Smart Communication System Using Sign Language Interpretation

Divyansh Bisht, Manthan Kojage, Manu Shukla, Yash Prakash Patil, Priyanka Bagade
Indian Institute of Technology, Kanpur, India
{dbisht21,manthank21,manushukla,yashpp21,pbagade}@iitk.ac.in

***Abstract*—Although sign language has become more widely used in recent years, establishing effective communication between mute/deaf people and non-signers without a translator remains a barrier. There have been multiple methods proposed in the literature to overcome these challenges with the help of Sign Language Recognition (SLR) using methods based on arm sensors, data glove and computer vision. However, the sensor-based methods require users to wear additional devices such as arm bands and data-glove. The sensor-free vision-based methods are computationally intensive and sometimes less accurate as compared to the wearable sensor-based methods. In this paper, we propose a vision-based light weight web-based sign-language interpretation system. It provides two-way communication for all classes of people (deaf-and-mute, hard of hearing, visually impaired, and non-signers) and can be scaled commercially. The proposed method uses Mediapipe to extract hand features from the input image/video and then uses a light weight random forest classifier to classify the signs based on the extracted features with the accuracy of 94.69 %. The proposed model is trained on alphabets from American Sign Language. We developed a web-based user interface to remove for ease of deployment. It is equipped with text-to-speech, speech-to-text and auto-correct features to support communication between deaf-and-mute, hard of hearing, visually impaired and non-signers.**

I. Introduction

Sign language enables deaf, hard-hearing and mute people to communicate effectively with others [1]. Signs can be generated using hand movements and can be intercepted visually using eyes. However, most of the people do not have the domain knowledge to interpret the sign language. Traditionally, humans have been employed as sign language translators. However, it makes the communication inefficient and expensive making its accessibility poor. With the advancement in technology, various methods are proposed for Sign Language Recognition (SLR) using wearable IoT sensors, data-gloves and vision-based system [2]–[5]. These methods can potentially reduce the burden of having human translator for communication.

SLR with wearable IoT sensors and data gloves rely on capturing data from the IoT devices and interpret the hand gesture for sign recognition. Data-glove technology uses mechanical or optical sensors attached to a glove to convert finger flexions into electrical impulses to determine hand posture. Gloves worn by the users are connected to the computer using heavy cords that can obstruct the user's natural hand movements [6], [7]. Similarly with arm sensors-based method, users are required to wear EMG sensors on the arm making it an invasive system [2], [8]–[10]. Although these methods can provide good accuracy in signs prediction, the requirement of additional hardware makes them unsuitable to be used in daily conversations. The accelerated research

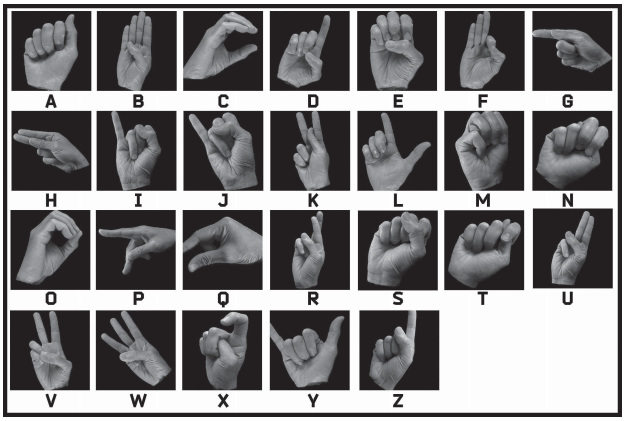

Fig. 1. American Sign Language (A.S.L.) signs

in AI and Computer vision led to developing new methods and models for SLR. These methods are non-invasive and convenient since the user does not need to wear any additional hardware to interact with the computer. Convolutional Neural Networks (CNN) are widely used to interpret the signs. Their ability of automatic feature extraction makes them attractive to be implemented without instrumenting additional algorithms for image processing. However, CNNs are computationally expensive models to be used in real time communication e.g. in video conferencing. This leads to the need of light weight models to support real time recognition for SLR.

In this paper, we propose a computer vision-based SLR method to assist deaf, mute hard-hearing and visually impaired people to communicate with non-signers. Plenty of sign languages are used globally, like American sign language, Indian sign language, German sign language, Etc. The American Sign Language (A.S.L.) is the most widely used among all of them. Therefore, this work uses the basics of A.S.L. as a communication medium using sign languages. The dataset used in this work consists of A.S.L. signs of 26 alphabets (as shown in Figure 1) primarily and words used in daily communication such as thank you, please, hello, yes etc. Users can generate any possible words from these basic 26 alphabets and even use predefined words to save time.

The proposed work is intended to design a highly accurate

real-time communication model that is light weight and easily scalable with the web-based User Interface (UI). The proposed UI also supports speech to text, text to speech and auto completion to allow visually impaired people to use the framework efficiently as well. It enables deaf/mute person to communicate with the visually impaired person possible without any middle man or a translator. The proposed method uses Mediapipe to extract the features from the image/video and uses random forest classifier to classify them. Since both Mediapipe and random forest classifier require less compute power, this method becomes suitable for establishing real time communication. The detailed study of the state of the art is reviewed in section II. It compares the proposed approach with the currently available solutions for sign language interpretation. Sections III and IV focus on the overview of the proposed research methodology. Performance analysis and UI features are described in Results section V. Section VI discusses the future work on how the current methodology can be improved further to achieve better accuracy.

II. Related Work

Approaches for sign language interpretation in human-computer interaction can be divided into two types, wearables-based methods and vision-based methods. (a) Wearables-based method: It includes both data-glove and arm sensor technology. They both rely on capturing data from the wearable devices and interpret the hand gesture for sign recognition. Data-glove technology uses mechanical or optical sensors attached to a glove to convert finger flexions into electrical impulses to determine hand posture. Gloves worn by the users are connected to the computer using heavy cords that can obstruct the user's natural hand movements [6], [7]. Similarly with arm sensors-based method, users are required to wear EMG sensors on the arm making it an invasive system [2], [8]–[10]. (b) Vision-based methods: Computer vision techniques are non-invasive and are based on how people interpret information in their environment. This method is more convenient since the user does not wear any additional hardware to interact with the computer.

Since we propose a computer vision-based approach for sign language interpretation, we will discuss the state of the art in vision-based methodologies only. Table I shows the comparison of the currently available vision-based methods and the proposed method for SLR. There have been multiple methods utilizing Convolutional Neural Network based or compute intensive models [3], [11]–[16] and light-weight models [4], [17]–[20] for Sign Language Recognition(SLR) from input video/image.

A. *SLR using CNN-based models*

Kshitij Bantupalli et al. [3] utilized a custom recorded American Sign Language dataset using four synchronized cameras for Sign Language Recognition (SLR) using Convolutional Neural Network (CNN), Long Short-Term Memory (LSTM) and Recurrent Neural Network (RNN) models. These models extract spacial features and temporal information from the custom dataset. Although the method achieved 90% accuracy the dataset is confined to 150 words only. Further the method does not work well with different skin tones, different background and when a face is a part of the frame. Shikhar et al. [11] employed a dataset of 24 ASL alphabets (J and Z were excluded) with depth and color maps, feature descriptors, and CNN to achieve real-time SLR and accuracy 91.4%. The model does not support signs represented using both hands and did not include the formation of complete sentences. The model did not have the feature for working on unique and more interactive words with gestures other than ASL.

Koller et al. [13] employed a mixture of CNN and Hidden Markov Model (HMM) to recognize continuous sign language. The CNN model was used to recognize the hand gesture, while the HMM model was utilized to analyze the results supplied by the CNN model in real-time. Further, Garcia et al. [14] also use CNN with the transfer learning approach for SLR. However, both of these models perform poorly for similar gestures and have an accuracy of 79.1%. Use of 3D-CNN is has also been evaluated for SLR with Italian gesture dataset [15] resulted in low accuracy of 78.9%. Kang et al. [16] also proposed using CNN to solve SLR. For hand segmentation, the method used depth maps, and the cropped image was then fed into CNN (AlexNet) for classification. However, the difficulty of recognizing alphabets for a different signer was not addressed, and color map compatibility was not included. The authors were able to reach an accuracy of 74.1% only. Rioux et al. [4] used Deep Belief Network (DBN) to classify different types of feature extraction methods on ASL dataset with accuracy 76.1%. DBN is a classification algorithm that uses both depth and intensity data to classify images. It works only for static images and can recognize signs from one hand. They have proposed a highly advanced network that can reach an accuracy of 76.1% and also takes around 2.69sec to recognize the gesture. To reduce the computations in feature extraction, Mediapipe framework has been used for Vietnamese sign language detection [5] using RNN. However, RNN is a compute intensive model as compared to the light weight models mentioned in the next section.

B. *SLR using light-weight models*

Wang H. et al. [12] have proposed an advanced framework based on HMM called Light-HMM. It uses low-rank approximation to identify the keyframes and adaptively determine hidden states in HMM. The model's accuracy drops from 93% to 77% when tested on public users with different skin tone, background and clothing colors. Further, Kishore PVV et al. [18] proposed a model that incorporates the Elliptical Fourier Descriptor (EFD) for feature extraction and Artificial Neural Network (ANN) for feature recognition trained on Indian Sign Language with an accuracy of 90.12%. This methods requires a four-camera paradigm for identifying Indian sign language motions. HSV and YCbCr color models were able to achieve accuracy of 93% with 24 ASL alphabets (excluding J and Z). However, the model could only take image as input making it inefficient to use for real time recognition [17]. A

TABLE I. COMPARATIVE ANALYSIS OF RELATED WORK

Research Work	Method	Model used	Accuracy	Real time recognition	Background dependency	User interface	Shortcomings/Features
[3]	Covers 150 common gestures including ASL alphabets.	CNN, RNN, LSTM	90	Yes	Skin tone dependent.	No	Background noise like face or clothing decrease real time accuracy.
[11]	24 ASL alphabets using depth and color map	CNN	91.4	Yes	Background independent	No	Fails for 2 hand gestures, sentence formation feature missing, text to speech and vice versa missing.
[17]	24 ASL alphabets using HSV color model where images were first converted to grayscale	HSV, YCbCr color model	93	No	Background dependent	No	Very static model, lacks realtime interpretation and works only on static images.
[12]	ASL alphabets using light HMM model	HMM	93	Yes	Background dependent	No	Accuracy suffers when tested in signer independent situation, just detects signs and lacks advanced features(autocorrect, text to speech etc.).
[13]	CNN and HMM for continuous sign language	CNN, HMM	79.1	Yes	Background dependent	No	Low accuracy, sentence formation and other advanced features missing.
[14]	Transfer learning using GoogLeNet	Transfer learning	79.1	Yes	Background dependent	No	Accuracy is low and advanced features are missing.
[15]	3D-CNN for upper body and hand extraction	3D-CNN	78.9	Yes	Background dependent	No	Low accuracy
[16]	CNN and depth maps for sign language	CNN	74.1	Yes	Background dependent	No	Low accuracy, different signer problem not addressed, color map compatibility not included.
[18]	ANN and Elliptical Fourier Descriptor for ISL	ANN, EFD	90.12	Yes	Background dependent	No	Setup not scalable and model lacks dynamic features.
[4]	Deep Belief Network for ASL	DBN	76.1	Yes	Background depdendent	No	Works only for 1 hand, not dynamic, low accuracy and slow recognition in real time
[19]	SVM for ISL	SVM	97.5	Yes	Background dependent	No	Slow recognition in real time.
[20]	Dynamic signs predictions using human pose estimation, implements lightweight model on front end, with balance of speed and accuracy.	LSTM	91.5	Yes	Background independent	Yes	Requires dedicated GPU in front end(client-side).
Our method	Custom ASL and ISL alphabets along with gestures for some common words using mediapipe.	LwP	94.69	Yes	Background independent. Works well even with face, clothes etc.	Yes. User can know what features are fed to model at real time.	High accuracy, fast at real time, simple model used so it's easily deployable, sentence formation, text to speech and vice versa, autocorrect feature present, very dynamic and can adapt to new signs.

while back, JL Raheja et al. [19] employed SVM to classify the feature vector of the retrieved characteristics, including Hue-moments, which are position, angle, and form invariant moments, fingertips, and trajectory tracking. The system used the spatial domain approach to collect characteristics from the fingertips. They achieved an accuracy of 97.5%, which cannot be generalized as they tested their model only on four signs of ISL (A, B, C, and Hello).

Currently, Google [20], [21] is working on sign language recognition system to predict dynamic signs in video conferences. Their current model gives accuracy of 91% with 285 fps which is the fastest in all available methods.The only limitation of system is the client system should have dedicated GPU for faster predictions.

C. Comparison of the proposed method w.r.t. the state of the art

Our proposed method tries to fill the gaps in the currently available methods. Apart from reaching high accuracy, the model works very fast in real-time, using a Random forest classifier, one of the light weight models in the field. It works for different signers, different background, with and without face. It also supports an interactive UI. The model can form complete sentences after recognizing signs and has some custom and previously used default signs for more everyday words so that the user does not have to generate them again. An auto-correct feature has also been added to avoid the model's accuracy getting in the way of making sentences. The model has advanced speech-to-text and text-to-speech recognition to help visually impaired people. Our model has been trained on a mixture of ASL and ISL datasets. It helps to adapt our model to different conditions like one hand or

two hand gestures. Only a few ISL alphabets are added to the majority ASL dataset to make our model more flexible and accurate. Other works till now have not developed models on such hybrid datasets yet as we have performed. We have discussed the details of our approach in the section III.

III. Proposed Framework for Sign Language Interpretation System

Figure 2 shows the steps involved in the vision-based interpreter for the sign language recognition. The approach can be divided into three sub-parts, a)dataset creation, b) model training, and c) UI functionality. For creating dataset, the hand features are extracted using mediapipe from the static images of the ASL dataset. The final dataset is prepared with lettersets having highest accuracy. We trained different classification models as shown in Table II for sign recognition using the prepared dataset and selected the best performing model, random forest for inference. Then the trained model is tested in real time to classify letters. As a part of the UI functionality, the proposed framework also includes advanced features such as auto-correct, speech to text and text to speech conversion to help visually impaired people. This framework can be easily deployable on the web. It does not need any additional OS specific instrumentation.

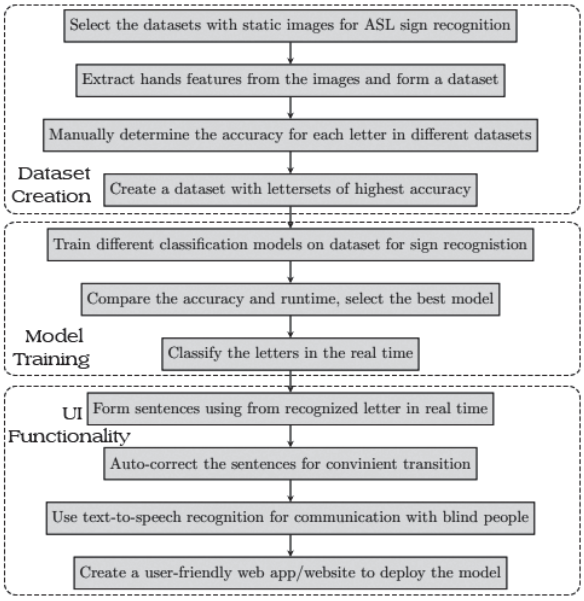

Fig. 2. Flowchart for the proposed research on vision-based sign language interpretation system

IV. Methodology

The approach to carry out this work includes data-set collection, pre-processing data, training data and then testing on real time data. All these processes are explained in detail below. This work extensively uses the concept of mediapipe framework. A brief introduction to Mediapipe framework is given below.

A. Mediapipe

Mediapipe is a highly efficient framework developed by Google that helps in effective tracking of hands and fingers using these 21 3-D landmarks from a single hand. It detects a hand from the real time images and generates a set of 21 points. These points are interconnected with lines that are termed as connections in mediapipe. Figure 5 shows a glimpse of these landmarks and connections. This hand extraction technique is highly effective even with the background noise unlike the conventional techniques. Even if we have our face behind our hand while facing the webcam, then too mediapipe will detect the hands with no serious issues. We just need to tune the minimum detection confidence of mediapipe hands model to obtain good hand detection feature. If the minimum detection confidence is 0.4, then mediapipe will show the landmarks only if it's atleast 40 percent sure that there is a hand in the image. If minimum detection confidence is very low, the mediapipe might detect everything as hands. If the minimum detection confidence is very high, then mediapipe might not detect the hand at all even if the hand is placed in front of the camera. So, we need to keep its value accordingly depending upon the conditions. If the user has a very high-definition webcam, then the value of minimum detection confidence can be kept on a higher side. But, if the user has a low-quality webcam, then the value of minimum detection confidence must be kept low. So, our model works well for any camera resolution. It's independent of camera quality. The values of minimum detection confidence must be tuned at the front end to adjust our model. Still, a good resolution camera will produce better accuracy overall. We have used webcam of laptop which is of average quality and our model worked well on it after adjusting minimum detection confidence value. This work gives access to the user to tune this parameter according to their needs in real time on the web app we've built for this model. Mediapipe helps us to detect correct signs quickly in real time. Since, our main aim is achieving good accuracy on real time data, we'll be testing each alphabet or word 4 times only for calculating accuracy. It's mainly because we want our model to produce correct results within 4 guesses only. If it's producing correct outputs for first 4 inputs in real time, then it will definitely produce accurate results even after 4 results too. So, we decided to test for 4 times only per alphabet/word.

B. Datasets Used

For training the ML model, we have used two ASL datasets from Kaggle. The symbols of a few alphabets in the ASL are very similar, like 'M' and 'N.' So, after training, our model was getting confused in a few alphabets and producing incorrect predictions. To tackle this problem, we tried mixing out two different datasets. It helped us in improving the accuracy of predictions. We even added some ISL alphabets for training that were manually captured using the webcam

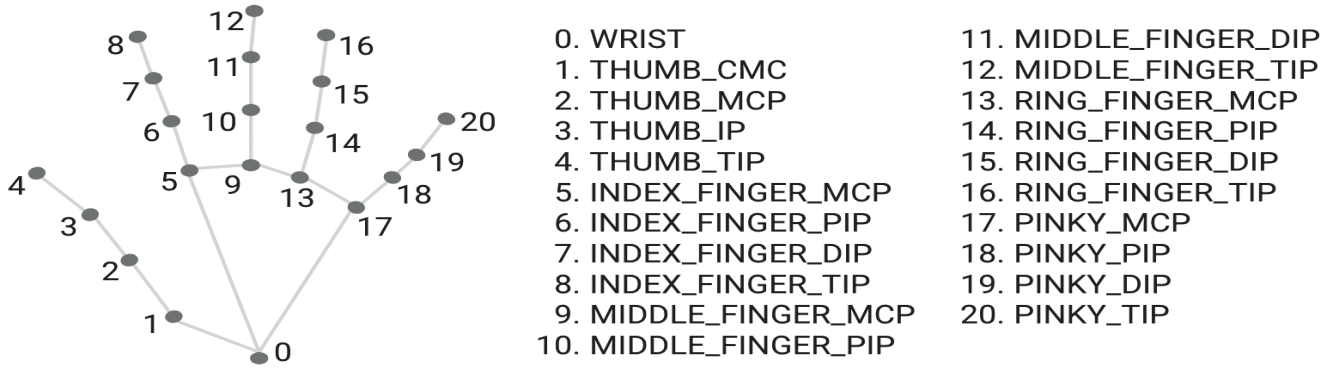

Fig. 3. Mediapipe hand landmarks

for better generalization and adaptibility. This was done to improve accuracy and make the model more flexible for future use. We chose ISL alphabets because they had two hand gestures for some alphabets which ASL was lacking. For training on words, we created a custom dataset while ensuring the ASL words requirements. After shuffling, mixing, and filtering the datasets, we obtained a dataset consisting of 26 alphabets, 7 words, and 3 special symbols (delete, space, and enter). The delete sign is used to remove any letter from the screen. It is helpful to remove any letter in case the model incorrectly predicts it. Space symbol adds a space after any word. It is helpful to form sentences. The 'enter' sign is used to generate audio for the currently spelled words. The dataset preparation step was carried out several times and cross verified with the predictions of the ML model. Finally, the best possible dataset was created that performed well compared to all others in terms of real-time accuracy. The training was performed on this final dataset to generate our final model. The datasets we used were Kaggle datasets named Dataset-1 [22], Dataset-2 [23] and Dataset-3 [24].

C. Data Pre-processing

Once the image is fed, we create a hand extractor using the Mediapipe framework. The images are preprocessed using the Mediapipe framework. The Mediapipe extracts 42 landmarks and their specific connections from the image as shown in Figure 5. Hence, the actual image becomes useless once its features are extracted. But, these features need to be processed before training model over them, because they are useless in their raw/natural form as these are mere co-ordinates indicating different points on hand in a certain 2D frame. Feature derivation involves evaluating angles and the length of fingers. Each finger has three segments like (5,6), (5,7), (5,8) for index finger, where 5 is point on palm and 8 is tip of the finger. Hence we have 15 segments for 5 fingers. But, the thumb has 4 segments (0,1), (0,2), (0,3), (0,4). So, the point 2 is removed as thumb has 3 segments in real. It keeps in maintaining balance and accounting for 15 segments. We keep the longest segment of index finger as our reference segment, as its most volatile. All angles and lengths of other segments will be calculated relative to the reference segment. For measuring the lengths of other segments, we will use ratios. For example, if the length of the reference segment is 1, then the length of the middle finger's largest segment may be 1.4, the length of the thumb's largest segment may be 0.5, and so on. We prefer ratios over absolute length due to a fundamental reason. If the hand is very close to the camera, then the absolute length of fingers will be larger compared to the case where the hand is far from the camera. However, whatever be the distance between hand and camera in real-time, the ratios will always be the same. So, using ratios rather than absolute lengths is better in real-time usage. The same is the reason for using a relative reference line for calculating angles. If the reference for calculating angles is fixed, then the real-time performance will be very rigid. If the user performs the correct sign, but his hand is at a slight angle from the reference line, the model will mispredict it. If we keep the reference line relative, the user can focus on performing correct signs without worrying about the angle. Our final model would give better-generalized results. The ratios and angles are calculated for all fingers (except the index finger, which is the reference) and appended in a pandas dataframe. This process is repeated for the entire dataset, and a final dataframe is created. The dataframe consists of an additional column about the labels, hand(left or right) of the trained image. The first letter of each image name in the dataset tells us the label name. We extract this label using string slicing and store it in the dataframe under the 'label' column, and for hand we flip the same image so that we can get dataset for both hands. So user can use any hand for communicating.The dataframe contains 10 columns named slope0, slope1, slope2, slope3 slope4, dist01, dist02, dist03, dist11, dist12, dist13, dist21, dist22, dist23, dist31, dist32, dist33, dist41, dist42, dist43, slope01, slope02 slope03, slope11, slope12, slope13, slope21, slope22, slope23, slope31, slope32, slope33, slope41, slope42, slope43, hand, label. Where slopei is slope of i^{th} finger, distij is distance ratio of j^{th} segment of i^{th} finger to reference segment, slopeij is slope difference between j^{th} segment of i^{th} finger and reference segment, hand is left,right, label is letter/word the data denotes. For the words and some other letters dataset, where two hands are involved, we create a different dataframe with more columns as the number of

TABLE II. ACCURACY ON DIFFERENT MODELS

Classifier algorithms	Accuracy of one hand dataset(%)	Accuracy of two hand dataset(%)
RFC	90.12	94.58
NBC	86.59	79.74
LRC	82.59	89.95
KNN	82.12	89.15
SVC	80.47	90.91
LwP	72.96	58.45

features is doubled for two hands. We generate 2 CSV files using the two data frames we created for one hand and two hand data. These calculated parameters are finalized for training the machine learning model. This final dataset will be fed to the ML model to train. Our ML algorithm works on this preprocessed data.

D. Machine Learning Model Training

We trained various Machine Learning classifiers, Learning with Prototype (LwP), Logistic regression classifier (LRC), Support-vector machines classifier (SVC), Random forest classifier (RFC), K-nearest neighbour (KNN) and Naive bayes classifier (NBC) with the dataset prepared using steps explained in the previous section. We evaluated the performance of each algorithm using test data. As we need a light-weight(i.e fast working) algorithm we didn't include the neural network algorithms. We tested different hyper-parameters of each of these algorithms and selected the best parameters that works in favor of both being lightweight and classify at better accuracy.

E. Testing phase based on train test split:

For testing the accuracy of different algorithms, we shuffled and split the data in two parts (2/3 - training data, 1/3 - testing data). The algorithm is trained using training data and accuracy is calculated based on prediction of testing data. Table II displays the performance of each algorithm on two datasets (one hand, two hand).

Hence, we implemented Random Forest Classifier (RFC) for classification in our system. The RFC distributes the data in more than one samples and fits the decision tree on these samples and averages them to improve the prediction accuracy. The sample distribution helps in reducing the overfitting effect. The hyper-parameters are tuned and final values set are max_depth=12, min_sample_split=4, random_state=101.

F. Testing phase on real time data

We created a new python file named handextractor.py that performs real-time testing on data. We read the CSV files created in the training phase and store them as data frames. In this phase, OpenCV is used to capture frame-by-frame information of real-time data. For every frame, we first do feature extraction using the Mediapipe hand's package. The landmarks and connections fed from the frame are sent to our model. The model calculates the ratios and angles of different fingers' segments, w.r.t. the index finger's largest segment. The trained algorithm is used to predict the letters for given input. However, using this technique, many words are printed on the screen in a single second. The reason for it is that if OpenCV reads real-time data at 30 fps, then per second, 30 output labels will be generated. It is too fast to handle by the user in real-time. So, we set a threshold value for it. We will print the output label only if we have generated 40 labels of a certain letter. We will store these labels in a dictionary and output that label which has occurred a maximum number of times. So even if our model incorrectly predicts a few times, it will still not affect the overall output. We will print the output generated by the model on the screen itself using the cv2 method putText(). We also have some special signs like 'Space', 'Delete' and 'Enter.' These are not supposed to get printed. We allow them their tasks using if, elif, and else methods. If the sign detected is 'Space,' then append " " to the current string printed on the screen. If the sign detected is 'Delete,' remove the last alphabet of the string printed on the screen. If the sign detected is 'Enter,' feed the word printed on-screen to the gTTS module and generate audio for the currently spelled word. We have also added the autocorrect feature to our model. It helps generate logically correct sentences even if the user misspelled 2 or 3 alphabets wrongly in a word. We have also implemented text to speech module to complete the 2-way communication. This method will take audio input from the normal user and convert it to text that a deaf and mute person could read and understand. A brief overview of all the processes we discussed till now can be inferred from figure 4.

V. RESULTS

This section discusses the user interface and the performance results in terms of accuracy for interpreting signs for all English alphabets and some common words, thank you, please, hello etc.

A. UI for Sign Language Interpretation System

We developed a web application using streamlit. It is a light weight UI framework that eases of deployment of our model in real world applications. In real time, if the user is unable to form correct signs, then 1-2 alphabets of any word may get incorrectly classified by our model. To avoid that, we have added auto-correct feature that predicts the correct word even if 1-2 alphabets are incorrect. The UI has a sidebar having options: About App, Info about Signs, Run on Image, Run on Video and Info about project group members. The "About App" section contains an overview of the sign language communication. In "Info about Signs" section, the user can see a reference table containing all signs of respective images. "Run on Image" section helps to classify letters on static images if we upload them. "Run on Video" section helps to classify letters on static videos if we upload them. To run on real time dynamic gestures, the user has to use "Use webcam" button. It will ask for permission to open the webcam and start detection and word formation in real time. The user can set the values of "maximum number of hands" that the model can detect too. This value is set to 2 by default. The minimum

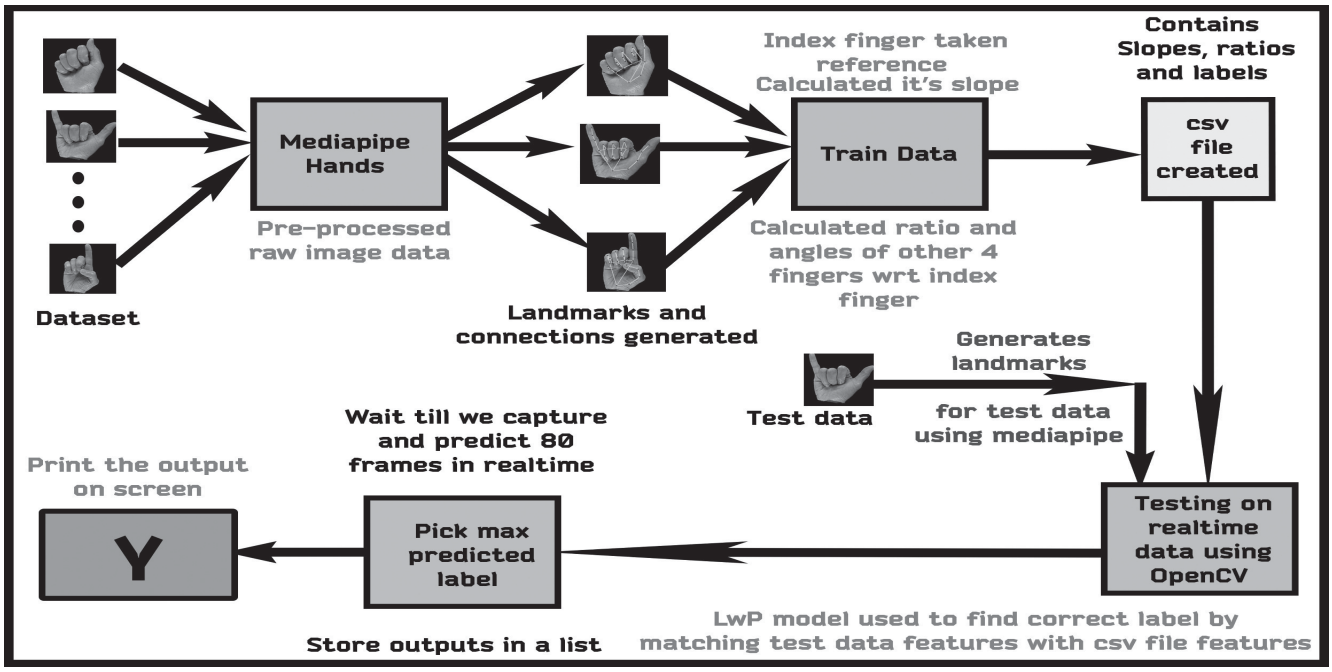

Fig. 4. Basic workflow of the model

detection confidence can also be adjusted by the user in the UI itself. If user puts his hand in front of webcam, then mediapipe will show lines only if its x percent confident that the object is a hand. Here, x is the minimum detection confidence value in percentage representation. The value of minimum detection confidence lies in between 0 and 1. To convert it to percentage representation, we multiply it by 100. If it's value is 0.6 (value in percentage will be 60), then it will show lines only if the model is 60 percent sure that it's a hand. It's value can be adjusted as per video camera quality. If quality of webcam is good, then detection confidence can be set high. Otherwise, it must be kept low. The user has control over it and can modify it according to the requirements.

B. Performance Analysis

All these experiments were performed on a laptop having Intel(R) Core(TM) i3-6100U CPU @ 2.30GHz and 8GB RAM. The proposed method is designed specifically for good performance on real-time data. Conventional methods generally provide excellent accuracy on static data, but their accuracy falls significantly when tested on real-time data. We tested our proposed model on real-time data by testing for each label four times and recording the predicted outputs. Then the accuracy would be calculated as follows:

$$Accuracy = \frac{Number\ of\ signs\ correctly\ predicted}{Total\ number\ of\ predictions\ made}$$

We also calculated the accuracy for each test using the same formula. There will be 5 tests overall. The final accuracy obtained overall is calculated as follows:

$$FinalAccuracy = \frac{\sum_{i=1}^{5} Accuracy(Test\ i)}{4}$$

TABLE III. THE PREDICTION TIME

Frame Rate(fps)	Frame window for testing	Time taken(sec)
30	1	0.033
60	1	0.016
30	10	0.33
60	10	0.16
30	20	0.67
60	20	0.33
30	40	1.33
60	40	0.67
30	80	2.67
60	80	1.33

Table IV shows the accuracy related to each test run.

We obtained an overall accuracy of 94.54 percent on real time data. The accuracy keeps increasing as the number of tests increase. It's so because the user can figure out what features are getting extracted and fed to the model. If he gets any incorrect prediction, then the user can make necessary changes in his sign as he can see what features are being extracted.

LwP works very fast, giving the recognized letter in every frame, making the time to recognize the gesture to be 1/30 (0.033) sec for a 30fps system, 1/60 (0.016) sec for a 60 fps system which is very fast even for real-time. However, as explained in the methodology, we have created a frame window of 80 frames for the words formation and letters printing part. The majority of the classification for letter/word will take for the final output. It will reduce the chances of

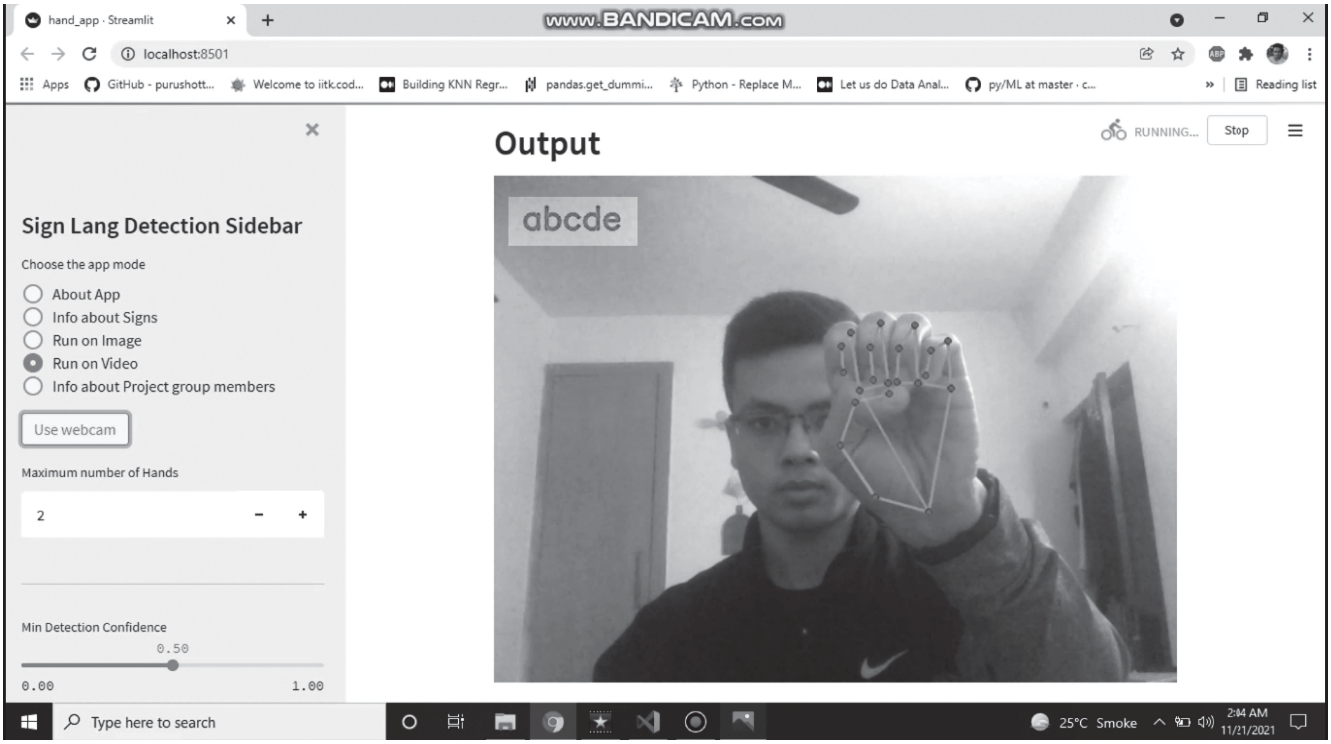

Fig. 5. User Interface

error in the final prediction even if the user has made wrong gestures in some frames. The overall frame window will have the gestures which are correctly classified. Thus, while making the gestures, the user can create wrong gestures for some frames, But eventually, the user will tend to make the fitting gesture. So for this, there should be a time frame for the user to make the right gestures. If the time frame is small, the user will not always make the fitting gesture in that short time frame. Thus it will not be user-friendly. If the time frame is long, the user has to hold the gesture he wants to make for longer. So for this, we have to set the time frame appropriately, i.e., neither less nor more, so that there can be sufficient time in which the user can make the gesture without the effort to hold the hand for long. So for setting this, our testing system's webcam works at 30 fps, i.e., the webcam captures 30 frames per second. We had to set the frame window appropriately, i.e., neither less nor more. We have come up with the length of the frame window as 80 frames for testing our system. For this, we are getting 2.67 secs to make a single gesture. We have tested it for 30 frames per window (1-sec time frame), 50 frames per window (1.67 secs time frame), and 70 frames per window (2.33 secs time frame) also. One can say that there is a trade-off between the accuracy and speed of recognition. However, the good part is that adjusting the frame window entirely depends upon the user. i.e., how fast he/she can make the gestures. Furthermore, as the user gets familiar with all the gestures for the letters/words, he will get used to making the hand signs. Thus, he can set the frame window at a lower value also. (i.e., 30 frames or 50 frames). It depends entirely on the user's ability to make gestures, i.e., how fast he/she can make them. The table related to the same is plotted as Table 3.

VI. Discussion and Future Work

The proposed method performs very well with different users, various backgrounds and even with two hands. It can be further extended by adding complex gestures to the dataset apart from basic alphabets and signs. Since there is no readily available dataset for the complex gestures, creating a corresponding dataset is one of the critical challenges. These gestures will make use of hands, arms, face, and body. Mediapipe pose can be effectively used for the same. Once the new model has been trained, it is required to combine the usage of the Mediapipe pose and the Mediapipe hands and produce a very generalized model. The benefit of including these gestures would be that the user can directly generate that word from a single pose instead of typing alphabets to form words. Nevertheless, it will also have some limitations as all words cannot be included in our model. We will include common words only. If any word is not trained on our model, it can be formed using sign language letters. Further, to create a better front-end mechanism, we plan to build an authentication system too. Users can log in to the portal and carry out video calls using sign language. We also intend to provide users with the option of generating their own custom gestures for quick

TABLE IV. THE LETTER-WISE ACCURACY FOR EACH DATASET IN PERCENT

Word	Test-1	Test-2	Test-3	Test-4	Acc
a	a	a	a	a	**100**
b	b	b	b	b	**100**
c	c	c	c	c	**100**
d	d	d	d	d	**100**
e	e	e	e	e	**100**
f	f	f	f	f	**100**
g	g	g	g	g	**100**
h	h	h	h	h	**100**
i	j	i	i	i	**75**
j	j	j	i	j	**75**
k	k	k	k	k	**100**
l	l	l	l	l	**100**
m	m	m	m	m	**100**
n	m	n	n	n	**75**
o	o	o	o	o	**100**
p	p	p	p	p	**100**
q	q	q	q	q	**100**
r	u	r	r	r	**75**
s	s	s	s	s	**100**
t	m	t	t	t	**75**
u	u	u	u	u	**100**
v	v	k	v	v	**75**
w	w	v	w	w	**75**
x	x	x	x	x	**100**
y	y	y	y	y	**100**
z	z	z	z	z	**100**
hello	hello	hello	hello	hello	**100**
please	please	please	please	please	**100**
thankyou	thankyou	thankyou	thankyou	thankyou	**100**
bathroom	bathroom	bathroom	bathroom	bathroom	**100**
family	family	family	family	family	**100**
friend	friend	friend	friend	friend	**100**
yes	yes	yes	yes	yes	**100**
Acc	**87.87**	**93.93**	**96.96**	**100**	**94.69**

communication once they log in to the site. It would be an excellent application for the public to use.

VII. CONCLUSION

In this paper, we proposed a light weight method for Sign Language recognition using Mediapipe and random forest classifier trained on American Sign Language. The proposed work successfully performed very well on real-time data attaining an accuracy score of 94.69 %. Conventional methods using a computer vision-based approach are unable to obtain such high accuracy on real-time data. Some of them can achieve such accuracy but are deeply affected by background noise. The proposed method performs well and is unaffected by background noise. The user can even have many different things in the frame; still, the Mediapipe effectively detects the hands. Also, the web-based user interface is a significant improvement here. The user can look at what features are being extracted from their hands in real-time using the Mediapipe. It helps the user to correct their mistakes while creating hand signs. It also has an autocorrect feature enabled along with speech to text and text to speech conversion. The user can adjust the values of minimum detection confidence of the Mediapipe hands using the frontend slider option. In the UI, the user has three options to test our model. The user can upload any picture, upload any video, or test on real-time data. The light weight models used in the proposed method makes it suitable for the real time recognition systems.

REFERENCES

[1] W. Sandler and D. Lillo-Martin, *Sign language and linguistic universals*. Cambridge University Press, 2006.

[2] H.-D. Yang, "Sign language recognition with the kinect sensor based on conditional random fields," *Sensors*, vol. 15, no. 1, pp. 135–147, 2015.

[3] B. Kshitij and Y. Xie, "American sign language recognition using deep learning and computer vision," pp. 4896–4899, 12 2018.

[4] L. Rioux-Maldague and P. Giguère, "Sign language fingerspelling classification from depth and color images using a deep belief network," pp. 92–97, 05 2014.

[5] B. Duy Khuat, D. Thai Phung, H. Thi Thu Pham, A. Ngoc Bui, and S. Tung Ngo, "Vietnamese sign language detection using mediapipe," (New York, NY, USA), Association for Computing Machinery, 2021.

[6] S. A. Mehdi and Y. N. Khan, "Sign language recognition using sensor gloves," pp. 2204 – 2206 vol.5, 12 2002.

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

[8] J. Wu, L. Sun, and R. Jafari, "A wearable system for recognizing american sign language in real-time using imu and surface emg sensors," *IEEE Journal of Biomedical and Health Informatics*, vol. 20, no. 5, pp. 1281–1290, 2016.

[9] P. Paudyal, J. Lee, A. Banerjee, and S. K. Gupta, "A comparison of techniques for sign language alphabet recognition using armband wearables," *ACM Transactions on Interactive Intelligent Systems (TiiS)*, vol. 9, no. 2-3, pp. 1–26, 2019.

[10] J. J. Bird, "Statistical and spatio-temporal hand gesture features for sign language recognition using the leap motion sensor," *arXiv preprint arXiv:2202.11005*, 2022.

[11] S. Sharma, K. Kumar, and N. Singh, "Deep eigen space based asl recognition system," *IETE Journal of Research*, pp. 1–11, 07 2020.

[12] H. Wang, X. Chai, Y. Zhou, and X. Chen, "Fast sign language recognition benefited from low rank approximation," 07 2015.

[13] O. Koller, S. Zargaran, H. Ney, and R. Bowden, "Deep sign: Hybrid cnn-hmm for continuous sign language recognition," 09 2016.

[14] B. Garcia and S. A. Viesca, "Real-time american sign language recognition with convolutional neural networks," *Convolutional Neural Networks for Visual Recognition*, vol. 2, pp. 225–232, 2016.

[15] L. Pigou, S. Dieleman, P.-J. Kindermans, and B. Schrauwen, "Sign language recognition using convolutional neural networks," vol. 8925, pp. 572–578, 03 2015.

[16] B. Kang, S. Tripathi, and T. Nguyen, "Real-time sign language fingerspelling recognition using convolutional neural networks from depth map," pp. 136–140, 11 2015.

[17] S. S and D. S, "American sign language recognition system: An optimal approach," *International Journal of Image, Graphics and Signal Processing*, vol. 10, 08 2018.

[18] P. Kishore, M. Prasad, P. Ch.Raghava, and R. Rahul, "4-camera model for sign language recognition using elliptical fourier descriptors and ann," 01 2015.

[19] J. Raheja, A. Mishra, and A. Chaudhary, "Indian sign language recognition using svm," *Pattern Recognition and Image Analysis*, vol. 26, 04 2016.

[20] C. Linder, "How google is making sign language more accessible in video calls." https://www.popularmechanics.com/technology/design/a28981613, 2021.

[21] A. Moryossef, I. Tsochantaridis, R. Y. Aharoni, S. Ebling, and S. Narayanan, "Real-time sign language detection using human pose estimation," 2020. https://research.google/pubs/pub49425/.

[22] A. Thakur, "american sign language dataset kaggle." https://www.kaggle.com/ayuraj/american-sign-language-dataset, 2018.

[23] Akash, "Asl alphabet, image data set for alphabets in the american sign language kaggle." https://www.kaggle.com/grassknoted/asl-alphabet, 2017.

[24] D. Rasband, "Asl alphabet images with a variety of backgrounds for validating a model." https://www.kaggle.com/danrasband/asl-alphabet-test, 2017.