGE RECOGNITION

A PROJECT REPORT

Submitted to

ASSAM DON BOSCO UNIVERSITY

by

LAXMAN BASUMATARY

DC2019BTE0103

NONGTHOMBAM GUNSUN

DC2019BTE0030

KAISER MISAAL

DC2019BTE0078

in partial fulfilment for the award of the degree

of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE&ENGINEERING
SCHOOL OF TECHNOLOGY
ASSAM DON BOSCO UNIVERSITY
AZARA,GUWAHATI 781 017,
ASSAM, INDIA.

BATCH (2019- 2023)

CERTIFICATE

This is to certify that the project report entitled "**SIGN LANGUAGE RECOGNITION**" submitted by **LAXMAN BASUMATARY (DC2019BTE0103), NONGTHOMBAM GUNSUN (DC2019BTE0030)** and **KAISER MISAAL (DC2019BTE0078)** to the Assam Don Bosco University, Guwahati, Assam, in partial fulfilment of the requirement for the award of Degree of Bachelor of Technology in Computer Science and Engineering is a bonafide record of the project work carried out by them under my supervision during the year 2023.

Supervisor:

Dr. Alexy Bhowmick
Assistant Professor (Stage III)
Department of Computer Science and Engineering
School of Technology,
Assam Don Bosco University.

CERTIFICATE

This is to certify that the Project Report entitled "**SIGN LANGUAGE RECOGNITION**" submitted by **LAXMAN BASUMATARY, NONGTHOMBAM GUNSUN** and **KAISER MISAAL** bearing Roll Number **DC2019BTE0103, DC2019BTE0030** and **DC2019BTE0078** respectively to the Assam Don Bosco University, Guwahati, Assam, in partial fulfilment of the requirement for the award of Degree of Bachelor of Technology in Computer Science and Engineering is a bonafide record of the project work carried out by them during the year 2022-2023.

Dr. Bobby Sharma
Head of the Department
Dept. Of CSE, School of Technology,
Assam Don Bosco University
Date: .....

Prof. Manoranjan Kalita
Director, School of Technology
Assam Don Bosco University

Date: .....

EXAMINATION CERTIFICATE

This is to certify that **LAXMAN BASUMATARY, NONGTHOMBAM GUNSUN** and **KAISER MISSAL** bearing Roll Number **DC2019BTE0103, DC2019BTE0030** and **DC2019BTE0078** respectively of the **Department of Computer Science & Engineering** has carried out the project work in a manner satisfactory to warrant its acceptance and also defended it successfully.

We wish them all the success in their future endeavours.

Examiners:

01. External Examiner:

02. Internal Examiner:

03. Internal Examiner:

DECLARATION

We hereby declare that the project work entitled "**SIGN LANGUAGE RECOGNITION**" submitted to the Assam Don Bosco University, Guwahati, Assam, in partial fulfilment of the requirement for the award of Degree of Bachelor of Technology in Computer Science and Engineering is an original work done by us under the guidance of **Dr. Alexy Bhowmick (*Assistant Professor*, *Dept of Computer Science & Engineering, School of Technology, Assam Don Bosco University)*** and has not been submitted for the award of any degree.

(Signature of the student)
LAXMAN BASUMATARY
DC2019BTE0103
Department of Computer Science & Engineering
School of Technology, Assam Don Bosco University.

(Signature of the student)
NONGTHOMBAM GUNSUN
DC2019BTE0030
Department of Computer Science & Engineering
School of Technology, Assam Don Bosco University.

(Signature of the student)
KAISER MISAAL
DC2019BTE0078
Department of Computer Science & Engineering
School of Technology, Assam Don Bosco University.

ACKNOWLEDGEMENT

Firstly, we would like to thank Assam Don Bosco University for providing us with proper education and knowledge during our course of Computer Science Engineering with its amazing faculty members and a great environment where everyone can learn the skills they need to for a bright future.

Secondly, we would like to thank our guide, Dr. Alexy Bhowmick, for sharing with us, an enormous amount of information regarding our project, from her past experiences in this field, which has helped us more than anything in the whole process of completing this project.
We would like to thank Dr. Bobby Sharma, Head of the Department, Dept. of CSE, School of Technology, Assam Don Bosco University, who has given us this golden opportunity to complete this project.

We would also like to express our gratitude to our Project Co-ordinators Dr. Syed Sazzad Ahmed, Assistant Professor, Dept of CSE, School of Technology and Mr. Alok Choudhury, Assistant Professor, Dept of CSE, School of Technology for their constant support and guidance.

Lastly, we would like to thank our friends, for their relentless support in augmenting the value of work, our families, for being considerate and appreciative throughout.

ABSTRACT

Sign language recognition refers to the process of interpreting and translating sign language gestures into written or spoken language. This study proposes a Deep Learning-based model that recognizes the words from a person's hand gestures. Deep learning models, namely, LSTM (Long Short Term Memory) and Bi-LSTM (Bidirectional Long Short Term Memory), is used to recognize signs from isolated Indian Sign Language (ISL) video frames. In the Bi-LSTM model we added a dropout layer to prevent over fitting and improve the model's ability to generalize well to unseen data. The trained LSTM model could recognize eleven different signs with an accuracy of 80% and the Bi-LSTM yielded an accuracy of 87% on the Isolated Indian Sign Language (IISL) dataset.

In future research, the developed model could be utilized for interpreting continuous sign language instead of isolated signs. Large scale sign language datasets can provide more diverse examples for the model which can improve model performance.

Keywords: *Indian Sign Language, LSTM, Bidirectional LSTM, Key-frame, Deep Learning.*

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

ISL	Indian Sign Language
ASL	American Sign Language
SSD	Single Shot-Multibox Detector
CNN	Convolutional Neural Network
RNN	Recurrent Neural Network
B-RNN	Bidirectional Recurrent Neural Network
LSTM	Long Short Term Memory
BI-LSTM	Bidirectional Long Short Term Memory
GRU	Gated Recurrent Units
TP	True Positive
TN	True Negative
FP	False Positive
FN	False Negative
RGB	Red Blue Green

CONTENTS

CHAPTER 1

INTRODUCTION

Communicating with the hearing and speech-impaired population has been a problem for people as they primarily rely on sign language for communicating. According to World Health Organization over 5% of the world's population require rehabilitation to address their 'disabling' hearing loss (432 million adults and 34 million children). These people deal with difficulties in interacting with others especially when joining workforce, education, healthcare, and transportation. Underprivileged areas and remote areas, however are difficult to assign and train interpreters. Which means those population are lacking a vital necessity that all human beings require in order to live a normal life.

Therefore, bridging the gap between these two worlds is of utmost necessity. This project aims to capture and recognize sign language using Deep Learning methods and provide the output as English text.

1.1. Title: Sign Language Recognition

1.2. Objective

Our objective in this project is to develop a sign language recognition system that can easily recognize a variety of different hand gestures signs and translate it into text.
The objectives of the project are listed below:

- Detect hand(s) present in the frame.
- Detect gestures: Identify the gesture of the detected hand(s).
- Translate and view the meaning of the gesture in text.

1.3. Problem definition

There are various sign languages like ASL (American Sign Language), ISL (Indian Sign Language), etc. which have different patterns and ways of communication. Sign languages, like any other language, have their own grammar, vocabulary and expressions. So,

the problems mainly will be differentiating between the types of sign languages and translating it into an understandable form such as text. The gestures can vary greatly between signers even for the same words or phrase. This variability makes it difficult for recognition systems to generalize different signers. Sign languages are dynamic and involve continuous motion which makes it a difficult challenge to process gesture in real time. Various environmental factors such as lighting and occlusion of signers affect the performance of the model.

Advances in computer vision and deep learning have made it possible to face those challenges. Hence, assessments have to be made and an efficient technique must be chosen to translate whatever is conveyed by the signers.

Challenges:

- Sign languages comprises of different complex hand gestures to represent a word. Therefore, the core problem lies in distinguishing the different types of signs and translating them into an understandable form such as text.

- The dynamic nature of sign languages and the constant motion they involve makes it a difficult challenge to process gesture in real time. Various environmental factors such as lighting and occlusion of signers also affects the performance of the model.

- Deep Learning models need a large amount of data to recognise sign language. Collecting and annotating sign language data requires time and effort which limits the ability to use available data.

1.4. Motivation

Deafness is not a choice and it should not be a factor which hampers communication. Everyone should be equal and one should be able to communicate distinctly in whichever way one likes. We have been fascinated by Artificial Intelligence and Machine Learning since we were introduced to these terms and with these domains. Going through various reports we found out that there are various projects based on Sign Language Recognition and that we can also chip in a little effort in making communication easier among differently able people.

Chapter 2

Literature Review

2.1. SSD: Single Shot Multibox Detector [1]

SSD is a method for detecting objects in images using a single deep neural network. SSD discretizes the output space of bounding boxes into a set of default boxes over different aspect ratios and scales per feature map location. At prediction time, the network generates scores for the presence of each object category in each default box and produces adjustments to the box to better match the object shape. The network combines predictions from multiple feature maps with different resolutions to naturally handle objects of various sizes. SSD is simple relative to methods that require object proposals because it completely eliminates proposal generation and subsequent pixel or feature resampling stages and encapsulates all computation in a single network. This makes SSD easy to train and straightforward to integrate into systems that require a detection component. Experimental results on the PASCAL VOC, COCO, and ILSVRC datasets confirm that SSD has competitive accuracy to methods that utilize an additional object proposal step and is much faster, while providing a unified framework for both training and inference. For 300 × 300 input, SSD achieves 74.3% mAP on VOC2007 test at 59 FPS on a Nvidia Titan X and for 512 × 512 inputs, SSD achieves 76.9% mAP, outperforming a comparable state-of-the-art Faster R-CNN model. Compared to other single stage methods, SSD has much better accuracy even with a smaller input image size.

2.2. Real Time Hand Gesture Recognition for Human Computer Interaction [2]

Most of the human computer interaction interfaces that are designed today require explicit instructions from the user in the form of keyboard taps or mouse clicks. As the complexity of these devices increase, the sheer amount of such instructions can easily disrupt, distract and overwhelm users. A novel method to recognize hand gestures for human computer interaction, using computer vision and image processing techniques, is proposed in this paper. The proposed method can successfully replace such devices (e.g. keyboard or mouse) needed

for interacting with a personal computer. The method uses a commercial depth RGB camera called Senz3D, which is cheap and easy to buy as compared to other depth cameras. The proposed method works by analysing 3D data in real time and uses a set of classification rules to classify the number of convexity defects into gesture classes. This results in real time performance and negates the requirement of any training data. The proposed method achieves commendable performance with very low processor utilization.

This paper makes use of morphological operation for hand detection. Morphological Operations is a broad set of image processing operations that process digital images based on their shapes. In a morphological operation, each image pixel is corresponding to the value of other pixel in its neighbourhood. By choosing the shape and size of the neighbourhood pixel, you can construct a morphological operation that is sensitive to specific shapes in the input image.

Types of Morphological operations are as follows:

- **Dilation:** Dilation adds pixels on the object boundaries.
- **Erosion**: Erosion removes pixels on object boundaries.
- **Opening**: The opening operation erodes an image and then dilates the eroded image, using the same structuring element for both operations.
- **Closing**: The closing operation dilates an image and then erodes the dilated image, using the same structuring element for both operations.

The number of pixels added or removed from the object in an image depends on the shape and size of the structuring element used to process the image. In the morphological dilation and erosion operations, the state of any given pixel in the output image is determined by applying a rule to the corresponding pixel and its neighbours in the input image. The rule used to process the pixels defines the morphological operation as dilation or erosion.

2.3. Finger spelling recognition in the wild with iterative visual attention [3]

Sign language recognition is a challenging gesture sequence recognition problem, characterized by quick and highly coarticulated motion. This paper focuses on recognition of fingerspelling sequences in American Sign Language (ASL) videos collected in the wild,

mainly from YouTube and Deaf social media. Most previous work on sign language recognition has focused on controlled settings where the data is recorded in a studio environment and the number of signers is limited. This work aims to address the challenges of real-life data, reducing the need for detection or segmentation modules commonly used in this domain. This paper proposes an end-to-end model based on an iterative attention mechanism, without explicit hand detection or segmentation. The approach dynamically focuses on increasingly high-resolution regions of interest. It outperforms prior work by a large margin. It also introduces a newly collected dataset of crowdsourced annotations of fingerspelling in the wild, and show that performance can be further improved with this additional data set.

2.4. Bidirectional Recurrent Neural Networks [4]

The paper presents a study on bidirectional recurrent neural networks (BRNNs), which are an extension of regular recurrent neural networks (RNNs). Unlike RNNs, which can only use input information up to a certain point in time, BRNNs can be trained in both positive and negative time direction, giving them the ability to use information from the entire sequence. The paper explains the structure and training process of BRNNs and demonstrates their superiority over other approaches through regression and classification experiments on artificial and real data. The second part of the paper shows how the BRNN structure can be modified to estimate the conditional posterior probability of complete symbol sequences without making any assumptions about the shape of the distribution. The experiments on real data show the effectiveness of this approach.

2.5. Long Short-Term Memory [5]

LSTM (Long Short-Term Memory) is a type of Recurrent Neural Network (RNN) that is widely used for processing sequential data, such as time series, text, and speech. The LSTM architecture was specifically designed to handle the problem of vanishing gradients in traditional RNNs, where information from the early part of the sequence becomes increasingly difficult to propagate through the network as the sequence becomes longer.

In an LSTM cell, information can be stored in the cell state and controlled by the use of gates (input, forget, and output gates) that can be opened or closed. This allows the LSTM

to maintain a persistent memory of the relevant information from the previous time steps and pass it along to the next time step, effectively capturing long-term dependencies in the data.

LSTMs are widely used for tasks such as language modelling, sentiment analysis, and speech recognition, where the data is sequential in nature and the previous information is important for understanding the current input. Additionally, LSTMs are commonly used in deep learning frameworks, making them a popular tool for developers and researchers in the field.

2.6. Deep sign: Sign Language Detection and Recognition Using Deep Learning [6]

In this paper, the authors explore the application of LSTM and GRU for recognizing Indian Sign Language (ISL) gestures using the IISL2020 dataset. The study demonstrates that the proposed model surpasses existing methods in terms of performance when recognizing commonly used words in ISL, such as ‘hello’, ‘good morning’, and ‘work’. By increasing the number of layers in LSTM and GRU, as well as utilizing a sequential combination of LSTM followed by GRU, the model achieves higher accuracy in ISL recognition. The proposed model, consisting of a single layer of LSTM followed by GRU, achieves around 97% accuracy over 11 different signs.

Currently, the developed models focus on isolated signs; however, there is potential to extend this approach to interpret continuous sign language, which would involve generating syntax, particularly within the context of ISL. Additionally, incorporating vision transformers can potentially yield more precise results compared to feedback-based learning models.

2.7. MediaPipe’s Landmarks with RNN for Dynamic Sign Language Recognition [7]

This paper focuses on Dynamic Sign Language Recognition, employing three RNN models (GRU, LSTM, BILSTM) and utilizing the DSL10-Dataset. The feature extraction phase makes use of the MediaPipe framework. Through two comprehensive experiments, the paper demonstrates the superior performance of the proposed method in recognizing DSL from two different perspectives.

To implement the proposed method, three fundamental steps are outlined. Firstly, it is necessary to prepare a dataset comprising videos with an equal number of frames. Next, the

input data is passed through the MediaPipe framework, which extracts essential hand, face, and pose keypoints from each video frame. Finally, during the training phase, the extracted keypoints are fed into one of the pre-existing RNN models: GRU, LSTM, or BILSTM.

The findings reveal interesting patterns. GRUs exhibit better performance than LSTM networks on sequences with low complexity, whereas LSTMs excel on sequences with high complexity. It is worth noting that LSTM and BILSTM models require more parameters and nodes compared to GRU. Moreover, the study demonstrates that the inclusion or exclusion of face keypoints has minimal impact on train and test accuracy. However, including face keypoints significantly increases the size of the keypoints, which negatively affects prediction and learning time.

In their research, the authors conducted two experiments on their dataset, DSL10-Dataset, using RNN models to evaluate the accuracy of dynamic sign language recognition. The purpose was to compare the performance of the model with and without the inclusion of face keypoints. The results of the experiments revealed that their model achieved an accuracy rate exceeding 99%.

Year	Title	Authors	Summary	Dataset used	Reference No.
2016	*SSD: Single Shot MultiBox Detector*	Liu, Wei, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg.	This paper introduces SSD, a fast single-shot object detector for multiple categories. A key feature of this model is the use of multi-scale convolutional bounding box outputs attached to multiple feature maps at the top of the network.	PASCAL VOC 2007 dataset	1
2016	*Real Time Hand Gesture Recognition for Human Computer Interaction*	Agrawal, Rishabh, and Nikita Gupta	The proposed method for real time hand gesture recognition produces commendable results with high accuracy and precision and can be used in a real-world scenario for interaction with a computer.	CMU Graphics Hand Gesture Database	2
2019	*Fingerspelling recognition in the wild with iterative visual attention*	Shi, Bowen, Aurora Martinez Del Rio, Jonathan Keane, Diane Brentari, Greg Shakhnarovich, and Karen Livescu	The proposed model outperforms prior work on fingerspelling recognition in the wild by a large margin. The iterative attention mechanism is effective at dynamically focusing on the most relevant regions of interest in the input video. The newly collected data set of crowdsourced annotations of fingerspelling in the wild is a valuable resource for future research on sign language recognition.	ChicagoFSWild& ASL-MNIST	3
1997	*Bidirectional recurrent neural networks*	Schuster, Mike, and Kuldip K. Paliwal.	This paper proposed bidirectional structure can be easily modified to allow efficient estimation of the conditional posterior probability of complete	The Wall Street Journal corpus,The Switchboa	4

Year	Title	Authors	Summary	Dataset used	Reference No.
			symbol sequences without making any explicit assumption about the shape of the distribution.	rd corpus& The Penn Treebank corpus	
1997	*Long short-term memory*	Hochreiter, Sepp, and Jürgen Schmidhuber	The authors then introduce the LSTM network, which is a new type of RNN that is able to learn long-term dependencies by using a special type of unit called a gated recurrent unit (GRU). GRUs are able to control the flow of information through the network, which helps to prevent the vanishing gradient problem.	The German Traffic Sign Recogniti on Benchmar k (GTSRB), The Penn Treebank corpus& The Wall Street Journal corpus	5
2022	*Deep sign: Sign Language Detection and Recognition Using Deep Learning*	Deep Kothadiya, Chintan Bhatt Krenil Sapariya Kevin Patel, Ana-Belén Gil-González 2 and Juan M. Corchado	This paper introduces a model for Indian Sign Language (ISL) recognition using LSTM and GRU on the IISL2020 dataset. The proposed model outperforms existing methods, especially for common words like 'hello' and 'good morning'. Increasing the layer count and using LSTM followed by GRU improves the accuracy of ISL recognition.	IISL2020	6
2022	*MediaPipe's Landmarks with RNN for Dynamic Sign Language Recognition*	Gerges H. Samaan, Abanoub R. Wadie, Abanoub K. Attia, Abanoub M. Asaad,	This paper presents Dynamic Sign Language Recognition using GRU, LSTM, and BILSTM models on DSL10-Dataset. MediaPipe framework extracts features. Results show superior performance in two	DSL10	7

Year	Title	Authors	Summary	Dataset used	Reference No.
		Andrew E. Kamel 1, Salwa O. Slim, Mohamed S. Abdallahand Young-Im Cho	experiments. The method involves dataset preparation, feature extraction with MediaPipe, and training with GRU, LSTM, or BILSTM. GRUs excel with low complexity sequences, while LSTMs perform better with high complexity sequences. Face keypoints slightly affect accuracy but increase computation time.		
2016	*You Only Look Once: Unified, Real-Time Object Detection*	Joseph Redmon, Santosh Divvala, Ross Girshick and Ali Farhadi	The main contribution of this paper: Introduce YOLO, a unified model for object detection. Our model is simple to construct and can be trained directly on full images. Unlike classifier-based approaches, YOLO is trained on a loss function that directly corresponds to detection performance and the entire model is trained jointly.	PASCAL VOC 2007 dataset	8
2022	*Chinese Sign Language Recognition with Batch Sampling ResNet-Bi-LSTM*	Chung, Wan-Young, Haokai Xu, and Boon Giin Lee	A study aimed to develop a sign language recognition framework using a multi-modal approach and spatio-temporal features to address the lack of sign language interpreters in Chinese society. The framework consisted of handshape recognition, movement tracking, and sign recognition components. The proposed model achieved a true recognition rate of 98.75% on a dataset of continuous Chinese sign language,	Chinese sign language (CSL) dataset	9

Year	Title	Authors	Summary	Dataset used	Reference No.
			outperforming existing methods. The application of this framework in public service sectors like banks, hospitals, and police stations could help bridge the communication gap between the Deaf community and hearing individuals.		

Table 2.1. Literature summary

Chapter 3

Feasibility Study and Requirement Analysis

3.1. System Requirements

3.1.1. Software Requirement (Developing)

Components	Name	Version
Operating System	Windows	10 and above.
Platform	Jupyter Notebook	5.0
Machine Learning FrameWork	Tensorflow SckitLearn OpenCV Keras NumPy Matplotlib MediaPipe	2.10.0 0.20 4.5.5 2.5.0 1.19.5 3.6.2 0.8.11
Programming Language	Python 3	3.10

Table 3.1. Software requirement

3.1.2. Hardware Requirement (Developing)

Components	Specification	Version/Generation
Processor	Intel i5 Ryzen 5	8^{th} Gen and above. 3^{rd} Gen and above.
RAM	8 GB	DDR4
GPU	GTX 1650 and above	GDDR5
Camera	Laptop Webcam	720p HD

Table 3.1. Hardware requirement

3.2. Scheduled Feasibility

The estimated time is illustrated with the help of work break down structure and Gantt chart provided below.

3.2.1. Work Breakdown Structure

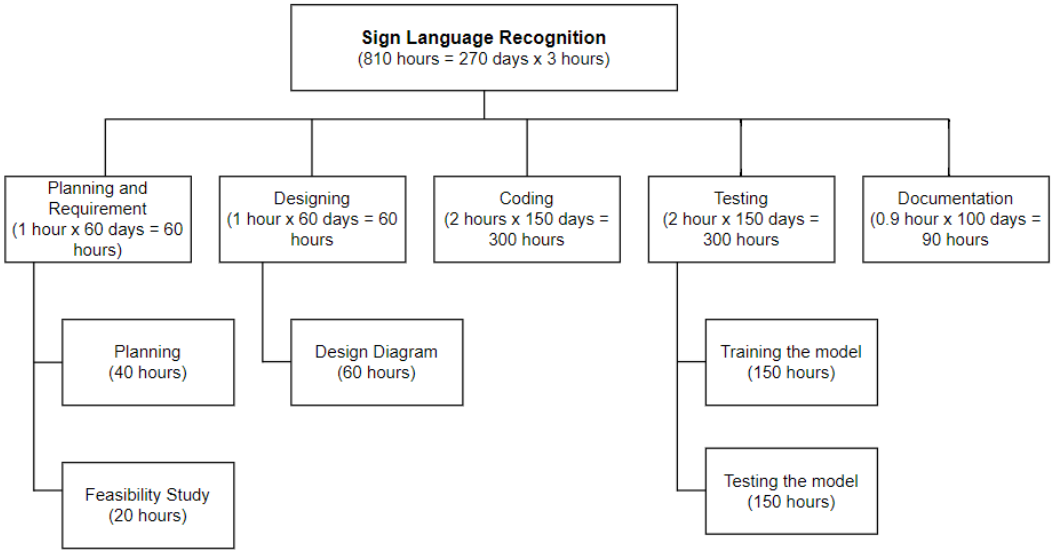

Fig 3.1. Work Breakdown Structure

3.2.2. Gantt Chart:

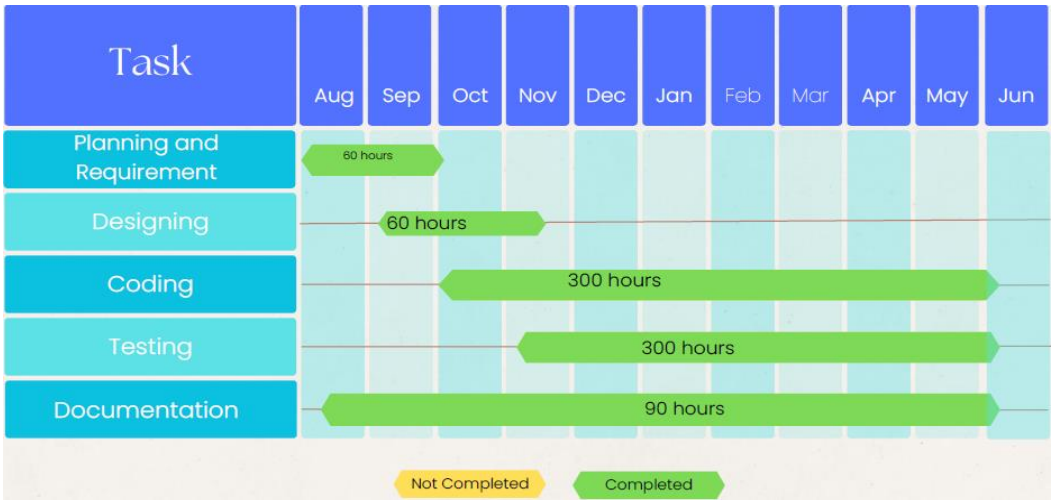

Fig 3.2. Gantt Chart

3.3. Feasibility Study

3.3.1. Economic Feasibility

Our project is economically feasible as we used open-source resources and trained on your own machines. Open-source software and resources provide significant cost savings, as they allow to utilize existing tools, libraries, and frameworks without having to invest in proprietary solutions. Hence it is economically feasible.

3.3.2. Operational Feasibility

This model was trained on a small dataset hence it requires less computation compared to other models and the requirements to use it is going to be really low and thus anyone can make use of it with ease with lower end devices, easing the process of communication.

3.3.3. COCOMO Model

The Constructive Cost Model (COCOMO) is a procedural software cost estimation model. The model parameters are derived from fitting a regression formula using data from historical projects.

The basic COCOMO equations take the form:

- Effort Applied (E) = $a(KLOC)^b$ [person-months]
- Development Time (D) = c(Effort Applied) d [months]
- People Required (P) = Effort Applied/Development Time [Count] where, KLOC is the estimated number of delivered lines (expressed in thousands) of code.

The constants a, b, c and d are given in the following table:

Software Project	a	b	c	D
Organic	2.4	1.05	2.5	0.38
Semi-detached	3	1.12	2.5	0.35
Embedded	3.6	1.2	2.5	0.32

Table 3.3. COCOMO model

Our project type is SEMI-DETACHED.

Estimated LOC = 6000

Now basic COCOMO equation of our project is:

Effort Applied (E) = a (KLOC)^{b} person/month

= 3.0 (6)1.12 person-month

= 22 person-month 15

Development Time (D)= c (Effort Applied)d [months]

= 2.5 (22) 0.35 [months]

= 7.35 months (approximately)

People Required (P) = Effort Applied/Development Time [count]

= 22 / 7.35 [count]

= 3 [count]

Chapter 4

Proposed Plan

4.1. Methodology

In the first proposed method, we use the combinational capabilities of MediaPipe Hands library and LSTM (Long Short-Term Memory) for hand sign recognition. MediaPipe Hands is a finger tracking library developed by Google to support real-time hand and finger tracking for various application. MediaPipe Hands can help us track hand and finger movements and extract the hand landmarks or keypoints from a sequence of video frames which is used to train the LSTM model.

In the second method we followed the same steps as above method for pre-processing the image dataset. We then trained the Bi-LSTM (Bidirectional Long Short Term Memory) model to recognize signs from isolated Indian Sign Language (ISL) video frames.

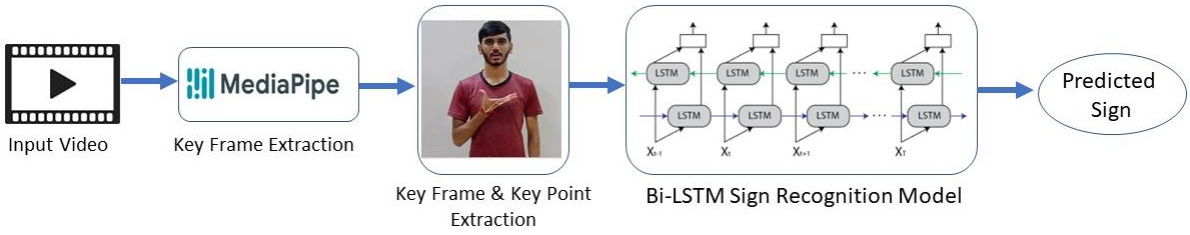

Fig 4.1. Workflow for the proposed Indian Sign Language (ISL) Recognition system

4.1.1. Hand detection:

MediaPipe Hands used SSD (Single Shot Detection) architecture to locate the hands from the RGB input image. In the hand segmentation stage the library uses a combination of colour and depth information to separate the hand regions from the background by producing a binary mask that indicates the pixels which belongs to the hand.

4.1.2. Feature Extraction:

After detecting the hands we need to extract the hand landmarks from the input image. The library uses a deep neural network to predict the locations of keypoints on the hand such

as fingertips and the wrist. This provides us a compact representation of the hand shape and position. Using these hand landmarks, we can track the fingers and estimate the hand posture. Various gestures were detected using this method and the co-ordinates of the key points were stored in a NumPy array.

The hand landmarks are shown below in the figure which was taken from the official mediapipe webpage.

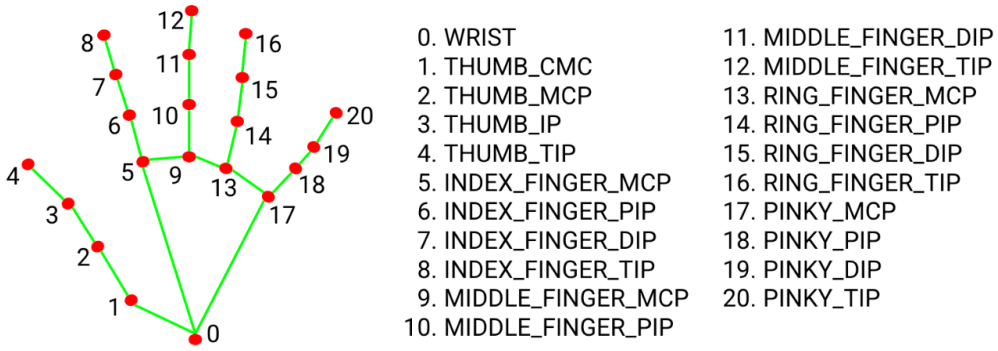

Fig 4.2. Hand Landmarks[1]

4.1.2. Sign recognition:

To perform sign recognition, we used the hand tracking solution from the MediaPipe library to detect the user's hand and its position in real time. We have extracted the keypoints of 26 frames from video sequences. The dataset, on which this model was trained, consists of eleven words (bye, good, hello, house etc.) and for each word we had 58 video sequences.

Using this dataset, we trained the LSTM and Bi-LSTM model to recognize different signs based on the hand position and movements.

We feed the keypoints from each frame of the input video to train both models.

4.2. Dataset Description:

We used the Isolated Indian Sign Language (IISL2020)[2] dataset for training and testing. This dataset was collected from 16 subjects aged between 20 and 25 and included both female and male participants. All the video samples have an average of 28 FPS and a

[1] https://mediapipe.dev/images/mobile/hand_landmarks.png

[2] https://github.com/DeepKothadiya/Custom_ISLDataset/tree/main

resolution of 1920 × 1080, with an average length of 2 seconds. The dataset consists of 11 words and for each word, about 1100 video samples were created with the 16 subjects. This dataset was created under natural conditions—without any extra brightness, orientation, Kinect sensor, background adjustments, gloves, etc.

4.3. Evaluation Metrics

Confusion Matrix - A confusion matrix is a specific table layout that allows visualization of the performance of an algorithm. Each row of the matrix represents the instances in an actual class while each column represents the instances in a predicted class, or vice versa. For a multi-class classification problem, we need a multiclass confusion matrix to evaluate the model. The multi-class classification confusion matrix does not contain any positive or negative classes, like binary classification.

True Positives (TP) - These are the data points whose actual outcomes were positive and the algorithm correctly identified it as positive.

True Negatives (TN) - These are the data points whose actual outcomes were negative and the algorithm correctly identified it as negative.

False Positives (FP) - These are the data points whose actual outcomes were negative but the algorithm incorrectly identified it as positive.

False Negatives (FN) - These are the data points whose actual outcomes were positive but the algorithm incorrectly identified it as negative.

Accuracy - Classifier accuracy is measured by the fraction of samples correctly classified by the model.

Here is the formula for calculating accuracy:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

Precision - It tells you the proportion of positive predictions that were actually correct. Here is the formula for calculating precision:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

Recall - Recall indicates the percentage of positive samples that were correctly classified as positive by the classifier. Here is the formula for calculating recall:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

Accuracy & Loss Curve - The performance of a deep learning model can be illustrated by accuracy and loss curves.

F1-Score or F-measure is an evaluation metric for a classification defined as the harmonic mean of precision and recall. It is a statistical measure of the accuracy of a test or model. Mathematically, it is expressed as follows,

$$\text{F1} = 2 * \frac{\text{Recall} * \text{Precision}}{\text{Recall} + \text{Precision}}$$

Here, the value of F-measure(F1-score) reaches the best value at 1 and the worst value at 0. F1-score 1 represents the perfect accuracy and recall of the model.

4.4. Data Pre-processing

To extract information from the hand gestures we use the Mediapipe Hands library. A series of images(26 frames) were extracted from the input video stream which was taken from IISL2020 dataset. We then set up the necessary configuration and initialize the Mediapipe Hands library to detect hands. On each frame we identify the bounding boxes or region of

interest when hands are present. For each detected hand, this library provides a set of predefined keypoints which represent various points on the hand such as finger joints and palm centre.

There are 21 points for each hand. These landmarks are represented as (x, y) coordinates on the 2D space. To work with these points we store it in a NumPy array which makes processing easier.

The figure below shows us a snippet of the training dataset.

```
1 X = np.array(sequences)
2 y = to_categorical(labels).astype(int)
```

```
1 X
```

```
array([[[ 0.00000000e+00,  0.00000000e+00,  0.00000000e+00, ...,
          3.12457442e-01,  2.73301363e-01, -8.35394859e-03],
        [ 0.00000000e+00,  0.00000000e+00,  0.00000000e+00, ...,
          2.75213510e-01,  2.77769804e-01, -8.59703217e-03],
        [ 0.00000000e+00,  0.00000000e+00,  0.00000000e+00, ...,
          2.37468719e-01,  3.02107632e-01, -1.15115708e-02],
        ...,
        [ 0.00000000e+00,  0.00000000e+00,  0.00000000e+00, ...,
          1.90866113e-01,  3.38213027e-01, -8.77804402e-03],
        [ 0.00000000e+00,  0.00000000e+00,  0.00000000e+00, ...,
          1.89844593e-01,  3.40147018e-01, -9.21134744e-03],
        [ 0.00000000e+00,  0.00000000e+00,  0.00000000e+00, ...,
          1.88894540e-01,  3.41293335e-01, -1.06528774e-02]],

       [[ 0.00000000e+00,  0.00000000e+00,  0.00000000e+00, ...,
          3.32989037e-01,  2.69391418e-01, -1.14208460e-02],
        [ 0.00000000e+00,  0.00000000e+00,  0.00000000e+00, ...,
          3.32257271e-01,  2.67875910e-01, -8.84444732e-03],
        [ 0.00000000e+00,  0.00000000e+00,  0.00000000e+00, ...,
          3.27096045e-01,  2.70512760e-01, -9.16940719e-03],
        ...,
```

Fig 4.3. Co-ordinates of the key points

4.5. Model

4.5.1. LSTM

Long Short Term Memory networks (LSTM) are a special kind of Recurrent Neural Network (RNN), capable of learning long-term dependencies. LSTMs are explicitly designed to avoid the long-term dependency problem.

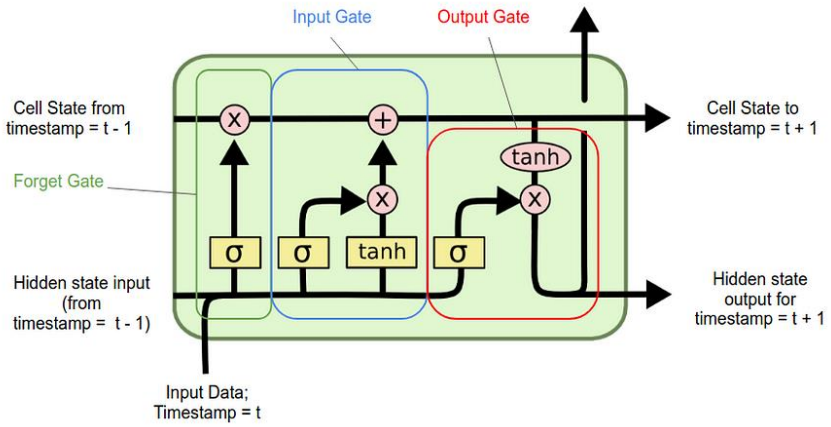

Fig 4.4. A single LSTM Cell[3]

In an LSTM network the cell state acts as a conveyor belt that carries information throughout the sequence. The cell state is modified by gates, which are composed of sigmoid layers and pointwise multiplication operations.

The forget gate layer determines which information should be discarded from the cell state by using a sigmoid layer that takes the previous hidden state and current input. A value of 0 means the information is completely forgotten, while a value of 1 means it is retained entirely.

The input gate layer consists of a sigmoid layer and a tanh layer. The sigmoid layer decides which values in the cell state should be updated, and the tanh layer generates new candidate values. These two components are combined to create an update to the cell state. To update the cell state, the LSTM multiplies the old state by the forget gate values to discard irrelevant information and adds the element-wise multiplication of the new candidate values and an update gate value. This process allows the LSTM to update the cell state with relevant information.

The output gate determines which parts of the cell state should be included in the output. It uses a sigmoid layer to decide the output content and passes the cell state through a

[3] https://miro.medium.com/v2/resize:fit:828/format:webp/1*ahafyNt0Ph_J6Ed9_2hvdg.png

tanh function to scale the values. The scaled values are then multiplied element-wise by the output gate values to produce the final output.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 26, 64)            48896

 lstm_1 (LSTM)               (None, 64)                33024

 dense (Dense)               (None, 64)                4160

 dense_1 (Dense)             (None, 11)                715

=================================================================
Total params: 86,795
Trainable params: 86,795
Non-trainable params: 0
_________________________________________________________________
```

Fig 4.5. LSTM model summary

4.5.2. Bi LSTM

The Bidirectional Long Short Term Memory (Bi-LSTM) extends the LSTM architecture by processing input sequences in both forward and backward directions simultaneously, capturing information from both past and future contexts. It consists of two separate LSTM networks which are the forward LSTM and the backward LSTM. Each LSTM processes the input sequence in one direction, with its own set of parameters.

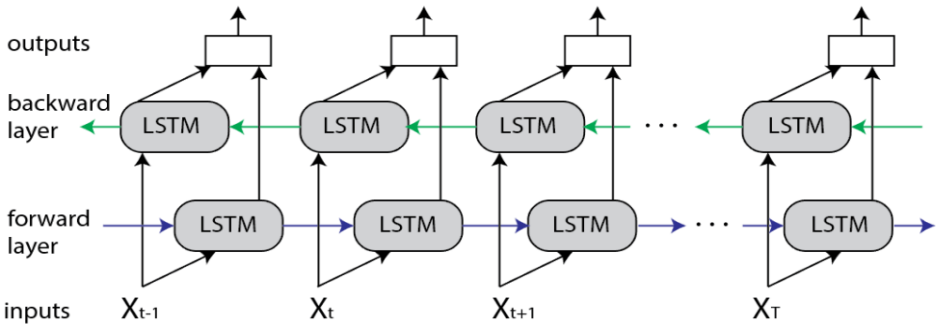

Fig 4.6. Unrolled Bi_LSTM[4]

[4] https://www.baeldung.com/wp-content/uploads/sites/4/2022/01/bilstm-1-768x288.png

The forward and backward LSTMs consists independent hidden states. The forward LSTM processes the input sequence from the beginning to the end, while the backward LSTM processes it in the opposite direction. By combining the forward and backward LSTMs, BiLSTM leverages both past and future contextual information at each time step. This allows the model to capture dependencies that span both earlier and later parts of the sequence.

By considering information from both past and future contexts, BiLSTM can make more accurate predictions or classifications compared to standard LSTMs, which only consider past information.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 bidirectional (Bidirectiona  (None, 26, 256)          261120
 l)

 dropout (Dropout)           (None, 26, 256)           0

 bidirectional_1 (Bidirectio  (None, 256)              394240
 nal)

 dropout_1 (Dropout)         (None, 256)               0

 dense (Dense)               (None, 64)                16448

 dropout_2 (Dropout)         (None, 64)                0

 dense_1 (Dense)             (None, 11)                715

=================================================================
Total params: 672,523
Trainable params: 672,523
Non-trainable params: 0
_________________________________________________________________
```

Fig 4.7. Bi-LSTM model summary

Chapter 5

IMPLEMENTATION

The figure below shows all the classes with the label map.

```
label_map = {label:num for num, label in enumerate(actions)}
label_map
```

```
{'bye': 0,
 'good': 1,
 'hello': 2,
 'house': 3,
 'morning': 4,
 'nice': 5,
 'no': 6,
 'thankyou': 7,
 'welcome': 8,
 'work': 9,
 'yes': 10}
```

Fig 5.1. Label map for all the classes

The code below prints the shape of the training and testing data which is used or training.

```
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(551, 26, 126)
(98, 26, 126)
(551, 11)
(98, 11)
```

Fig 5.2. Shape of the training and testing data

5.1. LSTM

```
model = Sequential()
model.add(LSTM(64, return_sequences=True, activation='relu', input_shape=(26, 126)))
model.add(LSTM(64, return_sequences=False, activation='relu'))
model.add(Dense(64, activation='relu'))
model.add(Dense(actions.shape[0], activation='softmax'))

model.compile(optimizer='Adam', loss='categorical_crossentropy', metrics=['accuracy'])
keras.utils.plot_model(model, to_file='lstm_drop_graph.png', show_shapes=True)

model.summary()
```

Fig 5.3. LSTM model code

The first layer is an LSTM layer with 64 units, which represent the number of memory cells or hidden units in the layer. To return the full sequence of outputs for each timestep we set the return_sequences parameter to True. For this layer we specified the activation function as rectified linear unit (ReLU). ReLU helps to reduce the vanishing gradient problem that can occur during training.

The second layer is also an LSTM layer with 64 units. We set return sequences to False to return only the last output of the sequence rather than the full sequence.

The dense layer is fully connected) layer which connects each unit to every unit in the previous layer. It has 64 learnable weights and biases.

The softmax activation function converts the output into a probability distribution over the actions. Each unit in this layer represents the probability of choosing a specific action.

The model is trained using the Adam optimizer with a categorical cross-entropy loss function, which is suitable for multi-class classification problems. The accuracy metric is also computed during training to monitor the model's performance.

To split the data we used the train-test split. We set the test size to 10% of the data and the remaining 90% for training. The model has been trained for 100 epochs.

```
history = model.fit(X_train, y_train, epochs=100,  validation_data=(X_test, y_test))
```

```
8462
Epoch 95/100
19/19 [==============================] - 0s 26ms/step - loss: 0.0634 - accuracy: 0.9760 - val_loss: 0.9513 - val_accuracy: 0.
8308
Epoch 96/100
19/19 [==============================] - 0s 25ms/step - loss: 0.0898 - accuracy: 0.9743 - val_loss: 1.1143 - val_accuracy: 0.
8154
Epoch 97/100
19/19 [==============================] - 0s 26ms/step - loss: 0.0762 - accuracy: 0.9726 - val_loss: 0.8813 - val_accuracy: 0.
8462
Epoch 98/100
19/19 [==============================] - 0s 26ms/step - loss: 0.1751 - accuracy: 0.9623 - val_loss: 0.7581 - val_accuracy: 0.
8154
Epoch 99/100
19/19 [==============================] - 0s 26ms/step - loss: 0.0918 - accuracy: 0.9692 - val_loss: 1.2625 - val_accuracy: 0.
7077
Epoch 100/100
19/19 [==============================] - 0s 26ms/step - loss: 0.1955 - accuracy: 0.9401 - val_loss: 0.9046 - val_accuracy: 0.
7846
```

Fig.5.4. LSTM Training

5.2. Bidirectional LSTM

```
from tensorflow import keras
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Bidirectional, LSTM, Dense, Dropout

model = Sequential()
model.add(Bidirectional(LSTM(128, return_sequences=True, activation='relu'), input_shape=(26, 126)))
model.add(Dropout(0.2))
model.add(Bidirectional(LSTM(128, return_sequences=False, activation='relu')))
model.add(Dropout(0.2))
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(actions.shape[0], activation='softmax'))

model.compile(optimizer='Adam', loss='categorical_crossentropy', metrics=['accuracy'])
#keras.utils.plot_model(model, to_file='bi_drop_graph.png', show_shapes=True)

model.summary()
```

Fig 5.5. Bi-LSTM model code

This architecture is similar to the LSTM model except we added a dropout layer to avoid overfitting. The first layer is a Bidirectional LSTM layer with 128 units with return sequence set to true. The input shape for this layer is (26, 126), input sequences of length 26 with 126 features.

The dropout layer with a rate of 0.2 is added after every layer except for the last layer.

The final layer is a dense layer with the number of units same as the number of actions. It also uses softmax activation to predict the output.

For the Bi-LSTM model we set the test size to 15% and the rest 85% for training. We trained this model for 90 epochs.

```
1 history = model.fit(X_train, y_train, epochs=90,  validation_data=(X_test, y_test))
```

```
19/19 [==============================] - 2s 123ms/step - loss: 0.3269 - accuracy: 0.8784 - val_loss: 0.5768 - val_accuracy:
0.8154
Epoch 46/90
19/19 [==============================] - 2s 125ms/step - loss: 0.4021 - accuracy: 0.8596 - val_loss: 0.5374 - val_accuracy:
0.8308
Epoch 47/90
19/19 [==============================] - 2s 124ms/step - loss: 0.2973 - accuracy: 0.8990 - val_loss: 0.7056 - val_accuracy:
0.8000
Epoch 48/90
19/19 [==============================] - 2s 122ms/step - loss: 0.2798 - accuracy: 0.9058 - val_loss: 0.4592 - val_accuracy:
0.8308
Epoch 49/90
19/19 [==============================] - 2s 120ms/step - loss: 0.2708 - accuracy: 0.9024 - val_loss: 0.6490 - val_accuracy:
0.8462
Epoch 50/90
19/19 [==============================] - 2s 120ms/step - loss: 0.3902 - accuracy: 0.8613 - val_loss: 0.5354 - val_accuracy:
0.8462
Epoch 51/90
19/19 [==============================] - 2s 121ms/step - loss: 0.3331 - accuracy: 0.8921 - val_loss: 0.6163 - val_accuracy:
0.8462
```

Fig 5.6. Bi-LSTM Training

Using the OpenCV library we can visualize the results. The snapshots of the results are shown below for two classes (work and good) for the Bi-LSTM model. The model was able to predict correctly for the two classes.

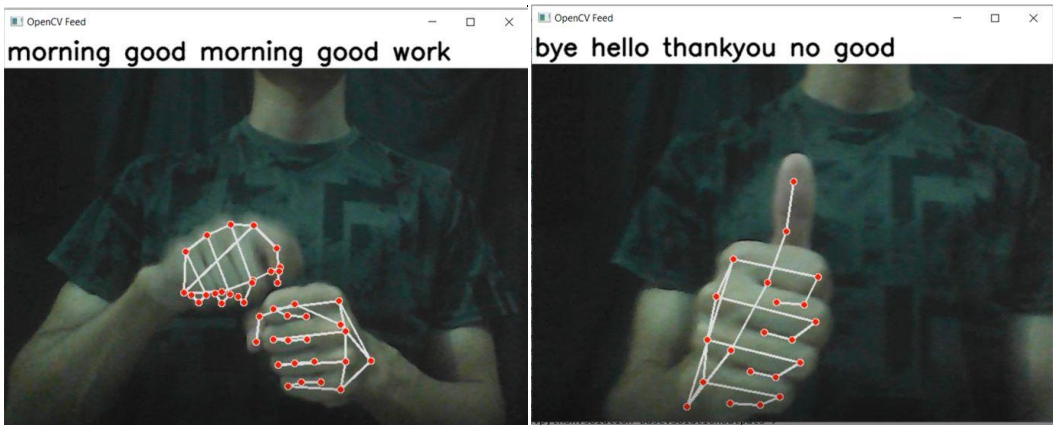

Fig 5.7. Bi-LSTM results

Chapter 6

RESULTS AND DISCUSSION

6.1. LSTM

The plot below shows the accuracy and loss of training and validation over 100 epochs.

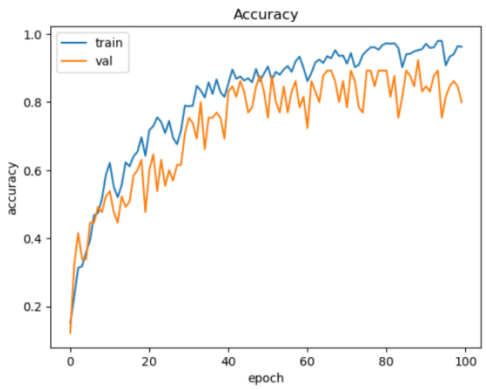

Fig 6.1(a). LSTM accuracy graph

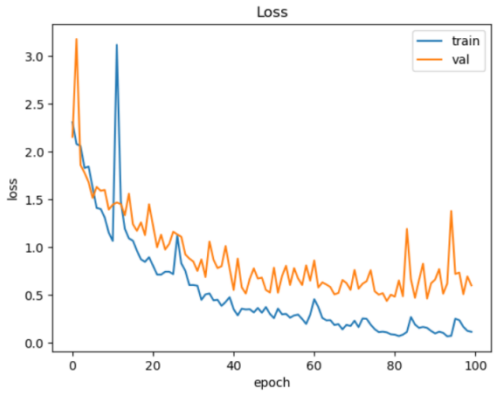

Fig.6.1(b). LSTM loss graph

The graph shows that the model was able to learn properly since the training accuracy increases with every epoch and the loss also decreases. Although, there is some overfitting. Since our dataset was not large enough, there are some spikes.

The confusion matrix of the LSTM models is shown in figure 6.1.3.

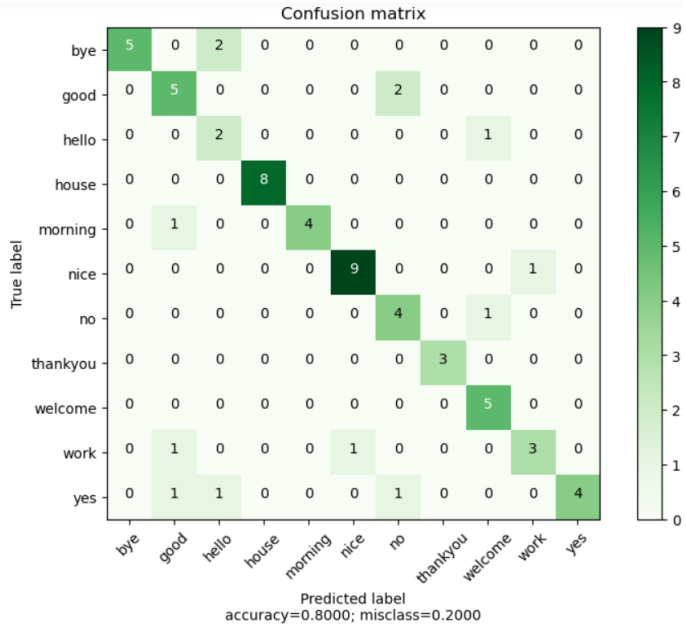

Fig 6.2. Confusion matrix for LSTM

6.2. Bi-LSTM

The plot below shows the accuracy and loss of training and validation over 90 epochs.

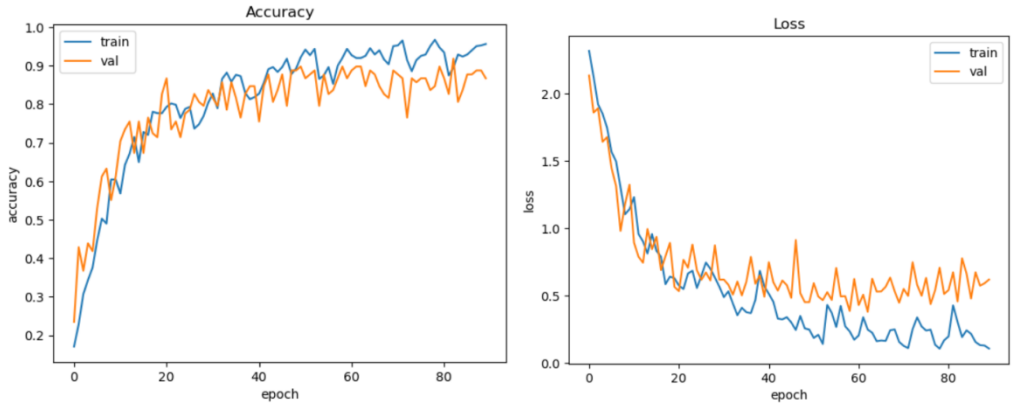

Fig 6.3(a). Bi-LSTM accuracy graph

Fig 6.3(b). Bi-LSTM loss graph

The graph for the Bi-LSTM model shows that it has better results than the LSTM model. We have reduced overfitting by introducing dropout layers.

The confusion matrix of the LSTM models is shown in figure 6.2.3.

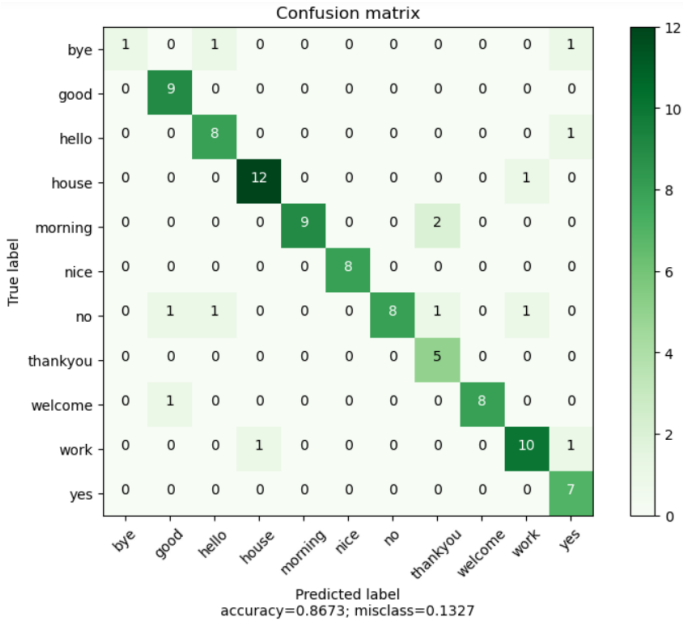

Fig 6.4. Confusion matrix for Bi-LSTM

6.3 Model Comparison

	IISL2020 Dataset			
Model	**Accuracy**	**Precision**	**Recall**	**F1-Score**
LSTM	0.8	0.81	0.8	0.79
Bi-LSTM	0.86	0.88	0.85	0.84

Table 6.1. Model Comparison

From table 6.3.1 we can see that the Bi-LSTM yields better results than the LSTM model. Bi-LSTMs capture dependencies from both past and future contexts which allow the model to have a comprehensive understanding of the input sequence.

Chapter 6

Conclusion

Communication is very important as it allows us to share our thoughts, feeling, emotion with our friends and family. Conversing with people having a hearing disability is a major challenge. Deaf and Mute people use hand gesture sign language to communicate, hence normal people face problems in recognizing their language by signs made. Hence there is a need for systems that recognize the different signs and conveys the information to normal people. Sign language recognition is a challenging problem in computer vision and pattern recognition due to high variability and complexity of sign languages.

However, advances in deep learning and computer vision have enabled the development of sign language recognition systems that can accurately recognize and translate sign language to text or speech. With the integration of MediaPipe and sequence learning methods like LSTM and Bi-LSTM, it has enabled us to create a model which can predict eleven different words from the IISL2020 dataset.

The work attempts to provide a vision-based solution for Indian Sign Language recognition problem. In future we aim to improve our model by incorporating an attention mechanism and report results on larger Indian datasets.

REFERENCES

[1] Liu, Wei, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. "Ssd: Single shot multibox detector." In European conference on computer vision, pp. 21-37. Springer, Cham, 2016.

[2] Agrawal, Rishabh, and Nikita Gupta. "Real time hand gesture recognition for human computer interaction." 2016 IEEE 6th International Conference on Advanced Computing (IACC). IEEE, 2016.

[3] Shi, Bowen, et al. "Fingerspelling recognition in the wild with iterative visual attention." Proceedings of the IEEE/CVF International Conference on Computer Vision. 2019.

[4] Schuster, Mike, and Kuldip K. Paliwal. "Bidirectional recurrent neural networks." IEEE transactions on Signal Processing 45.11 (1997): 2673-2681.

[5] Hochreiter, Sepp, and Jürgen Schmidhuber. "Long short-term memory." Neural computation 9.8 (1997): 1735-1780.

[6] Kothadiya, Deep, et al. "Deepsign: Sign language detection and recognition using deep learning." Electronics 11.11 (2022): 1780.

[7] Samaan, Gerges H., et al. "MediaPipe's Landmarks with RNN for Dynamic Sign Language Recognition." Electronics 11.19 (2022): 3228.

[8] Redmon, Joseph, et al. "You only look once: Unified, real-time object detection." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

[9] Chung, Wan-Young, Haokai Xu, and Boon Giin Lee. "Chinese Sign Language Recognition with Batch Sampling ResNet-Bi-LSTM." SN Computer Science 3.5 (2022): 414.