GesSpy: ML Driven Real Time Sign Language Detection

Nusrat Ansari
Department of Computer Engineering
Vivekanand Education Society's
Institute of Technology
Mumbai, India
nusrat.ansari@ves.ac.in

Siddhi Awari
Department of Computer Engineering
Vivekanand Education Society's
Institute of Technology
Mumbai, India
2022.siddhi.awari@ves.ac.in

Sri Haritha Movva
Department of Computer Engineering
Vivekanand Education Society's
Institute of Technology
Mumbai, India
2022.sriharitha.movva@ves.ac.in

Ananya Parthasarathy
Department of Computer Engineering
Vivekanand Education Society's
Institute of Technology
Mumbai, India
2022.ananya.parthasarathy@ves.ac.in

Srushti Poriwade
Department of Computer Engineering
Vivekanand Education Society's
Institute of Technology
Mumbai, India
2022.srushti.poriwade@ves.ac.in

Abstract—Various sign language translation technologies are used to ease the work of the sign language interpreters. Laying a stress on this, a real-time sign language detection system has been designed which employs OpenCV, MediaPipe, TensorFlow, Scikit-Learn and CNN to construct a system that accurately detects signs in real-time and further obtains the textual representation of the hand gestures. This system efficiently interprets the hand gestures used by the sign language users and converts it into its relevant text translation. This system is integrated in a web application for more interactive communication. This is done by integrating the Flask web API into android studio. System exhibits significant accuracy in detecting the gestures appropriately. Envisioning the goal to empower people with disabilities by enabling an inclusive environment. The proposed system is an efficient tool to increase awareness about ISL and enable non-ambiguous interactions between the deaf and hearing community.

Keywords–- Indian Sign Language, Real-time, OpenCV, Tensorflow, MediaPipe, Matplotlib

I. INTRODUCTION

Sign language is a fully developed language which has its grammar, syntax and vocabulary. Sign language is not just a communication tool but it is integral to deaf culture. They are highly trained and experienced professionals who play the role of paraphraser between them. They translate the spoken language to sign language and vice versa. There are almost more than 300 sign languages in the world which are used by around 72 million deaf or hard-of-hearing communities. Recognizing the diversity of languages, fosters respect for different forms of expression and promotes effective communication across diverse communities. Many organizations have also contributed to the study and development of ISL. Indian Sign Language gained legal status by the passing of the "Rights of persons with disabilities Act, 2016" in India. This Act recognizes ISL as the language of the deaf community. ISL is commonly used in schools and educational institutions. Students often communicate in ISL as a part of their education. Despite progress, challenges exist, including the need for wider awareness.

The need to make the life of deaf and mute individuals a bit easier and to boost the awareness of sign language. Therefore, in order to facilitate communication between normal and disabled individuals, a sign language recognition system is utilized to translate sign language [1]. This system contributes to the goal of reducing inequalities through advancing inclusive technology and fostering effective communication through real-time sign language recognition. Emphasizing on the conversion of signs used by the users in their day to day life to communicate in real time.

This system provides an innovative strategy to recognize the signs using landmark-based features and Tensorflow Object Detection API. Discussion begins with the problems faced by traditional methods. Followed by methodology, dataset, feature extraction, model selection, evaluation and results are then outlined.

II. LITERATURE REVIEW

This section explains the literature review in detail. A system proposed by Soma Shrenika and Myneni Madhu Bala [2] uses image processing techniques. Image captured is converted to grayscale, followed by removal of noise and edges are tracked using canny edge detection (CED). The comparison of image with the dataset using a sum of absolute difference method which produces the best match for the sign and displays the equivalent label. But this system highly relies on the quality of the captured image.

The authors of [3] and [4] paper implement a system using tensorflow to recognize sign language and use python and openCV for data acquisition. Grayscale image dataset is imported which is split into separate training and testing subsets and relevant labels are assigned to the images. The input image is processed through CNN (Convolution neural network) in order to produce a feature map and the model is trained using keras, a deep learning library for python.

The system mentioned in [5] uses a geometric approach for sign detection. Pre-processed images, that is a dataset containing MINIST (Modified National Institute of Standards and Technology database) and ASL, after noise reduction, passed through gamma correction to modify the brightness of the light. After converting the image to binary, morphological operators are applied, and then directed by directional region detection and localization of extreme points of the hand detected. Landmark-based features are identified and classified using CNN.

Another approach for sign detection is the sensor-based technique [6] and [7] which utilizes sensory glove devices to capture position and movement of the hand. The use of flex sensors is to adjust its resistance in response to the bending of fingers at a certain angle. As the gestures contain movement of the wrist as well, it includes an accelerometer sensor's features to recognize the rotation. But this technique is bulky as it requires a broad range of sensors to be worn and sensors being highly sensitive increases the errors.

In the reference paper [8], the real-time feed is sent to the Django server for preprocessing. The python script then detects the hand gestures and the edges are detected using canny edge detection CED algorithm. This image is sent to the trained CNN model for prediction and the equivalent word is transformed to voice.

Three deep learning models for sign language recognition namely the Dynamic Motion Network (DMN), the Accumulative Motion Network (AMN), and the Sign Recognition Network (SRN) have been proposed in paper [9]. Key postures are extracted with the help of the DMN, sign motion is encoded into a single image by the AVM, and input is accepted by the SRN as fused characteristics of the DMN and AMN.

The paper [10] and [11] tackles the challenge of sign language recognition, particularly in modeling hand movements. It presents an effective method using Hidden Markov Models (HMMs) to derive hand movement information irrespective of the sign language. The approach created a sign language recognition system by combining data collection, feature extraction, subunit-derived HMMs, model training, and evaluation.

Paper [12] serves as a survey conducted on various methodologies and models used for sign-language translation, particularly focusing on American Sign Language (ASL). The objective is to create a real-time ASL translation product that combines static and dynamic hand positions with Computer Vision and ML models and recognizes the demand for contactless systems.

A system utilizing a neural network is proposed in paper [13] to interpret sign language without the need for an interpreter. The system uses the system camera to capture user gestures in real-time, detect hand movements, crop images, and apply the canny edge detector algorithm to scale and modify images. The model utilizes the Google Text-to-Speech API to translate the recognized gestures into words, which are subsequently converted into appropriate voice.

The recognition system mentioned in [14] and [15] combines a LSTM neural network with an advanced CNN for generation and detection. It has a PyQt GUI interface that lets users take photos using OpenCV.

III. METHODOLOGY

This section provides a detailed explanation of the suggested system methodology. Tensorflow and Scikit-Learn are both open-source machine libraries which are utilized in the training process of the machine learning algorithm for recognition of sign languages.

A. *Tensorflow*

OpenCV and TensorFlow's object detection API are utilized in this model. TensorFlow provides deep learning capabilities, while OpenCV offers an extensive toolkit for image and video processing. The implementation is entirely in Python. A pre-trained TensorFlow object detection model is employed. The model is loaded, live images are captured from the camera, and a straightforward structure is used to perform object detection. The cv2.VideoCapture class is used to access the video stream. The following are the subdivisions of the TensorFlow-based model:

1) *Gather and annotate pictures:-* Initialize the camera using OpenCV by calling cv2.VideoCapture(0). The argument 0 specifies the default camera. This step is crucial for capturing live video feed from the camera. Use nested loops to capture a series of images for each class label. The outer loop iterates through a list of class labels, while the inner loop captures a specified number of images for each label. This method ensures a balanced dataset for each class. Use cv2.imshow to display the captured image in a window. Each image is saved with a unique filename using the uuid.uuid4() function to ensure there are no duplicates. This uniqueness is critical for managing and organizing the dataset. Use the LabelImg tool for annotating the captured images. LabelImg is a graphical image annotation tool commonly used in machine learning projects to create datasets for object detection. It allows manual drawing of bounding boxes around objects in images and labeling them accordingly.

2) Train the model:- The collected photos are manually divided into training and testing folders. This ensures that the model can be evaluated on unseen data during training. The pipeline configuration file specifies various parameters needed for training, such as paths to training and testing data, model architecture, and hyperparameters like learning rate and number of training steps. Use the model_main_tf2.py script from TensorFlow's Object Detection API to begin the training process. This script reads the pipeline configuration file, initializes the model, and starts the training loop. It makes use of the model_main_tf2.py script to set the pipeline configuration file, model directory, and number of training steps. Following training, the pipeline setup is loaded and the

TensorFlow object detection API is used to generate a detection model. This code records live video from a camera by using OpenCV. To detect objects, convert each frame to a TensorFlow tensor. To detect objects in the input image, call detect_fn. Analyze the detection results and see the image's bounding boxes and labels. Show the window with the annotation in iteration till the q key is pressed.

B. *Scikit-learn*

The training procedure for the Scikit-Learn Model is divided into four phases. First, images were collected using OpenCV, a computer vision library, which was used to collect, sort and save the pictures on disk. Several classes use the webcam to take pictures, which are then saved in the data directory. Second, a dataset for identifying hand landmarks in photos is created from scratch using the MediaPipe package. The coordinates of those landmarks are then extracted and normalized. Matplotlib is used for visualization, and Pickle is used for object serialization. Labels and the normalized coordinates are kept in a pickle file.

Thirdly, the RandomForestClassifier gets trained using pickle, which divides the data into sets for training and testing, analyzes the model on the test set, and stores the model. The last classifier is the inference classifier, which uses the pre-trained model's ability to recognize hand motions to recognize and display characters.

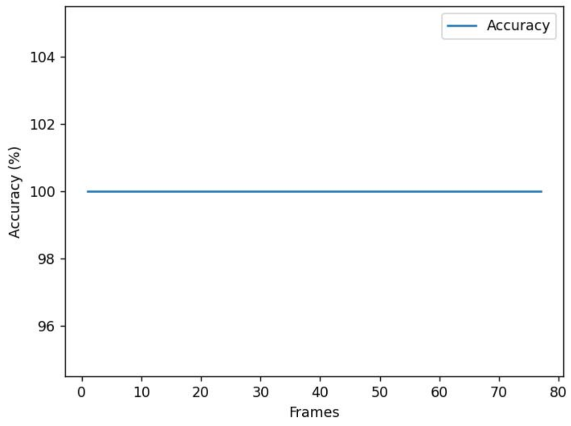

Fig.1: Accuracy for Scikit-Learn

C. *CNN*

A large dataset (100 images for each label) is collected. The collected data is preprocessed, which includes resizing images, converting them to grayscale, normalization, etc. Images may come in various sizes and resolutions, which can affect the training process. Resizing images to a standard size (48X48 pixels) ensures uniformity and simplifies processing. Additionally, images are often converted to grayscale to reduce computational complexity and remove color variations that may not be relevant for sign language detection. Normalizing the pixel values of images helps in reducing the impact of illumination differences and improves convergence during training. This typically involves scaling pixel values to a range (e.g., [0, 1] or [-1, 1]).Normalizing means adjusting the pixel values of images so they fall in a similar range, typically between 0 and 1. This helps the CNN learn faster and perform better. This process ensures consistent data scaling, making training more stable and efficient.

By ensuring consistent data scaling, this procedure improves the stability and effectiveness of training. Edge detection algorithms, like Sobel or Canny filters, are used in preprocessing to draw attention to key structural elements in the pictures. These features are vital for recognizing the forms and contours of hand motions that are necessary for classifying sign language. A CNN architecture is trained with the gathered dataset to identify various sign language movements. It is intended to understand characteristics of sign language identification. A different test set is used to assess the model's performance in order to guarantee accurate classification. To improve the performance and resilience of the model, strategies like dropout, learning rate scheduling, cross-validation, and data augmentation are used. While data augmentation broadens the model's potential applications, cross-validation guarantees the model's strong generalization to new data. Dropout prevents overfitting by randomly deactivating a fraction of neurons during training, and learning rate scheduling adjusts the learning rate to fine-tune the convergence process. These strategies collectively contribute to a more accurate and reliable sign language detection system, which can then be deployed in a web application for real-time use.

D. *GUI*

A prototype web application called "GesSpy" is developed using "Figma". Various frame arrangements were made. After that, the frames were joined in a way that followed the flow of the application, and the Figma prototype function was utilized to figure out how the program functioned.

The trained gesture detection CNN model extracts the input features to make accurate predictions on the basis of the features. Using the flask framework the keras (.hdf5) format was deployed to make it accessible for the users. The flask web application serves as the interface between the model and the users. The inputs are processed according to the requirements of the model like resizing, normalizing and other transformations. The live feed of the camera is passed into the model after processing. The model passes the label for that gesture which is displayed on the user screen. Thus availing the real time facility of the model, the output label changes as per the gesture detected by the model.

"GesSpy" starts its user experience with a simple login/signup page. By offering a customized experience, this vital entry point guarantees that users can safely access the application's capabilities. By integrating Firebase Authentication, the application was able to securely store and manage user credentials, including email addresses and passwords, in Firebase's backend infrastructure. By incorporating Firebase Authentication into the application, developers could focus on building the core functionality of the application without having to worry about the intricacies of user authentication.

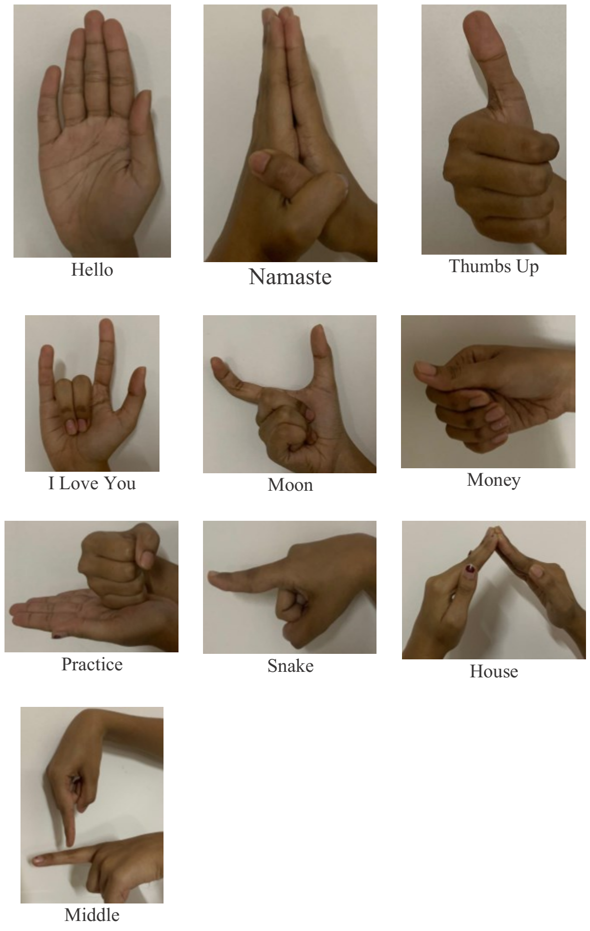

Fig.2:Sign Chart

E . Algorithm

1. User Authentication: The user is prompted to either sign up or log in to the application. This step ensures that the system knows who is using it and can personalize the experience if needed.
2. Gesture Conversion Choice: After authentication, the user is presented with an option to interact with the application. They are given the choice to either convert hand gestures into text or hand gestures from text.
3. Camera Activation: Once the user selects the gesture conversion option, the camera is activated. This allows the application to access real-time camera input from the user's device.
4. Real-time Sign Detection: The camera captures the user's hand gestures in real time and extracts them by eliminating the background as much as possible for accurate detection.
5. Comparison with Dataset: The captured sign is then compared with a pre-existing dataset of known sign language gestures. This dataset contains information about the visual features of various signs and their corresponding textual translations.
6. Translation and Display: If a match is found in the dataset, the corresponding translation in text is retrieved. Additionally, an option for speech synthesis can be provided, where the translation is also spoken out loud through text-to-speech technology.
7. Accuracy and Recognition: The system relies on accurate recognition to ensure that the correct sign is being captured and translated.
8. Display to User: The translated text and optional speech output are displayed on the user's screen in real time as well as it displays the accuracy of detection.

IV. RESULTS

The outcomes of this system are covered in this section. Landmarks that are generated systematically to ensure precision. The creation process involves identifying key points on each sign that are essential for accurate recognition. These key points typically correspond to critical features such as corners, edges, or distinctive patterns. Once identified, these points are annotated and used as reference markers in the system. The landmarks utilized for each sign are depicted in the pictures below [Fig.3, Fig.4, Fig.5]. The textboxes for the individual signs are displayed in the figures [Fig.6, Fig.7, Fig.8]. In general, the proposed system offers accurate interpretation of the outcomes. The figures [Fig.9, Fig.10, Fig.11] are the outcomes of the TensorFlow Model, which are trained using the pre-trained TensorFlow API.

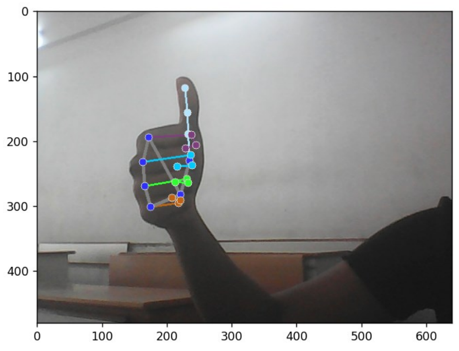

Fig.3: Landmarks for ThumbsUp

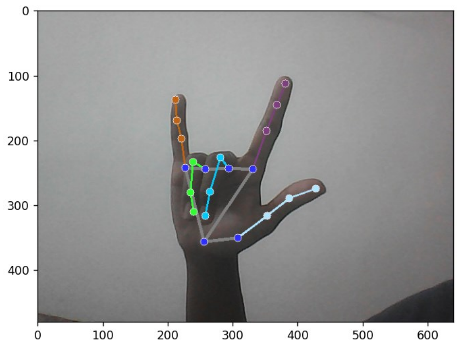

Fig.4: Landmarks for I Love You

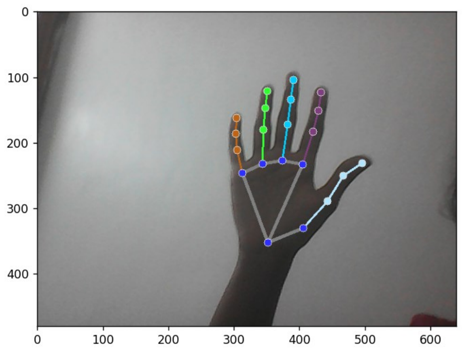

Fig.5: Landmarks for Hello

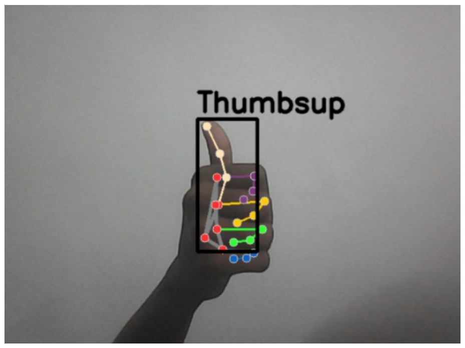

Fig.6: Label for ThumbsUp

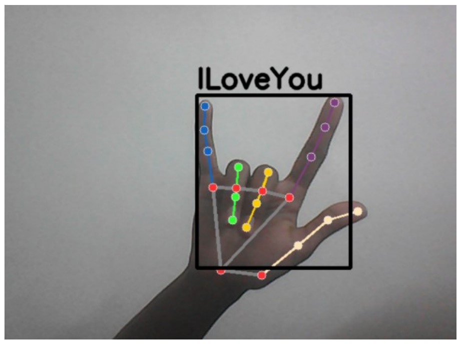

Fig.7: Label for I Love You

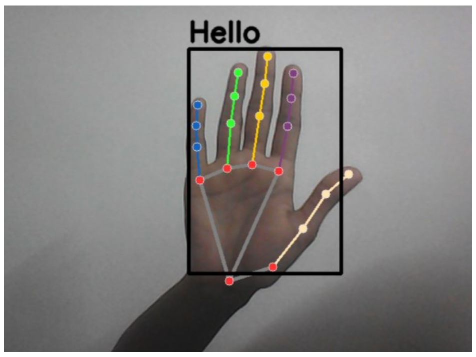

Fig.8: Label for Hello

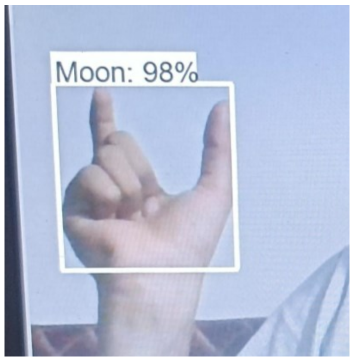

Fig.9: Label for Moon

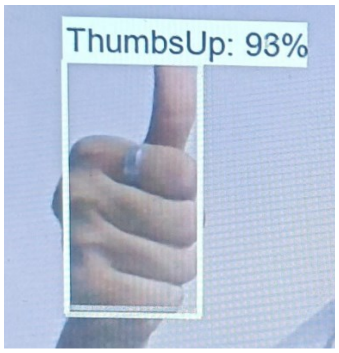

Fig.10: Label for ThumbsUp

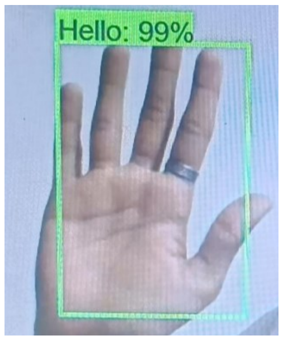

Fig.11: Label for Hello

V. CONCLUSION

Tensorflow is renowned for its adaptability and strength in applications involving deep learning. It is ideal for image identification and natural language processing because it offers a flexible foundation for creating and enhancing neural networks. The simpler machine learning library sci-kit, on the other hand, was created for conventional machine learning techniques. When handling large-scale, complex neural networks, Tensorflow is helpful, whereas Sci-kit is best suited for rapid implementation. The precision of Tensorflow is an essential measure of the model's effectiveness. The model accurately identifies the sign gesture by a bounding box around the sign gesture and the translated text is displayed on the top of the border box with the accuracy of the detection. The proposed system exhibits an accuracy range of 70-90%, contingent upon the interplay of multiple parameters and the system's implementation. As advancements continue, dynamic signs can be incorporated to increase the system's flexibility, reinforce its capability and make it robust for various applications.

In contrast, traditional models often rely on basic algorithms or specialized hardware, such as pattern matching, geometric techniques, or those that use sensory gloves. These traditional techniques can provide varying levels of accuracy, often less than deep learning models. The real-time processing capabilities of a TensorFlow-based system may not be available to traditional models, but may also be limited to static indicators and pre-processed images. Although TensorFlow requires moderate to large hardware resources, especially when using GPU acceleration, the hardware requirements of traditional techniques can vary depending on their methodology. Although TensorFlow is difficult to implement, its features are better than traditional models, which are simpler but less flexible and scalable. Large data sets can be processed with a TensorFlow-based system, which is also easy to use with web-based work environments. CNN models implemented with TensorFlow offer superior accuracy, real-time processing capabilities, and scalability compared to traditional techniques. Although they require moderate hardware resources, their advanced features make them a compelling choice for many applications where accuracy and real-time performance are critical.

REFERENCES

[1] M. R. Chilukala and V. Vadalia, "A Report on Translating Sign Language to English Language," 2022 International Conference on Electronics and Renewable Systems (ICEARS), Tuticorin, India, 2022, pp. 1849-1854, doi: 10.1109/ICEARS53579.2022.9751846.

[2] S. Shrenika and M. Madhu Bala, "Sign Language Recognition Using Template Matching Technique," 2020 International Conference on Computer Science, Engineering and Applications (ICCSEA), Gunupur, India, 2020, pp. 1-5, doi: 10.1109/ICCSEA49143.2020.9132899.

[3] S. Kurundkar, A. Joshi, A. Thaploo, S. Auti and A. Awalgaonkar, "Real-Time Sign Language Detection," 2023 2nd International Conference on Vision Towards Emerging Trends in Communication and Networking Technologies (ViTECoN), Vellore, India, 2023, pp. 1-4, doi: 10.1109/ViTECoN58111.2023.10157784.

[4] Mohit Titarmare, Gaurav Wankar, Gaurav Thapliyal, Yash Raut, Dr.Rahila Shah "Sign Language Detection" International Journal for Research in Applied Science andTechnology.

[5] H. Ansar et al., "Hand Gesture Recognition for Characters Understanding Using Convex Hull Landmarks and Geometric Features," in IEEE Access, vol. 11, pp. 82065-82078, 2023, doi: 10.1109/ACCESS.2023.3300712.

[6] Z. R. Saeed, Z. B. Zainol, B. B. Zaidan and A. H. Alamoodi, "A Systematic Review on Systems-Based Sensory Gloves for Sign Language Pattern Recognition: An Update From 2017 to 2022," in IEEE Access, vol. 10, pp. 123358-123377, 2022, doi: 10.1109/ACCESS.2022.3219430.

[7] Jong-Sung Kim, W. Jang, Z. Bien "A dynamic gesture recognition system for the Korean sign language (KSL)" IEEE Trans. Syst. Man Cybern. Part B

[8] K. Tiku, J. Maloo, A. Ramesh and I. R., "Real-time Conversion of Sign Language to Text and Speech," 2020 Second International Conference on Inventive Research in Computing Applications (ICIRCA), Coimbatore, India, 2020, pp. 346-351, doi: 10.1109/ICIRCA48905.2020.9182877.

[9] H. Luqman, "An Efficient Two-Stream Network for Isolated Sign Language Recognition Using Accumulative Video Motion," in IEEE Access, vol. 10, pp. 93785-93798, 2022, doi: 10.1109/ACCESS.2022.3204110.

[10] S. Tornay, M. Razavi and M. Magimai.-Doss, "Towards Multilingual Sign Language Recognition," ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Barcelona, Spain, 2020, pp. 6309-6313, doi: 10.1109/ICASSP40776.2020.9054631.

[11] Rung-Huei Liang, Ming Ouhyoung"A real-time continuous gesture recognition system for sign language" Proceedings Third IEEE International Conference on Automatic Face and Gesture Recognition

[12] D. Sau, S. Dhol, M. K and K. Jayavel, "A Review on Real-Time Sign Language Recognition," 2022 International Conference on Computer Communication and Informatics (ICCCI), Coimbatore, India, 2022, pp. 1-5, doi: 10.1109/ICCCI54379.2022.9740868.

[13] S. Patil, S. Gulave, V. Gawai, P. Gode and P. Mudme, "Conversion of Indian Sign Language to Speech by Using Deep Neural Network," 2022 6th International Conference On Computing, Communication, Control And Automation (ICCUBEA, Pune, India, 2022, pp. 1-6, doi: 10.1109/ICCUBEA54992.2022.10011043.

[14] Vishwa Hariharan Iyer, U. M. Prakash, Aashrut Vijay, P. Sathishkumar "Sign Language Detection using Action Recognition" 2022 2nd International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE).

[15] Wanbo Li, Hang Pu, Ruijuan Wang "Sign Language Recognition Based on Computer Vision" 2021 IEEE International Conference on Artificial Intelligence and Computer Applications (ICAICA).