Towards Adaptive Real-Time Hand Sign Language Interfaces Addressing User Independence, Pose and Occlusion Variability with Machine Learning

Leena Chandrashekar
Dept of ECE
RNS Institute of Technology
Bangalore, India

Sanjay S.B.
Dept of ECE
RNS Institute of Technology
Bangalore, India

Raghavendra M. Hegde
Dept of ECE
RNS Institute of Technology
Bangalore, India

Samarth Shinnur
Dept of ECE
RNS Institute of Technology
Bangalore, India

Samrudh B.R.
Dept of ECE
RNS Institute of Technology,
Bangalore, India

ABSTRACT

Human-Computer Interaction (HCI) has emerged as a critical component of navigating and connecting with the digital world as technology has advanced. Hand gesture recognition has received substantial interest as a natural and intuitive communication interface. This paper describes the design and implementation of a real-time hand gesture detection system for supporting HCI, with a special focus on assisting the hearing and speech challenged. The proposed system captures hand gestures using a real-time video camera and creates a bespoke dataset that is robust by accounting for user, posture, and occlusion variability. A Convolutional Neural Network (CNN) is used to extract features, with 21 important features identified for each hand gesture. These features are then classified using a Random Forest method, which achieves an overall accuracy of 94.58% over several instances. Recognized gestures are translated into text and speech, allowing for efficient and convenient communication. The method allows you to combine various gestures to make whole sentences, which are often used in regular interactions. Performance assessment under different lighting circumstances reveals a PSNR of 3 to 4.27 dB, suggesting robustness to illumination fluctuations. A graphical user interface (GUI) with a feedback system allows for seamless two-way interaction, which improves usability and accessibility.

Keywords

Human Computer Interaction, Sign Language, Hand Gesture Recognition, User and Pose Independence, Occlusion Variability

1. INTRODUCTION

In today's world, the interaction between humans and computers has become indispensable due to continuous technological advancements. This interaction, referred to as Human Computer Interaction (HCI) encompasses various methods facilitating communication with and control of computers. From utilizing a mouse to tapping on a touchscreen, HCI plays a vital role in navigating the digital realm. Hand gesture recognition has emerged as a focal point of interest among researchers within the broader scope of HCI. It provides a natural and intuitive means of interacting with computers, free from the complexities associated with traditional input devices. Unlike devices such as joysticks or remote controls, hand gestures feel instinctive and user-friendly [1].

Among the diverse aspects of HCI, hand gesture recognition has emerged as a prominent area of interest for researchers. It offers a natural and intuitive means of interacting with computers, devoid of the complexities associated with traditional input devices. Unlike requiring training for devices like joysticks or remote controls, using hand gestures feels instinctive and user-friendly. Its applications span across numerous domains, from controlling home appliances to guiding robots and aiding in medical procedures [2]. However, despite its potential benefits, hand gesture recognition presents challenges, particularly in ensuring accurate recognition across varying environmental conditions. In critical fields such as healthcare, where precision is paramount, errors in gesture recognition can have serious consequences. Researchers are actively striving to develop robust systems that can reliably interpret gestures under diverse circumstances, but achieving this remains a formidable task. Leveraging advanced techniques in image processing and machine learning, these systems interpret sign language gestures, enabling participation across education, healthcare, and workplaces, fostering inclusivity and understanding. Furthermore, these systems reduce reliance on costly human interpreters, offering a scalable and cost-effective solution for interpretation services. By ensuring wider accessibility, they contribute to a more inclusive society, minimizing communication barriers and enhancing participation for deaf individuals. Ultimately, the goal is to enhance communication, accessibility, and inclusion, empowering deaf individuals to lead fulfilling lives and engage fully in societal activities.

The process of hand gesture recognition involves following stages - gesture capture, hand position identification, feature extraction, and gesture classification. Use of technologies like cameras and sensors, these stages enable the detection and analysis of both simple and complex gestures. In our paper, we aim to advance the field of hand gesture recognition by proposing innovative solutions to address existing challenges. Additionally, our efforts aim to streamline communication between humans and computers, facilitating smoother interactions. Additionally, we seek to contribute to a future where human-computer interaction is more intuitive and efficient. By bridging the gap between humans and machines, we define the below objectives create a more interconnected and accessible digital world –

- Create a comprehensive dataset of hand signs by converting video sequences into frames.
- Identify a suitable feature extraction technique to extract the significant key points from the hand images that represent the hand sign effectively.
- Identify and train the machine learning model with the goal of optimizing the model's accuracy.
- Develop a trained model for text-to-speech application that can interpret written text and generate corresponding spoken output.

2. RELATED WORK

Hand Gesture Recognition (HGR) systems have been to make human-computer interaction more natural, efficient, and real, especially for persons who only use hand gestures to talk. Even though computer vision has come a long way, it is still quite hard to automatically and accurately recognize hand motions. Mohammed Alonazi and others looked at changes in computer vision and sensor technology and gave a detailed look at HGR methods and data modalities from 2014 to 2024. They expressed the necessity for examination utilizing diverse modalities, including RGB, Skeleton, Depth, Audio, EMG, EEG, and Multimodal approaches. They looked at more than 200 research from reliable databases [1] that focused on collecting data, establishing data, and showing gestures. Miah et al. put out a multi-branch attention-based graph and a universal deep-learning model to address generalization issues through the detection of hand movements by extracting all potential skeleton-based characteristics [2]. They proposed a general neural network channel and two graph-based neural network channels within a multi-branch architecture. The temporal-spatial, spatial-temporal, and general characteristics are put together and delivered to the fully connected layer to make the final feature vector. To keep track of the node's order and minimize the system's computing cost, they added position embedding and mask operation to both the spatial and temporal attention modules. The MSRA, DHG, and SHREC'17 benchmark datasets were used to test this model, and its accuracy was 94.12%, 92%, and 97.01%, respectively.

Christine Dewi et al. essentially investigated CNN-based object identification methods utilizing the Yolov7 and Yolov7x models with 100 and 200 epochs on the Oxford Hand Dataset. Performance metrics include GFLOPS, mAP, and detection time. This study found that Yolov7x with 200 training epochs is the most reliable method. In training, it had 84.7% precision, 79.9% recall, and 86.1% mAP. Moreover, Yolov7x achieved the highest average mAP score of 86.3% during testing. While performance improves with epoch, processing time also increases dramatically. They recommended federated learning and hand detection to improve HGR systems [3]. Jungpil Shin employed the Media pipe method to identify American sign characters by looking at webcam photos of hand joints. The calculated joint coordinates produced two classification features: vector-3D axis angles and distances between joint points. SVM and LGBM classifiers categorized characters. The ASL Alphabet, Massey, and Finger spelling A character files were used to recognize each character. The Massey dataset gave a score of 99.39%, the ASL Alphabet gave a score of 87.60%, and the Finger Spelling A gave a score of 98.45%. The automatic American Sign Language identification architecture has done better than earlier experiments, is cheap to run, and doesn't need special sensors or equipment. This technique can also be employed for aerial writing and sign language recognition [4]. Real-world human–computer interaction hand gesture recognition using augmented YOLOv5 overcomes latency and low accuracy with complex backdrops. R Chen et al. suggested replacing the CSP1_x module in the YOLOv5 backbone network with an efficient layer aggregation network to improve gradient pathways. This improves network expression, learning, and recognition. The CBAM attention mechanism filters channel and spatial gesture features. This makes the network less susceptible to complex backgrounds and gesture images. Detailed backdrop gesture datasets EgoHands and TinyHGR were used. Using 640 × 640 input photos, mAP0.5:0.95 achieved 75.6% and 66.8% identification accuracy and 64 FPS recognition speed. The suggested method is more accurate and resilient than YOLOv5l, YOLOv7, and other algorithms, enabling fast and accurate movement recognition against complex backdrops [5]. Shashidhar et al. examined Indian Sign Language (ISL) for 24 English alphabets excluding J and Z. Recognition of 4972 static hand signs is achieved. A deep learning-based application uses the "Google text to speech" API to translate Indian Sign Language into text, allowing signers and non-signers to communicate. They used Kaggle's public dataset. The customized convolutional neural network solution was 99% accurate [6].

In the past, it was only possible to manually add unique hand gestures when the application was very limited. Jeong-Seop Han and others created a versatile and efficient graphical user interface that lets users specify their own hand movements. Their method customizes hand movements by creating a camera-based model that recognizes hand gestures depending on user data. They used a Multilayer Perceptron architecture based on contrastive learning to cut down on the amount of data and training time needed compared to older recognition models that need huge training datasets. The experimental findings indicate that the recognition model converges rapidly and precisely [7]. A user study is performed with initial user feedback on the implemented system.

The recommended procedures in the pertinent articles have a variety of limitations. Apart from their weak explainability and generality, none of the papers have offered any real-time analysis. Furthermore, the majority of the solutions merely address ASL detection; they don't offer real-time feedback. The system's accuracy can be increased with real-time feedback. Furthermore, there is no consideration paid to user, position, or occlusion fluctuation. Furthermore, the focus on generating sentences by integrating distinct signs has not yet been examined. The majority of the gestures are static and obtained from publically accessible web sources; real-time dynamic continuous gestures and their generalization to diverse settings or users are not covered.

3. DATASET AND PREPROCESSING

3.1 Dataset Collection

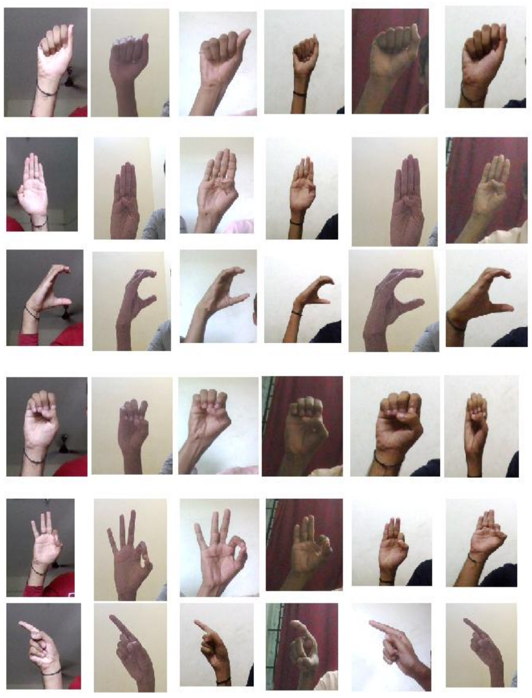


Figure 1 Hand Signs for A, B, C, D, E, F and Z obtained from 6 different individuals

There are Indian, Korean, and Indonesian sign languages, but American Sign Language is the most used and standardized. There are more e-resources for ASL than for other sign languages. So, we make an HGI system that uses ASL hand signs. Setting up a directory structure to store photos is the first step in collecting a dataset. This methodical way of doing things made sure that everything was stored in an organized way and could be found easily later on in the project. The system smoothly switched to real-time webcam image recording when data gathering for a hand sign lesson began. There are 26 classes in the dataset that stand for the letters of the ASL alphabet. There are 2,600 photos in all, with 100 pictures in each class. The JPG photos are 640x340 pixels and have 96 dots per inch and 24 bits of bit depth. We took pictures of four people and hand signs in different lighting situations to make sure the dataset was varied. We also looked at Indian skin tones to make sure our dataset had a wide spectrum of differences. Adding more picture data gives us 5200 photos to help the model work better and generalize. This technique includes flipping, rotating, and changing the size of photos. Figure 1 shows different users making hand signs for A, B, C, D, E, F, and Z. This strategy got the user to be flexible, occluded, and in a good posture. Different people encode ASL signals in different ways. A live video camera records these so that they can capture moving and continuous hand actions in diverse lighting conditions.

3.2 Feature Extraction

The Media Pipe hand identification and feature extraction approach is better than CNN, PCA, and machine learning [15–17]. The input image is analyzed to find hand landmarks. Media Pipe's hand tracking model leverages SSD to find single photos. The SSD architecture is a powerful object detection algorithm that looks at the input image at different scales to find hand areas based on the form, texture, and color of the skin.

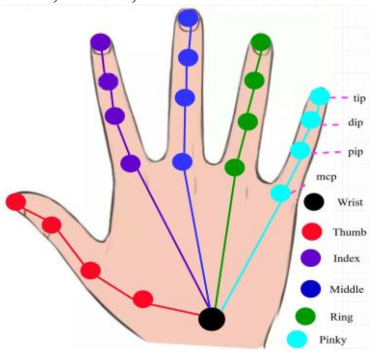


Figure 2 The 21 Key points identified [2]

A base CNN takes feature maps from the input image at different scales. The SSD network predicts bounding boxes with different aspect ratios and sizes and hand presence confidence scores for each site in these feature maps. Non-maximum suppression gets rid of predicted bounding boxes that have low confidence ratings or a lot of overlap with boxes that score higher, leaving only the boxes that are most likely to contain hands. The module detects hand landmarks once it finds hand areas. There are 21 markers in Figure 2. The module gets 42 coordinate values for each input image, 21 for the x-axis and 21 for the y-axis. These values include important hand features including the fingertips, knuckles, palm center, and wrist. Landmark localization uses CNN layers and regression algorithms. Convolutional layers process the input image by calculating dot products between the filter weights and the input values and creating feature maps. Pooling layers take data from neighboring places and combine it to make the spatial dimensions smaller. This protects feature map sampling from tiny translations and distortions. Fully connected layers use complicated non-linear transformations and flatten final feature maps to guess where landmarks are. So, the hand landmarks object has the normalized x and y coordinates for each landmark. The data preparation function goes through all the recognized hands and landmarks to get their normalized x and y coordinates. To keep all values between 0 and 1, the x and y coordinates are kept in separate arrays and normalized by taking away the lowest value. The collection has pictures of hands of different sizes and angles, therefore this normalization is necessary. The data aux array has normalized single-hand feature coordinates.
Data serialization adds features and labels that have been extracted to data and label lists. After processing all of the photographs, the pickle module turns the data and labels lists into a file called test. pickle. The x and y coordinates of a hand landmark for one hand instance are shown by each pair of consecutive elements in the data aux list. If data aux has [x1, y1, x2, y2,..., x21, y21], the first two items (x1, y1) are the coordinates of the first landmark, the second two (x2, y2) are the coordinates of the second landmark, and so on for 21 landmarks. The data. pickle file has a dictionary with two keys: "data" and "labels." The "data" key points to a list of data aux lists that show the features of a single hand instance, while the "labels" key points to a list of class names for each hand sign. Pickle files are used to train machine learning models to process, analyze, and recognize hand signs. Table 1 shows the 21 important points from Sign—B, C, E, F, and Z.

Table 1 21 Key point features extracted from Sign – B, C, E, F & Z in form of X and Y Coordinates

X	B	C	E	F	Z	Y	B	C	E	F	Z
x_0	0.0411	0.0000	0.0478	0.1296	0.2060	y_0	0.1681	0.0151	0.0000	0.1558	0.1348
x_1	0.6074	0.3361	0.3641	0.5792	0.3283	y_1	0.1020	0.1609	0.0832	0.1691	0.0691
x_2	0.1124	0.0859	0.1071	0.1928	0.2142	y_2	0.0757	0.0000	0.0711	0.0729	0.1905
x_3	0.5641	0.2736	0.3334	0.5354	0.2891	y_3	0.1036	0.2097	0.0850	0.1737	0.0920
x_4	0.1531	0.1294	0.1561	0.2526	0.1772	y_4	0.0000	0.0143	0.1373	0.0000	0.2252
x_5	0.4320	0.2201	0.2457	0.4700	0.2371	y_5	0.0454	0.0398	0.0390	0.1033	0.1002
x_6	0.1208	0.1683	0.1490	0.2980	0.1230	y_6	0.3111	0.1243	0.1236	0.2948	0.2044
x_7	0.3216	0.2031	0.1800	0.4177	0.2143	y_7	0.0499	0.0937	0.0350	0.0928	0.0537
x_8	0.0567	0.2155	0.0972	0.2900	0.0716	y_8	0.1840	0.0330	0.0167	0.1749	0.1869
x_9	0.2929	0.1762	0.1716	0.3693	0.2081	y_9	0.0605	0.1526	0.0455	0.0909	0.0664
x_10	0.1339	0.0546	0.1246	0.1930	0.1457	y_10	0.0991	0.0136	0.0786	0.0985	0.2372
x_11	0.3211	0.1008	0.1267	0.3107	0.1258	y_11	0.0644	0.1974	0.0521	0.0917	0.0937
x_12	0.1391	0.1010	0.1469	0.2424	0.0807	y_12	0.0308	0.0112	0.1431	0.0286	0.2620
x_13	0.1860	0.0287	0.0187	0.2438	0.0628	y_13	0.0000	0.0473	0.0000	0.0609	0.0887
x_14	0.1378	0.1463	0.1380	0.2664	0.0360	y_14	0.3349	0.1540	0.1402	0.3334	0.2370
x_15	0.1018	0.0031	0.0674	0.2913	0.0293	y_15	0.0050	0.0863	0.0023	0.0297	0.0513
x_16	0.1331	0.1877	0.1226	0.2716	0.0000	y_16	0.2347	0.0754	0.0628	0.2460	0.2320
x_17	0.0304	0.0012	0.1309	0.3450	0.0000	y_17	0.0126	0.1227	0.0138	0.0118	0.0629
x_18	0.0871	0.0438	0.0789	0.1480	0.1211	y_18	0.1635	0.0338	0.0944	0.1867	0.2618
x_19	0.3035	0.1057	0.1182	0.2855	0.1639	y_19	0.0175	0.1533	0.0203	0.0000	0.0900
x_20	0.0962	0.1008	0.0824	0.1582	0.0640	y_20	0.1000	0.0072	0.1425	0.1303	0.2779

3.3 Design of Random Forest Classifier

The Random Forest Classifier works by creating multiple decision trees from randomly selected subsets of the training data. For each decision tree it randomly selects a subset of features from the total features when growing the tree [18-19]. Each decision tree grows to its maximum depth without any pruning. To make a prediction for a new data instance the Random Forest passes the instance to each of the decision trees. The key advantages of Random Forest Classifiers include high accuracy due to combining multiple decision trees robustness to noise and outliers, ability to handle high dimensional data and relative simplicity compared to other ensemble methods [12, 13]. Additionally, Random Forests provide feature important estimates which can help identify the most relevant features in a dataset. The Random Forest works best on selecting the below model parameters -

Splitting Criterion: Decision trees split nodes based on impurity measures. `Gini' is one of the criteria used to measure node impurity. For this model, the splitting criterion has been set to `Gini'.

Maximum Depth: This parameter controls the maximum depth of the decision trees. For this model, the max depth parameter has been set to "none", which allows the trees to expand until all leaves are pure or contain the minimum number of samples specified by min_samples_split.

Maximum Features: This parameter determines the number of features to consider when looking for the best split at each node. For the model proposed in the paper, the max_features parameter has been set to `sqrt', which means that the square root of the total number of features is considered.

Minimum Leaf Nodes: This parameter specifies the minimum number of leaf nodes that must be present in each decision tree. For the proposed model, we set the minimum leaf nodes to 1 meaning that each decision tree in the Random Forest must have at least one leaf node.

Max Leaf Nodes: This parameter specifies the maximum number of leaf nodes that can be created in each decision tree. Setting this parameter can help control the size and complexity of the individual trees in the Random Forest ensemble. We set Max Leaf Nodes to 2, the ensures each decision tree in the Random Forest can have a maximum of 2 leaf nodes.

Number of Estimators: This parameter defines the number of decision trees in the Random Forest ensemble. For this model, we set it to 100, which means the Random Forest ensemble consists of 100 decision trees.

For other parameters their default values are used, like bootstrapping "True" and class weights "balanced", to promote model diversity and reduces biases towards dominant classes. These settings enhance robustness and generalization by incorporating randomness in tree construction and adjusting class weights based on frequency. The Random Forest classifier operates using an ensemble learning technique called Bagging which is combination of Bootstrap and Aggregating. The process begins with Bootstrap sampling where multiple decision trees are trained on different bootstrap samples of the training data. In bootstrap - sampling data points are randomly selected with replacement resulting in some points appearing multiple times while others may not appear at all in a sample. This creates diversity within the ensemble as each tree is trained on a slightly different subset of the original data [18].

Feature selection or feature bagging is another crucial step where a random subset of features from the input dataset is selected for each decision tree. This reduces correlation between trees and helps prevent overfitting. The number of features considered at each split point is controlled by the max_features parameter, which can be set to "sqrt", "log2" or a specific integer value. In the tree construction phase each decision tree is built using a bootstrap sample. At each node the algorithm searches for the best split among the subset of features determined by max_features. The quality of the split is evaluated using criteria such as Gini impurity or entropy. Gini impurity measures the probability of incorrectly classifying a randomly chosen element if labeled according to the class distribution in the set with lower values indicating more homogeneous sets. The formula for Gini impurity represented as I_g for a set of data with K classes as shown in equation (1)

$$I_g = 1 - \sum_{i=1}^{K} P_i^2 \qquad (1)$$

P_i is the probability of randomly selecting an element of class i from the set. The sum is taken over all K classes. This formula calculates the Gini impurity by summing the squared probabilities of each class and subtracting the result from 1. A lower Gini impurity indicates a more homogeneous set of samples with respect to the target variable. Entropy is a measure of impurity or disorder in a set of data points. It quantifies the uncertainty associated with a given set of data points. The formula for Entropy I_h for a set of data with K classes is given by below Equation (2)

$$I_h = -\sum_{i=1}^{K} P_i \log_2(P_i) \qquad (2)$$

Minimizing the entropy of the resulting subsets is equivalently to maximizing the information gain. Gini impurity is generally faster to compute than entropy because it does not involve logarithmic calculations. This makes the tree-building process quicker. Hence Gini is preferred over entropy. Throughout the prediction phase each tree in the forest independently predicts the class of the input sample. For classification tasks the result (most frequent class) of the predictions from all the trees is taken as the final prediction. This process is called aggregation. In other words the class that receives the most votes among all the trees is selected as the predicted class.

4. HAND SIGN RECOGNITION SYSTEM

A comprehensive workflow for the proposed Hand Sign Recognition System is shown in the Figure 3. The workflow begins with importing the necessary dependencies, such as Media Pipe (version 0.9.0.1), Scikit-learn (version 1.2.0), OpenCV (version 4.7.0.68), and Pandas (version 2.0.3). These libraries provide functionalities for hand tracking, machine learning, image processing, and data manipulation, respectively. The next step is capturing a custom dataset, which is a crucial component for training the hand gesture recognition model. The dataset consists of 26 classes, one for each alphabet of the American Sign Language (ASL). Pre-processing steps are applied to the extracted features, including converting the image color space from BGR to RGB and normalizing the feature values. Feature extraction is a critical step in the workflow, where relevant information is extracted from the input images to represent the hand gestures effectively. The Media Pipe Library employed include transfer learning models, hand models, and various backend models like SSD, YOLO, and CNN. These models facilitate hand detection, landmark localization, and drawing utilities. The static images are processed to extract 21 key points of the hand, and a total of 42 features are derived from each hand, representing the x and y coordinates of the landmarks. Post-processing operations involve data splitting, where the dataset is divided into training and testing subsets, and data serialization, where the features and corresponding labels are stored in a suitable format for further processing.

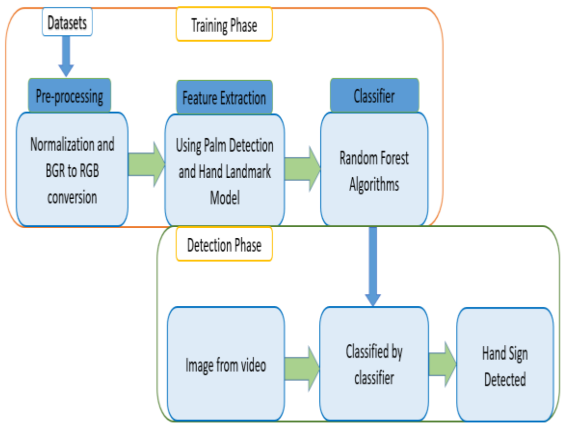


Figure 3 Methodology for Hand Sign Recognition System

The next stage is training a classifier using the extracted features and labels. The Random Forest Classifier with an Ensemble approach is chosen. Specific hyper parameters are provided, including the train test split ratio (80:20), the impurity measures, the number of decision trees, the maximum samples for splitting nodes and max samples leaf, bootstrapping, and the random state are chosen. Model evaluation is an essential step to assess the performance of the trained classifier. Metrics such as accuracy, precision, recall, and F1-score are computed, and a detailed classification report is generated. After training and evaluation, the workflow proceeds to testing, where the trained model is loaded and initialized using the Media Pipe hands module. During testing, the model predicts the hand gestures based on the input images, and the predictions are visualized, allowing for the display of unknown gestures and their corresponding probabilities.

5. EVALUATION OF MODEL

The evaluation of the proposed system relies on two fundamental components: the Stratified K-Fold function and the GridSearchCV function, both pivotal for robust model assessment and hyper parameter tuning. Stratified K-Fold is a cross-validation technique widely used in machine learning to ensure fair and unbiased model evaluation, especially with imbalanced datasets. It divides the dataset into `k' folds while maintaining the original class distribution across each fold. This ensures that each fold has a proportional representation of the different classes, minimizing biased evaluations. The value for k=5 the dataset is split into 5 folds, each having a class distribution similar to the original dataset.

The process begins with shuffling the dataset to eliminate any potential ordering bias that might exist. Once shuffled, the data is partitioned into `k' approximately equal-sized folds. Unlike standard k-fold cross-validation, where data is partitioned randomly into folds without considering class distribution, Stratified K-Fold ensures that each fold contains a balanced representation of classes. During each iteration of the cross-validation process, one of the folds is held out as the validation set, while the remaining folds are used for training the model. This process is repeated `k' times, with each fold being used once as a validation set. By rotating the roles of training and validation sets across folds, Stratified K Fold trains and evaluates the model multiple times on different data subsets. Its key advantage is providing reliable performance estimates across various class distributions, crucial for assessing the model's generalization to unseen data, especially with imbalanced classes. Maintaining a representative mix of classes in each fold helps detect issues like overfitting or under fitting.

GridSearchCV is a technique used for hyper parameter tuning in machine learning, essential for optimizing model performance by systematically exploring a predefined grid of hyper parameters. The process involves exhaustively searching through all possible combinations of hyper parameters and evaluating each combination using cross-validation to identify the set that produces the best results. Firstly a grid of hyper parameters and their corresponding values given in Table 2, are defined. For instance, in a decision tree classifier, parameters like maximum depth, minimum samples split, and criterion are included in the grid. GridSearchCV then performs cross-validation by splitting the training data into `k' folds. It trains the model on `k-1' folds and evaluates its performance on the remaining fold, repeating this process `k' times. During each iteration, GridSearchCV tests every combination of hyper-parameters from the defined grid. The model is trained and evaluated using each combination, and the performance metric specified (such as accuracy, precision, or F1-score) is computed. This comprehensive evaluation allows GridSearchCV to identify the hyper-parameter combination that yields the best performance across all folds.

Combination 4 in Table 3 represents the best combination, showcasing a strategic approach to constructing a decision tree ensemble. Notably, it sets the maximum depth to `None' enabling trees to dynamically adjust their complexity, guarding against overfitting without arbitrary depth limitations. Additionally it establishes a minimum leaf size of 1 fostering refined decision boundaries while maintaining model interpretability. This combination strikes a balance between complexity and accuracy, making it the optimal choice for model performance and generalization. Combination 4 strategically selects the ‘log2’ criterion for feature selection, balancing randomness and model performance. This criterion allows the algorithm to explore an optimal number of features without overwhelming computational resources or risking overfitting. Employing the `gini' criterion prioritizes splits that minimize impurity, enhancing the model's generalization capabilities. Furthermore, Combination 4 utilizes 200 estimators, leveraging the power of ensemble learning to mitigate biases and errors through averaging. This approach bolsters robustness and generalization without imposing excessive computational burdens.

Table 2 Hyper Parameters for Random Forest Classifier

Hyper parameters	Values
n_estimators	50, 100, 200
Max_depth	None,10,20
Min_samples_split	2,5,10
Min_samples_leaf	1,2,4
Max_features	Sqrt,log$_2$
Criterion	Gini, Entropy, Log loss

The method was effectively implemented to extract 21 key points from hand images using Media Pipe's hand module which utilizes machine learning techniques to accurately identify and extract key points representing hand landmarks in each image. These extracted key points provide effective features for representing sign language gestures. The dataset consists of 26 classes, each representing a different sign language gesture, with 200 images per class, resulting in a total of 5,200 images. For each image, the Media Pipe hand module extracts 21 key points, with each key point having both x and y coordinates, resulting in 42 features per image. Consequently, for the entire dataset, the total number of features extracted is 2, 18,400 (5,200 images multiplied by 42 features per image). This detailed representation of hand landmarks, comprising 21 x-coordinates and 21 y-coordinates for each image, enables precise and reliable recognition of sign language gestures.

Table 3 GridSearchCV Result

Parameters	Combination 1	Combination 2	Combination 3	Combination 4
n_estimators	100	200	50	**200**
Max_depth	10	20	None	**None**
Max Leaf	5	10	100	**2**
Min leaf	2	4	1	**1**
Max_features	sqrt	Log2	sqrt	**Log2**
Criterion	Entropy	Gini	Entropy	**Gini**

5.1 Performance Metrics

Accuracy is the ratio of correctly predicted instances to the total instances in the dataset as shown with Equation 3. It is a measure of the overall correctness of the model across all classes. Our model attained an accuracy of 95.67% with 200 estimators, demonstrating its high performance in hand sign recognition. Precision is the ratio of correctly predicted positive observations to the total predicted positive observations as shown with Equation 4. It measures the proportion of correctly identified positive cases among all cases that were predicted as positive. Our model attained a Precision of 95.83\% with 200 estimators.

$$Accuracy = \frac{\text{Number of Correct Preditions}}{\text{Total Number of Predictions}} \quad (3)$$

$$Precision = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \quad (4)$$

Recall also known as Sensitivity or true positive rate, is the ratio of correctly predicted positive observations to all actual positives in the dataset as given in Equation 5. With 200 estimators, our model achieved an impressive accuracy of 95.67%. The F1 score is the harmonic mean of precision and recall as shown with Equation 6. It provides a balance between precision and recall and is often used as a single metric for evaluating classification models, especially when there is an uneven class distribution proposed technique attained a score of 95.74%.

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \quad (5)$$

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad (6)$$

Table 4 presents the overall results of a proposed technique using different numbers of estimators 50, 100 and 200 in a machine learning model. As the number of estimators increases from 50 to 200, we observe an increasing trend across various evaluation metrics. This indicates a balanced improvement in the model's ability to correctly classify positive and negative instances.

A confusion matrix is a table that is often used to describe the performance of a classification model on a set of test data for which the true values are known. It allows visualization of the performance of an algorithm and helps in understanding how well the model is performing in terms of classifying different categories. In the presented confusion matrices, each corresponding to varying numbers of n-estimators (200, 100 and 50) the performance of the classification models is evaluated based on their ability to correctly classify samples. The x-axis and y-axis of the matrix represent the predicted labels and true labels respectively. Each row in the matrix corresponds to an actual label while each column represents a predicted label. The diagonal elements (from the top-left to the bottom-right) of the matrix represent the correctly classified samples or true samples where the predicted label matches the true label. These values are typically highlighted or colored differently to make them visually distinct. The off-diagonal elements represent the misclassified samples or false samples. The values in these cells indicate the number of instances that were incorrectly classified as belonging to a different class. The confusion matrix for 200 n-estimators is shown in Figure 4.

Table 4 Performance Metrics for the proposed Hand Sign Recognition System

Estimators	True Samples	False Samples	Accuracy	Precision	Recall	F1 Score
50	3979	181	0.9404	0.9421	0.9404	0.9387
100	3958	202	0.9483	0.9497	0.9483	0.9489
200	3944	216	0.9767	0.9583	0.9567	0.9574

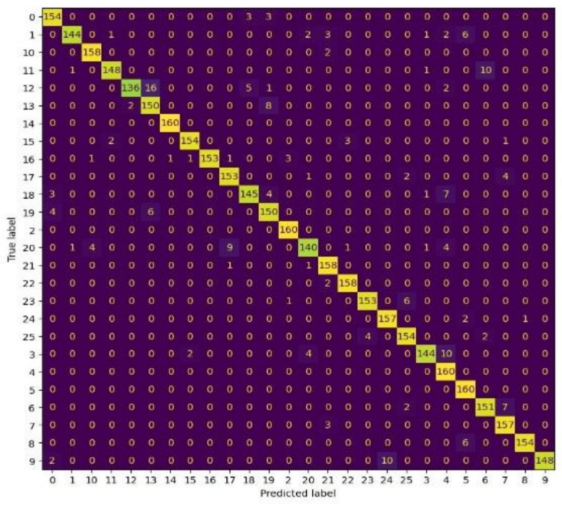


Figure 4 Confusion Matrix for Proposed Hand Sign Recognition System

5.2 Comparative Analysis across Different Lighting Conditions

In order to make the proposed model robust, we tested it to operate in different light conditions and with different individuals. Three different test scenarios aimed at assessing the detection probability across different lighting conditions like Light, Dim, Dark are identified. The quality of lighting conditions were evaluated using different image quality descriptors like Luminance, Brightness, Contrast, Noise level.

Brightness represents the pixel intensity in range of 0 to 255 for 8-bit color images. The brightness is calculated as the mean value of all pixels in the image. Contrast measured as the standard deviation of pixel intensity values. The contrast is calculated by first computing a histogram of grayscale pixel values using, and then taking the standard deviation of the histogram. Theoretically, its range spans from 0 to approximately 73.74 for 8-bit images, as the standard deviation can extend to the value of a uniform distribution ranging from 0 to 255. Contrast reflects the extent to which pixel values deviate from the mean pixel value within the grayscale histogram. Higher contrast values indicate more pronounced deviations and greater image contrast. The noise level is estimated by calculating the standard deviation of the difference between the grayscale image and a blurred version of the image using a 5x5 Gaussian kernel. This metric estimates the variability in pixel values attributed to noise. The brightness, contrast, and noise level considered for 3 cases are tabulated in Table 5.

Case 1

The hand signs are detected in daytime conditions with sufficient luminance or brightness levels showing a person displaying three different hand signs labeled as B, C and F illustrated in Figure 5. The luminance or brightness values range from 159.29 to 167.54, indicating good overall brightness. The contrast values show a wide variation from 2020.20 to 402.21 and 464.55. This variation in contrast could indicate differences in the lighting conditions or scene compositions within this case. The noise levels ranging from 3.27 to 4.46, which is acceptable because it can be compensated with high luminance and brightness. Overall, hand Signs in Case 1 appear to have very high luminance and brightness compared to other cases, but the wide variation in contrast values and slightly elevated noise levels suggest potential challenges in achieving consistent image quality across this case. Also indicated is a probability value enclosed in parentheses which represents the model's confidence in classifying that particular hand sign. Results for hand sign B, C and F are 0.59, 0.73 and 0.73 respectively.

Case 2

The dim lighting conditions are considered to acquire the images in Case 2 illustrated in Figure 6. The luminance values range from 144.08 to 146.05, and the brightness values range from 143.57 to 145.70, which are slightly lower than Case 1 but still within an acceptable range. However, the contrast values are exceptionally high, ranging from 1136.97 to 1201.20. Such high contrast values suggest the presence of very distinct bright and dark regions within the images, which could potentially lead to loss of detail in the adequate brightness or darkness areas. The noise levels are relatively low, ranging from 3.56 to 3.58, indicating that these images have minimal noise or unwanted artifacts. However an important factor to note is that this person's dataset was not included in the training data used to build the model. This means that the model has not been explicitly trained on this individual's hand gestures or appearances. Figure 5 shows results for B, C and F with confidence of 0.45, 0.65 and 0.63 respectively.

Case 3
The hand signs in Case 3 represent low-light or dark conditions. The luminance values range from 75.37 to 79.34, and the brightness values range from 75.59 to 79.40, which are significantly lower than the previous two cases, indicating darker overall conditions. Interestingly, the contrast values for this case are relatively high, ranging from 1144.59 to 1175.62, which could be attributed to the presence of both bright and condition dark regions within the low-light scenes. The noise levels are the lowest among the three cases, ranging from 3.00 to 3.36, which might be due to the lower overall brightness levels in these images. Overall, the images in Case 3 as shown in Figure 7 exhibit low luminance and brightness levels, as expected for dark conditions. The high contrast values could pose challenges in preserving detail in both bright and dark areas, while the relatively low noise levels are a positive aspect. The results for B, C and F are show as 0.59, 0.48 and 0.61.

Overall, the analysis revealed that daytime images (Case 1) had high brightness with variable contrast, dim lighting images (Case 2) had high contrast and low noise, while dark condition images (Case 3) exhibited low brightness but relatively high contrast and low noise. The model's classification confidence varied with lighting conditions, generally decreasing in dimmer settings. Though, the confidence levels decreased with varying lighting conditions, the proposed system is successful in detecting the hand signs.

Table 5 The Lighting Conditions for 3 cases.

	Brightness	Contrast	Noise Level
Case 1	159.29 to 167.54	402.21 - 2020.20	3.27 to 4.46
Case 2	143.57 to 145.70	1136.97 to 1201.20	3.56 to 3.58
Case 3	75.59 to 79.40	1144.59 to 1175.62	3.00 to 3.36

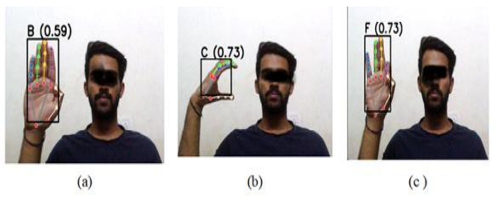


Figure 5 Case 1 Hand Sign detected in Daytime Condition

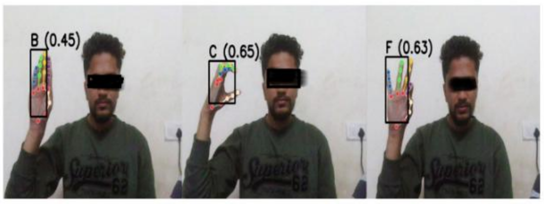


Figure 6 Case 2 Hand Sign detected in Moderate light

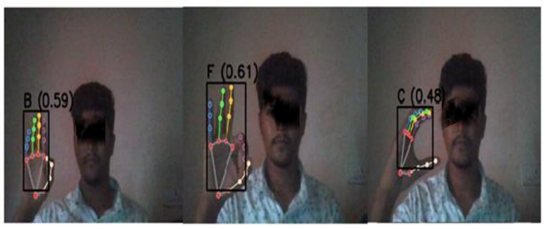


Figure 7 Case 3 Hand Sign detected in Dark light condition

6. USER INTERFACE TO CONVERT HAND SIGN TO TEXT AND SPEECH

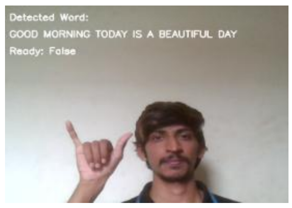
The primary objective of this paper is to build a user interface to enable individuals who are unable to speak or hear, to express themselves effectively using the proposed sign recognition system. The proposed model is interfaced with a webcam to acquire hand sign inputs from users in real-time. These are translated to speech and text in real-time. This innovative approach not only facilitates communication with others but also enhances accessibility and inclusivity in various social and professional settings [16].

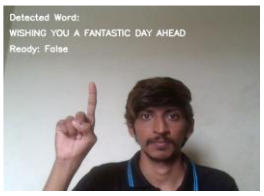
The OpenCV library is employed for capturing frames from the webcam, processing these frames, and displaying them with overlays such as text annotations. Additionally, OpenCV facilitates color space conversions and basic image manipulations. MediaPipe library is utilized for hand landmark detection. This library offers pre-trained machine learning models to accurately locate key points on the hand in real-time, enabling precise tracking of hand gestures and movements. Typically, this involves training a model on a dataset of hand gesture images annotated with their corresponding labels (e.g., letters or words). The pyttsx3 library is utilized for text-to-speech conversion. This library provides an interface to the platform-specific TTS engines installed on the system, allowing the script to generate spoken output from text strings. The script interacts with the user through keyboard commands. These commands check for system readiness, trigger speech synthesis, add spaces, deselect words and delete characters from the constructed word.

The proposed user interface is demonstrated for 2 sample gestures as shown in the Figure 8 Sample gesture (a) and (b). With the implementation of key strokes like backspace key likely allows the user to delete the last entered character, the delete key enables the user to remove the entire detected text. The “Ready” state displayed in the images suggests the presence of a key or command that toggles the system's readiness to capture and interpret hand gestures. When set to “True” the application actively monitors and translates the user's hand signs into text. The inclusion of a key or command that triggers the text-to-speech functionality is designed. This would allow the detected text to be vocalized, providing an audible output for improved accessibility.

In the Figure 8(a) the detected text based on the hand gestures reads “good morning today is a beautiful day". However, the “Ready” state is set to False indicating that the system is not actively capturing or interpreting hand signs at that moment. Figure 8(b) shows a different sentence detected as “wishing you a fantastic day ahead” again with the “Ready” state set to False.

Overall, this system integrates computer vision for hand gesture detection, machine learning for gesture classification, and text-to-speech technology for spoken output. It offers a practical framework for real-time translation of hand signs into audible speech, facilitating communication for individuals with hearing impairments.





(a) (b)

Figure 8 Sample Gestures identified by the Proposed System

7. COMPARISON WITH STATE-OF-ARTS-TECHNIQUES

Jindi Wang et. al. have built a camera-based hand gesture recognition model by training the model for a particular user only. They employed a lightweight Multilayer Perceptron architecture based on contrastive learning, reducing the size of the data needed and the training timeframes. Experimental results demonstrated rapid convergence and 94.25% accuracy of the recognition model[9]. In 2023, Alonazi et al [2] employed Deep Belief Nets and Convolutional Neural Networks (CNNs) on a custom Hand Gesture Dataset, attaining an accuracy of 90.73%.

Table 6 Comparatively Analysis with state-of-art techniques

S.no	Year	Author	Dataset	Feature Extraction	Detection Technique	Accuracy
1.	2025	Proposed Work	Custom ASL & Gesture Dataset	Media Pipe	Random Forest	95.73%
2.	2024	Jindi Wang et. al. [9]	Custom Gesture Dataset	Media Pipe	Multi-Layer Perceptron	94.25%
3.	2023	Mohammed Alonazi et. al.[1]	Custom Hand Gesture	CNN	Deep Belief Network and CNN	90.73%
4.	2023	Christine Dewi et. al. [3]	Oxford Hand Dataset	Yolo7	Yolo7	86.3%
5.	2023	Jyotishman Bora et. al. [10]	Assamese Sign Langauge	Media Pipe	Custom Feedforward Network	99%
6.	2023	Chen et. al. [5]	EgoHands & Tiny HGR datasets	Yolo5l	Yolov5l	75.6% and 66.8%
7.	2022	Shashidhar R [6]	Indian Sign Language	CNN	CNN	99%
8.	2021	Shin et. al. [4]	ASL Dataset, Massey Dataset and Finger Spelling A dataset	Media-pipe API	SVM and GBM	87.60%, 87.60%, 98.45%
9.	2022	V Radhika[22]	ASL Dataset		SVM, KNN, CNN	97%, 95%, 98.49%

The field of sign language recognition has seen significant advancements in recent years, with the application of image processing and machine learning techniques. Based on the results obtained from the implementation of various n-estimators in the random forest algorithm for hand sign recognition, it is evident that increasing the number of estimators generally improves the accuracy of classification. The highest accuracy of 95.67% was achieved with 200 n-estimators, showcasing the robustness of the model in distinguishing between different hand signs. However, it's important to note that while higher n-estimators may lead to improved accuracy, there might be a trade-off with computational resources and efficiency. Furthermore, the analysis of confusion matrices revealed consistent trends across different numbers of n-estimators, with the majority of samples correctly classified as true positives. This indicates the effectiveness of the random forest algorithm in accurately recognizing hand signs.

Jyotishman Bora et. al. implemented a Assamese Sign recognition model by using Media Pipe hand tracking and detection API as feature extractor and a custom made Multi-

Layer Perceptron Network. They achieved an accuracy 94.25% [9]. Dewi et al. [3] used Yolov7 and Yolo models on the American Sign Language (ASL) dataset, yielding a mean average precision of 86.3% and recall of 79.9%. In 2021, G. Pala

et al. applied Support Vector Machines (SVM), K-Nearest Neighbors (KNN), and Convolutional Neural Networks (CNN) on the American Sign Language (ASL) Dataset, achieving an accuracy of 98.49\% for CNN, 83% for SVM, and 93% for KNN[26]. Chen et al. proposed YOLOv5 in 2023, achieving accuracies of 75.6\% and 66.8\% on EgoHands and TinyHGR datasets, respectively[5]. Shin et al.[13] employed Support Vector Machines (SVM) and Gradient Boosting Machine (GBM) on ASL dataset, Massey dataset, and Finger Spelling A dataset in 2021, achieving an average accuracy of 96% for SVM and 93% for GBM. In 2022, A.Pothuri et al. [19] proposed Random Forest techniques on the Indian Sign Language (ISL) dataset, achieving accuracies of 96% for ISL. Sangum et al. [11] used Random Forest on a Custom Hand Gesture Dataset in 2015, attaining an accuracy of 90% for 160x120 resolution and 94% for 640x480 resolution.

8. CONCLUSION

In conclusion, the project successfully demonstrated the feasibility and effectiveness of using the random forest algorithm for hand sign recognition, laying the foundation for further research and development in this field. The insights gained from this project contribute to the advancement of gesture recognition technology, with potential applications in human-computer interaction, assistive technology, and accessibility solutions.

Integration of hand sign recognition capabilities into wearable devices such as smart glasses or wristbands could enable hands-free interaction and communication for individuals with disabilities. Additionally incorporating hand sign recognition into Internet of Things (IoT) devices could enable gesture-based control of smart home appliances, entertainment systems and other connected devices [22, 23]. While there is great potential for hand sign recognition systems in assistive technology applications, including communication aids for individuals with disabilities the accessibility and affordability of such technologies may limit their widespread adoption.

Machine learning-powered educational applications for sign language learning and training hold promise creating immersive virtual reality (VR) or augmented reality (AR) environments with realistic hand sign recognition capabilities may still be challenging. Developing adaptive learning platforms that cater to individual learning styles and abilities also requires sophisticated machine learning algorithms and user interface design. Although real-time hand sign recognition systems have potential applications in healthcare and rehabilitation integrating these systems into clinical practice may face regulatory hurdles and require rigorous validation and testing. Additionally ensuring the accuracy, reliability and safety of machine learning algorithms in critical healthcare settings remains a significant challenge

While these future scopes represent exciting opportunities for advancing real-time hand sign recognition systems using machine learning, addressing the associated challenges will require collaborative efforts across multiple disciplines, including computer science, engineering, healthcare, and social sciences. Continued research innovation and investment in these areas are essential to realizing the full potential of hand sign recognition technology in improving accessibility, communication and quality of life for individuals worldwide.

9. DECLARATION

All authors declare that they have no conflicts of interest.

10. ACKNOWLEDGMENTS

Our thanks to the experts who have contributed towards development of the template.

11. REFERENCES

[1] Alonazi, Mohammed, Hira Ansar, Naif Al Mudawi, Saud S. Alotaibi, Nouf Abdullah Almujally, Abdulwahab Alazeb, Ahmad Jalal, Jaekwang Kim and Moohong Min. “Smart Healthcare Hand Gesture Recognition Using CNN-Based Detector and Deep Belief Network.” IEEE Access 11 (2023): 84922-84933

[2] Abu Salem Musa Miah, Md. Al Mehedi Hasan and J. Shin, "Dynamic Hand Gesture Recognition Using Multi-Branch Attention Based Graph and General Deep Learning Model," in IEEE Access, vol. 11, pp. 4703-4716, 2023, doi: 10.1109/ACCESS.2023.3235368.

[3] Dewi, Christine, Abbott Po Shun Chen, and Henoch Juli Christanto, "Deep Learning for Highly Accurate Hand Recognition Based on Yolov7 Model" Big Data and Cognitive Computing 2023, 7(1):53. https://doi.org/10.3390/bdcc7010053.

[4] Shin J, Matsuoka A, Hasan MAM, Srizon AY. American Sign Language Alphabet Recognition by Extracting Feature from Hand Pose Estimation. Sensors. 2021; 21(17):5856. https://doi.org/10.3390/s21175856.

[5] Chen, Renxiang, and Xia Tian, "Gesture Detection and Recognition Based on Object Detection in Complex Background," 2023 Applied Sciences 13, no. 7: 4480. https://doi.org/10.3390/app13074480.

[6] Shashidhar R, A. S. Manjunath and B. N. Arunakumari, "Indian Sign Language to Speech Conversion Using Convolutional Neural Network," *2022 IEEE 2nd Mysore Sub Section International Conference (MysuruCon)*, Mysuru, India, 2022, pp. 1-5, doi: 10.1109/MysuruCon55714.2022.9972574.

[7] Han, Jeong-Seop, et al. "A study on real-time hand gesture recognition technology by machine learning-based mediapipe." Journal of System and Management Sciences 12.2 (2022): 462-476.

[8] Sung, George, et al. "On-device real-time hand gesture recognition." arXiv preprint arXiv:2111.00038 (2021).

[9] Wang, J., Ivrissimtzis, I., Li, Z. et al., “Hand gesture recognition for user-defined textual inputs and gestures. Univ Access Inf Soc ., 2024. https://doi.org/10.1007/s10209-024-01139-6.

[10] Jyotishman Bora & Dehingia, Saine & Boruah, Abhijit & Chetia, Anuraag & Gogoi, Dikhit, “Real-time Assamese Sign Language Recognition using MediaPipe and Deep Learning,” 2023 Procedia Computer Science. 218. 1384-1393. 10.1016/j.procs.2023.01.117.

[11] Sangjun, O. & Mallipeddi, Rammohan & Lee, Minho, “Real Time Hand Gesture Recognition Using Random Forest and Linear Discriminant Analysis”, In Proceedings of the 3rd International Conference on Human-Agent Interaction (HAI '15). Association for Computing Machinery, New York, NY, USA, 279–282. https://doi.org/10.1145/2814940.2814997.

[12] Bhushan, Shashi, Mohammed Alshehri, Ismail Keshta, Ashish Kumar Chakraverti, Jitendra Rajpurohit, and Ahed Abugabah. 2022. "An Experimental Analysis of Various Machine Learning Algorithms for Hand Gesture Recognition" *Electronics* 11, no. 6: 968. https://doi.org/10.3390/electronics11060968.

[13] Shin, Jungpil & Miah, Abu Saleh Musa & Akiba, Yuto & Hirooka, Koki & Hassan, Najmul & Hwang, Yong. (2024). Korean Sign Language Alphabet Recognition through the Integration of Handcrafted and Deep Learning-Based Two-Stream Feature Extraction Approach. IEEE Access. PP. 1-1. 10.1109/ACCESS.2024.3399839.

[14] Mohammadi, Zahra, Alireza Akhavanpour, Razieh Rastgoo, and Mohammad Sabokrou. "Diverse hand gesture recognition dataset." Multimedia Tools and Applications 83, no. 17 (2024): 50245-50267.

[15] Suharjito, Suharjito & Wiryana, Fanny & Zahra, Amalia. (2018). Feature Extraction Methods in Sign Language Recognition System: A Literature Review. 11-15. 10.1109/INAPR.2018.8626857.

[16] S. Adhikary, A. K. Talukdar and K. Kumar Sarma, "A Vision-based System for Recognition of Words used in Indian Sign Language Using MediaPipe," *2021 Sixth International Conference on Image Information Processing (ICIIP)*, Shimla, India, 2021, pp. 390-394, doi: 10.1109/ICIIP53038.2021.9702551.

[17] Antonio Guadalupe Cruz Bautista, Jose-Joel Gonzalez-Barbosa, Juan Bautista, Hurtado-Ramos, Francisco-Javier Ornelas-Rodriguez, Erick-Alejandro Gonzalez-Barbosa, "Hand Features Extractor Using Hand Contour - A Case Study", Automatika - Journal for Control, Measurement, Electronics, Computing and Communications, Vol. 61, No. 1, pp. 99-108, 2020.

[18] Ribó, Alba & Warchoł, Dawid & Oszust, Mariusz, "An Approach to Gesture Recognition with Skeletal Data Using Dynamic Time Warping and Nearest Neighbor Classifier," Journal of Intelligent Learning Systems and Applications, 2016, Vol. 8. pp. 1-8. 10.5815/ijisa.2016.06.01.

[19] A. S, A. Potluri, S. M. George, G. R and A. S, "Indian Sign Language Recognition Using Random Forest Classifier," *2021 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT)*, Bangalore, India, 2021, pp. 1-6, doi: 10.1109/CONECCT52877.2021.9622672.

[20] Joshi, Harita & Golhar, Vaibhav & Gundawar, Janhavi & Gangurde, Akash & Yenkikar, Anuradha & Sable, Nilesh. (2024). Real-Time Sign Language Recognition and Sentence Generation. SSRN Electronic Journal. 10.2139/ssrn.4992818.

[21] V. Bansal, S. Sinha, R. Astya, A. K. Sagar and K. Sahu, "A Hybrid Approach to Sign Language Recognition Using MediaPipe and Machine Learning," *2025 IEEE International Students' Conference on Electrical, Electronics and Computer Science (SCEECS)*, Bhopal, India, 2025, pp. 1-6, doi: 10.1109/SCEECS64059.2025.10940312.

[22] V. Radhika, C. R. Prasad and A. Chakradhar, "Smartphone-Based Human Activities Recognition System using Random Forest Algorithm," *2022 International Conference for Advancement in Technology (ICONAT)*, Goa, India, 2022, pp. 1-4, doi: 10.1109/ICONAT53423.2022.9726006.

[23] B. Ben Atitallah *et al.*, "Hand Sign Recognition System Based on EIT Imaging and Robust CNN Classification," in *IEEE Sensors Journal*, Vol. 22, no. 2, pp. 1729-1737, 15 Jan.15, 2022, doi: 10.1109/JSEN.2021.3130982.

[24] Mohit Patil, Pranay Pathole, Hrishikesh Patil, Ashutosh Raut, Prof. S S Jadhav. “Indian Sign Language Recognition,” International Journal of Scientific Research Engineering Trends, Volume 6, Issue 4, July-Aug-2020, ISSN (Online): 2395-566X.

[25] Mohammadi, Zahra, Alireza Akhavanpour, Razieh Rastgoo, and Mohammad Sabokrou. "Diverse hand gesture recognition dataset." Multimedia Tools and Applications 83, no. 17 (2024): 50245-50267.

[26] Pala, Greeshma, et al. "Machine learning-based hand sign recognition." 2021 International Conference on Artificial Intelligence and Smart Systems (ICAIS). IEEE, 2021.