Indian Sign Language Communication System

Ashish P [1]***, Madhura K***[1]***, Piyusha M***[1]***, Ritu R***[1]***, Shubham G***[1]

[1]*Student, Department of Computer Science and Engineering, Government College of Engineering Nagpur (GCOEN), Nagpur, Maharashtra, India.*

Corresponding Author: ashishpatel9119@hotmail.com

Abstract: - The main aim of this paper is overcoming the communication barrier between the auditory and profoundly deaf individuals. A lot of research has been done in the related field of American Sign Language (ASL), but unfortunately the same cannot be said for ISL. We used the INCLUDE dataset for the training and testing of our trained model. INCLUDE is recorded with the help of experienced signers to give close resemblance to natural conditions. Sign language recognition is a difficult problem if we consider all the possible combinations of gestures that a system of this kind needs to understand and translate. The best way to solve this problem is to split it into simpler problems, and the system presented here would correspond to a possible solution to one of them. We aim to create a sign detector, which detects numbers from 1 to 10 that can very easily be extended to cover a vast multitude of other signs and hand gestures including the alphabets.

Key Words: —*Communication, American Sign Language, sign detector.*

I. INTRODUCTION

A gesture is a pattern which may be static, dynamic or both, and is a form of nonverbal communication in which bodily motions convey data. Communication is an essential aspect when it comes to share or express information, feelings, and it brings people closer to each other with better understanding. When it comes to disabled persons for example deaf and dumb people, it becomes tougher for them to communicate using natural language. So, they use sign language to communicate with themselves and with entire world. But normal people find it hard to understand sign language as they do not have mostly any prior education or experience in this. Sign language is composed of visual gestures and signs, which are used by deaf and mute for their talking. It is a well-structured code gesture where every sign has a specific meaning allotted to it. These signs are not only used for alphabets or numeric but also for common expressions also for example greetings and sentences Communication is the way information is shared.

Manuscript revised April 15, 2022; accepted April 16, 2022.
Date of publication April 17, 2022.

This paper available online at www.ijprse.com

ISSN (Online): 2582-7898; SJIF: 5.59

It helps to understand people better by removing misunderstanding and creating clarity of thoughts and expression.

Profoundly deaf individuals communicate among themselves using sign language. But they find it difficult while communicating with normal people and most of the regular people not able to understand their sign language. Large work has been done on American sign language recognition, but Indian sign language differs significantly from American sign language. ISL uses two hands for communicating, whereas ASL uses a single hand for communicating. Using both hands often lead to obscurity of features due to overlapping of hands. A few research works have been published regarding Indian Sign Language (ISL). But instead of using high-end technology like gloves or Kinect, we goal to solve this problem using state of the art computer vision and machine learning algorithms.

This project goal is to take the simple step in connecting the social and communication bridge between regular people and the disabled people with the help of Indian Sign Language. A s our project only deals with alphabets and numeric in ISL, it can be extended to common expressions and also words which can be more effective for disabled and normal people in communication and understanding. As we live in a century where India is developing at a quick pace in terms of digital and technological advances, this project could be one of the steppingstones where technology meets humanity and help the hearing impaired and mute community.

II. Related Work

In earlier work on Double hand sign recognition, Agrawal et al. [1] proposed a two-stage recognition approach for 23 alphabets. Signs are performed by wearing red gloves on both hands for better segmentation. The segmented frames served as an input to the feature extraction and recognition phase. In stage 1, the hand features are extracted which describes the overall shape of gestures, recognition is done through training feature vectors without using any classifier. In stage 2, a recognition criterion was tough, and the feature vector had a binary coefficient. Finally, an output shows whether the gesture is correct or not.

In [2], a method for the recognition of 10 two-handed Bangla characters using normalized cross-correlation is proposed by Deb et al. A RGB color model is adopted to select heuristically threshold values for detecting hand regions, and template-based matching is used for recognition. However, this method does not use any classifier and is tested on limited samples.

M.A. Mohandes [3] proposed a method for the recognition of the two-handed Arabic signs using the Cyber Glove and support vector machine. Principal Component Analysis (PCA) feature is used for feature extraction. This method consists of 20 samples of 100 signs by one signer. 15 samples of each sign were used for training a Support Vector Machine to execute the recognition. The system was tested on the remaining 5 samples of each sign. A recognition rate of 99.6% on the testing data was obtained. When the number of signs in the vocabulary increases, the support vector machine algorithm must be parallelized so that signs are recognized in real-time. The drawback of this method was to employ 75% images for training and the remaining 25% for testing.

Work on two-handed signs has been done in Rekha et al. [4]. Here, Principal Curvature Based Region (PCBR) is used as a structure detector, Wavelet Packet Decomposition (WPD-2) is used to fine texture, and complexity defects algorithms are used for finding features of the finger. The skin color model used here is YCbCr fragmenting hand region. The classifier used is multi-class non-linear support vector machines (SVM). The accuracy for static signs is 91.3%. However, three dynamic gestures are also considered which use Dynamic Time Warping (DTW). The feature extracted is the hand movement trajectory forming the feature vector. The accuracy for the same is 86.3%.

Bauer and Heinz [5] introduced the basic problems and difficulties that arise while performing continuous sign language recognition. In this paper, manual sign parameter information such as hand structure, hand location, and orientation are extracted, forming the feature vector. The hand motion is not considered as it is handled separately by HMM topology, and hence not included in the feature vector. Gloves of different colors are worn by users to distinguish dominant and nondominant hands. For each sign, one HMM is modeled. The system experimented on 52 different signs of German Sign language (GSL) and 94% accuracy was found for all features. For 97 signs, this accuracy drops to 91.7%.

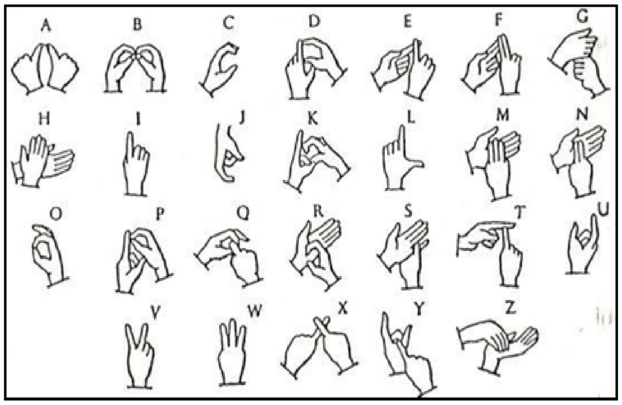


Fig.1. Finger Spelling in ISL

III. Implementation

The algorithm implemented in this paper recognizes Indian Sign Language gestures taken from videos of ISL gestures the system comprises of several steps which are Dataset Collection, Training and Evaluation of ConvNet CNN and Video Classification.

The algorithm implemented in this paper recognizes Indian Sign Language gestures taken from videos of ISL gestures the system comprises of several steps which are Dataset Collection, Training and Evaluation of ConvNet CNN and Video Classification.

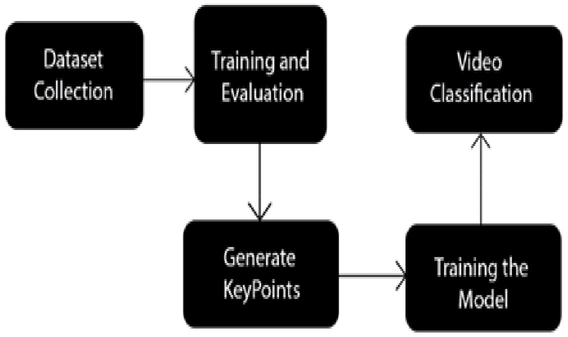


Fig.2. Overview of the System

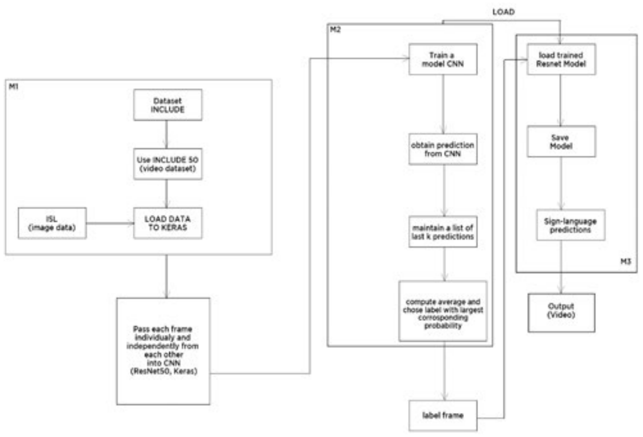

LOAD
M1
Dataset INCLUDE
Use INCLUDE 50 (video dataset)
ISL (image data)
LOAD DATA TO KERAS
Pass each frame individualy and independently from each other into CNN (ResNet50, Keras)
M2
Train a model CNN
obtain prediction from CNN
maintain a list of last k predictions
compute average and chose label with largest corresponding probability
label frame
load trained Resnet Model
Save Model
Sign-language predictions
M3
Output (Video)

Fig.3. Data Flow diagram of the Recognition System

3.1 Dataset Collection

Dataset used in our project is "INCLUDE: A Large-Scale Dataset for Indian Sign Language Recognition" [6]. They INCLUDE dataset has 4292 videos (the paper mentions 4287 videos, but 5 videos were added later). The videos used for training are mentioned in train.csv (3475), while that used for testing is mentioned in test.csv (817 files). Each video is a recording of 1 ISL sign, signed by deaf students from the St. Louis School for the Deaf, Adyar, Chennai.

3.2 Training and Evaluation

This phase is divided in multiple sections.

3.3 Generate Key Points

After the required dataset has been collected, we move to the next step of the process, i.e., to generate key points of the gestures using media pipe and the given dataset. To do this you will have to mention the directory where the key points must be saved and the directory of the dataset in the arguments.

This will start processing the landmarks in each frame of the video. The process also includes a function, wherein the landmarks after getting detected are swapped. If left hand distance is less than right hand distance it is marked as hand no. 2 and vice-versa.

3.4 Training the Model

After the media pipe landmarks have been generated successfully, the next step is to train a model using the train dataset and the generated landmarks. We will be omitting the pose landmarks and will use only hand landmarks for the training to save computer bottlenecking and faster processing. We will be including 2 methods to convert key points and landmarks to videos and generate embeddings:

- Using Augmented Data
- Using Convolutional Neural Network

Alongside these, we will be including 3 models, and the user will have the liberty to choose any one of the three methods to train depending on his/her computing needs. The methods are as follows:

- LSTM
- Transformer
- xgboost

The user will also be able to choose the number of Epochs that can be used to train the model as per his/her computing needs. Default (when not mentioned) no. of epochs is set to 20.

Maximum epochs recommended is 50 as we have successfully tested the accuracy scores which will be mentioned further in Results and Analysis chapter. Learning Rate i.e., the rate at which the network will train is set to default to 1e-4 and can be changed as per the user's needs.

If the user intends to use the transformer model, he/she needs to specify the size of the transformer model to be used. Default set to small. We have used CNN for generating embedding and have used the transformer model to train the CNN. Epochs have been set to 20 epochs for minimum CPU usage.

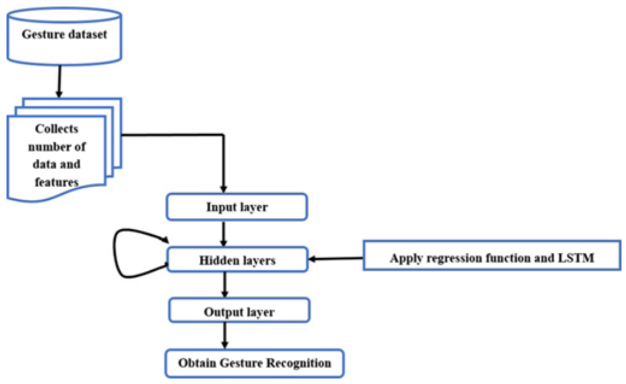

Gesture dataset
Collects number of data and features
Input layer
Hidden layers
Apply regression function and LSTM
Output layer
Obtain Gesture Recognition

Fig.4. Flow Diagram giving an overview how the recognition is done using LSTM.

3.5 Video Classification

After the generation of our CNN model, the next step is our video classification. For video classification, we can classify each frame individually and independently of each other and choose the label with the largest corresponding probability. Though there's a problem with this approach if we try to apply simple image classification to video classification, we are likely to encounter a sort of prediction flickering.

As we know that subsequent frames in a video are correlated concerning their semantic contents. We can take advantage of the same to improve our video classification results. To obtain accurate prediction, for instance, of choosing the label with the largest corresponding probability we have maintained a list of the last K predictions then compute the average of that prediction and choose the label with the largest corresponding probability.

The output obtained by this method does not show predictions flickering and has smoothed out our predictions. We will parse our trained CNN model, a path to the input video, path to our output video, and the size of the queue for rolling average. By default, the size value is set to 128. We have used a deque to implement our rolling prediction averaging. The deque, is initialized with a maxlen equal to the args["size"] value. This queue is updated each time prediction is made on the frame.

IV. RESULTS AND DISCUSSION

In the INCLUDE50 dataset we have used 25 videos for training, 19 videos for testing and 25 videos for validating the trained CNN model strengths, and weaknesses and also answer the 15 to16 questions. We have subdivided the INCLUDE50 dataset to *"include50_train.txt"*, *"include50_test.txt"*, and *"include50_val.txt"* files. We have trained the Convent model using these datasets.

Using 20 epochs our CNN model achieves a maximum accuracy of 92.5%, which was achieved at the epoch no. 19/20 as shown in Fig 5 below:

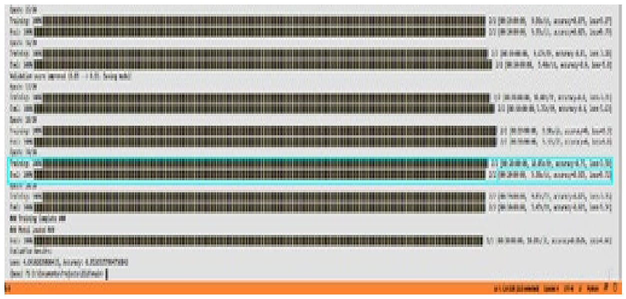


Fig.5. Screenshot showing accuracy – 0.925 at epoch 19/20

Also, we have implemented a system where when the CNN model achieves a validation accuracy of >= 5% accuracy, we will save the model for the first time and continue training on the saved model. When the accuracy hits 50% for the first time the model is again saved to disk and when accuracy reaches 90% the model is again saved, and this updated trained model is then used for training.

Fig.6. Screenshot showing model being saved when it reaches 5% accuracy score.

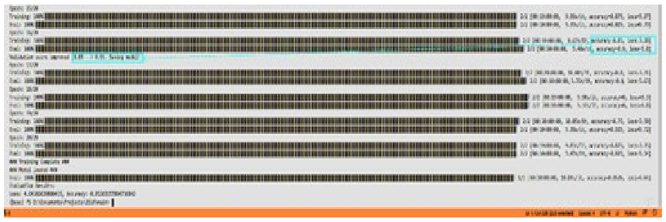


Fig.7. Screenshot showing model being saved when it reaches 90% accuracy score

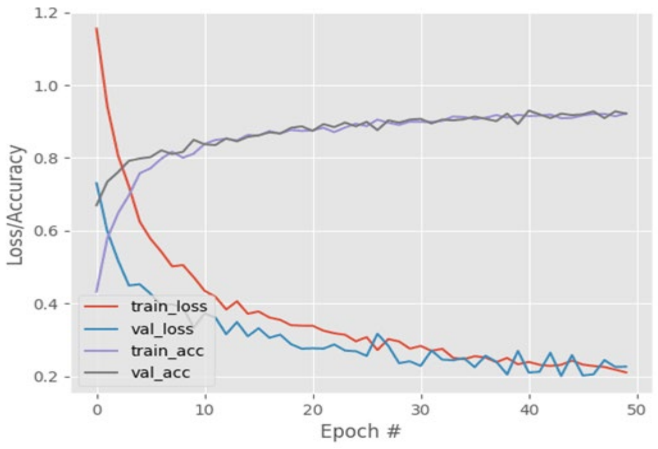
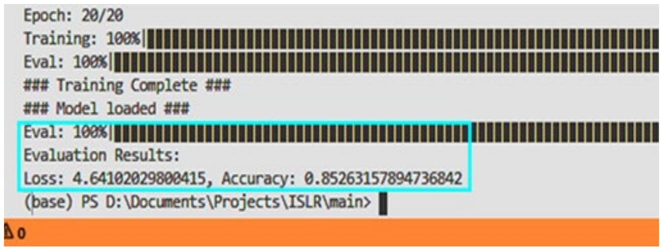


Fig.8. graph showing the loss/accuracy vs no. of epochs plot

From Fig. 7 one can easily conclude that our model can achieve a maximum of 94.5% accuracy with the INCLUDE50 dataset used and no. of epochs set to 50. Although we have used only 20 epochs for training and validating our CNN model, it can still achieve an accuracy score of 85.26% accuracy with a dataset of 88 videos.



```
Epoch: 20/20
Training: 100%|
Eval: 100%|
### Training Complete ###
### Model loaded ###
Eval: 100%|
Evaluation Results:
Loss: 4.64102029800415, Accuracy: 0.85263157894736842
(base) PS D:\Documents\Projects\ISLR\main>
```

Fig.9. Final Evaluation Results with accuracy score 85.26%

As shown in the above figure a loss of 4.64% is achieved and an Accuracy of 85.26% is achieved with our trained model.

After the training process the classification is processed. We have used the following labels to classify the videos:

['okay', 'peace', 'correct', 'disapprove', 'call me', 'stop', 'rock', 'live long', 'fist', 'smile']

Some examples of gestures shown in the following images: So, the final testing and evaluation process the results we see is that with 20 epochs and 88 videos used for training, testing, and evaluating video dataset we can achieve a maximum 92.5% accuracy and an average 85.26% accuracy using the INCLUDE50 dataset which is a very good result. Also, from Fig 5 we can see that training the model itself requires approximately 40 mins and evaluating the videos will take a further 400 seconds as calculated while using 20 epochs. Expect about 1000 seconds of wait time in the evaluation process when using 50 epochs. Also, expect processing time >2hrs if using more than 100 videos

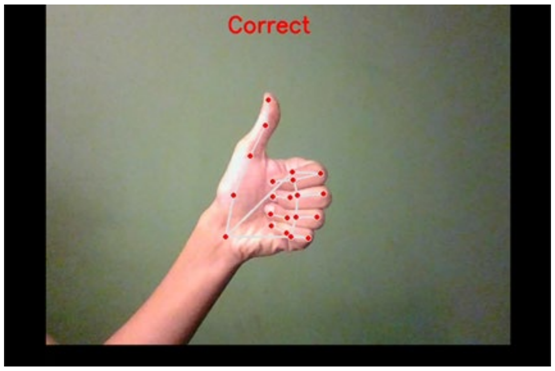


Fig.10. gesture recognised successfully for *"correct"*.

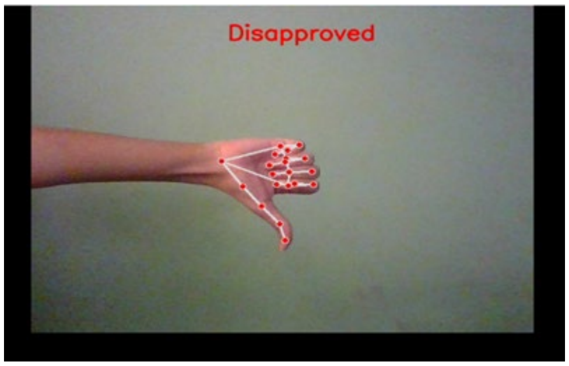


Fig.11. gesture recognized successfully for *"disapproved"*.

So, the final testing and evaluation process the results we see is that with 20 epochs and 88 videos used for training, testing, and evaluating video dataset we can achieve a maximum 92.5% accuracy and an average 85.26% accuracy using the INCLUDE50 dataset which is a very good result. Also, from Fig 5 we can see that training the model itself requires approximately 40 mins and evaluating the videos will take a further 400 seconds as calculated while using 20 epochs. Expect about 1000 seconds of wait time in the evaluation process when using 50 epochs. Also, expect processing time >2hrs if using more than 100 videos. These times greatly vary from system to system as using a CPU one can expect extremely low levels of parallelism. To counter the problems, we have added a variety of arguments which the client or user can choose from, depending on his or her processing needs and capability. We recommend using a GPU enabled system for faster processing and better parallelism.

V. Conclusion

This work presents two approaches to Sign Language recognition, dataset creation and real-time implementation. In the proposed system, we have created a sign detector, which detects a vast multitude of signs and hand gestures including the alphabets. The best performing model achieves an accuracy of 94.5% on the INCLUDE-50 dataset and 85.6% on the INCLUDE dataset. This model uses a pre-trained feature extractor and encoder and only trains a decoder. Currently we are using webcam live streaming for the recognition, in which we will perform various gesture in front of the webcam, which will detect the gesture we are performing in the ISL and will show us the name of the action or gesture on the screen.

References

[1]. I. Agarwal, S. Johar, and Dr. J. Santhosh, "A Tutor for The Hearing Impaired (Developed Using Automatic Gesture Recognition)," International Journal of Computer Science, Engineering and Applications (IJCSEA) vol.1, no.4, 2011.

[2]. K. Deb, H. P. Mony, and S. Chowdhury, "Two-Handed Sign Language Recognition for Bangla Character Using Normalized Cross Correlation," Global Journel of Science and Technology, vol. 12, Issue 3, 2012. Williams, Mickey (2002), 'Microsoft Visual C# .NET', Redmond, Wash.: Microsoft Corporation.

[3]. M. A. Mohandes," Recognition of Two-handed Arabic Signs using the CyberGlove," The Fourth International Conference on Advanced Engineering Computing and Applications in Sciences, 2010

[4]. J. Rekha, J. Bhattacharya, and S. Majumder, "Shape, Texture and Local Movement Hand Gesture Features for Indian Sign Language Recognition," in 3rd International Conference on Trends in information science and computing, pp. 30-35, 2011.

[5]. H.D. Yang, S. Sclaroff, and S.W. Lee, “Sign Language Spotting with a Threshold Model Based on Conditional Random Fields,” IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 31, pp. 1264-1277, 2009.

[6]. O. Aran, I. Ari, L. Akarun, B. Sankur, A. Benoit, A. Caplier, P. Campr, A. H. Carrillo, and F. Xavier Fanard, “SignTutor: An Interactive System for Sign Language Tutoring,” IEEE feature article, pp. 81-93, 2009.

[7]. B. Lekhashri and A. ArunPratap, “Use of motion-print in sign language recognition,” IEEE National Conference on Innovations in Emerging Technology (NCOIET), pp. 99-102, 2011.

[8]. A. Nandy, J. S. Prasad, S. Mondal, P. Chakraborty, and G.C. Nandi, “Recognition of Isolated Indian Sign Language Gesture in Real Time,” BAIP, Springer LNCSCCIS, Vol. 70, pp. 102-107, 2010. References Dept. of CEA, GLAU 41.

[9]. Masood, S., Thuwal, H.C., Srivastava, A.: American sign language character recognition using convolutional neural networks. In: Proceedings of Smart Computing and Informatics, pp. 403–412. Springer, Singapore (2018).

[10]. Sridhar, Advaith, Ganesan, Rohith Gandhi, Kumar, Pratyush, & Khapra, Mitesh. (2020). INCLUDE: A Large-Scale Dataset for Indian Sign Language Recognition [Data set]. ACM Multimedia 2020 (ACMMM2020).

[11]. Xiaolong Wang, Abhinav Gupta. Unsupervised Learning of Visual Representations using Videos. In ICCV, 2015.