WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

Social Interaction Web App for Differently Abled using Machine Learning

[1]Kiran Kumari, [2]Aabha Borle, [3]Anam Choulkar, [4]Varsha Pandit, [5]Sahil Hate

[1,2,3,4,5]Department of IT, K.J.S.C.E, Mumbai University, Mumbai, Maharashtra, India

[1]kirankumarisinha@somaiya.edu, [2]aabha.borle@somaiya.edu, [3]anam.choulkar@somaiya.edu, [4]varsha.pandit@somaiya.edu, [5]sahil.hate@somaiya.edu

ABSTRACT:

Human beings communicate through language, be it verbal or sign language.. Hearing and Speech impaired people, not having a way to communicate verbally, make use of Sign Language. They show gestures using sign language in order to convey their message and effectively communicate with each other. As American Sign Language (ASL) is not known by most people, it becomes difficult for people with no impairments to fluently communicate with the Hearing and Speech impaired community. This paper proposes an ASL gesture recognition system along with text-to-speech using pyttsx3 and vice versa in order to decrease this communication gap. The dataset consists of images of ASL gestures which are performed by different subjects. The proposed system uses OpenCV for image processing and MediaPipe for extraction of key points. Once the extraction of keypoints is done, Long Short Term Memory (LSTM) is used for classification of gestures. The LSTM model classifies the ASL gesture as to which sign it belongs to. Conversion of Text/Speech to ASL is carried out using the NLTK python package for NLP and WebKit Speech Recognition API.

Keywords: American Sign Language (ASL); OpenCV; Tensorflow; Key Points; Long Short Term Memory (LSTM); Natural Language Toolkit (NLTK).

I. INTRODUCTION

This project proposes a platform that can provide social interaction means for the differently abled with each other as well as with people with no impairment. The application takes as input the sign language gestures, voice messages as well as text messages and converts them into text, speech or image format as required by the end user. A machine learning model is used to determine the accuracy of the recognized sign language gesture and convert it as required by the user.

The existing systems use some specialized hardware sensors, gloves, or IR Based optical markers. These hardware devices are at the moment expensive and unmanageable. It is very common for people nowadays to carry hand held devices like Mobile phones, Tablets, Laptops etc. and these devices have become the communication medium between people. Hence a system that can make use of these devices to recognise the signs is more convenient than having a specialized hardware around.

The paper is organized as follows: The section I tell us about the introduction of sign language. In section II, the analysis of literature survey is mentioned which talks about the works that are already done along with the drawbacks in it. Section III mentions the proposed methodology. In section IV, implementation and results are discussed. Section V mentions the future work and section VI is the conclusion. The references are mentioned at the end of the paper.

II. LITERATURE SURVEY

Numerous authors have proposed a variety of image processing and Machine Learning (ML) techniques for sign language recognition. In the mid-1970s, Myron W. Krueger introduced gesture recognition as an innovative communication method between humans and computers. With the recent advancements in computer hardware and vision systems, it has become a significant research area. A. A. Abdulhussein and F. A. Raheem [1] discussed the use of grayscale images and edge detection techniques for hand gesture detection. However, grayscale images have limitations as they only represent 2D data, making it challenging to extract key hand features. Teak-Wei Chong and Boon-Giin Lee [2] presented a technique using a leap motion controller for hand gesture recognition, which requires additional expensive hardware. Matteo Rinalduzzi [3] proposed a glove-based approach with hand sensors for gesture recognition, but the sensors are costly and add weight, limiting hand mobility. Rajesh George Rajan and Dr. M. Judith Leo [4] employed deep learning techniques with a public dataset from Kaggle to recognize American Sign Language (ASL) gestures using various ML algorithms.

Aashni Hariaa et al. [9] utilized color features and contour extraction for hand gesture recognition. However, this approach is more suitable for finger counting rather than recognizing different hand gestures. Pei Xu [10] presented a real-time hand gesture recognition approach using Convolutional Neural Networks (CNN) for human-computer interaction, but it focuses on general hand signs for human-computer interface rather than sign language gestures. Abhishek Bet al. [11] proposed a 3D CNN-based algorithm for hand gesture detection, but it does not discuss how to detect gestures involving motion. Gongfa Li et al. [12] described a CNN-based method for hand gesture detection using Kinect, but Kinect is expensive. Noorkholis Luthfil Hakim et al. [13] introduced an approach using 3D CNN and LSTM for recognizing dynamic hand gestures, but it is only specific to military gestures rather than alphabet-based sign language.

Sourav Bhowmick, Sushant Kumar, and Anurag Kumar [5] introduced a technique for ASL gesture recognition using Deep Learning for two-letter combinations. However, this approach relies on still images and may struggle to accurately track gestures involving motion. Onamon Pinsanoh, Yuttana Kitjaidure, and Ariya Thongtawee [6] proposed a technique that divides the frame into left and right sides for gesture recognition, but it increases the complexity of feature extraction. Chinmaya R. Naguri et al. [7] presented a technique for detecting dynamic hand gestures using 3D motion and a Leap motion controller. Kai Li, Qieshi Zhang, Jun Cheng, and Jianming Liu [8] demonstrated the use of hand tracking and gesture recognition for Human-Computer Interaction (HCI) through hand region segmentation. Thakur et al. [15] focused on real-time sign language recognition and speech generation using a CNN algorithm. They replaced motion-based signs with static images.

From the literature review, it is evident that gesture recognition research remains an active area, and there is a need for new methods to effectively leverage diverse computer vision algorithms.

III. PROPOSED METHODOLOGY

Recognizing sign language using a 2-dimensional RGB camera is a difficult task. We extracted the important and noiseless features out of the videos that would be captured for recognition purposes. 3-dimensional cameras which can capture depth information would do this job easily. But to make the system accessible for everyone and cheaper we chose to stick with the 2-dimensional camera. The features to be extracted could be then used for the further classification purposes. The classification was carried out using a simple 'LSTM' architecture.

A. For ASL to Text/Speech

The proposed system for the ASL alphabet recognition is shown in Figure 1.

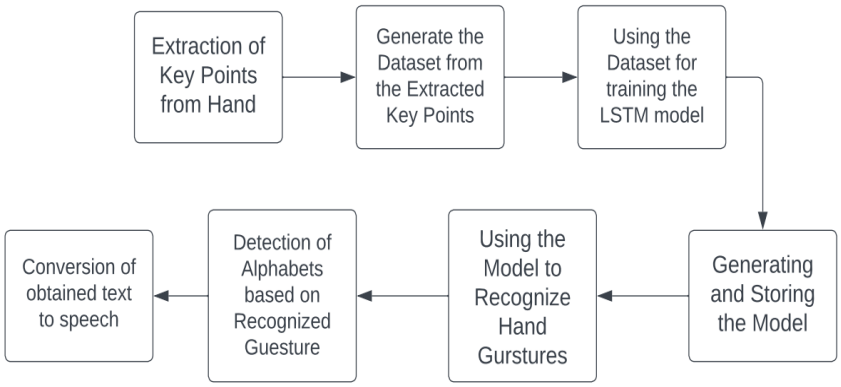

Figure 1: Architecture of ASL to Text/Speech Conversion

i. OpenCV for image processing

In sign language detection, OpenCV is useful for tasks such as hand detection and tracking, feature extraction, and image pre-processing. For example, OpenCV can be used to detect the hand region in a video stream and extract features such as hand shape and orientation. This information will then be fed into a machine learning model such as an LSTM network for sign language recognition.

Preprocessing the input images or video frames is critical so as to improve the accuracy of LSTM models for sign language recognition. OpenCV provides a range of image processing functions that can be used to improve the quality of input images. Using OpenCV, image processing and feature extraction can be performed in real-time, enabling sign language recognition in live video stream format.

ii. MediaPipe for extraction of the key points

MediaPipe provides a pre-built hand detection and tracking module that can detect and track the movements of the signer's hands. This can provide valuable information about the signer's hand gestures and movements, which can be used as input to LSTM models for sign language recognition. (Figure 2)

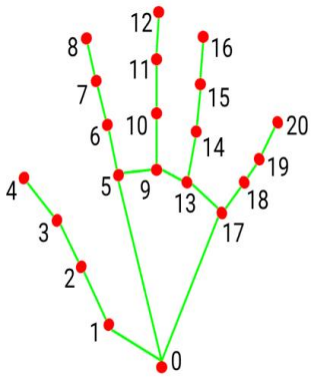

Figure 2: Hand Landmarks representation used in MediaPipe [16]

iii. Training the model using LSTM

The LSTM model is trained using the dataset in the next step. The advantage of using LSTM is that it allows not only single data points i.e images, but it also allows the entire sequences of data[14].

Keras provides a high-level API for building LSTM models, making it easy for developers to create custom models for sign language recognition. TensorFlow provides a lower-level API for building LSTM models, which provides greater flexibility and control over the model architecture and training process.

Once the LSTM model has been built using Keras for TensorFlow, it can be trained on a dataset of sign language images. The training process involves feeding the pre-processed images into the model, computing the model's output, and adjusting the model's parameters to minimize the difference between the model's output and the desired output.

After training, the LSTM model can be evaluated on a separate dataset of sign language images to assess its performance. Keras and TensorFlow provide tools for computing various metrics such as accuracy, precision, recall, and F1 score.

Once an LSTM model has been trained and evaluated, it can be deployed in a sign language detection system. The LSTM model can take pre-processed images as input, and output a predicted sign language gesture.

iv. Using pyttsx3 library

Pyttsx3 is a Python library that provides a simple and straightforward way to convert text to speech. Here, the text obtained from ASL conversion is used as an input to convert the resulted text into speech.

B. For Speech/Text to ASL

The proposed system for the conversion of Text/Speech to ASL is shown in Figure 3.

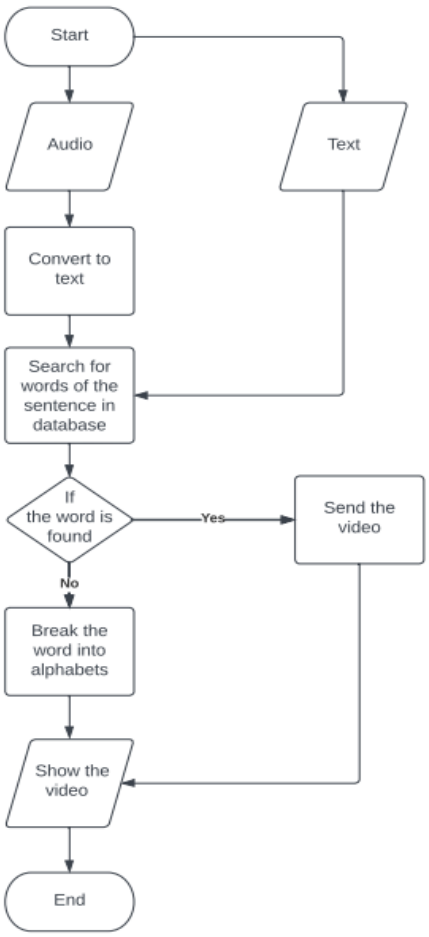

Figure 3: Flowchart for Text/Speech to ASL Conversion

i. NLTK python package for NLP

Since NLTK provides tokenization functionality, it is useful for breaking down the input text into individual words or sentences. Tokenization is an important initial step in NLP, as it helps in further processing and analysis of text data.

NLTK includes part-of-speech (POS) tagging capabilities, which assign grammatical tags (e.g., noun, verb, adjective) to words in a sentence. POS tagging can aid in identifying the syntactic structure of the input text and can be useful for generating accurate sign language translations.

It also provides tools for lemmatization and stemming, which are techniques used to reduce words to their base or root form. These techniques help in normalizing the text and reducing inflectional forms, which is beneficial for sign language translation.

It includes WordNet, a lexical database that provides information about words and their semantic relationships. WordNet can be used for synonym detection, finding word definitions, and exploring semantic connections between words. This information can be helpful in generating appropriate sign language equivalents for words.

ii. WebKit Speech Recognition API

The WebKit Speech Recognition API which is a JavaScript API, is used to convert speech into text. It provides speech recognition capabilities in web browsers based on the WebKit engine. It is used to integrate speech recognition into the web application, enabling users to interact with the application using their voice.

IV. IMPLEMENTATION AND RESULTS

A. ASL to Text/Speech

To create our own dataset, a file “collectdata.py” was created which requires two modules- os and cv2. Cv2 is used to perform image processing. A folder for all 26 letters of alphabets and a few words are created inside an Images folder. Once the data is organized, we can run the script "collectdata.py" to collect and label the data, capturing images and assigning appropriate labels for the taken image.

The script "function.py" is created to define common functions that will be used throughout the project. These functions are used for mediapipe detection, drawing styled landmarks, and to extract key points.

As the collected data is in raw form, it needs to be converted into an appropriate format for further processing. This step involves converting the image data into arrays, which can be done using libraries like OpenCV or NumPy.

The script "data.py" is responsible for extracting data from the images. It processes the images to extract landmarks or key points of the hand, which are important features for sign language recognition. The extracted landmarks or keypoints are then saved in the .npy format, typically in a folder called "MP_data". This step creates an organized dataset consisting of arrays representing the key points of hand gestures. (Figure 4)

Figure 4: Collecting frames for alphabets/words

The script "trainmodel.py" is used to train the LSTM model for sign language recognition. It imports the necessary libraries, including TensorFlow, and sets up the model architecture. The “sequential()” model typically consists of three LSTM layers followed by three dense (fully connected) layers (Figure 5). Once the model is trained on the dataset, it is saved as "model.h5" and "model.json" for future use. The script also provides an evaluation of the model's accuracy after training.

Finally, the system is executed that utilizes the trained model. It takes input, typically in the form of images and uses the LSTM model to recognize sign language gestures. The output can be displayed on the screen or processed further. (Figure 6)

After the conversion of ASL to text, the resulting text is converted into audio using pyttsx3 library.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm (LSTM)                  (None, 30, 64)            442112
_________________________________________________________________
lstm_1 (LSTM)                (None, 30, 128)           98816
_________________________________________________________________
lstm_2 (LSTM)                (None, 64)                49408
_________________________________________________________________
dense (Dense)                (None, 64)                4160
_________________________________________________________________
dense_1 (Dense)              (None, 32)                2080
_________________________________________________________________
dense_2 (Dense)              (None, 3)                 99
=================================================================
Total params: 596,675
Trainable params: 596,675
Non-trainable params: 0
```

Figure 5: Summary of the sequential model

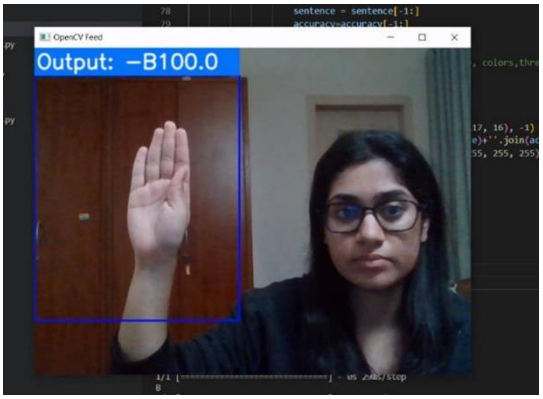

Figure 6: Output with accuracy

B. Text/Speech to ASL

For this part of the project, the Django framework and NLP packages are utilized to convert text into (ASL). Necessary packages for NLP and other functionalities are imported and loaded within the project.

The text input is converted to lowercase to standardize the text. Then, the text is tokenized using the `word_tokenize()` function from the NLTK package. Tokenization breaks the sentence into individual words, allowing for further analysis.

Stop words, which are commonly used words with little semantic meaning, are removed from the text. This process helps to eliminate noise and focus on the more meaningful words. Lemmatization, a process that reduces words to their base or root form, is performed using the `WordNetLemmatizer()` function. The lemmatized words are appended to a `filtered_text()` list or data structure, which contains the processed and filtered words.

The words are then mapped to corresponding videos or animations of sign language gestures. The appropriate sign language gesture or animation is selected based on the words and their context. The selected videos are then displayed using an HTML template called `animation.html`, allowing users to visualize the sign language representation of the input text.

Speech-to-text conversion is achieved using the WebKit Speech Recognition API, a JavaScript API for speech recognition in web browsers. The API captures the user's speech input, which is then converted into text. The obtained text is further processed using the NLP techniques described above to convert it into ASL. (Figure 7)

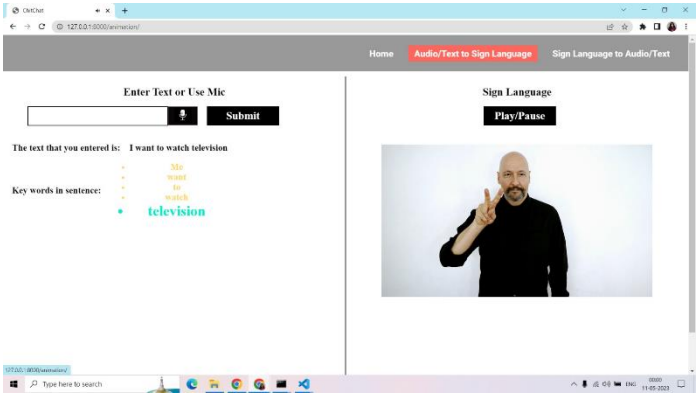

Figure 7: Text/Speech to ASL

Further, the integration of ASL to text/speech and Text/Speech to ASL is done using Flask API.

V. FUTURE SCOPE

- Building larger and more diverse datasets specific to ASL sign language will enhance the training and performance of LSTM models. Improving the speed and efficiency of ASL sign recognition systems which is crucial for real-time applications.
- Investigating transfer learning techniques that leverage pre-trained LSTM models on large-scale sign language datasets or related tasks can accelerate the development of ASL sign recognition systems. Few-shot learning approaches can also be explored to improve the model's ability to recognize new signs with limited training data.

- Developing user-friendly interfaces and applications for ASL sign recognition can facilitate effective communication between sign language users and non-signers. This involves designing intuitive and accessible user interfaces and incorporating feedback mechanisms for real-time corrections and suggestions.
- Optimizing ASL sign recognition models for deployment on mobile devices or edge computing platforms will enable on-device inference and promote accessibility. This can involve model compression techniques, quantization, and efficient architecture design to ensure accurate and efficient sign recognition on resource-constrained devices.

VI. CONCLUSION

In conclusion, our project successfully developed a comprehensive system that bridges the gap between speech and American Sign Language (ASL) by integrating speech-to-text, text-to-ASL, ASL-to-text, and text-to-speech functionalities. Through this project, we aimed to enhance communication and inclusivity for individuals who use ASL as their primary means of communication. Through the integration of these modules using a Flask API framework, we have created an interactive platform that supports bidirectional communication between spoken language and ASL. This project opens up new possibilities for individuals with hearing impairments to communicate effectively with the wider community and bridges the communication gap between the hearing, deaf and mute communities.

REFERENCES

[1] A. A. Abdulhussein and F. A. Raheem, “Hand gesture recognition of static letters American sign language (ASL) using deep learning,” Engineering and Technology Journal, Vol. 38, No. 06, pp. 926-937, 2020.

[2] Teak-Wei Chong and Boon-Giin Lee, American Sign Language Recognition Using Leap Motion Controller with Machine Learning Approach, Department of Electronics Engineering, Keimyung University, Daegu 42601, Korea, October 2018.

[3] Matteo Rinalduzzi, Alessio De Angelis, Francesco Santoni, Emanuele Buchicchio, Antonio Moschitta, Paolo Carbone, Paolo Bellitti, Mauro Serpelloni, Gesture Recognition of Sign Language Alphabet Using a Magnetic Positioning System, June 2021.

[4] Rajesh George Rajan, Dr.M.Judith Leo, American Sign Language Alphabets Recognition using Hand Crafted and Deep Learning Features IEEE Xplore Part Number:CFP20F70-ART published on 2020

[5] Sourav Bhowmick, Sushant Kumar and Anurag Kumar, Hand Gesture Recognition of English Alphabets using Artificial Neural Network, published in 2015.

[6] Onamon Pinsanoh, Yuttana Kitjaidure, Ariya Thongtawee. A Novel Feature Extraction for American Sign Language Recognition Using Web- cam. Published in 2018.

[7] Chinmaya R. Naguri, Razvan C. Bunescu.Recognition of Dynamic Hand Gestures from 3D Motion Data using LSTM and CNN architectures. Published in 2017.

[8] Kai Li, Qieshi Zhang, Jun Cheng, Jianming Liu .Hand Gesture Tracking and Recognition based Human-Computer Interaction System and Its Applications published in 2018 .

[9] Aashni Hariaa, Archanasri Subramaniana, Nivedhitha Asokkumara, Shristi Poddara, Jyothi S Nayaka, Hand Gesture Recognition for Human Computer Interaction. 7th International Conference on Advances in Computing Communications, ICACC- 2017, 22- 24 August 2017, Cochin, India.

[10] Pei Xu, Department of Electrical and Computer Engineering, University of Minnesota, Twin Cities. A Real-time Hand Gesture Recognition and Human- Computer Interaction System.

[11] Abhishek B, Kanya Krishi, Meghana M, Mohammed Daaniyaal, Anupama H S , BMS Institute of Technology, Bangalore, India. Hand gesture recognition using machine learning algorithms. Computer Science and Information Technologies.

[12] Gongfa Li, · Heng Tang, · Ying Sun, · Jianyi Kong, · Guozhang Jiang, · D u Jiang · Bo Tao, ·Shuang Xu, · Honghai Liu. Hand gesture recognition based on convolution neural network. part of Springer Nature 2017.

[13] Noorkholis Luthfil Hakim, Timothy K. Shih, Sandeli Priyanwada Kasthuri Arachchi, Wisnu Aditya, Yi-Cheng Chen and Chih-Yang Lin. Dynamic Hand Gesture Recognition Using 3DCNN and LSTM with FSM Context-Aware Model.

[14] Sepp Hochreiter ,Fakultat fur Informatik, Technische Universitat Munchen, 80290 Munchen, Germany. LONG SHORT-TERM MEMORY, Neural Computation 9(8):1735-1780, 1997

[15] Thakur, Amrita, Pujan Budhathoki, Sarmila Upreti, Shirish Shrestha, and Subarna Shakya. ”Real Time Sign Language Recognition and Speech Generation.” Journal of Innovative Image Processing 2, no. 2 (2020): 65-76.