Indian Sign Language Interpretation Integrating Pose Detection and Speech Synthesis

Abhishek Lal – C23039

Anubhab Koley – C23006

Sayan Ghosh Roy – C23026

Shivam – C23028

Acknowledgement

We take great pleasure in extending our profound gratitude and sincere thanks to the esteemed supervisor, **Dr. Subhasis Dasgupta, Professor of Praxis Business School in Kolkata**, for his priceless advice, insightful commentary, and continuous assistance and encouragement during the project's duration. His endurance, vision, and comprehension during the study's multiple crises were fortunate to study, grow, and absorb the values of research, independent thought and time management under his prestigious guidance. It has been a very rewarding and interesting experience working with him.

Abstract

Sign language is a cardinal element for communication between deaf and dumb community. Indian Sign Language is a complete language with its own grammar, syntax, vocabulary and several unique linguistic attributes. For understanding a sign Language there need to be a lot more focus not only on the hand gestures but also on the facial expressions.

Our objective is to capture the various hand gestures and also the facial expressions correspondingly described for a particular word and train a sequence model to give us a correct prediction by observing the gestures done by a person.

Contents

Chapter 1

Introduction

Hearing loss is the most common sensory deficit in humans today. As per WHO estimates in India, there are approximately 63 million people, who are suffering from Significant Auditory Impairment; this places the estimated prevalence at 6.3% in Indian population. As these speech impairments and deaf people need a proper channel to communicate with normal people there is a need for a system. Not all normal people can understand the sign language of impaired people. Our project hence is aimed at converting the sign language gestures into text that is readable for normal people.

1.1 Sign Language

For communication purposes, deaf and hard-of-hearing individuals utilize sign language (SL), a visual-gestural language.

Significations are communicated through the use of three-dimensional spaces, hand gestures, and other body components. It contains a unique vocabulary and syntax that are wholly distinct from spoken and written languages. The sounds that are mapped against certain words and grammatical combinations in spoken languages are produced by the oratory faculties in order to transmit meaningful information. The auditory faculties then receive and analyse the oratory materials as necessary. Unlike spoken communications, sign language makes use of the visual senses. Rules are used in spoken language to create understandable messages, and sign language follows a similarly intricate grammar. An easy, effective, and portable technique for recognizing sign language

1.2 Indian Sign Language

The principal means of communication for the Deaf people in India is Indian Sign Language (ISL), which was granted formal recognition in 2019. It is a unique language with unique syntax, grammar, and regional differences. ISL incorporates an alphabetic finger spelling system in addition to using hand forms, gestures to convey message.

In schools for the Deaf, ISL is taught, allowing Deaf students to interact with peers and teachers and receive an education. It is essential to the preservation and advancement of Deaf culture because it makes storytelling and cultural activities possible.

To reduce communication barriers between the hearing public and the Deaf community, ISL interpreters are crucial. However, there are still issues facing the Deaf people in India, like restricted access to jobs and education. Many advocacy groups and organizations strive to improve opportunities and quality of life for the nation's Deaf people by promoting and increasing knowledge of ISL. A critical first step in promoting inclusiveness and protecting the rights of the Deaf community is recognizing ISL as a language.

1.3 LSTM

The LSTM network concept was initially presented in the literature (Hochreiter & Schmidhuber, 1997) in order to solve the vanishing and decreasing gradient issues with RNNs. A memory cell is the building block of an LSTM network. A memory cell is essentially a neural network's hidden layer. Every memory cell has a recurrent edge. The values carried by the recurrent edge are known as the states of the cell. The recurrent edge should ideally be associated with a link weight w = 1 to ensure that the vanishing and exploding gradient problems do not occur. An exploded form of an LSTM cell is presented in the figure.

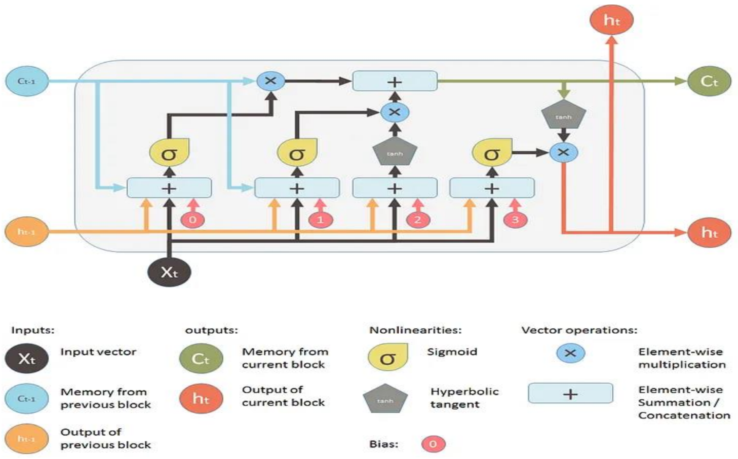

h_t
C_{t-1}
C_t
tanh
σ
0
1
2
3
h_{t-1}
h_t
X_t
Inputs:
X_t Input vector
C_{t-1} Memory from previous block
h_{t-1} Output of previous block
outputs:
C_t Memory from current block
h_t Output of current block
Nonlinearities:
σ Sigmoid
tanh Hyperbolic tangent
Bias: 0
Vector operations:
× Element-wise multiplication
+ Element-wise Summation / Concatenation

Diagram by Shi Yan

A dedicated memory cell, gating mechanisms, and the capacity to address the vanishing gradient issue—which can impede learning in conventional RNNs—are important characteristics of LSTMs. To capture intricate dependencies in data, the LSTM memory cell's ability to store and retain information across lengthy durations is crucial. Input, output, and forget gates are examples of gating devices that regulate the flow of data into and out of the cell, enabling the network to govern what data is updated and remembered at each time step.

The state information at the previous time slot, C(t-1), is transformed into the cell state at the current time slot C (t) via a number of intermediate operations instead of being directly multiplied by a weight factor (Sebastian & Mirjalili, 2019). The information flow in the memory cell of an LSTM node is controlled by some gates, which will be discussed in what follows.

A typical LSTM node consists of four different types of gates – the forget gate, the input

gate, the input modulation gate, and the output gate. The functionalities of these gates are

discussed below.

1. **Forget gate:** This gate enables the node to reset the state of the node by discarding past information, which is not of much use currently. It effectively decides which information from the previous state is to be allowed to get into the node and which information is to be discarded. The output of the forget gate at time slot t, ft:

$$ft = \sigma(Wxf * x(t) + Whf * h(t-1) + bf)$$

σ represents the sigmoid activation function, Wxf denotes the weight matrix for the input and the forget gate, Whf is the weight matrix for the previous hidden layer and the forget gate, and the bf is the bias of the forget gate node, x(t) is the input at time slot t.

2. **Input gate and Input modulation gate:** The input gate and the input modulation gate update the cell state of the LSTM node. The outputs of the input gate (it) and the input modulation gate (gt)

$it = \sigma(Wxi * x(t) + Whi * h(t-1) + bi)$

$gt = \tanh (Wxg * x(t) + Whg* h(t-1) + bg)$

the state of the node at time slot t is computed

$C(t) = (C(t-1) \Theta ft \bigoplus (it \Theta gt)$

3. **Output gate:** The output yielded by the output gate at time slot t is computed using

$ot = \sigma(Wxo * xt + Who * h(t-1) + bo)$

Finally, using the values of ot , the hidden units at time slot t are computed

$h(t) = ot \Theta \tanh(C(t))$

Chapter 2

Methodology

Video data has been used to train a sequencial model. The model would be capable to understand the gestures and give the correct word which it’s trying to predict

2.1 Workflow

The total work has been done following the below five steps:

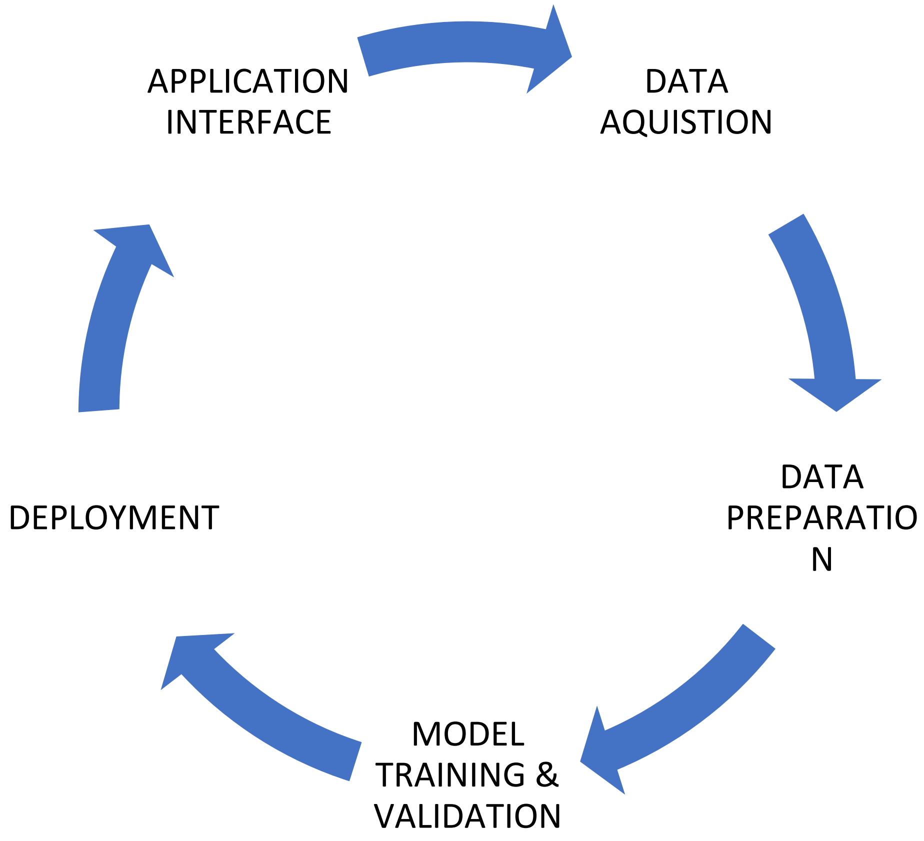
I. **Data Acquisition:** Video dataset has been acquired from a well maintained repositiory of **“Indian Sign Language Research and Training Centre”**

II. **Data Preparation:** Sequential data was required to be created for good prediction

III. **Model Training & Validation:** LSTM model has been used for prediction

IV. **Deployment:** Streamlit has been used for deployment of the model

V. **Application Interface:** An application Interface can be designed which will be capable to take a video as input and predict the correct word and also to play an audio of the predicted word.



2.2 Data Acquisition

A set of 1300 videos has been used for this purpose. The data has been taken from **“Indian Sign Language Research and Training**

Centre”. They have a very well maintained Indian sign language dictionary of about 10000 terms.

Indian Sign Language Research and Training Centre visions –

To build an inclusive society in which equal opportunities are provided for the growth and development of Persons with Disabilities so that they can lead productive, safe and dignified lives.

Play / Pause
COVID -19 - Mental Health Helpline -Toll Free Number 080-46110007
Sign Learn App
Google Play
Indian Sign Language Research and Training Centre
Department of Empowerment of Persons with Disabilities (Divyangjan)
Ministry of Social Justice & Empowerment, Government of India
Azadi Ka Amrit Mahotsav
Skip to Main Content
Screen Reader Access
Font Size A- A A+
English हिन्दी
HOME
ABOUT US
ACADEMIC
RESOURCES
ISL DICTIONARY
REPORT
STUDENT
GALLERY
TENDER
SIGN LANGUAGE DAY 2023
LIBRARY
RECRUITMENT
CORONA UPDATES
ISL DICTIONARY WEBSITE
YOUTUBE LINK TO ISL DICTIONARY
ISL DICTIONARY 10000 TERMS
ISL DICTIONARY LAUNCH
Students in Deaf Friendly Library
Previous
Pause
Next
Indian Sign Language Research and Training Center (ISLRTC)
https://drive.google.com/drive/folders/1U-Pr4r1-cupgNOOg9NH_uTsQnPSVEKco?usp=sharing

2.3 Dataset

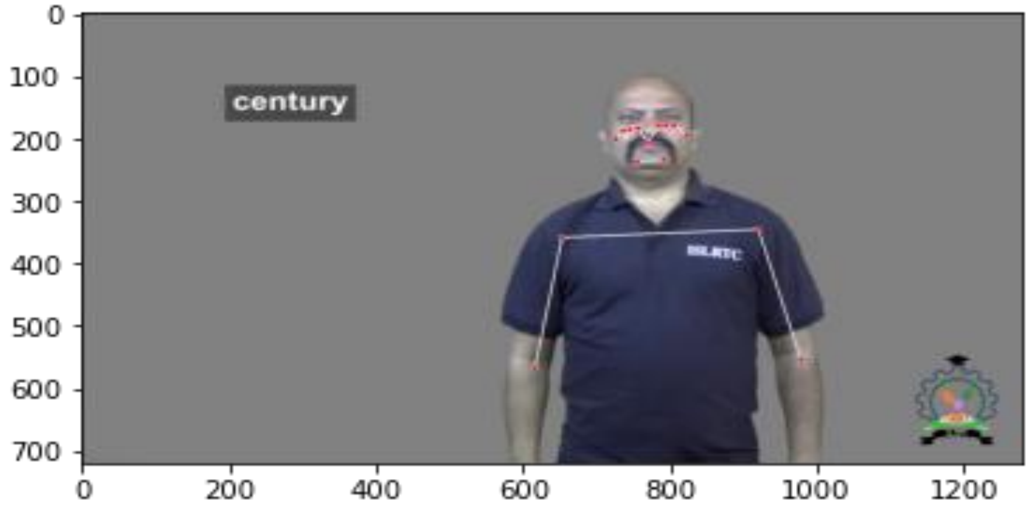
Out of the acquired 1300 videos, due to resource crunch we were able to use only 796 videos.

Dataset was created out the videos. Different words are expressed differently. Hence, it is required to keep a track of all the hand movements and facial expressions as a part of the gestures to understand a word.

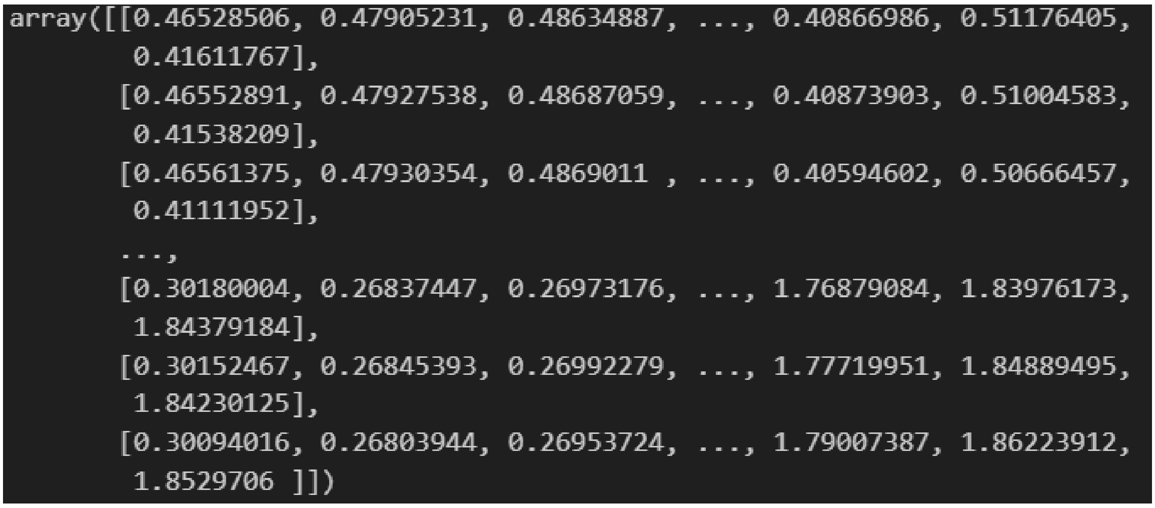
We have used CV2(OpenCV) and Mediapipe tools, which helped us to open a video file and use pose detection on the video to keep a track of the gestures done.

Mediapipe’s pose detection kept a track of total 33 key points and several features of each key point for each frame. Here, we considered only two points of x- coordinate and y-coordinate for each key points in each frame.

Hence, for each frame we have taken total 66(33x2) features.

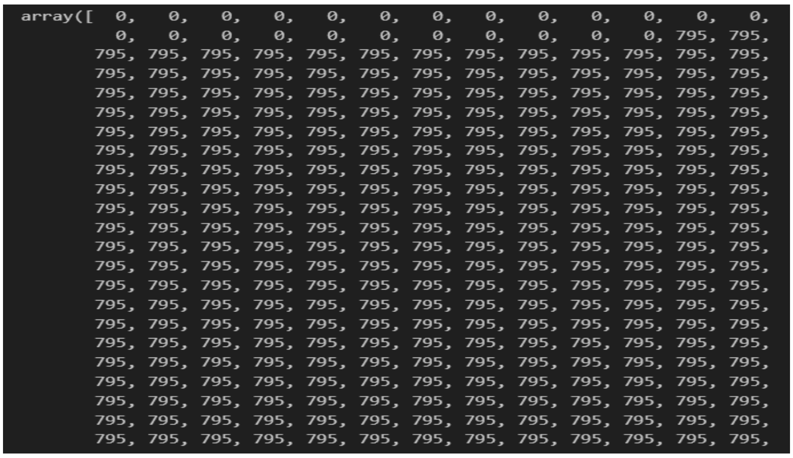


We have created two sets of lists, one contains information of all the keypoints extracted from the video, so a list of 2d arrays is created and the other contains the name or index of the processed video arranged accordingly as a video gets processed.



```
array([[0.46528506, 0.47905231, 0.48634887, ..., 0.40866986, 0.51176405,
         0.41611767],
       [0.46552891, 0.47927538, 0.48687059, ..., 0.40873903, 0.51004583,
         0.41538209],
       [0.46561375, 0.47930354, 0.4869011 , ..., 0.40594602, 0.50666457,
         0.41111952],
       ...,
       [0.30180004, 0.26837447, 0.26973176, ..., 1.76879084, 1.83976173,
         1.84379184],
       [0.30152467, 0.26845393, 0.26992279, ..., 1.77719951, 1.84889495,
         1.84230125],
       [0.30094016, 0.26803944, 0.26953724, ..., 1.79007387, 1.86223912,
         1.8529706 ]])
```

This is the list of 2d arrays containing the information of all the frames of the keypoints.



```
array([  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
         0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0, 795, 795,
       795, 795, 795, 795, 795, 795, 795, 795, 795, 795, 795, 795, 795,
       795, 795, 795, 795, 795, 795, 795, 795, 795, 795, 795, 795, 795,
       795, 795, 795, 795, 795, 795, 795, 795, 795, 795, 795, 795, 795,
       795, 795, 795, 795, 795, 795, 795, 795, 795, 795, 795, 795, 795,
       795, 795, 795, 795, 795, 795, 795, 795, 795, 795, 795, 795, 795,
       795, 795, 795, 795, 795, 795, 795, 795, 795, 795, 795, 795, 795,
       795, 795, 795, 795, 795, 795, 795, 795, 795, 795, 795, 795, 795,
       795, 795, 795, 795, 795, 795, 795, 795, 795, 795, 795, 795, 795,
       795, 795, 795, 795, 795, 795, 795, 795, 795, 795, 795, 795, 795,
       795, 795, 795, 795, 795, 795, 795, 795, 795, 795, 795, 795, 795,
       795, 795, 795, 795, 795, 795, 795, 795, 795, 795, 795, 795, 795,
       795, 795, 795, 795, 795, 795, 795, 795, 795, 795, 795, 795, 795,
       795, 795, 795, 795, 795, 795, 795, 795, 795, 795, 795, 795, 795,
       795, 795, 795, 795, 795, 795, 795, 795, 795, 795, 795, 795, 795,
       795, 795, 795, 795, 795, 795, 795, 795, 795, 795, 795, 795, 795,
       795, 795, 795, 795, 795, 795, 795, 795, 795, 795, 795, 795, 795,
       795, 795, 795, 795, 795, 795, 795, 795, 795, 795, 795, 795, 795,
       795, 795, 795, 795, 795, 795, 795, 795, 795, 795, 795, 795, 795,
       795, 795, 795, 795, 795, 795, 795, 795, 795, 795, 795, 795, 795,
       795, 795, 795, 795, 795, 795, 795, 795, 795, 795, 795, 795, 795,
       795, 795, 795, 795, 795, 795, 795, 795, 795, 795, 795, 795, 795,
```

List of the indexes of the processed video

2.4 Data Preparation

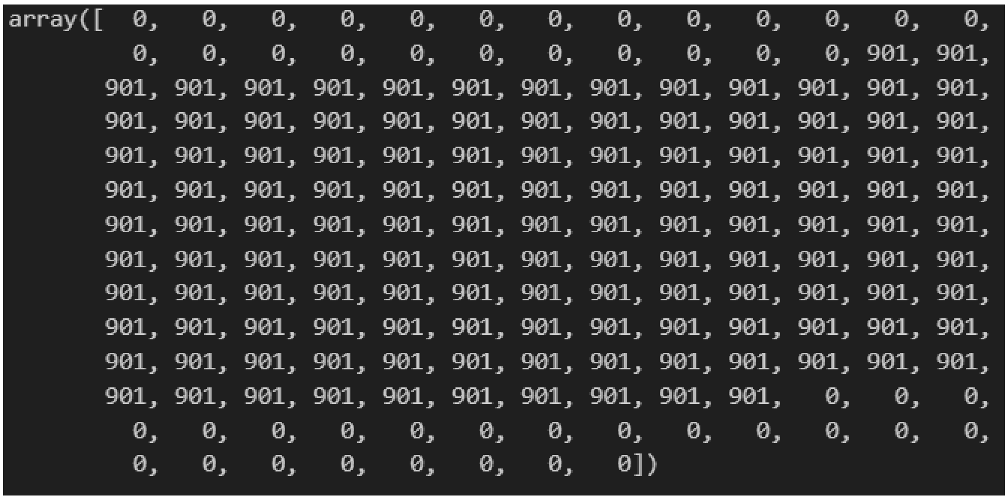
Data Preparation is one of the crucial part of this process. Since the plan is to read the data in a sequence and ultimately give a prediction, so the data needs to be prepared in that manner.

The videos that are used here have 24 frames per second. So if a video is of 5 seconds, there are a total of 120 frames (24x5) in a video.

The plan here is to take a sequence of 15 frames and the 16th index as the target. Then this sequence is continued for the complete length of a video with a stride of 1. Thus a new dataset is prepared.

The index list for a video is also prepared differently. We observed that in most of the videos for the first and last 1 second there was no gestures shown. We only want our model to see which video it is only when there is some movement.

Hence, the first and last 24 indexes were made 0 as it will be denoting that it has nothing.

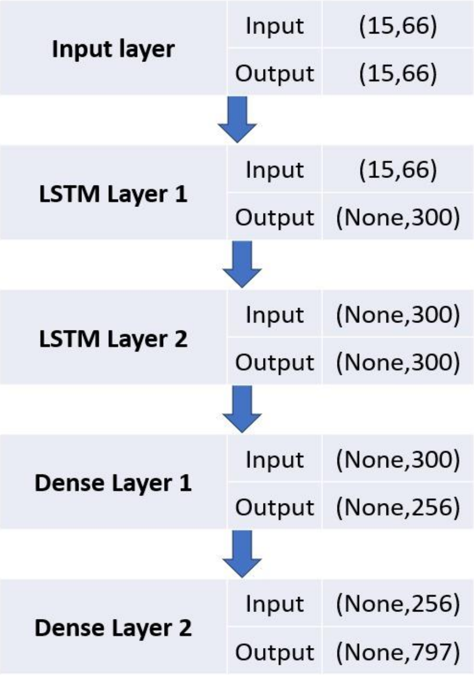


```
array([  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
         0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0, 901, 901,
       901, 901, 901, 901, 901, 901, 901, 901, 901, 901, 901, 901, 901,
       901, 901, 901, 901, 901, 901, 901, 901, 901, 901, 901, 901, 901,
       901, 901, 901, 901, 901, 901, 901, 901, 901, 901, 901, 901, 901,
       901, 901, 901, 901, 901, 901, 901, 901, 901, 901, 901, 901, 901,
       901, 901, 901, 901, 901, 901, 901, 901, 901, 901, 901, 901, 901,
       901, 901, 901, 901, 901, 901, 901, 901, 901, 901, 901, 901, 901,
       901, 901, 901, 901, 901, 901, 901, 901, 901, 901, 901, 901, 901,
       901, 901, 901, 901, 901, 901, 901, 901, 901, 901, 901, 901, 901,
       901, 901, 901, 901, 901, 901, 901, 901, 901, 901, 901, 901, 901,
       901, 901, 901, 901, 901, 901, 901, 901, 901, 901,   0,   0,   0,
         0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
         0,   0,   0,   0,   0,   0,   0,   0])
```

Out of all the videos, we have only considered the videos which are upto 350 frames as the videos bigger than that have mostly longer parts where there is no related information.

2.5 Modelling

For modelling we have used LSTM (Long Short Term Memory) model. It’s capable of reading the data sequentially and give the prediction.



Input layer	Input	(15,66)
	Output	(15,66)
LSTM Layer 1	Input	(15,66)
	Output	(None,300)
LSTM Layer 2	Input	(None,300)
	Output	(None,300)
Dense Layer 1	Input	(None,300)
	Output	(None,256)
Dense Layer 2	Input	(None,256)
	Output	(None,797)

We have used a two layer LSTM model. It has an input size of 66 and a hidden size of 300. 20% dropout has been used to try and make the model a little robust.

While training the data has been shuffled between the videos and the model has been trained on the shuffled videos.

300 epochs have been used for training purpose.

Input Layer: At a given time a sequence of 15 frames each containing 66 features are passed as input to the model.

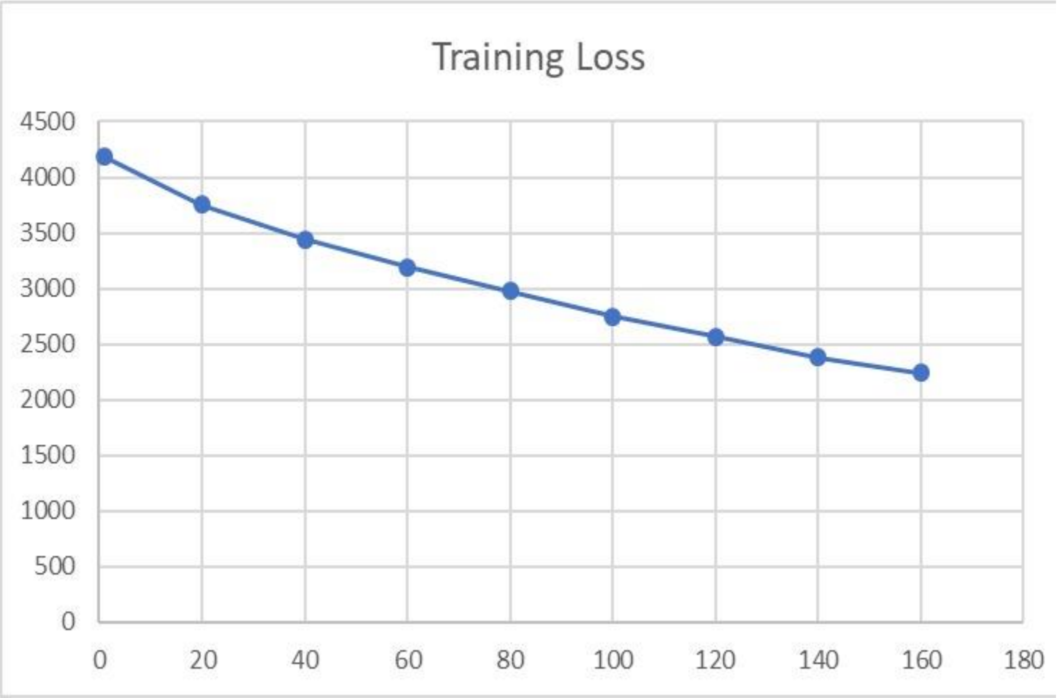
LSTM Layer 1: It takes the input and gives an output containing 300 features.

LSTM Layer 2: It takes the input from the previous LSTM layer and gives an output of 300 features.

Dense Layer 1: It takes the input from the second LSTM layer and tries to give a linear output of 256 features.

Dense Layer 2: This layer finally gives the out of 797 classes. (796 different videos and a 0 for classifying null).

The training loss is depicted here for each epoch.



2.6 Challenges

As in the project the dataset involves video data, so there were several challenges in the part of data preparation, modelling and deployment.

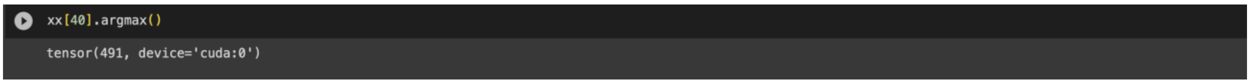
1. After acquiring the video data it was challenging to choose and implement the correct movement detection technique to keep a track of the hand movements. Finally, pose detection in Mediapipe was chosen.

2. Understanding and preparing the data was actually challenging. The correct format of data was required so that a model can be trained and we can get the correct output as per our requirement. It took maximum time of the project.
3. The modelling needed to be done on the GPU. We were using google colaboratory vm for work. As colaboratory gives limited sessions for using GPU, so we needed to perform maximum of the task in a given session.
4. We were trying to deploy the model in the local environment using Streamlit. Hence, issues were faced for making the gpu trained model compatible on the cpu.

Chapter 3

Results

The trained model is now used for the testing purpose. A video is given as an input to the model. The model gives an output of probabilities for 797 classes for all the set of sequence of a video. Then by using “argmax()” function we can actually find out the index of the input video which was used during training.

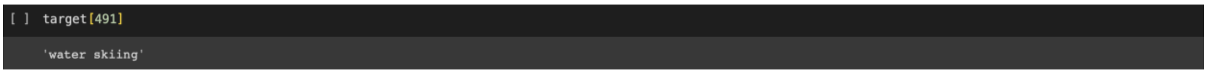


```
xx[40].argmax()

tensor(491, device='cuda:0')
```

From the above code, it’s found that when a video has been passed on to the model, the model is able to predict the index from the input videos. In this case the index is 491.

Now, from the target column, we will match the index and try to find the predicted word.

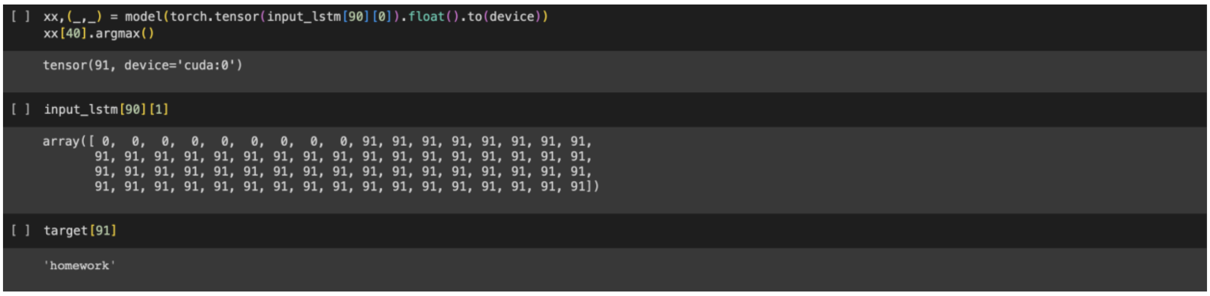


```
[ ] target[491]

'water skiing'
```

Thus, we find that the model has predicted the correct word which in this case is “**water skiing**”.

<u>TEST-II</u>



```
[ ] xx,(_,_) = model(torch.tensor(input_lstm[90][0]).float().to(device))
    xx[40].argmax()

    tensor(91, device='cuda:0')

[ ] input_lstm[90][1]

    array([ 0,  0,  0,  0,  0,  0,  0,  0,  0, 91, 91, 91, 91, 91, 91, 91, 91,
           91, 91, 91, 91, 91, 91, 91, 91, 91, 91, 91, 91, 91, 91, 91, 91, 91,
           91, 91, 91, 91, 91, 91, 91, 91, 91, 91, 91, 91, 91, 91, 91, 91, 91,
           91, 91, 91, 91, 91, 91, 91, 91, 91, 91, 91, 91, 91, 91, 91, 91, 91])

[ ] target[91]

    'homework'
```

Similar testing has been done for another video.

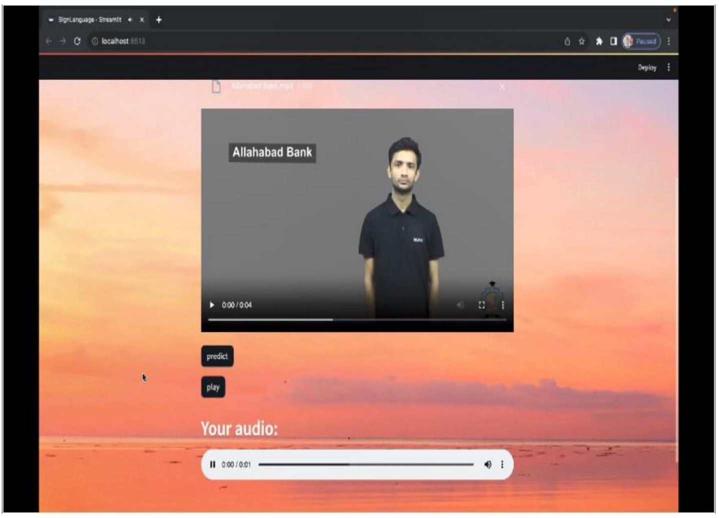
Here also it’s found that the predicted index and the input video has matched.

Thus, the conclusion is that the model has been working perfectly. The prediction by the model is matching from the training data.

Chapter 4

Model Deployment

The model which has been trained for predicting the gestures of sign language was tried to be deployed behind an application interface. The application will be able to take an input video. The model will then predict what the video is trying to say. The output of the prediction will be then passed through a text to speech converter to play an audio of the predicted text.



Streamlit is an open-source Python library that makes it easy to create and share beautiful, custom web apps for machine learning and data science.

The deployed model in Streamlit is hosted locally. The interface looks like below.

Chapter 5

Future Scope of Work

- Different video renditions of the same word expose language models to a broad spectrum of contextual and linguistic subtleties. By exposing models to a variety of accents, tones, and speaking styles, they become more adept at comprehending and processing spoken language overall. In the end, it strengthens their resilience in practical uses such as automated transcription services and voice assistants.
- By extending this goal to sentence prediction, language models are able to produce meaningful and contextually relevant sentences by understanding the nuances of language structure and coherence. This increases their suitability for tasks involving sophisticated language generation and comprehension.
- Further, the work can be extended to detect the subject from a chaos background and understand the gesture done by them to predict the word.

Chapter 6

Bibliography

1. National Health Mission. Available online (https://nhm.gov.in/index1.php?lang=1&level=2&sublinkid=1051&lid=606)
2. Advaith Sridhar, Rohith Gandhi Ganesan, Pratyush Kumar, Mitesh Khapra INCLUDE: A Large Scale Dataset for Indian Sign Language Recognition (https://dl.acm.org/doi/10.1145/3394171.3413528) 12 October 2020
3. Elakkiya R, NATARAJAN B ISL-CSLTR: Indian Sign Language Dataset for Continuous Sign Language Translation and Recognition (https://data.mendeley.com/datasets/kcmpdxky7p/1) 22 January 2021
4. Manasi Malge, Vidhi Deshmukh, Prof. Harshwardhan Kharpate Indian Sign Language Recognition (IJARSCT) Volume 2, Issue 2, March 2022
5. Indian Sign Language Research and Training Centre (https://www.islrtc.nic.in/)
6. Shi Yan Understanding LSTM and its diagrams ML Review (https://blog.mlreview.com/understanding-lstm-and-its-diagrams-37e2f46f1714) Mar 14, 2016

7. Jaydip Sen* and Sidra Mehtab LSTM Networks: Architectures and Applications in Stock Price Prediction

8. Deep Kothadiya, Chintan Bhatt, Krenil Sapariya, Kevin Patel, Ana-Belén Gil-González, Juan M. Corchado Deepsign: Sign Language Detection and Recognition Using Deep Learning Published: 3 June 2022

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

10. Le, X.-H.; Hung, V.; Ho, G.L.; Sungho, J. Application of Long Short-Term Memory (LSTM) Neural Network for Flood Forecasting. Water 2019, 11, 1387. [Google Scholar] [CrossRef][Green Version]