Sign Language Recognition

Draft Paper

Hritika Doshi - A010
Jitesh Mishra - A020
Romil Shah - A030
Vasav Patel- A044

Faculty Mentor: Harsh Taneja

Abstract

Sign language is one of the oldest and most natural forms of communication, but since most people do not know sign language and interpreters are extremely difficult to find, we have developed a real-time neural network-based fingerspelling system for Indian sign language. In our method, the hand is first processed by a filter and then by a classifier that predicts the class of the hand gestures. Our method has an accuracy of 98.24% for the 26 letters of the alphabet.

Introduction

Sign language is a complete convoluted language based on computer vision that engrosses signs formed by hand movements in conjunction with facial expressions. It is a natural language used for communication by people who have poor or no hearing. A sign language can be used to communicate letters, words, or sentences by using various hand gestures. This type of communication allows hearing-impaired people to express themselves more easily and helps to bridge the communication gap between hearing-impaired people and others. Much research has recently been conducted in order to develop systems capable of classifying signs from various sign languages. Such systems have found use in games, virtual reality environments, robot controls, and natural language communications. The automatic recognition of human signs is a difficult multidisciplinary problem that has yet to be solved. Several approaches involving the use of machine learning techniques have been used in recent years for sign language recognition. There have been attempts to recognise human signs since the advent of deep learning techniques.

Deep network training is typically done layer by layer and is based on more distributed features found in the human visual cortex. The abstract features from the collected signs in the first layer are grouped into primary features in the second layer, which are then combined into more defined features in the following layer. These features are then combined into more engrossing features in the following layers, which aid in the recognition of various signs.

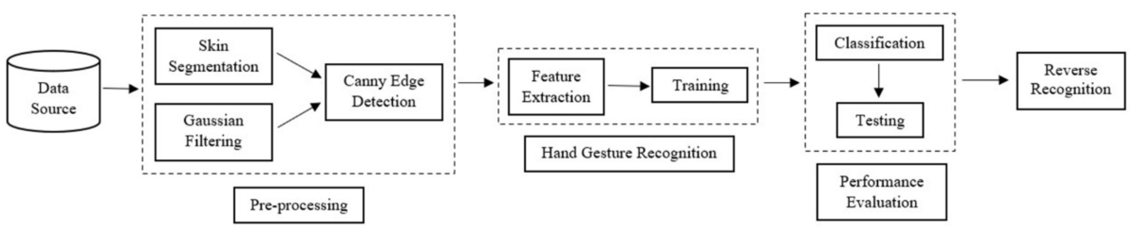
The majority of deep learning-based sign language recognition research is done on sign languages other than Indian Sign Language. This area has recently gained popularity among research experts. The earliest work on sign language recognition was based primarily on machine learning techniques. The idea is to learn a set of features from raw data that can be used in sign language recognition automatically. By learning as a set of features automatically, it avoids the manual process of handcrafted feature engineering.

The goal of this paper is to recognise alphabets in Indian Sign Language by using the corresponding gesture. The identification of gestures and sign languages is a well-researched subject in American Sign Language, but it has gotten little attention in Indian Sign Language. We want to solve this problem, but instead of using high-end technologies like gloves or the Kinect, we want to recognise gestures from photographs (which can be accessed via a webcam) and then use computer vision and machine learning techniques to extract and classify specific features.

Literature survey

The literature survey in this paper focuses on the research done on conversion of various sign languages to text and the methodology that they have used make better, accurate and precise prediction.

Many advances have made in the field of sign language recognition initially various studies were of the mind that for effective translation useres might have to wear gloves for computer to analyse the sign made by the user. However, V. Adithya et al [1] propose a vision based approach for the recognition of fingerspelling in Indian sign language. It uses digital image processing techniques and feed forward neural networks for recognizing different signs. It uses skin color segmentation for identification of hand in frame. It proposes a distance transformation method using euclidean distance for feature transformation and this is used to separate the hand from background. Making advances on this theory Shagun Katoch et al [2] proposed to have built and used a custom data set for effectively classification and recognition of sign language. The image below describes the flow of data in the model.



They have manually recorded signs of alphabets ranging from A-Z. Moreover numerical signs from 0 to 9 were also included from three different individuals whilst they found that position of camera is very crucial for quality of pictures and noise elimination. For identification of hands they have adopted two options for capturing images, one is basic methodology viz performing skin segmentation on image. The second method uses the philosophy of moving averages and takes the first 30 frames of a video as background and the difference in following frames is considered to be foreground. For feature extraction authors present the usage of Bag of Visual Words inspired by BOW from NLP.

Nevertheless, here instead of keywords they have used a 64 member vector representing features of interest. Furthermore to better recognize the hand gesture with the help of computer vision Ashish S. et al [3] presents a real time system for recognition of hand gesture on basis of detection of shape based features like orientation, the centroid of the mass, the condition of the fingers, and the thumb in relation to the elevated or folded fingers of the hand. With the help of contour analysis they considered vectors value and found it’s ease to recognize hand gestures irrespective of scaling and shape. Thus coming to conclusion, that recognition can be done by matching templates of hand by considering Contour curve shapes.

Along with efficient ways to recognize the hand gestures some researchers also focused on making an effecive dataset to enhance the accuracy and precision of the sign language recognization systems. Sharvani Srivastava et el [4] presents a technique for generating an Indian Sign Language dataset using a camera, followed by the training of a TensorFlow model utilising transfer learning to provide a real-time Sign Language Recognition system. There are two types of SLR systems, both continuous and isolated. Camera is the primary input device used in SLR systems. The system is trained to recognise a single gesture in isolated SLR. Each image is identified as standing for a letter of the alphabet, a number, or a particular motion. In contrast to single gesture classification, continuous SLR is continuous. Instead of just one gesture, the technology can recognise and translate entire sentences in continuous Flow. Despite the small size of the dataset, the system still performs well. Python and OpenCV are used to capture photos from a camera for data collection. In this study, TensorFlow's object identification API has been used to translate Indian sign language motions into commonly used language. The alphabet dataset for Indian Sign Language was used to train the system. The technology instantly recognises sign language. Python and OpenCV have been used to capture images from a camera for data acquisition, which lowers the cost. The created system displays an 85.45% average confidence level. Although the system has a high average confidence rate, its training dataset is tiny and has certain limitations. It is simple to create, train, and use an object detection model thanks to the open-source framework

TensorFlow object detection API. Their system, the TensorFlow detection model zoo, provides a number of detection models that have already been pre-trained using the COCO 2017 dataset. SSD MobileNet v2 320x320 is the pre-trained TensorFlow model that is being used. Using training images scaled to 320x320, the SSD MobileNet v2 Object identification model is integrated with the FPN-lite feature extractor, shared box predictor, and focal loss. The SSD MobileNet v2 Object detection model is combined with the FPN-lite feature extractor, shared box predictor, and focal loss with training images scaled to 320x320. A very unique and substantial development in creating of dataset was made by Samuel Albanie et al [5] with their BSL-1K dataset The general methodology for visually recognizing SLR is identifying letters then words and then sentences however it creates a large and very complex data set. Moreover it was identified by the authors that in real life application sign language translators usually mouth some keywords along with signing them for effective understanding. However many researchers have forgotten to include the factor in dataset creation. The authors have used television shows with signers broadcasted on BBC Network for keyword spotting in order to map them with sign language and mouthing actions using subtitles. Since signers do not always mouth along with sign language and there is high probability of latency in both we cannot use visual lip reading. The effective way is to identify keywords in subtitles then extend the time frame of video by 4 seconds on either side in order to identify the mouthing action among various translators and can pinpoint the mouthing action for that keyword. The annotated keywords were then manually verified by BSL signers and right annotations comprising of 2,103 annotations covering 334 signs from the 1,064 sign vocabulary was kept as dataset. They have proposed a very unique dataset however due to it’s size the computational power required to use this dataset is not easily available and is not implemented by many.

The third pillar of researchers in this field are the ones who apart from exloring the ways to optimize dataset or gester recognization methodology or hardware are more focused towards an effective algorithm. Kshitij Bantupalli et al [6] have used Hidden Markov Models (HMM) which is a Machine Learning algorithm to recognize facial expressions from various video sequences combined with two other algorithms. They are Bayesian Network Classifiers & Gaussian Tree Augmented Naive Bayes Classifiers. The research paper methodology is to do frame extraction from video data, pre-processing the data then extracting key frames from the data followed by extracting other features, in the end recognition and optimization. Here, pre-processing was done by converting the video to a sequence of RGB frames. The CNN model employed in this system architecture was Inception, a model created by Google for picture recognition and largely recognised as the most effective image recognition neural network currently in use.

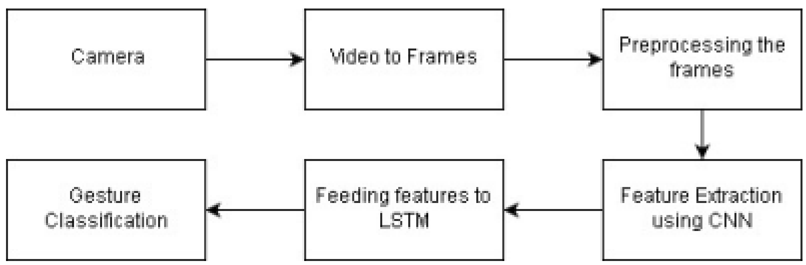


Fig. 3. High Level System Architecture.

In this paper, the author suggests us to use Capsule Networks for improvements instead of Inception methods regarding CNN model. Sarfaraz Masood et al [7] presents their findings were two approaches were used to train the model on the temporal and the spatial features i.e. Prediction Approach and Pool Layer Approach. In Prediction Approach, spatial features for individual frames were extracted using inception model (CNN) and temporal features using RNN. Each video was then CNN's predictions for each of their distinct frames are shown in a list. The RNN received this as input. To avoid the model learning to recognise specific colours, frames from each video matching to each gesture were selected, and backdrop body parts other than the hand were eliminated to obtain a grayscale representation of the hands. Frames from the practise set were provided to CNN.to get a grayscale image of hands which avoided color-specific learning of the model. Frames of the training set were given to the CNN model for training on the spatial features. The obtained model was then used to make and store predictions for the frames of the training and test data. The LSTM RNN model was then trained on the temporal features using the predictions corresponding to the frames of the training data. In Pool Layer Approach CNN was used to train the model on the spatial features and passed the pool layer output to the RNN before it is made into a prediction. The pool layer provides a 2048-dimensional vector, but no class, that depicts the intricate details of the image.prediction. Rest of the steps of this approach are same as that of first approach. Both approaches only differ in terms of input given to the RNN. The dataset used for both the approaches consists of Argentinean Sign Language (LSA) gestures, with around 2300 videos for 46 gestures. The prediction approach got an accuracy of 80.87% by recognizing 370 gestures correctly a test set of 460, while the pool layer approach scored 95.21% by recognizing 438 gestures. While most researchers have focused on webcam of computers Mehreen Hurroo et al [8] have focused on mobile computer vision. Moreover they have used MobileNet which is a pre-trained lightweight CNN model. Moreover they have used TensorFlow API for object detection techniques. The data set is collected by different AzSL signers on telegram. However it resulted in poor performance because of usage of pre-trained CNN method. Moreover, using CNN Rachana Patil et al [9] performed an analysis using a web camera, and the CamShift Algorithm was utilised to identify and display real-time hand gestures. CNN was educated to recognise the guesture, and in the end the model was successful. The model has reached an accuracy level of approximately 95%.

Sr no.	Existing Work	Data Used	Algorithm Used	Performance Measure	Best Algorithm	Issue Addressed
1	G.Anantha Rao et al [10]	Real time hand gestures	CNN	Accuracy (92.88%)	CNN	Recognize and detect Sign Language from Hand Gestures
2	Sanket Bankar et al [11]	Hand Gestures from Real time videos	CNN & YOLOv5	Accuracy (88.4%)	YOLOv5	Recognize Sign Language from trained Dataset
3	MUHAMMAD AL-QURISHI et al[12]	Hand Gestures and still images	CNN & SVM,LSTM	Accuracy	CNN	Recognize and detect Sign Language
4	Sharvani Srivastava et al [4]	Images by Web Cam	Tensor Flow and OpenCV	Accuracy (85.45%)	Tensor Flow and OpenCV	Recognize and detect Sign Language
5	Mehreen Hurroo et al [8]	Hand Gesture Images	CNN	Accuracy (90%)	CNN	Recognize and detect Sign Language
6	Lionel Pigou, et al [13]	Hand Gestures	CNN and GPU acceleration	Accuracy	CNN	Recognize and detect Sign Language

The table lists six existing works or studies that focus on recognizing and detecting sign language from hand gestures. The first study used real-time hand gestures, the second used real-time videos, and the third used hand gestures and still images. The primary issue addressed was recognizing and detecting sign language from hand gestures. The study used CNN, SVM, LSTM, Tensor Flow, OpenCV, hand gesture images, and GPU acceleration to recognize and detect sign language. The performance measure used was accuracy reaching upto 92.88%, and the best algorithm identified was the CNN.

Proposed Work

The system employs a vision-based methodology. The issue of using any artificial technology for interaction is eliminated because all of the indications are represented with just the naked hands.

Generation of Data Sets

We looked for pre-made datasets for the project but were unable in finding any that were in the form of raw photos and met our specifications. The datasets in the form of RGB values were the only ones we could locate. Hence, we made the decision to compile our own data set. The procedures we used to produce our data set are listed below.To create our dataset, we utilised the Open Computer Vision (OpenCV) library.

Initially, for training purposes, we took around 100 photographs of each ISL symbol.We begin by taking a picture of each frame produced by our machine's webcam. As seen in the image below, each frame has a region of interest (ROI) that is indicated by a blue-bounded square.

We took our ROI, which is RGB, and turned it into a grayscale image, as seen below, by extracting it from the entire image.

In order to extract different aspects from our image, we then apply our gaussian blur filter to it. Below is the resultant image after applying gaussian blur.

GESTURE CLASSIFICATION

The approach which we used for this project is:

The strategy we employed for this project is as follows: To forecast the user's final symbol, our strategy employs two levels of algorithm.

Algorithm Layer 1:

1. To obtain the processed image after feature extraction, apply the gaussian blur filter and threshold to the frame captured with opencv.
2. The CNN model is given this processed image for prediction, and if a letter is found in more than 50 frames, it is printed and taken into account while creating the word.
3. With the blank symbol, the space between the words is taken into account.

Algorithm Layer 2:

1. We identify alternative sets of symbols that produce comparable outcomes when recognised using the second algorithmic layer.

2. Using classifiers designed specifically for those sets, we then categorise between those sets.

Layer 1: CNN

Model :

1. First Convolution Layer: The input image has a 128x128 pixel resolution. It is first processed using 32 filter weights in the first convolutional layer (3x3 pixels each). A 126X126 pixel image, one for each of the Filter-weights, will be produced as a consequence.
2. First Pooling Layer: The images are downsampled using maximum 2x2 pooling, meaning the highest value in each 2x2 square of the array is retained. As a result, our image has been downscaled to 63x63 pixels.
3. Second Convolution Layer: The 63 x 63 pixels from the first pooling layer's output are now used as the input for the second convolution layer.

   32 filter weights are used in the second convolutional layer of processing (3x3 pixels each). A 60 by 60 pixel image will be produced as a consequence.
4. Second Pooling Layer: The second pooling layer reduces the output images to a resolution of 30 x 30 with a maximum pool size of 2x2.
5. First Densely Connected Layer: The output of the second convolutional layer is reshaped into an array of 30x30x32 = 28800 values, and these images are now utilised as an input to a completely connected layer with 128 neurons. This layer receives a 28800 value array as input. The second densely connected layer receives the output of these layers. To prevent overfitting, we are utilising a dropout layer with a value of 0.5.
6. Second Densely Connected Layer: The output from the 1st Densely Connected Layer is now sent into a layer with 96 neurons that is fully connected.
7. Final layer: The second densely connected layer's output feeds into the final layer, which has as many neurons as classes being classified (alphabets plus a blank sign).

Activation Function:

Rectified Linear Unit (ReLu) has been employed as the activation function in each layer (convolutional as well as fully connected neurons). Max(x,0) is calculated by ReLu for each input pixel. This gives the formula nonlinearity and aids in learning more intricate features. By cutting down on computing time, it aids in removing the vanishing gradient problem and expediting training.

Pooling Layer :

Using the relu activation function and a pool size of (2, 2), we perform Max pooling to the input image. This lowers the number of parameters, which lowers the cost of computing and lowers overfitting.

Dropout Layers:

The overfitting issue occurs when the weights of the network are so closely tuned to the training examples that they get after training that they perform poorly when given fresh instances. A random set of activations in that layer are "dropped out" by this layer by being set to zero. Even if some activations are dropped out, the network ought to be able to deliver the correct categorization.

Optimizer:

To update the model in response to the loss function's output, we employed the Adam optimizer. Adam combines the benefits of adaptive gradient algorithm (ADA GRAD) and root mean square propagation, two extensions of two stochastic gradient descent techniques (RMSProp).

Layer 2:

We are using two layers of algorithms to verify and predict

To come as near to accurately identifying the symbol displayed, we use two layers of algorithms to forecast and validate symbols that are more similar to one another. During our testing, we discovered that the following symbols weren't displaying correctly and were also providing other symbols:

1. For D : R and U
2. For U : D and R
3. For I : T, D, K and I
4. For S : M and N

Hence, in order to categorise these sets in the aforementioned scenarios, we created three distinct classifiers:

1. {D,R,U}
2. {T,K,D,I}
3. {S,M,N}

Finger spelling sentence formation

Implementation:

1. We display the letter and add it to the current string whenever the count of a letter detected exceeds a certain number and no other letter is within a certain distance of it (In our code we kept the value as 50 and difference threshold as 20).
2. If not, we delete the current dictionary, which records the number of times the current symbol has been detected, to reduce the likelihood that the projected letter would be

incorrect.

3. If the current buffer is empty and the number of blanks (plain background) detected exceeds a certain value, no spaces are recognised.
4. In the other scenario, it prints a space to indicate the end of the word and appends the current to the sentence below.

Training and Testing:

In order to reduce extra noise, we turn our RGB input photos into grayscale and apply gaussian blur. To separate our hand from the backdrop, we use adaptive threshold. Afterwards, we resize our photos to 128×128.

We run all the aforementioned procedures on the input photos before preprocessing them and feeding them into our model for training and testing. The prediction layer calculates the likelihood that the image will belong to a particular class. In order for the output to be normalised between 0 and 1, each class's sum of values must equal 1. We did this by utilising the softmax function. The prediction layer's output will initially deviate somewhat from the true value. We used labelled data to train the networks to improve the system. A performance metric utilised in the categorization is cross-entropy. It is a continuous function that is zero precisely when it equals the labelled value and positive at values that are not the same as the labelled value. In order to maximise the cross-entropy, we reduced it as much as possible. We modify the weights of our neural networks at our network layer to achieve this. The cross entropy can be calculated using a built-in feature of TensorFlow.

Since we discovered the cross entropy function, we used gradient descent to optimise it; in fact, the best gradient descent optimizer is known as Adam Optimizer.

Result & Discussion

Result:

The sign language detection system achieved a high level of accuracy for static images with an optimal accuracy of 98.24% and an average accuracy of 98.01%. These results indicate that the system can successfully recognize and detect sign language from static images with a high level of precision. The high accuracy achieved by the system suggests that it has the potential to be a valuable tool in aiding individuals with hearing or speech impairments in communicating with others.

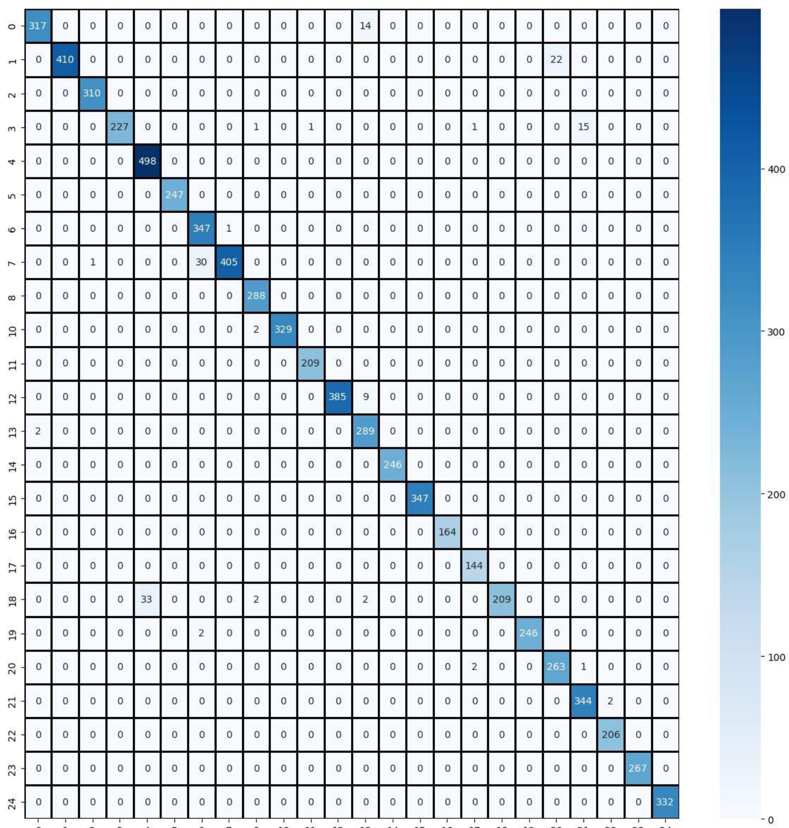


Fig 1. Confusion Matrix

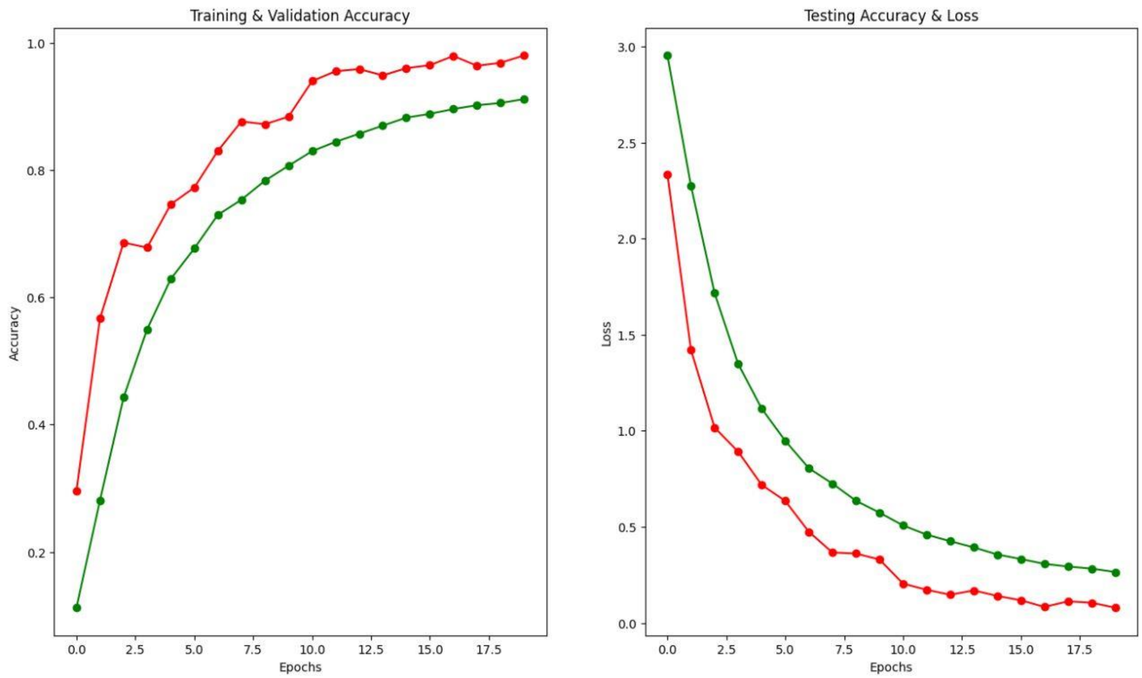


Fig 2. Accuracy and Precision Graph

The sign language detection system's performance metrics, including confusion matrix(Fig 1) , accuracy, and precision, were calculated using scikit-learn Python library. The scikit-learn library provides several functions in Python to evaluate machine learning models' performance, including confusion matrix, accuracy, and precision.(Fig 2). The confusion matrix function returns a matrix that shows the number of true positives, false positives, true negatives, and false negatives for your model's predictions. The accuracy score function returns the accuracy of your model's predictions, which is the proportion of correct predictions. The precision score function returns the precision of your model's predictions, which is the proportion of true positives among all positive predictions.

```
225/225 [==============================] - 3s 13ms/step
              precision    recall  f1-score   support

           0       0.99      0.96      0.98       331
           1       1.00      0.95      0.97       432
           2       1.00      1.00      1.00       310
           3       1.00      0.93      0.96       245
           4       0.94      1.00      0.97       498
           5       1.00      1.00      1.00       247
           6       0.92      1.00      0.95       348
           7       1.00      0.93      0.96       436
           8       0.98      1.00      0.99       288
           9       1.00      0.99      1.00       331
          10       1.00      1.00      1.00       209
          11       1.00      0.98      0.99       394
          12       0.92      0.99      0.96       291
          13       1.00      1.00      1.00       246
          14       1.00      1.00      1.00       347
          15       1.00      1.00      1.00       164
          16       0.98      1.00      0.99       144
          17       1.00      0.85      0.92       246
          18       1.00      0.99      1.00       248
          19       0.92      0.99      0.95       266
          20       0.96      0.99      0.97       346
          21       0.99      1.00      1.00       206
          22       1.00      1.00      1.00       267
          23       1.00      1.00      1.00       332

    accuracy                           0.98      7172
   macro avg       0.98      0.98      0.98      7172
weighted avg       0.98      0.98      0.98      7172
```

Fig 1.

The results portray performance of model through these parameters: accuracy, precision, recall & F1-Score for every characters as seen above.

Discussion:

The base paper G.Anantha Rao et al [10] discusses sign language recognition (SLR) as an evolving research area in computer vision, and describes the challenges involved in SLR, including video trimming, sign extraction, modelling of the sign video background, representation of the sign feature, and categorization of the sign. They also describe the creation of a benchmark dataset for Indian sign language and the use of a visual attention-based framework to extract information frames from the input video. Our paper discusses the goal of recognizing alphabets in Indian sign language using corresponding gestures. It mentions that while the identification of gestures and sign languages is a well-studied topic in American sign language, it has received little attention in Indian sign language.

G.Anantha Rao et al [10] describes a model of Convolutional Neural Network (CNN) applied to a sign language database for classification. The authors created a dataset of 200 Indian sign language words performed by 5 different signers in 5 different orientations. The dataset was pre-processed and used to train the proposed CNN architecture. The paper presents the results of the training in three batches, and provides visualizations of the feature maps obtained at different layers of the CNN. The text focuses on the technical details of the model and its training process, and its goal is to demonstrate the effectiveness of the proposed CNN architecture for recognizing sign language. The base paper is more technical and focused on the details of the model and its training process.

Our paper describes a sign language detection system's performance in recognizing and detecting sign language which are already pre-trained. The system achieved a high level of accuracy which is 98.24% and has the potential to be a valuable tool in aiding individuals with hearing or speech impairments. The text presents the performance metrics of the system, including the confusion matrix, accuracy, and precision, and explains how they were calculated using the scikit-learn Python library. The text focuses on the system's performance evaluation and its potential applications in assisting individuals with hearing or speech impairments. Our paper is more focused on the system's performance evaluation and its potential applications. It also acknowledges the potential limitations of the system's performance in different datasets or real-time scenarios.

Conclusion:

In this report, a functional vision based Indian sign language recognition for D&M people have been developed for ISL alphabets. We achieved final accuracy of 98.24% on our dataset. We are able to improve our prediction after implementing two layers of algorithms in which we verify and predict symbols which are more similar to each other. This way we are able to detect almost all the symbols provided that they are shown properly, there is no noise in the background and lighting is adequate.

Future Scope:

We are planning to achieve higher accuracy even in case of complex backgrounds by trying out various background subtraction algorithms. We are also thinking of improving the preprocessing to predict gestures in low light conditions with a higher accuracy.

References

1. V. Adithya, P. R. Vinod and U. Gopalakrishnan, "Artificial neural network based method for Indian sign language recognition," 2013 IEEE Conference on Information & Communication Technologies, Thuckalay, India, 2013, pp. 1080-1085, doi: 10.1109/CICT.2013.6558259.
2. Katoch, Shagun, Varsha Singh, and Uma Shanker Tiwary. "Indian Sign Language recognition system using SURF with SVM and CNN." *Array* 14 (2022): 100141.
3. Nikam, Ashish S., and Aarti G. Ambekar. "Sign language recognition using image based hand gesture recognition techniques." *2016 online international conference on green engineering and technologies (IC-GET)*. IEEE, 2016.
4. Srivastava, Sharvani, et al. "Sign language recognition system using TensorFlow object detection API." *Advanced Network Technologies and Intelligent Computing: First International Conference, ANTIC 2021, Varanasi, India, December 17–18, 2021, Proceedings*. Cham: Springer International Publishing, 2022.
5. Albanie, Samuel, et al. "BSL-1K: Scaling up co-articulated sign language recognition using mouthing cues." *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XI 16*. Springer International Publishing, 2020.
   Bantupalli, Kshitij, and Ying Xie. "American sign language recognition using deep learning and computer vision." *2018 IEEE International Conference on Big Data (Big Data)*. IEEE, 2018.
6. Masood, Sarfaraz, et al. "Real-time sign language gesture (word) recognition from video sequences using CNN and RNN." *Intelligent Engineering Informatics: Proceedings of the 6th International Conference on FICTA*. Springer Singapore, 2018.
7. *Mehreen Hurroo , Mohammad Elham, 2020, Sign Language Recognition System using Convolutional Neural Network and Computer Vision, INTERNATIONAL JOURNAL OF ENGINEERING RESEARCH & TECHNOLOGY (IJERT) Volume 09, Issue 12 (December 2020).*
8. Patil, Rachana, et al. "Indian sign language recognition using convolutional neural network." *ITM Web of Conferences*. Vol. 40. EDP Sciences, 2021.
9. Rao, G. Anantha, et al. "Deep convolutional neural networks for sign language recognition." *2018 conference on signal processing and communication engineering systems (SPACES)*. IEEE, 2018.
10. Bankar, Sanket, et al. "Real Time Sign Language Recognition Using Deep Learning." (2022).
11. Al-Qurishi, Muhammad, Thariq Khalid, and Riad Souissi. "Deep learning for sign language recognition: Current techniques, benchmarks, and open issues." *IEEE Access* 9 (2021): 126917-126951.
12. Pigou, Lionel, Sander Dieleman, Pieter-Jan Kindermans, and Benjamin Schrauwen.: "Sign language recognition using convolutional neural networks", In Workshop at the European Conference on Computer Vision 2014, pp. 572–578. Springer International Publishing.