Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 218 (2023) 1384–1393

Procedia Computer Science

www.elsevier.com/locate/procedia

International Conference on Machine Learning and Data Engineering

Real-time Assamese Sign Language Recognition using MediaPipe and Deep Learning

Jyotishman Bora, Saine Dehingia, Abhijit Boruah* Anuraag Anuj Chetia, Dikhit Gogoi

Department of CSE, Dibrugarh University Institute of Engineering and Technology, Dibrugarh 786004, India

Abstract

People lacking the sense of hearing and the ability to speak have undeniable communication problems in their life. People with hearing and speech problems communicate using sign language with themselves and others. Sign language is not essentially known to a more significant portion of the human population who uses spoken and written language for communication. Therefore, it is a necessity to develop technological tools for interpretation of sign language. Much research have been carried out to acknowledge sign language using technology for most global languages. But there are still scopes of development of tools and techniques for sign language development for local dialects. There are 22 modern Indian languages and more than 19000 languages that are spoken regionally as mother tongue. This work attempts to develop a technical approach for recognizing Assamese Sign Language, which is one of the 22 modern languages of India. Using machine learning techniques, this work tried to establish a system for identifying the hand gestures from Assamese Sign Language. A combination of two-dimensional and three-dimensional images of Assamese gestures has been used to prepare a dataset. The MediaPipe framework has been implemented to detect landmarks in the images. An Assamese Sign Language dataset of 2094 data points has been generated, including nine static gestures with vowels and consonants (অ, ই, ঈ, উ, এ, ও, ক, জ, ল) from the Assamese Sign Language. The dataset was used for training of a feed-forward neural network. The model yielded an accuracy of 99%. The results reveal that the method implemented in this work is effective for the recognition of the other alphabets and gestures in the Assamese Sign Language. This method could also be tried and tested for the recognition of signs and gestures for various other local languages of India.

Keywords: Sign Language Recognition; Assamese Sign Language; Gesture recognition; 3D Image Recognition; MediaPipe; Neural network;

* Corresponding author.
E-mail address: abhijit.btcs06@gmail.com

1877-0509
10.1016/j.procs.2023.01.117

1. Introduction

The Census of 2011 mentions that out of the population of disabled people of 2.68 crores (2.21% of the total population) in India, there are around 1027835 people, including 545179 male and 482656 female, who suffer from hearing and speech impairment. This group of people have developed their language to communicate among themselves, known to us as Sign language. Sign languages use visual human body movements and gestures to express one's thoughts. Sign languages vary from region to region. For example, there is American Sign Language in the United States of America, whereas Indian Sign Language in India. There is also a part in different sign languages where native language alphabets are expressed by fingerspelling hand signs. Assamese Sign Language is such a sign language where there are hand signs for different individual Assamese alphabets.

Over the past years, we can find many trials to develop sign language recognition (SLR) systems. Although much work has been done in different sign languages worldwide, there seems to be significantly less work done in regional dialects of India like the Assamese language. There are two types of SLR architecture on the basis of input data: glove-based [1], [2], [3] and vision-based [4], [5]. Glove-based SLR architecture involves usage of smart gloves to measure position, orientation, velocity etc. of hands using microcontrollers and sensors. Vision-based SLR techniques use cameras to detect hand gestures. Computer vision-based SLR systems are usually based on extracting features such as edge detection, skin colour segmentation, gesture detection, hand shape detection etc. But most of these solutions are too computing power consuming to run in real-time on low-end computation devices like mobile phones and hence are limited to platforms with high computing power. Moreover, the hit and miss of hand-tracking mechanisms is almost evident in all of these approaches.

This work contributes to a deep learning powered Assamese Sign Language Recognition system focused on recognizing a few fundamental letters of the Assamese Alphabet. The approach includes a methodology that involves implementation of a hand tracking solution provided by Google's open-source project, MediaPipe [6]. Along with it, a deep learning algorithm is implemented on this solution to produce a fast, inexpensive, lightweight, and easy-to-deploy system that can be used as the core of a complete sign language recognition system.

This work adopts the hand landmarks detection approach at the core of the complete model. Using MediaPipe's hand tracking model, 21 hand landmarks in each image containing hand signs from Assamese Sign Language have been detected. The landmarks are then collected as coordinate points, normalized, and saved in a .csv file as data-points. A feedforward neural network model than takes these data-points as input, and after training the model, real-time hand sign recognition with OpenCV has been implemented using the trained model. Fig. 1 gives an overview of the complete project. The prepared model for 9 Assamese alphabets is highly efficient, accurate, portable, and lightweight. This paper discusses the entire procedure of developing our model, along with the results and analysis of performance.

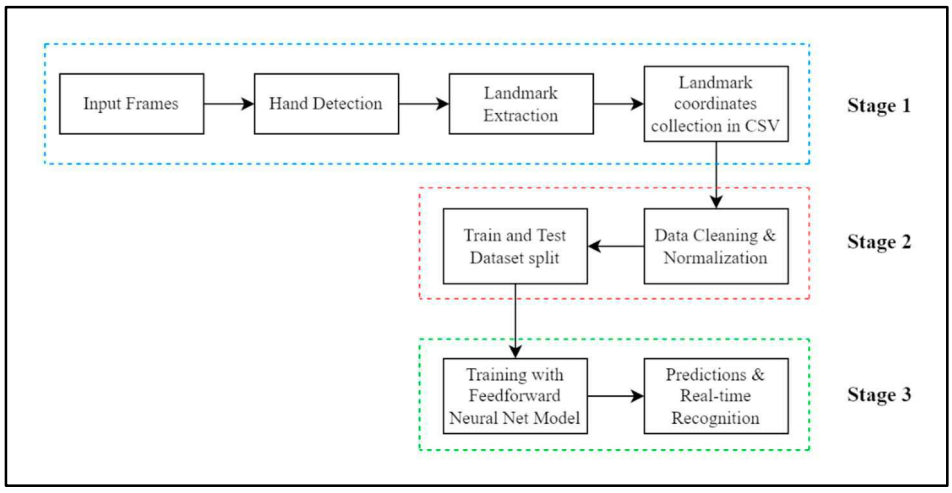


Fig. 1. Flowchart of the complete project

2. Literature Review

Das et al. [5] have researched on deep learning-based sign language recognition system with processed static images implemented on American Sign Language gestures. The dataset consisted of 24 labels of static gestures of alphabets from A to Z, excluding J. There were approximately 100 images per class in the dataset captured on an RGB sensor. The methodology included using of Inception V3 convolutional neural network model to train the model. After training and testing the model, the average accuracy rate of validation was over 90%, with the most outstanding validation accuracy being 98%. They concluded that the relatively new Inception V3 model could be an appropriate model for static sign language detection when provided with a dataset of properly cropped images.

Sahoo [7] used machine learning to work on Indian sign language recognition. This research focused on the static hand gestures in Indian sign language for the numeric values (0-9). A digital RGB sensor was used to capture the images of the signs to build the dataset. The dataset contained 5000 total images, with 500 images for each digit from 0 to 9. Two classifiers were used based on the supervised learning technique to train the model: Naïve Bayes and k-Nearest Neighbor. K-Nearest Neighbor technique slightly performed better than the Naïve Bayes classifier in this research as the average accuracy rates of k-NN and Naïve Bayes were 98.36% and 97.79%, respectively.

Ansari and Harit [4] researched classifying Indian sign language static gestures using images with 3d depth data. The images were captured using Microsoft Kinect, which enables 3d depth information capturing along with the 2D image. The dataset totaled 5041 images of static hand gestures and was labeled with 140 classes. K-means clustering was used to train the model. In the research, they were able to score a 90% accuracy rate for 13 signs and 100% accuracy for three signs making it up to 16 alphabets (A, B, D, E, F, G, H, K, P, R, T, U, W, X, Y, Z) recognition with an average accuracy rate of 90.68%.

Rekha et al. [8] worked on 23 static and three dynamic signs of the Indian Sign Language dataset. They used skin color segmentation to locate hands. Edge orientation and texture were used as features to train a multiclass SVM on which they achieved a success rate of about 86.3%. However, their approach was too slow to implement as a practical gesture recognition algorithm.

Bhuyan et al. [9] used a dataset of 8 gestures from Indian Sign Language consisting of 400 images. They used a skin colour based segmentation technique for hand detection, then used nearest neighbor classification, and finally achieved a recognition rate above 90%.

Pugeault & Bowden [10] worked on a real-time recognition system for recognition of the alphabets in American Sign Language. A dataset consisting of 24 classes with 48,000 3D depth images captured using a Kinect sensor was used. Gabor filters and multi-class random forests were used and highly accurate classification rates were achieved.

Keskin et al. [11] used an approach involving object recognition by parts to recognize signs indicating digits of American Sign Language. Their dataset had ten classes with a total of 30,000 3d depth images captured using a Kinect sensor, and they achieved about 99% accuracy rate.

Ren et al. [8] used a threshold-based technique for segmentation of hands from Kinect captured images. They had ten classes with 1000 images in total in their dataset. The accuracy rate achieved was about 93%.

Halder and Tayade [12] used the MediaPipe framework to get multi-hand landmarks and used Support Vector Machine (SVM) for Real-time detection of hand signs. The average accuracy achieved was about 99%.

3. Data Collection Procedure

3.1. Simulation Environment

KScan3D software was used for 3D image capturing with the Microsoft Kinect sensor. OpenCV library was used to capture 2D image using RGB webcam and visualization of 3D hand landmarks extracted by MediaPipe hand tracking model. Deep learning was implemented on the Jupyter Python notebook web application.

3.2. Image collection

Images of subjects posing with gestures of Assamese Sign Language were captured using a Microsoft Kinect sensor and an RGB webcam. The advantage of using a Microsoft Kinect sensor is that it collects depth data for individual

pixels of the image, and using that data 3D model of a person can be created. Further, the 3D depth image captured with the Microsoft Kinect sensor is used to produce multiple images of a person from different angles. This process also makes the model more robust and adaptable to varying angles of the hand signs after training. Along with the images produced from the Microsoft Kinect sensor, 2D RGB images of the signs were also captured with an RGB webcam. Sample of 3D and 2D images collected during the process are shown in Fig. 2(a) and Fig. 2(b) respectively. The ratio of the number of images from the Kinect sensor to the number of images from the webcam is approximately 1:1.

a.  b. 

Fig. 2. (a) 3D Image captured with Microsoft Kinect; (b) 2D Image captured with RGB webcam.

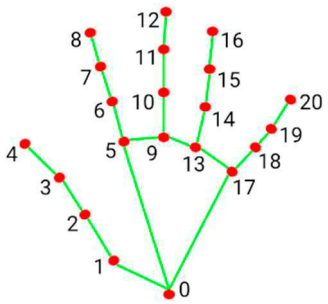
3.3. Detection of hand landmarks using MediaPipe

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.

MediaPipe toolkit altogether consists of the MediaPipe framework and MediaPipe solutions. The MediaPipe framework is developed using C++, Java, and Objective C programming, which consists of 3 key APIs –

- Calculator API
- Graph Construction API and
- Graph Execution API

MediaPipe solutions comprise 16 pre-trained TensorFlow and TensorFlow Lite models on top of the MediaPipe framework built for specific use cases. This work leveraged the MediaPipe solution to infer hand landmarks from images of hand signs. This hand-tracking model outputs 21 3D landmark points (as shown in Fig. 3) on a hand from a single frame. To achieve this, two dependent models are used simultaneously. First, a Palm Detection Model detects the palms of hands in the images since it is easier to detect rigid objects like palms and fists rather than a complete hand. Cropped palm images from this model are passed onto the next model, which is the Hand Landmark Model. This model precisely detects 21 3D hand landmark points in the detected hand region using regression. The complete model is trained on approximately 30,000 manually annotated real-world images. The model is very well-trained and robust and hence it can detect and map hand landmark points accurately, even on partially visible hands in most cases.



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

Fig. 3. 21 3D hand landmarks localized by MediaPipe hand tracking model

The hand tracking model from MediaPipe was passed through all the collected images of hand signs from Assamese Sign Language. The model output x, y, and z coordinates of the hand landmark points from the images, among which only x and y coordinates are necessary and sufficient for training the final model. Therefore, the z coordinates were eliminated, and the remaining coordinate points (x and y) of the hand landmarks for different hand signs are stored in a .csv file. Fig. 4 shows how the milestones are saved as coordinates. These coordinates were the data points using which the final dataset was generated and used to train the final model.

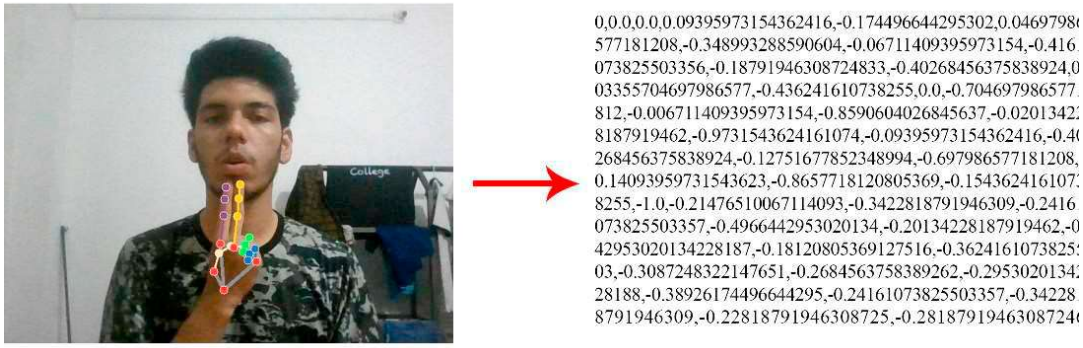


Fig. 4. Coordinates of the hand landmark points excluding z coordinates

3.4. Data cleaning and normalization

The MediaPipe hand landmarks model gives the coordinates indicating hand landmark points based on the placement of the pixels containing the landmark points in the image. Therefore, the coordinates of two images of the same hand sign with different arrangements in the frame can be vastly distant. This increases the difficulty of training the model. To solve this problem, the wrist's landmark point has been considered (donated by index 0 in the hand landmarks list) to have x and y coordinates as (0,0) and accordingly adjusted the coordinates of all the other landmark points relative to the wrist point. Then we further normalized the coordinate values to be in the range of [0,1] by dividing all the coordinates by the largest absolute coordinate value obtained by relative adjustment in the landmarks list. After normalization of the coordinates, they were collected in the .csv file. Fig. 5 demonstrates the coordinate normalization procedure.

After collecting coordinates in the .csv file, it was passed through a pandas' library function to detect null entries. Sometimes in blurry images, the model fails to detect hands, producing void entries. Cleaning of these void entries is necessary to train an unbiased model. Therefore, we detected the null entries and removed them from the file using their indices.

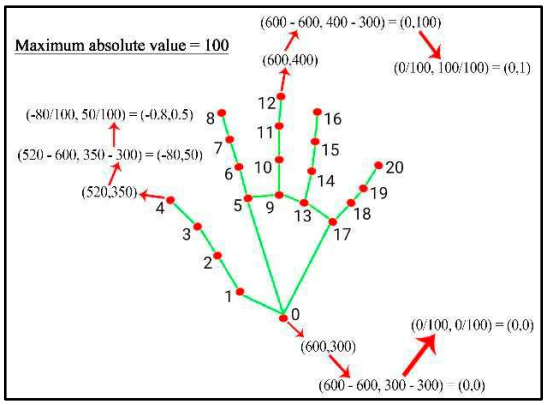


Fig. 5. Hand landmarks coordinates normalization example

3.5. Generated Data

Initially, nine fingerspelling hand signs from Assamese Sign Language were selected, referring to 9 different Assamese alphabets developed by Vaani® Foundations in 2021. Since we could not find any previous work regarding Sign Language Recognition in Assamese Sign Language, we have started from the most basic signs: static, posed by one hand, and distinct from each other. The 9 selected alphabets are- অ, ই, ঈ, উ, এ, ও, ক, জ and ল.

The total data points (coordinate points) in the dataset was 2094, with 200-300 data points for each class. 75% of the data points were randomly selected and utilized as a training set, and the remaining data points were considered as the test dataset. The training process of the recognition model was initiated from 200 data points in each class, and subsequently, more data was fed into the model by experimenting with different data points added to different classes to maximize the model's efficiency and accuracy. Fig. 6 shows the dataset's exact number of data points for each category, along with sample pictures of different signs.

Fig. 6. Sample size and gesture for each category of the Assamese alphabet

4. Training the Model with Custom Feedforward Neural Network

After creating the dataset, the next task is the training of a feedforward neural network with the data. The neural network included one input layer and one output layer along with four hidden layers, including two dense layers and two dropout layers. The dense layers perform matrix-vector multiplication in the background, and the dropout layers

reduce the overfitting of the model by modifying outgoing edges of hidden layer neurons randomly and resetting it to 0 at each iteration during the training process. A ReLU activation function was used in the hidden layers, and a Softmax activation function was used in the output layer. Table 1 presents the configuration of the sequential model implemented in this work.

After construction, the model was compiled using an Adam optimizer which is computationally efficient and appropriate for model training with large data/parameters. The loss function was set to Sparse Categorical Crossentropy, which gives loss between actual and predicted labels. The accuracy metric was used for the evaluation of the model, which depicts how often the prediction is equal to the actual label. An early stopping function was used in training the model, which stops the training if there is not much variation in the calculated loss and accuracy in 20 consecutive training steps. Initially, the epoch value of the training was set to 1000, but the model's training was completed in 341 epochs and stopped using the early stopping function. Training accuracy obtained was 99.40%, and the calculated loss was 0.1090.

Table 1: Configuration of the used feedforward neural network

Model: "sequential"		
Layer (type)	**Output Shape**	**No of Parameters**
dropout (Dropout)	(None, 42)	0
dense (Dense)	(None, 20)	860
dropout_1 (Dropout)	(None, 20)	0
dense_1 (Dense)	(None, 10)	210
dense_2 (Dense)	(None, 10)	110
Total parameters: 1,180		
Trainable parameters: 1,180		
Non- trainable parameters: 0		

5. Results and Analysis

5.1. Quantitative Analysis of the Model

The classification_report and confusion_matrix libraries from python's scikit-learn toolkit were 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. Equation (1) shows the mathematical formulation of the accuracy matrix.

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. Equation (2) depicts the mathematical formula of the precision matrix.

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). Equation (3) is the mathematical formulation of the recall matrix.

F1 score provides a cumulative measure by combining recall and precision. Equation (4) gives the mathematical formulation of this matrix.

$$A\ (Accuracy) = \frac{TP+TN}{TP+TN+FP+FN} \tag{1}$$

$$P\ (Precision) = \frac{TP}{TP+FP} \tag{2}$$

$$R\ (Recall) = \frac{TP}{TP+F} \tag{3}$$

$$F\ (F1\ score) = \frac{2*P*R}{P+R} \tag{4}$$

In the equations (1), (2), (3), and (4), TP, TN, FP, and FN represent True Positive, True Negative, False Positive, and False Negative, respectively.

Table 2 shows the classification report of the implemented Feed-Forward model in detail. The average accuracy achieved by the model is 99%, as shown in the report.

Table 2. Classification report of the Feedforward model

Class label	Precision	Recall	F1 score	Support
1	1.00	1.00	1.00	57
2	1.00	1.00	1.00	55
3	0.96	1.00	0.98	26
4	0.99	1.00	0.99	72
5	1.00	0.99	0.99	70
6	1.00	1.00	1.00	50
7	0.98	0.98	0.98	48
8	1.00	0.99	0.99	68
9	1.00	1.00	1.00	53
accuracy			0.99	499
macro avg	0.99	0.99	0.99	499
weighted avg	0.99	0.99	0.99	499

The *confusion_matrix* library is used for the analysis of the real-time performance of the model. It provides a confusion matrix measuring the number of accurately predicted labels. It also enables visualization of deviation of prediction in case of false predictions by the model. Fig. 7 shows the confusion matrix of the model.

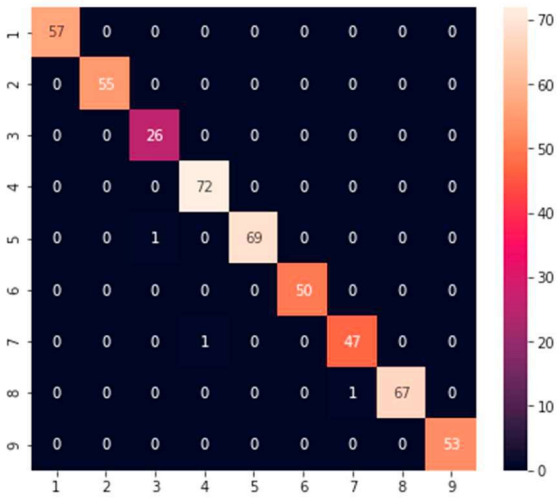


Fig. 7. Confusion matrix of the used feedforward neural network

5.2. Real-time recognition

Figure 8 presents the snapshots from real-time recognition system implemented with OpenCV open-source library using the model. The model is accurately able to recognize live gestures from the Assamese sign language both in 2D (Fig. 8 (a)) as well as 3D (Fig. 8 (b)) environments.

a.  b. 

Fig. 8. (a) Real-time sign recognition on webcam video stream; (b) Real-time sign recognition on 3D images from Kinect sensor.

5.3. Comparative Analysis

The comparative analysis of the results achieved in this work with the works in the literature is presented in Table 3 and Table 4. Table 3 presents the analysis of the works done with respect to the 2D image and vision based approach whereas Table 4 presents the analysis for 3D vision based approaches.

Table 3. Comparative analysis of this work and other 2D image based approaches

Sign Language	Authors	Approach	Accuracy	Year of Development
American Sign Language	Das et al. [5]	CNN based Inception V3 model	90%	2018
Indian Sign language	Rekha et al. [8]	Skin color segmentation with SVM	86.3%	2011
Indian Sign language	Sahoo [7]	k-NN and Naïve Bayes classifier	98%	2021
American, Indian, Italian and Turkey Sign Language	Halder and Tayade [12]	MediaPipe with SVM	99%	2021
Assamese Sign Language (this work)	This approach	MediaPipe with Feedforward neural network	99%	2022

Table 4. Comparative analysis of this work and other 3D image based approaches

Sign Language	Authors	Approach	Accuracy	Year of Development
Indian Sign language	Ansari and Harit [4]	k-means clustering	90.68%	2016
American Sign language	Keskin et al. [11]	Object recognition by parts	98%	2013
American Sign Language	Pugeault and Bowden [10]	MediaPipe with SVM	75%	2011
Indian Sign Language	Kumar et al. [13]	Spatio-temporal graph kernels	98.75%	2022
Assamese Sign Language (this work)	This approach	MediaPipe with Feedforward neural network	99%	2022

6. Conclusion

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 MidiaPipe. 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. The classification results claim that compared to other models in the literature, which require high computation power and longer training time, this approach of sign language recognition using the MediaPipe hand tracking solution is more effective and faster for classifying complex hand signs and gestures including alphabets. Moreover, the implementation of MediaPipe also ensures accurate tracking of the hand movements with different motions in the finger phalanges and finger joint deviations. Also, due to its lightweight characteristic, the model becomes more robust and can be implemented over various computing devices with different computing power without losing speed and accuracy. This work can be further extended to include recognition of more signs from the Assamese Sign Language, including dynamic gestures involved in daily life communications. Also, various deep learning techniques can be tested after implementing MediaPipe's hand tracking solution to increase the accuracy and efficiency of the model.

References

[1] A. Z. Shukor, M. F. Miskon, M. H. Jamaluddin, F. bin Ali, M. F. Asyraf, M. B. bin Bahar and others. (2015) "A new data glove approach for Malaysian sign language detection," *Procedia Computer Science,* vol. 76, p. 60–67.

[2] M. Mohandes, M. Deriche and J. Liu. (2014) "Image-based and sensor-based approaches to Arabic sign language recognition," *IEEE transactions on human-machine systems,* vol. 44, p. 551–557.

[3] N. M. Kakoty and M. D. Sharma. (2018) "Recognition of sign language alphabets and numbers based on hand kinematics using a data glove," *Procedia Computer Science,* vol. 133, p. 55–62.

[4] Z. A. Ansari and G. Harit. (2016) "Nearest neighbour classification of Indian sign language gestures using kinect camera," *Sadhana,* vol. 41, p. 161–182.

[5] A. Das, S. Gawde, K. Suratwala and D. Kalbande. (2018) "Sign language recognition using deep learning on custom processed static gesture images," in *International Conference on Smart City and Emerging Technology (ICSCET)*.

[6] C. Lugaresi, J. Tang, H. Nash, C. McClanahan, E. Uboweja, M. Hays, F. Zhang, C.-L. Chang, M. G. Yong, J. Lee and others. (2019) "Mediapipe: A framework for building perception pipelines," *arXiv preprint arXiv:1906.08172*.

[7] A. K. Sahoo. (2021) "Indian sign language recognition using machine learning techniques," in *Macromolecular Symposia*.

[8] J. Rekha, J. Bhattacharya and S. Majumder. (2011) "Shape, texture and local movement hand gesture features for indian sign language recognition," in *3rd international conference on trendz in information sciences & computing (TISC2011)*.

[9] M. K. Bhuyan, M. K. Kar and D. R. Neog. (2011) "Hand pose identification from monocular image for sign language recognition," in *2011 IEEE International Conference on Signal and Image Processing Applications (ICSIPA)*.

[10] N. Pugeault and R. Bowden. (2011) "Spelling it out: Real-time ASL fingerspelling recognition," in *2011 IEEE International conference on computer vision workshops (ICCV workshops)*.

[11] C. Keskin, F. Kıraç, Y. E. Kara and L. Akarun. (2013) "Real time hand pose estimation using depth sensors," in *Consumer depth cameras for computer vision*, Springer, p. 119–137.

[12] A. Halder and A. Tayad. (2021) "Real-time vernacular sign language recognition using mediapipe and machine learning," *Journal homepage: www. ijrpr. com ISSN,* vol. 2582, p. 7421.

[13] D. A. Kumar, A. S. C. S. Sastry, P. V. V. Kishore and E. K. Kumar. (2022) "3D sign language recognition using spatio temporal graph kernels," *Journal of King Saud University-Computer and Information Sciences*.

[14] Z. Ren, J. Yuan and Z. Zhang. (2011) "Robust hand gesture recognition based on finger-earth mover's distance with a commodity depth camera," in *Proceedings of the 19th ACM international conference on Multimedia*.