International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 11 Issue: 04 | Apr 2024 www.irjet.net p-ISSN: 2395-0072

SignSense – Sign Language Detection and Translation Software

Prof. Rahul Laxman Thorat[1], Atharva Kadam[2], Aditya Arkasali[3], Makarand Warade[4]

[1]*Professor, Dept. of Computer Engineering, TSSM Bhivarabai Sawant College of Engineering & Research Pune, India*
[2] *Dept. of Computer Engineering, TSSM Bhivarabai Sawant College of Engineering & Research Pune, India*
[3]*Dept. of Computer Engineering, TSSM Bhivarabai Sawant College of Engineering & Research Pune, India*

Abstract - ***Sign language is a vital means of communication for millions of individuals worldwide, particularly those who are deaf or hard of hearing. This software, aptly named SignSense, leverages cutting-edge computer vision and machine learning techniques to detect and translate sign language gestures into text and in real time. SignSense's core innovation lies in its ability to bridge the communication gap between the hearing and non-hearing communities. By harnessing the power of computer vision, it accurately interprets the intricate movements and expressions of sign language, transforming them into clear, concise text. Moreover, SignSense can be integrated into various platforms and devices, opening up endless possibilities for communication, from educational settings to daily interactions.***

Index Terms: Sign Language, Translation, ASL, Deep Learning.

1.INTRODUCTION

The primary aim of the SignSense project is to develop an advanced sign language detection and translation software that enhances accessibility and communication for individuals who are deaf or hard of hearing, ultimately fostering a more inclusive society. SignSense, a sign language detection and translation software, is born out of a deeply rooted commitment to address the communication barriers faced by individuals who are deaf or hard of hearing. The motivation behind this project stems from a profound understanding of the significance of inclusivity, as we aim to create a world where everyone can participate in conversations, education, and various facets of daily life without any restrictions.

1.1 Scope of Work

The scope of work for the SignSense project encompasses a multifaceted approach that aims to achieve a comprehensive and groundbreaking sign language detection and translation solution. The project will commence with in-depth research into sign language linguistics and recognition methodologies, laying the foundation for a robust sign language gesture detection system. This will involve the acquisition of a diverse dataset of sign language gestures, their annotation, and the development of machine learning models for real- time gesture recognition.

Following the research phase, the project will shift its focus to the development of the software's core components, including the sign language detection module, translation to text, and speech synthesis functionality. The software will be designed to run on multiple platforms, making it accessible to a wide range of users. The project's scope also extends to advocacy and awareness initiatives aimed at promoting the adoption of SignSense in educational institutions, workplaces, and public spaces. This holistic approach underscores the project's ambition to create a more inclusive society by breaking down the communication barriers faced by the deaf and hard of hearing communities. Sign language recognition is an area of research that involves pattern matching, deep learning, computer vision, natural language processing, and a design module or algorithm to identify sign language. It can be extended further to human-computer interaction without a voice interface. This system belongs to multidisciplinary content and the approach can be considered as a part of the Sign Language System. [1]

1.2 Outcomes

The SignSense project anticipates a range of significant outcomes that will contribute to its overarching mission of enhancing communication and accessibility for individuals who are deaf or hard of hearing. These outcomes include: Effective Sign Language Communication: SignSense will empower individuals who use sign language to communicate effectively with both the hearing and non-hearing communities, reducing the communication gap and fostering greater inclusivity. Improved Education Access: The software's integration into educational settings will enable students who are deaf or hard of hearing to access classroom discussions and educational resources in real time, thereby improving their educational outcomes.

Enhanced Workplace Inclusivity: By providing a tool for clear communication, SignSense will facilitate the inclusion of individuals with hearing impairments in the workplace, expanding their employment opportunities and contributing to diverse and productive work environments.

Increased Social Participation: Deaf and hard of hearing individuals will be able to actively participate in social interactions, contributing to a more inclusive society and reducing social isolation.

1.3 Literature Survey

SL recognition systems on movies typically comprise a temporal model that maps sequential representations to labels and a feature extraction module that extracts sequential representations to characterize gesture sequences. By using sequence learning models, sign language recognition seeks to understand the relationships between input sequences and sign labels. Vision-based continuous sign language recognition systems often learn to automatically output the gloss labels in the correct order using image sequences of signers' performances as input.

Deep et al. [1] have researched on deep learning-based Indian sign language recognition system with processed static images implemented on Indian Sign Language gestures. The proposed methodology has achieved 99.29% accuracy over the 36 image-based Indian sign language classes. The proposed methodology has achieved significant accuracy using a smaller number of attention layers in the encoding component of the transformer as well as a very small number of learning cycles.

Runpeng et al. [2] utilized DeepFlow method for computing optical flow. In instances where the original databases do not provide information about the dominant hand locations, the faster R-CNN framework is employed to detect the dominant hand in each frame of the video.
For training the feature extraction module based on alignment proposals, they divided the pairs of segments and gestural labels into training and validation sets at a ratio of 10:1. They utilized Adam as their stochastic optimization approach, maintaining a fixed learning rate of 5 × 10^(-5) and a mini-batch size of 20. The neural architecture is implemented in Theano, and experiments are conducted on NVIDIA Titan X GPUs.
It's noteworthy that their classifier, using color image modal, achieves a top-1 accuracy of 63.70% and a top-5 accuracy of 86.37%. Additionally, the classifier combining color and optical flow achieves a top-1 accuracy of 75.70% and a top-5 accuracy of 91.93% across 450 gestural classes.

Jyotishman et al. [3] This work attempts to provide a visual solution for the sign language recognition problem in the regional Indian language, where different sets of alphabets exist for each. A state-of-the-art methodology has been adapted to implement the solution using advanced tools like MediaPipe. Both real-time and static gestures have been tried to be recognized by training a self-generated 3D and 2D image dataset to a Feed Forward Neural Network. 94 data points has been generated, including nine static gestures with vowels and consonants from the Assamese Sign Language. The dataset was used for training of a feed-forward neural network. The model yielded an accuracy of 99%.

Prakhyath et al [4] employed the "Support Vector Machine" (SVM), a supervised machine learning approach frequently utilized for tackling classification problems, though it's also adaptable for regression tasks. Within SVM, each data point is depicted as a point within a multi-dimensional space, where the number of dimensions aligns with the number of features in the dataset. The value of each feature serves as the coordinate for that point. Subsequently, the algorithm discerns a hyperplane that proficiently distinguishes between the various classes present in the dataset.

Romala Sri Lakshmi Murali et al [6], The model is evaluated based on 10 alphabetic American sign language including: A, B, C, D, H, K, N, O, T and Y. They have used a total of 2000 images to train the Convolutional Neural Network. The dataset is split in the ratio of 80:20 for training and testing respectively. The results used in this paper gives us an accuracy of over 90%.

Ravikiran[5] et al. used an effective and rapid algorithm for determining the number of fingers displayed in a gesture representing a letter of American Sign Language. They achieve finger detection by employing Boundary Tracing and Finger Tip Detection techniques.

Hee-Deok Yang[7], In this paper, a novel method for recognizing sign language hand gestures was proposed. In order to detect 3D locations of the hands and face, depth information generated with Microsoft's Kinect was used. A hierarchical threshold CRF is also used in order to recognize meaningful sign language gestures using continuous hand motions. Then, the segmented sign was verified with the BoostMap embedding method. Experiments demonstrated that the proposed method recognized signs from signed sentence data at a rate of 90.4%. Near-term future work includes improving the detection accuracy of the upper body components.

Aman et al [8] utilized Convolutional Neural Networks (ConvNet/CNNs), which are a type of Deep Learning system, to analyze images and assign significance to different features within them. He accomplished this by training the network to learn weights and biases associated with various aspects or objects in the image.

ConvNets have the ability to learn filters and features automatically, removing the necessity for manual filter engineering. Designed as multilayer artificial neural networks, ConvNets are tailored to process either 2D or 3D data inputs. Within each layer of the network, there are multiple planes, which may be either 2D or 3D, and each plane holds numerous independent neurons. Neurons in adjacent layers are interconnected, while those within the same layer are not linked.

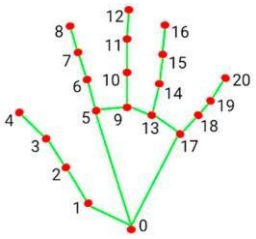
2. Materials & Methods

The proposed architecture uses a machine learning based classification methodology to effectively implement the project requirements. The project consists of three major parts: Detection, Identification & Translation.

2.1 Sign Detection –

MediaPipe framework is used to build machine learning pipelines for time-series data such as video, audio, etc. Google first introduced it to perform real-time video and audio analysis on YouTube. In 2019, MediaPipe's public release enabled researchers and developers to integrate and use this framework in their projects. Unlike most high computing power demanding machine learning frameworks, MediaPipe can run efficiently, maintaining its accuracy and robustness on devices with low computing power, such as androids and embedded IoT devices [3].

Fig. 1 – Localized Hand Landmarks by MediaPipe

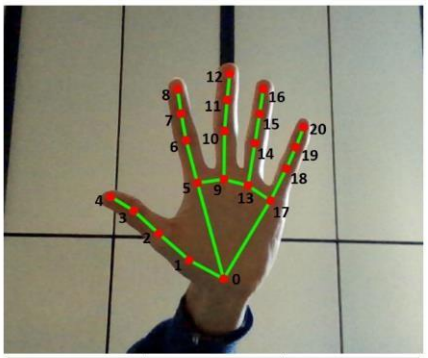


0. WRIST
1. THUMB_CMC
2. THUMB_MCP
3. THUMB_IP
4. THUMB_TIP
5. INDEX_FINGER_MCP
6. INDEX_FINGER_PIP
7. INDEX_FINGER_DIP
8. INDEX_FINGER_TIP
9. MIDDLE_FINGER_MCP
10. MIDDLE_FINGER_PIP
11. MIDDLE_FINGER_DIP
12. MIDDLE_FINGER_TIP
13. RING_FINGER_MCP
14. RING_FINGER_PIP
15. RING_FINGER_DIP
16. RING_FINGER_TIP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

This project used pictures of hand signs to infer hand landmarks using the MediaPipe technology. As it is simpler to identify stiff items like fists and palms than entire hands, a palm detection model first identifies the palms of the hands in the pictures. The Hand Landmark Model is the next model that receives cropped palm photos from this one. Approximately 29,000 hand annotated real-world photos are used to train the entire model. Since the model is highly reliable and well-trained, it can reliably identify and map hand landmark locations, even on hands that are often only partially visible.

2.2 Sign Identification –

MediaPipe employs compartmentalization techniques to isolate specific hand points, enabling precise segmentation of hand features. These segmented points are then subjected to a mapping process, yielding a comprehensive set of coordinates representing the hand's spatial configuration. Due to the dynamic nature of the input method, this dataset of coordinates is extensive, capturing diverse hand gestures and positions. The set is then cleaned, transformed and normalized to improve the training efficiency.



X	Y	Z
0.5232	0.9016	0
0.3951	0.8343	- 0.0488
0.2967	0.7415	- 0.0917
0.2132	0.6652	- 0.1347
0.1393	0.6288	- 0.1805
0.4076	0.5240	- 0.0844
0.3640	0.3575	- 0.1315
0.3386	0.2527	- 0.1708
0.3186	0.1652	- 0.1944
0.4881	0.5096	- 0.0826
0.4695	0.3106	- 0.1212
0.4590	0.1866	- 0.1680
0.4516	0.7980	- 0.2103
0.5619	0.5309	- 0.0777
0.5743	0.3544	- 0.1129
0.5813	0.2370	- 0.1491
0.5849	0.1331	- 0.1781
0.6297	0.5818	- 0.0748
0.6649	0.4524	- 0.1095
0.6902	0.3699	- 0.1381
0.7043	0.2881	- 0.1629

Fig. 2 – Hand Landmarks extracted by MediaPipe and coordinates

These coordinate sets are subsequently transmitted to the model for further translation and analysis, facilitating accurate interpretation of hand movements and gestures.

2.3 Sign Translation –

The dataset coordinates, extracted from the input data, serve as inputs for the machine learning model.

Fig. 2 – Sign Language Translation Results and Probability

This model has undergone rigorous training, involving extensive exposure to various hand gestures and positions during the learning process.
Finally, this character is transmitted back to the display screen, where it is rendered and presented to the user.

3. ARCHITECTURE OF THE SYSTEM

The system architecture comprises of data collection, model training, testing, classification, and web platform utilization. Beginning with data collection, raw input data is gathered, processed, and prepared for subsequent stages. Model training involves the extensive training of machine learning models using the collected data to learn patterns and relationships. Following training; testing and validation procedures are conducted to assess the performance and robustness of the trained models. Classification algorithms are then employed to interpret new data inputs and make predictions or classifications based on the learned patterns. Finally, the trained model is integrated into a web platform, allowing users to interact with the system via a user-friendly interface, enabling real-time data analysis and visualization.

3.1 Data Processing –

3. 1. 1. Data Collection –

In the data creation phase, a multi-frame video feed is utilized to capture images for analysis. The data acquisition process involves utilizing OpenCV to interface with a webcam, enabling real-time capture of video feed input. This feed is then processed to extract

individual frames, which are subsequently stored as data for further analysis. Through this method, the system generates over 1000 images for each character, ensuring a diverse and extensive dataset.

Fig. 3 – Sample Images

The images are generated from varied angles and different distances in order to randomize the model and provide better accuracy.

3.2.2 – Image Processing & Dataset Creation-

In the image processing and dataset creation phase, a series of steps are undertaken to prepare the collected images for analysis. Initially, the images undergo conversion from BGR to RGB color space as OpenCV images are loaded in BGR but other standard libraries including Matplotlib only accept images in RGB format, so conversion of the image ensures compatibility.

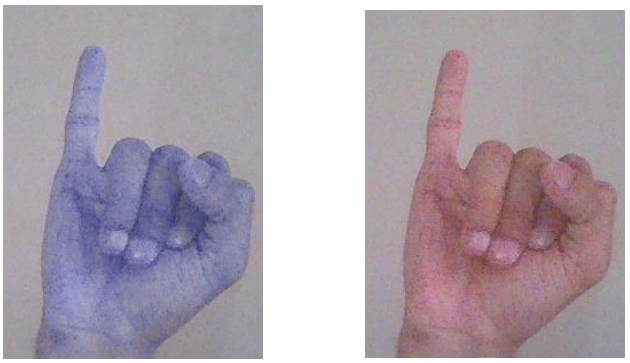


Fig. 4 – BGR to RGB Image Conversion

Utilizing MediaPipe, specific sections of the images, primarily focusing on hands and fingers, are targeted for further processing. Hand landmarks are detected within these isolated regions, and their coordinates are mapped accordingly. These coordinates are then organized into a labeled dataset, where each sample corresponds to a particular character. Finally, to facilitate convenient storage and accessibility, the dataset is serialized into a pickle file, ensuring integration into the subsequent stages of the pipeline.

3.2 – Comparing Multiple Classification Algorithms

In sign recognition, the choice of classification algorithm significantly impacts performance. The Classification algorithm most suitable for processing and predicted high dimensional data and the ones considered before training the model are – Convolutional Neural Networks, Support Vector Machine, & Random Forest.

Convolutional Neural Networks –

CNN or Convolutional neural network is a deep learning neural network i.e., we can think CNN as a machine learning algorithm that can take an input image, assign an importance to an object and then to be able to differentiate between one object and others. CNN works by extracting features from the images. Any CNN consist of three things which are – an input layer which is a grey scale image, then output layer which is the binary or multiclass labels and third hidden layers which contain convolution layer, RELU and then pooling layers and finally there is artificial neural network to perform the classification [9].

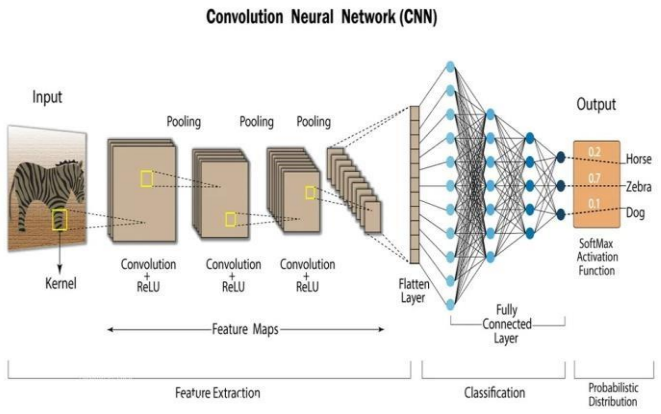


Fig. 5 – Convolutional Neural Networks

However, training a Convolutional Neural Network (CNN) requires copious amounts of labeled training data to capture intricate patterns directly from raw input, such as images. Moreover, the computational demands of CNN training are significant, owing to the numerous matrix operations involved in both the forward and backward passes through the network. This necessitates access to powerful hardware, such as GPUs or TPUs, to expedite computations and mitigate lengthy training times. During inference, CNNs may also exhibit longer detection times, particularly for deep architectures processing high-resolution images, as each layer sequentially processes the input data. Therefore, while CNNs offer state-of-the-art performance in various computer vision tasks, their data, computational, detection, and learning requirements make them resource-intensive solutions that may not always be feasible in environments with limited resources and for real time image processing.

Support Vector Machines

Support Vector Machines (SVMs) offer several advantages over deep learning models like Convolutional Neural Networks (CNNs). They are lightweight in terms of computational resources and boast a simpler architecture with fewer parameters to tune, facilitating easier training and deployment. Moreover, SVMs can yield satisfactory performance even with smaller datasets, which proves advantageous when data availability is limited.

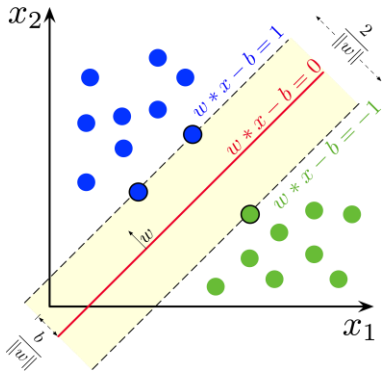


Fig. 6 – Support Vector Machine

However, it's important to consider that SVM training time may increase with dataset size, albeit remaining generally faster than training deep learning models. Additionally, while SVMs typically require fewer computational resources than CNNs, the choice of kernel function and hyperparameters can still impact training efficiency. Despite these considerations, SVMs remain a viable option for classification tasks, particularly in scenarios where computational resources are constrained or datasets are modest in size.

Random Forest –

Random Forest is a versatile machine learning algorithm that belongs to the family of ensemble methods. It is widely used for classification, regression, and anomaly detection Random Forest operates by constructing multiple decision trees during the training phase. Each decision tree is built using a random subset of the training data and a random subset of the features, which introduces diversity among the trees.

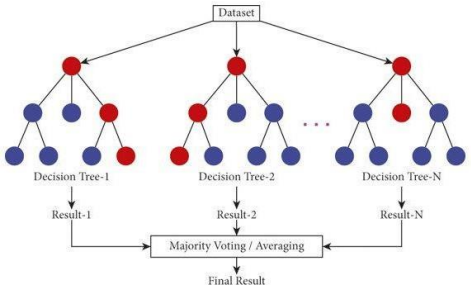


Fig. 7 – Random Forest

During prediction, the output of each decision tree is aggregated to make the final prediction. For classification tasks, the class with the most votes from the individual trees is chosen, while for regression tasks, the average prediction of all trees is taken. Random Forest is relatively easy to implement and less sensitive to hyperparameters compared to some other machine learning algorithms. It also handles missing values and outliers well, making it suitable for real-world datasets with noisy or incomplete data.

Random Forests, compared to deep learning models like Convolutional Neural Networks (CNNs), offer distinct advantages that make them favorable choices for classification tasks. One significant advantage is their computational efficiency, as they require fewer computational resources and boast a simpler architecture with fewer parameters to fine-tune. Additionally, Random Forests excel in handling smaller datasets, delivering robust performance even with limited data availability. Unlike SVMs, Random Forest training time remains consistently manageable, irrespective of dataset size. While SVMs may require careful selection of kernel functions and hyperparameters, Random Forests demonstrate reliability

without extensive parameter tuning, further underscoring their efficiency and effectiveness.
Overall, Random Forests present are better alternative to deep learning models, particularly in scenarios where computational resources are limited or datasets are relatively modest in size.

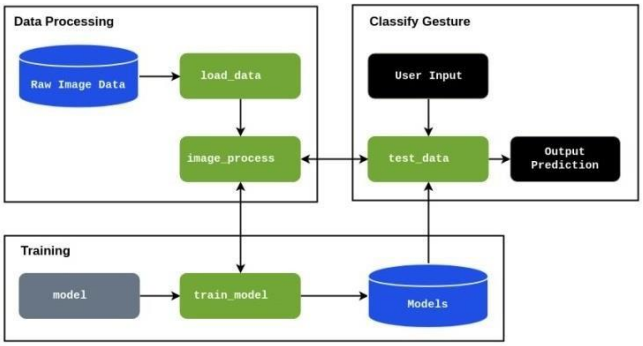


Fig. 8– General Architecture

3.3 – Model Testing and Training –

In the model training and testing phase, the process commences with the loading of the pickle dataset containing coordinates onto the designated machine learning model. Following this, a pivotal step involves the partitioning of the dataset into training and testing subsets, adhering to an 80-20 split ratio. This split ensures that both subsets represent the overall dataset adequately, with the 80% training portion utilized for model parameter optimization and the remaining 20% reserved for evaluating the model's generalization capability.
To mitigate the risk of overfitting, the data is shuffled prior to splitting. This randomization process disrupts any inherent sequence or pattern in the dataset, preventing the model from memorizing specific instances and instead encouraging it to learn generalizable patterns. Moreover, during this phase, the character data is encoded into numerical representations. This encoding enhances the model's ability to learn and generalize from the data by providing a structured numerical format for analysis.
During the testing phase, data is retrieved from the loaded file, and the model's predictions are checked. Each input-output relation is carefully assessed to decide the model's accuracy and effectiveness in predicting sign language gestures positions. This evaluation process aims to uncover any inaccuracies between the model's predictions and the ground truth, leading to the optimization of the model's parameters.
Finally, upon successful testing and validation, the finalized trained model is stored in the form of a .p file for future utilization. This file encapsulates the learned patterns and relationships acquired during the training phase, ensuring the model's portability and accessibility for subsequent deployments. Furthermore, the evaluation of the model reveals an impressive accuracy rate of 87%, underscoring efficiency in accurately predicting sign language gestures.

3.4 Deployment on Web Platform

In the development of a web platform for deploying the machine learning model, Flask served as the backbone for seamlessly integrating the browser interface with the underlying machine learning capabilities. Leveraging Flask's flexibility, the .p file containing the trained model was seamlessly incorporated, ensuring easy accessibility and utilization of the machine learning functionalities within the web environment.

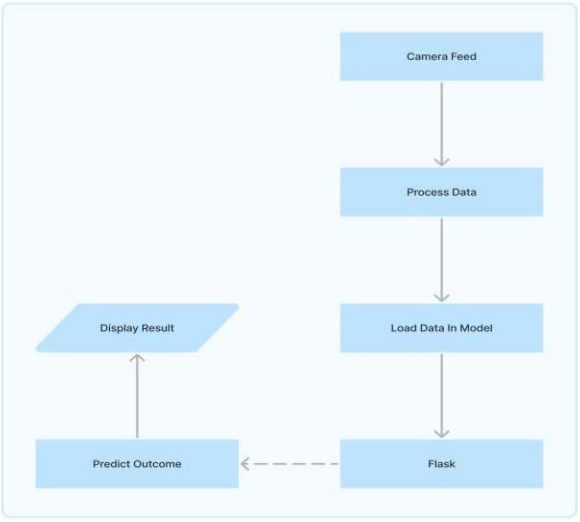


Fig. 9– Deployment Flow Chart

Platform's compatibility with a wide range of browsers improves the overall accessibility of the user, allowing them to interact with the machine learning model effortlessly across different browsing environments. By integrating live feed input and immediate display of predicted characters, the platform offers real-time feedback, further enhancing the user experience and usability.

4. RESULTS –

The classification_report library from python's scikit-learn toolkit was used for quantitative analysis of the test dataset. The classification_report library produces an evaluation report of our model with matrices - accuracy, precision, recall and F1 score. Along with them, the matrix 'support' represents the performance of the model in real-time recognition. The accuracy matrix measures correctly predicted labels by the model from the complete dataset. The accuracy matrix measures correctly predicted labels by the model from the complete dataset.

$$A\ (Accuracy) = \frac{TP+TN}{TP+TN+FP+FN}$$

The precision matrix measures the model's accuracy out of the predicted positives. It calculates the number of actual positives in the predicted positives. It is an excellent measure to consider when the False Positive (FP) cost is high.

$$P\ (Precision) = \frac{TP}{TP+FP}$$

The recall matrix measures how many predictions our model labels correctly as positives. It is a measure considered when the high cost is associated with False Negatives (FN).

$$R\ (Recall) = \frac{TP}{TP+F}$$

F1 score provides a cumulative measure by combining recall and precision. [3]

$$F\ (F1\ score) = \frac{2*P*R}{P+R}$$

The following table depicts the Accuracy, Precision, Recall and F1 Score of the developed Model.

Metric	Test
Accuracy	87.74%
Precision	0.89
Recall	0.91
F1-Score	0.89

Table 1 – Performance Metrics of the Model

5. FUTURE ENHANCEMENTS–

Opportunities for additional development and integration into other applications are presented by the created web platform. First, it may develop into a browser plugin that would increase its application and accessibility on many websites. Furthermore, because of its real-time nature, integration with video conferencing software may be able to provide smooth sign language interpretation for online meetings, improving accessibility to communication for those with hearing difficulties. Additionally, by utilizing its interactive characteristics, the platform may provide a basis for the development of cutting-edge instructional materials for the study of sign language. To engage users and promote successful learning, these technologies could include interactive lessons, quizzes, and gamified aspects. Additionally, investigating ways to include gesture recognition technology could improve the platform's functionality and enable uses outside of sign language interpretation.

6. CONCLUSION –

The advancement of sign language recognition technologies holds great promise for enhancing communication between the visually impaired and those with normal hearing. Through the incorporation of techniques such as brightness and angle adjustment, the system gains greater versatility. There are still issues, though, such real-time sign recognition in various contexts. Work is in progress to expand the system's understanding of continuous sign language and regional variances, as well as to remedy these concerns. Hand tracking is made simpler by programs like MediaPipe, which guarantees precise recognition on a range of platforms. Deep learning advancements and dataset growth will make these systems increasingly more efficient as research progresses, enabling improved communication among people with different linguistic backgrounds.

Communication between the visually impaired and those with normal hearing could be greatly enhanced by the development of sign language recognition technologies. To identify static sign motions, one method uses transformer-based models, which has demonstrated encouraging results with little training. The system gains versatility by integrating methods such as modifying angle and brightness. There are still difficulties, though, like identifying indications in various settings and in real time. In order to expand the system to comprehend continuous sign language and regional variances, efforts are still being made to overcome these problems. Accurate detection across several devices is ensured using hand tracking tools such as MediaPipe. Research on deep learning and dataset growth will make these systems increasingly more efficient as time goes on, facilitating improved communication between people with different linguistic backgrounds.

These issues are still being worked on in order to expand the system to understand continuous sign language and geographical variations. Hand tracking solutions like MediaPipe guarantee accurate detection across several devices. Over time, these systems will get more and more efficient due to research on deep learning and dataset expansion, which will increase communication amongst people with special needs.

7. References

[1] Deep R. Kothadiya, "SIGNFORMER: DeepVision Transformer for Sign Language Recognition," IEEE Explore,p. 10, 2022.

[2] Runpeng Cui, "A Deep Neural Framework for Continuous Sign Language Recognition by Iterative Training," IEEE, p. 12, 2018.

[3] Jyotishman Bora, Saine Dehingia, Abhijit Boruah* Anuraag Anuj Chetia, Dikhit Gogoi, "Real-time Assamese

Sign Language Recognition using MediaPipe and Deep Learning" ScienceDirect, p. 10, 2023

[4] Prakhyath Rai, Ananya Alva, Gautami K. Mahale, Jagruthi S. Shetty, Manjushree A. N., "Gesture Recognition System", IJCSMC, p. 12, 2018

[5] Ravikaran J, Kavi Mahes, Suhash Mahishi, Dheeraj R, Sudheender S, Nitin V Pujari, "Finger Detection for Sign Language Recognition,IMECS, p. 5, 2009

[6] Romala Sri Lakshmi Murali, L.D.Ramayya, V. Anil Santosh, " Sign Language Recognition System Using Convolutional Neural Network And Computer Vision", IJEIAT, p. 6, 2022

[7] Hee-Deok Yang, "Sign Language Recognition with the Kinect Sensor Based on Conditional Random Fields", MDPI, p. 13, 2014

[8] Aman Pathak, Avinash Kumar, Priyam, Priyanshu Gupta, Gunjan Chugh, "Real Time Sign Language Detection", IJMTST p. 7, 2022

[9] Sanket Bankar, Tushar Kadam, Vedant Korhale, Mrs. A. A. Kulkarni, "Real Time Sign Language Recognition using Deep Learning", IRJET p. 5, 2022