SPEECH/TEXT TO INDIAN SIGN LANGUAGE USING NATURAL LANGUAGE PROCESSING

Anannya Priyadarshini Neog[1]
Student, Department of Computing Technologies,
SRM Institute of Science and Technology,
Chennai, Tamil Nadu, 603203, India
anannya903@gmail.com

Arunabh Kalita[2]
Student, Department of Computing Technologies,
SRM Institute of Science and Technology,
Chennai, Tamil Nadu, 603203, India
arunabhkalita.coding@gmail.com

Ms. Nithyakani Pandiyarajan[3]
Assistant Professor, Department of Computing Technologies,
SRM Institute of Science and Technology,
Chennai, Tamil Nadu, 603203, India
nithyakp@srmist.edu.in

Abstract

Sign language is a way of communication that helps people exchange information by using hand and arm gestures, commonly used by individuals who have difficulty hearing. However, sign language isn't universal, because impaired individuals from different countries use their corresponding sign languages. Using sign language, it allows us to communicate with impaired individuals including our loved ones, students in mainstream/deaf schools/colleges, locals and company owners, etc. Studies say learning sign language makes it simpler for a person to grasp lip-reading along with their native language. Most research has been done on Sign Language Translation/Recognition; different sign languages are translated into a common spoken language. However, the inverse is less, meaning limited research has been done on converting spoken languages to sign languages. Focusing on this matter, this study aims to translate speech/text into Indian Sign Language using the basics of Natural Language Processing.

***Keywords*: Tokenization; Lemmatization; Buildozer.**

1. Introduction

Ever since Sign Languages were introduced into the world, it became much easier to communicate with people who suffer from hearing loss or are auditorily impaired by birth. The history of sign language introduction is not defined at a particular point in time, but there were various ages when people interacted using sign language differing from region to region. Since our world has a wide range of cultural diversity with over 6000+ spoken languages, there also exists a variety of sign languages around the world that use either single or both hands. An example of a single-hand user is American Sign Language (ASL) and a double-hand user is Indian Sign Language (ISL). According to the United Nations, there are more than 300 sign languages in the world. Indian Sign Language (ISL) is also one of them, which is the base of this study. According to WHO, approximately 6.3% of Indians suffer from Significant Auditory Impairment and most of them are minors. This is a considerable loss of physical and economic output. Rural areas and elderly populations have higher rates of the same. A significant portion of our population also suffers from unilateral (one-sided) hearing loss in lesser forms.

Although most impaired people are deaf from birth, there are individuals that tend to lose their hearing after growing up. So how do these individuals become susceptible to this? Research shows that aging and exposure to loud noises over an extended period of time can both lead to hearing loss. Moreover, temporary hearing loss can be caused by excessive ear wax. Other factors causing hearing loss can also be impacted by illnesses such as diabetes, hypertension, heart disease, brain injury, tumour, stroke, bacteria, and viruses. It's crucial to keep in mind that hearing loss might impact one's quality of life, particularly in elderly persons. It can hinder communication, leading to loneliness, isolation, reliance, frustration, depression, and difficulty in communication, all of which can contribute to the progression of cognitive decline. Gaming devices like Microsoft Kinect which is a motion-sensing device are there to understand human motions and gestures and act upon them. Humans are advancing towards more sophisticated technology, nevertheless. They often want a flexible system in which they can make the most of the resources and surroundings at hand. There are several methods that have been developed recently and are continuously being studied in order to simplify sign language translation. The goal of this study is to eliminate as many of these issues and correctly translate speech/text into signs.

2. Related Work

Various published papers that are pertinent to the subject of this study have been discussed in this section. These research articles have been gathered from a variety of reputable publications in online databases like IEEE, ResearchGate, etc. The majority of the studies collected date from the period 2015-2022. Let's find out what they actually reveal.

The end-to-end human interface framework proposed in Paper [1] enables a very realistic conversation between the deaf & mute community and the general population by first identifying and analyzing spoken language and then carrying out the relevant Indian Sign Language gestures. With the help of Microsoft Xbox Kinect 360s, they have captured motion data for all the different ISL gestures. After setting out each ISL gestures into animation using Unity3D, they combined everything into an Android application. They have self-collected their dataset. Over the course of a week of consistent use, application testing was successful in 91% of the circumstances. Due to the speech-to-text engine's limitations, a few words were mistranslated.

The goal of Paper [2] is to translate six regional Indian languages - Telugu, Hindi, Malayalam, Marathi, Kannada, and Tamil from speech to Indian sign language. The suggested model converts speech to a series of equivalent motions, which are shown as the output. The speech recognition process involves using of Wavelet-based MFCC (Mel-frequency cepstral coefficients) with GMM (Gaussian mixture model), translation of text is done by LSTM including text mapping with sign language. Speech corpus for the above mentioned languages are gathered from CommonVoice provided by Mozilla and for the purpose of training the model, MultiIndicMT by Kyoto University has been used. Text corpus for the same are retrieved from ManyThings.org. Anki. The accuracy obtained for each regional language is more than 80%.

Paper [3] focuses on to create a translating system made up of several modules where English audio is fed as input and translates the same into English text. This text is then parsed to create a structural grammar representation, on which Indian Sign Language grammar rules are applied. The reordered sentence does not contain any stop words. The supplied term will be changed into its root or original word using stemming and lemmatization because the Indian Sign Language does not permit word conjugation. Following this, every word is checked in a dictionary containing videos of each word. And if the algorithm is unable to locate the term in the dictionary, then the closest appropriate synonym will be replaced by the word.

The purpose of this paper [4] is to develop a portable system which is available 24x7 and can translate bidirectionally, that is, from voice to sign language and vice versa. Without the need for extra hardware, the flexibility of this system is maintained through the use of a mobile application that is always with the user. The smartphone application will produce audio and text as output when Argentinian Sign Language signs are fed as input. On the other hand, when audio and text are given as input, they are converted to signs in a 3D animation that has been created with the help of Unity3D. The system uses a pre-trained model created with the combination of CNN and RNN using machine learning. The dataset used for this study is of Argentinian Sign Languages. The sign to speech machine model gave an accuracy of 97% and conversion from speech to sign language obtained an accuracy of 95%.

Utilizing latest advancements in Neural Machine Translation (NMT), Generative Adversarial Networks, and motion generation, paper [5] introduces a unique method to produce automatic Sign Language. Using spoken language phrases, the system can generate sign videos. To train the model, the method needs minimum gloss and skeletal level annotations unlike the existing systems that need densely annotated data. The process is divided into specialized sub-processes. Initially, a combination of NMT network and Motion Graph has been used to interpret spoken language phrases into sign gestures. Following that, a generative model that can create realistic sign language video frames is trained using the poses obtained from the signed gestures. This method of continuous sign video generation doesn't apply any avatar techniques. The translation capabilities have been evaluated with the help of PHOENIX14T Sign Language Translation dataset. A benchmark has been set regarding text-to-gloss translation where a BLEU-4 score of 16.34/15.26 on dev/test sets was recorded. Further, with the help of broadcast quality assessment metrics, the video production abilities for both multiple signer and HD settings have been demonstrated qualitatively and quantitatively. Two types of multi-signer (MS) datasets have been used for this study purpose. One is PHOENIX14T that consists of German Sign Language (DGS) interpretations performed by 9 signers and the other one is called the SMILE Sign Language Assessment Dataset that contains 42 signers performing in Swiss German Sign Language (DSGS).

The objective of paper [6] is to create a software that partially compensates for the lack of educational resources and assistive technology that hearing-impaired people require for their learning and communication. Text may be converted into motions for a hearing person to comprehend using the translator. The software may be used profitably by those who have hearing loss to improve their written language abilities and can be used to teach sign language as well. The language chosen for this study Bangla Sign Language.

The motive of paper [7] is to develop a web application that can be used for translating input speech or text by a parser element into structural grammar representation. This structural grammar is then used by some other module that consists of an ISL grammatical tool. From the rearranged input, stop-words are eliminated. Here stemming and lemmatization have been used to replace the given words into their root words since ISL doesn't allow inflections. After that, all words are checked against a database by sentence filtration. This dictionary is nothing but a database of videos for each word. The programme can also replace those words which aren't available in the database by looking at its closest synonym. It is simple to access and utilize because this tool is web-based and can translate in real time. The language chosen for this study is ISL.

Paper [8] develops a program that can translate speech audio into ISL. With the help of speech-to-text API the audio consisting of the sentences or words are converted to text. To break the text into small words, Machine Learning is applied. Finally using Artificial Intelligence those converted text from the audio input will be interpreted into signs.

Paper [9] introduces an audio to ISL translation system. The system receives input in the form of speech and text and compares it to the videos stored in the authors' database. If they match, it produces matching sign gestures in the form of output depending on Indian Sign Language grammar otherwise; it proceeds through the tokenization and lemmatization stages. The basic technique applied in this system is natural language processing which provides access to tokenization, parsing, lemmatization, and part-of-speech tagging.

Paper [10] approaches to create an online platform for hearing impaired. The system's main components carry out tasks including speech-to-text conversion, parsing, lemmatization, stop words filtering, producing ISL syntax, and joining the clips. Indian Sign Language has been used for the purpose of this study. Accuracy achieved from this approach is 97.86%

Paper [11] uses techniques like central moments along with HU's moments for feature extraction and for classification; neural network and SVM have been used to recognize sign languages. In paper [12], methods like Bag of Visual Words model (BOVW) to recognize Indian sign language alphabets (A-Z) and digits (0–9), SURF (Speeded Up Robust Features), SVM and CNN. Paper [13] uses a little different approach compared to others, that is SIFT and HOG descriptors, K-Nearest Neighbor Classifier. Gradient based key frame extraction method, Orientation Histogram (OH) with Principal Component Analysis (PCA), DWT are applied in paper [14]. According to the suggested method in paper [15], double-handed ISL is collected in terms of pictures, processed using MATLAB, and then translated into voice and text.

3. Methodology

This section explains both the methodology and implementation of this study, where the implemented modules are also described. The proposed methodology describes in details of how speech/text has been converted into ISL signs using Natural Language Processing techniques. The procedure involves taking audio/text as input from the user using the device's default microphone/keyboard respectively. Then the system will try to recognize the words as signs which are stored in the database. After that, those signs will be converted to images/GIFs which are also stored as a separate file in the database and finally signs will be displayed accordingly on the screen. The steps have been explained in details in the following sections.

The implementation part explains how the proposed methodology is deployed into a mobile application for further use.

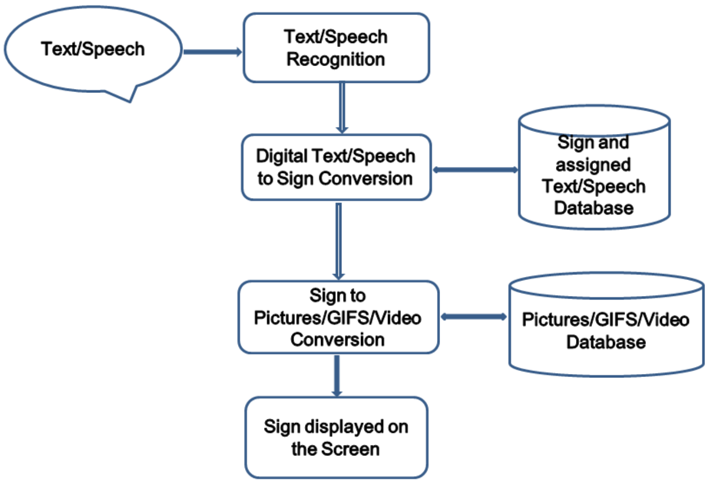

Text/Speech
Text/Speech Recognition
Digital Text/Speech to Sign Conversion
Sign and assigned Text/Speech Database
Sign to Pictures/GIFS/Video Conversion
Pictures/GIFS/Video Database
Sign displayed on the Screen

Fig. 1. Overview of the system

3.1. *Datasets*

The dataset used for the study of this paper has been collected from various sources and are stored in the local systems. This dataset consists of Indian Sign Language alphabets and various phrases or sentences that will be required to communicate and understand while taking input from the user. These files are stored in .jpg files and .gif files.

3.2. *Audio or Text as Input*

At first, the user speaks infront of the device's microphone and the audio will be recorded. This can be done by the default PC's microphone as well as the mobile microphone. This module has been carried out by importing libraries like speech recognition and pyaudio. Since this study uses voice as input, so the above mentioned libraries will help in capturing voice and identifying the speech and convert it to meaningful strings. They also help in reading audio/speech files.

3.3. *Text Preprocessing*

The conversion of speech to ISL signs has been by text preprocessing steps. Text preprocessing is the process of converting texts into a simple and readable format that can be fed into a model so that the model can understand and analyze them for further learning. It involves Tokenization, Lower casing, Stop words removal, Stemming, Lemmatization, Parsing and POS Tagging given in Fig.2.

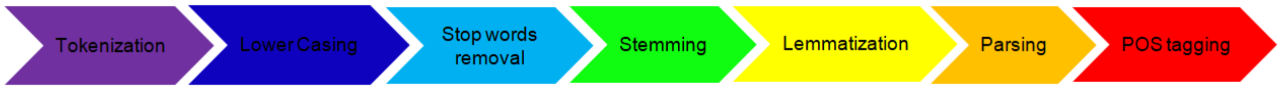


Fig. 2. Text Preprocessing Steps

3.3.1. Tokenization

Tokenization is the process of splitting a long text of string into smaller fragments called tokens. These tokens are basically nothing but words, phrases, alphabets or even numbers.

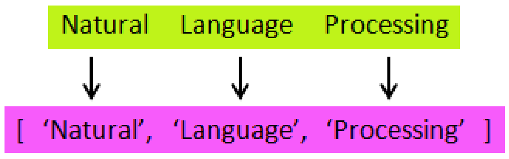

Natural Language Processing
['Natural', 'Language', 'Processing']

Fig. 3. Tokenization Example

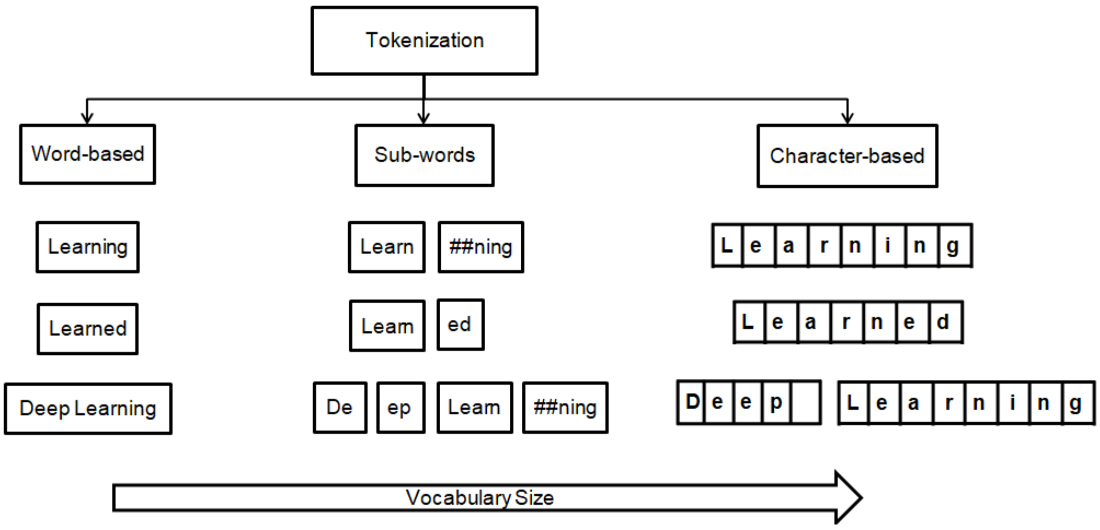


Fig. 4. Tokenization categories

3.3.2. Lower Casing

Changing the input case of the text into lower case. Despite the fact that the terms "HELLO", "Hello" and "hello" have the same meaning, the model treats them as three separate words if the lower casing is not applied.

Natural Language Processing -> natural language processing

ISL -> isl

Fig. 5. Lower Casing Example

3.3.3. Stop Words Removal

Stop words are nothing but the frequently occurred words in any text. These can be any articles, pronouns, prepositions, conjunctions and so on. In most cases, they don't significantly enrich the text. The words "a", "an", "the", "in", "on", "so", "and" & "what" are a few stop words in English. By getting rid of these terms, we can make the given text more focused on the key information by eliminating the secondary details. By removing these stop words, there remain fewer tokens that are involved in training and this significantly decreases the size of the dataset and, consequently, the training time.

Text with Stop Words	Text with Stop Words
Hello! Nice to meet you	Hello, Nice, meet
Can listening be exhausting?	listening, exhausting
Driving is my hobby, so I drive a lot	Driving, hobby, drive, lot

Table 1. Stop Words Removal Example

3.3.4. Stemming

It reverts a term to its original form that is the root word. These root words are called stems. Basically it cuts a word to its stem that is affixes -> suffixes & prefixes. The stemming algorithms are called stemmers. Various stemmers are there used for stemming like Porters Stemmer, Snowball Stemmer, Regex Stemmer, Lancaster Stemmer, Lovins Stemmer, Paice/Husk Stemmer, Dawson Stemmer, N-Gram Stemmer, HMM Stemmer, YASS Stemmer, Krovetz Stemmer, Xerox Stemmer, etc which are categorized based on Truncating, Statistical and Mixed Stemming Algorithms. Stemming example is given below.

Words = ['connects', 'connected', 'strange', 'is', 'am']

Stemmed = ['connect', 'connect', 'strang' , 'is', 'am']

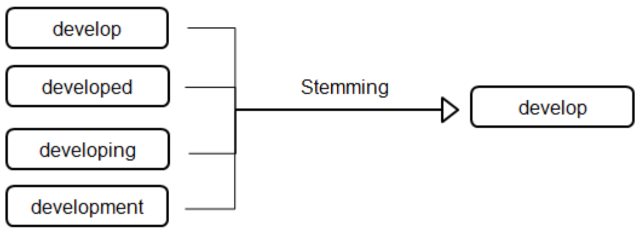


Fig. 6. Stemming Example

3.3.5. Lemmatization

A common technique used in text preprocessing which converts any sort of word to its basic root form. Although this technique doesn't differ much from the stemming process, lemmatization, in contrast to stemming, converts the words into a term that is present in the language dictionary. Here, the root word is called lemma. Since a lemmatizer gives a valid word, unlike a stemmer (given in Stemming's Example in Fig. 7), it tries to avoid the limitations of a stemmer. Therefore, a lemmatizer takes longer than a stemmer as it searches for proper words existing in the dictionary. There are various lemmatizers used in NLP like WordNet Lemmatizer, spaCy Lemmatizer, TextBlob Lemmatizer, etc. Example of Lemmatization is given below.

Words = ['connects', 'connected', 'strange', 'is', 'am']

Lemmatized = ['connect', 'connect', 'strange' , 'be', 'be']

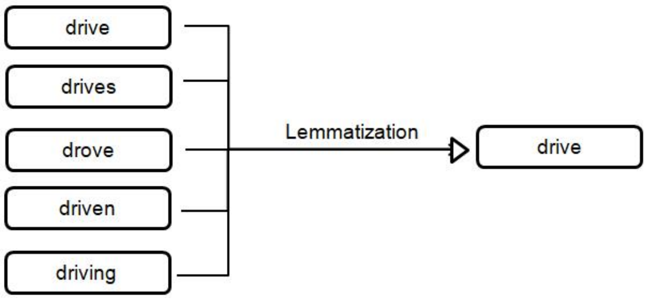


Fig. 7. Lemmatization Example

3.3.6. Parsing

It analyses the structure of the sentences and describes it in accordance with some grammatical forms. It is a typical task in NLP that breaks a given string of text to simple fragments depending on some constraints. In essence, it determines if the provided text uses good grammar. It also helps to modify text according to the grammar rules of the chosen language. The Stanford Parser is the most popular parsers of all. Here, the words arrangement and their relationships are also checked. Example of the same is given below.

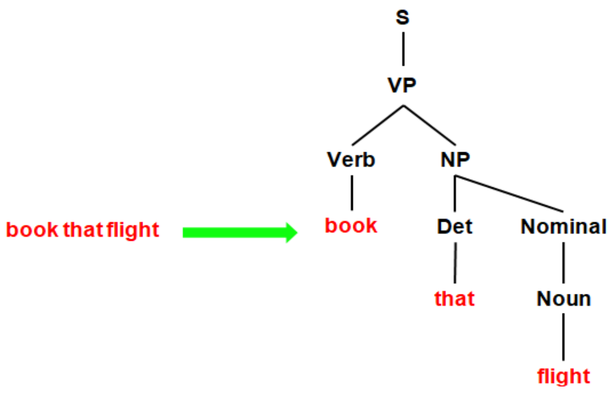


Fig. 8. Parsing Example

3.3.7. POS (Part-of-Speech) Tagging

POS (acronym for Part-of-Speech) tagging tries to determine the grammatical category a word falls under. e.g., a word could be a noun, verb, prepositions, adverb, adjective, etc. Each word in a phrase is assigned a suitable tag based on relationships found inside the sentence. An example is shown below for the same.

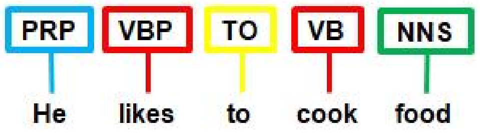
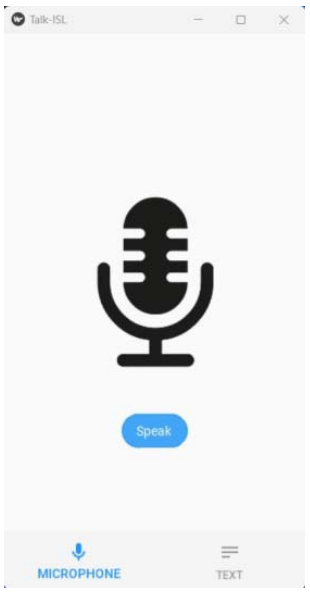


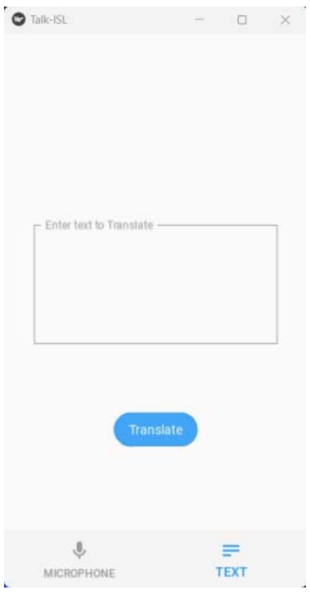
Fig. 9. POS Tagging Example

3.4. Deploying into a mobile application

This stage explains the deploying process of this project into a mobile application. To convert this into an application, Kivy and KivyMD have been used. Kivy is a cross-platform framework and KivyMD is an extension of the former one. Both give advanced-looking Graphic User Interfaces to transform the system into any form of application whether it be a mobile or a web application, allowing developers to deploy their models on any operating system like Android, iOS, Windows, Linux, and MacOS. Fig. 11 below shows the UI of the proposed application.



a)



b)

Fig. 10. a) User Interface for Speak mode; b) User Interface for Type mode

Once the application is built, it has been converted into an APK (Android Package Kit) package. The following five steps lead to creating the .apk file from .py file. First uploading the python file to Google colab directory followed by installing some dependencies. Only after performing this step, we can proceed further that is initializing the buildozer. It is nothing but a tool whose goal is to bundle mobile apps in a simple manner. The whole build procedure is automatic, leading to the prerequisites (like python-for-android, the Android SDK, NDK, etc) creation which later are downloaded. It stores a file called buildozer.spec in the project directory of Google colab that lists the requirements and preferences for the programme, including its title, icon, required modules, and other details. These details need modification according to the developer's aim to create the application which is the editing step in the below Fig. 12. Finally, it'll produce a package of .apk file format for iOS, Android, and other platforms using the specification file.

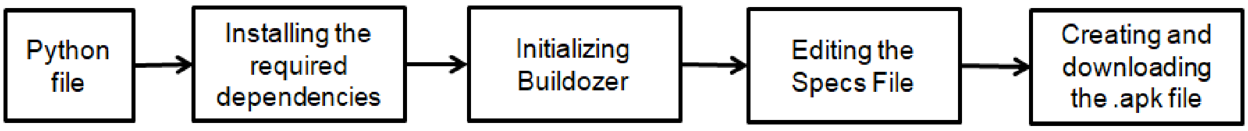

Python file
Installing the required dependencies
Initializing Buildozer
Editing the Specs File
Creating and downloading the .apk file

Fig. 11. Creating .py file to .apk file

4. Results and Discussions

The Speech/Text to Indian Sign Language has been performed using Deep Neural Network. To build the model, modules like Speech Recognition and Pyaudio have been used. The model is created using concepts of Text Preprocessing like tokenization, lemmatization, parsing, etc. Further to deploy the model, libraries like Kivy and KivyMD have been used to build this model into an android application. When the application receives audio input, it produces the assigned images/gifs based on the audio input.

For instance, if we take "Good Morning" and speak infront of the microphone, it shows the following GIF as output shown in Fig. 13.

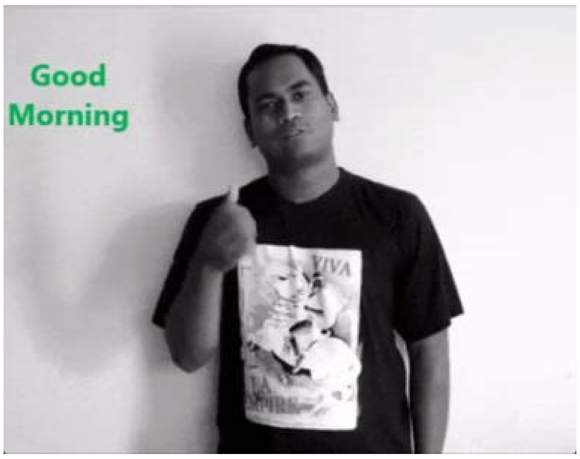


Fig. 12. Input "Good Morning" spoken infront of microphone and the model generating the assigned GIF.

Another example, if we take "How are you?" and speak infront of the microphone, it generates the following images as output shown in Fig. 14, since there isn't any gif of the same.

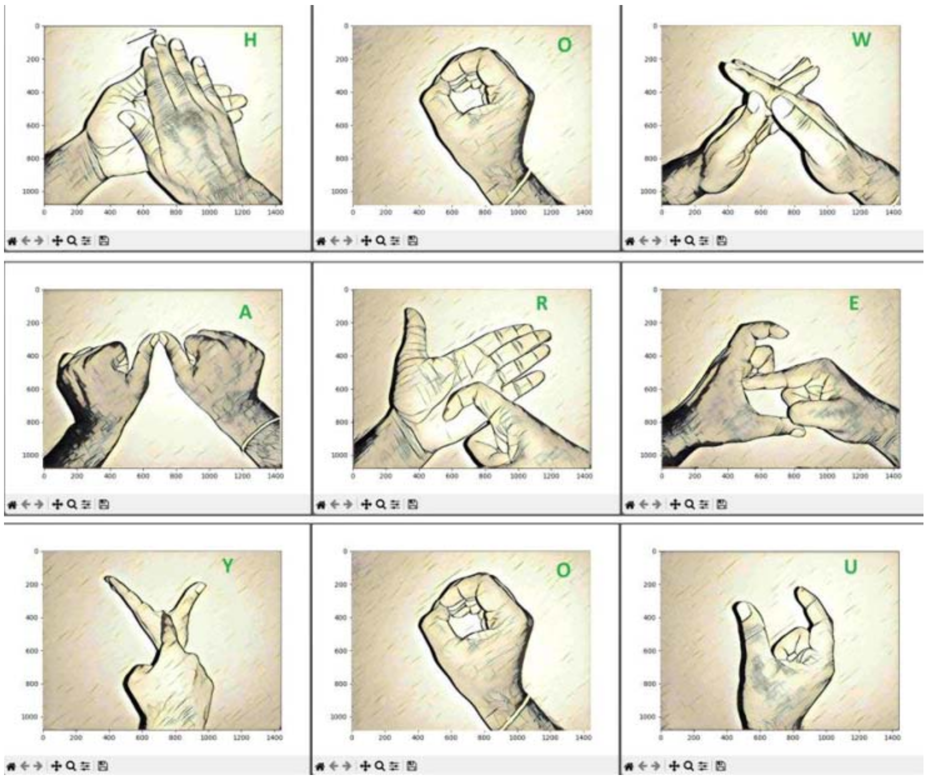
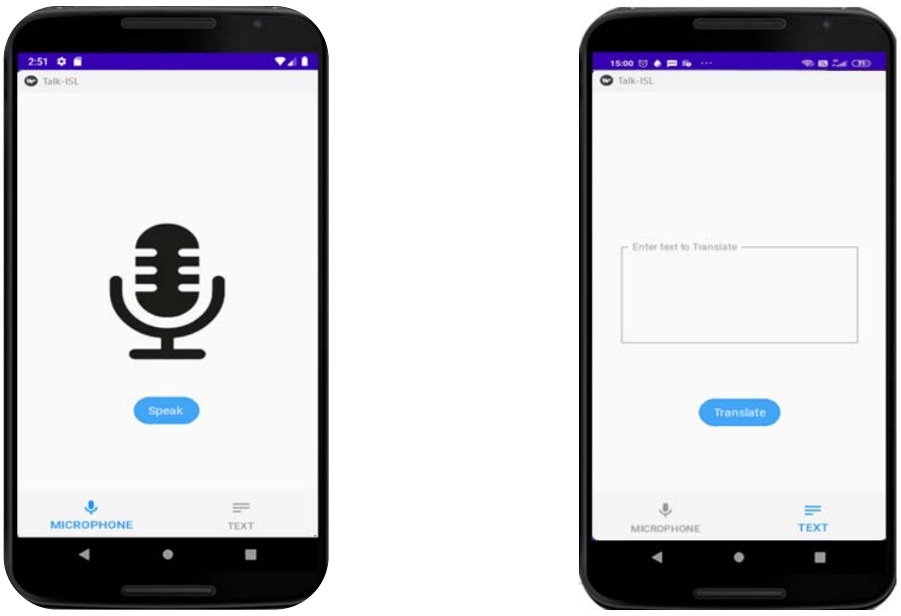


Fig. 13. Input "How are you" spoken infront of microphone and the model generating the assigned images.

The below Fig. 15 is the deployed application build from the above model which shows two screens for two different operations, that is Fig. 15 a) screen to use microphone mode and Fig. 15 b) screen to use Type (or text) mode.



a) b)

Fig. 14. Deploying it into a Mobile Application; a) Speak mode screen; b) Type mode screen

5. Conclusion and Future Scope

An application for translating Speech/Text to Indian Sign Language has been created by using NLP concepts. To build the model, python libraries like Speech Recognition and Pyaudio have been used. The model has been further transformed into a mobile application using Kivy and KivyMD frameworks. The application responds to most of the audio correctly by producing the assigned images/gifs. Additionally, this model can be enhanced by feeding more data like vocabulary in the form of images and GIFs to make it more dynamic and publishing it in the Google Playstore or App Store to reach it for national purpose.

Acknowledgments

We sincerely thank our Department of Computing Technologies and our institution SRM Institute of Science and Technology, for their help during our project work. Also, we would like to thank our parents, our family members and our friends for their unconditional love, constant support and encouragement.

Funding

No funding is provided for the preparation of manuscript.

Conflicts of interest

The authors have no conflicts of interest to declare.

References

[1] Sonawane, P., *et al.*, 2021, "Speech To Indian Sign Language (ISL) Translation System," *2021 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS)*, Greater Noida, India, doi: 10.1109/ICCCIS51004.2021.9397097.
[2] Peguda, J., *et al.*, 2022, "Speech to Sign Language Translation for Indian Languages," *2022 8th International Conference on Advanced Computing and Communication Systems (ICACCS)*, Coimbatore, India, doi: 10.1109/ICACCS54159.2022.9784996.
[3] Monga, H., *et al.*, 2021, "Speech to Indian Sign Language Translator," *Volume 39: Recent Trends in Intensive Computing*, doi: 10.3233/APC210172.
[4] Kanvinde, A., *et al.*, 2021, "Bidirectional Sign Language Translation," *2021 International Conference on Communication information and Computing Technology (ICCICT)*, Mumbai, India, doi: 10.1109/ICCICT50803.2021.9510146.
[5] Stoll, S., *et al.*, 2020, "Text2Sign: Towards Sign Language Production Using Neural Machine Translation and Generative Adversarial Networks," *International Journal of Computer Vision (volume 128, pages 891–908) (2020)*, doi: https://doi.org/10.1007/s11263-019-01281-2.
[6] B. Sarkar *et al.*, 2009, "A Translator for Bangla Text to Sign Language," *2009 Annual IEEE India Conference*, Ahmedabad, India, doi: 10.1109/INDCON.2009.5409449.
[7] Yadav, A., *et al.*, 2021, "Audio to Sign Language Translator Web Application," *2021 International Conference on Computational Performance Evaluation (ComPE)*, Shillong, India, doi: 10.1109/ComPE53109.2021.9751857.
[8] Jadhav, K.; Gangdhar, S. and Ghanekar, V., 2021, "Speech to ISL (Indian Sign Language) Translator," *International Research Journal of Engineering and Technology (IRJET) (Volume: 08 Issue: 04, Page 3696- 3698).*
[9] Sharma, P., *et al.*, 2022, "Translating Speech to Indian Sign Language Using Natural Language Processing," *Future Internet 2022, 14(9), 253*, doi: https://doi.org/10.3390/fi14090253.

[10] Kulkarni, A., *et al.*, 2021, "Speech to Indian Sign Language Translator," *3rd International Conference on Integrated Intelligent Computing Communication & Security (ICIIC 2021)*, doi: https://doi.org/10.2991/ahis.k.210913.035.
[11] Rokade and Jadav, 2017, "Indian Sign Language Recognition System," *International Journal of Engineering and Technology 9(3S):189-196*, doi: 10.21817/ijet/2017/v9i3/170903S030.
[12] Katoch, S.; Singh, V. and Tiwary, U. S., 2022, "Indian Sign Language recognition system using SURF with SVM and CNN," *Array 14 (2022): 100141*, doi: https://doi.org/10.1016/j.array.2022.100141.
[13] Gupta, B.; Shukla, P. and Mittal, A., 2016, "K-nearest correlated neighbor classification for Indian sign language gesture recognition using feature fusion," *2016 International Conference on Computer Communication and Informatics (ICCCI)*, Coimbatore, India, doi: 10.1109/ICCCI.2016.7479951.
[14] Tripathi, K.; Baranwal, N. and Nandi, G. C., 2015, "Continuous Indian sign language gesture recognition and sentence formation," *Eleventh International Multi-Conference on Information Processing-2015 (IMCIP-2015), Procedia Computer Science 54 (2015) 523 – 531*, doi: https://doi.org/10.1016/j.procs.2015.06.060.
[15] Dutta, K. K., *et al.*, 2015, "Double handed Indian Sign Language to speech and text," *2015 Third International Conference on Image Information Processing (ICIIP)*, Waknaghat, India, doi: 10.1109/ICIIP.2015.7414799.

Authors Profile

Anannya Priyadarshini Neog, received her B.Tech degree in Computer Science and Engineering from SRM Institute of Science and Technology, Chennai, Tamil Nadu, India. In addition to her research interests in Artificial Intelligence, she is interested in Application Development.

Arunabh Kalita, received her B.Tech degree in Computer Science and Engineering from SRM Institute of Science and Technology, Chennai, Tamil Nadu, India. In addition to working on full stack development, currently he is also learning machine learning and artificial intelligence.

Nithyakani Pandiyarajan, is pursing part time Ph. D and Assistant professor SRM Institute of Science and Technology, Chennai, India. She completed her bachelor of engineering in Information Technology in 2009 and her master's in Computer science and Engineering from PRIST University, Puducherry in 2013. Her research areas include Pattern Recognition, Biometrics, Gait Analysis and Medical Imaging.