1. INTRODUCTION

1.1 PROBLEM STATEMENT

Conversing with people having a hearing disability is a major challenge. Deaf and Mute people use hand gesture sign language to communicate, hence normal people face problems in recognizing their language by signs made.

Sign Language Recognition is a breakthrough for helping deaf-mute. Dynamic sign language recognition aims to recognize hand gestures of any person. The entire gesture of words or alphabets is recognized through video classification (Dynamic Input/Video Classification).

1.2 OBJECTIVE

The main objective of this project is to recognize word level dynamic sign language of any person and aid communication with deaf/dumb people.

Dynamic sign language recognition aims to recognize hand gestures of any person. Video contains both spatial and temporal features.

The proposed model uses CNN to extract the spatial features and RNN to extract the temporal features and find the corresponding word for the hand gesture.

1.3 DATASET - LSA64: A Dataset for Argentinian Sign Language

The sign database for the Argentinian Sign Language, created with the goal of producing a dictionary for LSA and training an automatic sign recognizer, is used. We consider 375 videos where 5 non-expert subjects executed 3 repetitions of 25 different types of signs. Signs were selected among the most commonly used ones in the LSA lexicon, including both verbs and nouns.

2. LITERATURE SURVEY

2.1 EXISTING WORK

Conversing with people having a hearing disability is a major challenge. Sign Language Recognition is a breakthrough for helping deaf-mute communicate with the rest of the world.

Video preprocessing is the first stage of Dynamic sign language recognition. It is done to extract the frames of the image in order to preprocess and send them for feature extraction. This part is mostly done with the help of OpenCv. Other papers which mainly focus on static hand sign recognition do not include this step since their input is of image format already.

After frame extraction, preprocessing is performed on the extracted frames. Different methods have been used to implement this process. Jinalee Jayeshkumar Raval et Al., [3] have converted the frames to HSV format and applied a series of dilation and erosion to fill the gaps and convert them into a row matrix. Kanchan Dabre., [4] preprocessed the images using color normalization, skin detection and blob analysis filtering. Kohsheen Tiku et Al., [5] have processed the frames using contour masking, skeletonization and canny edges. Neel Kamal Bhagat et Al., [6] have done effective real time background subtraction using depth perception techniques. Xinyun Jiang et Al., [8] have processed the frames using hand segmentation, image cropping, reorientation and normalization to find the Region of Interest.

Spatial Feature extraction is the next step after preprocessing of images. Kohsheen

Tiku et Al., [5] have used HOG or Histogram of Oriented Gradients to extract the features from images. HOG features are low level features compared to CNN. Aishwarya Sharma et Al., [2] and Jinalee Jayeshkumar Raval et Al., [3] have created CNN model to extract the spatial features. Aditya Das et Al., [1] have also employed CNN model using Inception V3 to extract features. Xinyun Jiang et Al., [8] have used Principal Component Analysis (PCA) to complete the transformation of pre-processed images into sets of feature vectors.

Those models which are used only to predict static hand gestures do not need the temporal feature extraction step. They can directly classify the images to the correct label. Kohsheen Tiku et Al., [5] and Xinyun Jiang et Al., [8] have used SVM for classification. Aishwarya Sharma et Al., [2], Jinalee Jayeshkumar Raval et Al., [3] and Aditya Das et Al., [1] have added an additional dense layer to CNN for classification purposes. Kanchan Dabre., [4] have classified using Haar Cascade Classifier to generate segmented output. Neel Kamal Bhagat et Al., [6] have done both static and dynamic gesture detection. For static gesture detection they have used CNN to extract features and for classification purposes.

Since Dynamic hand sign recognition involves video where each frame has different spatial features, we need to use temporal feature extraction methods to extract features in frames at different times. Suharjito et Al., [7] have extracted the spatiotemporal features and predicted the classes through the I3D model using transfer learning. It uses 3D convolution to learn spatiotemporal information directly from videos. Neel Kamal Bhagat et Al., [6] have used convolutional LSTM in case of dynamic hand gestures to extract the spatiotemporal features and prediction of the label.

2.2 LIMITATIONS OF EXISTING WORK

- Majority of the proposed models are limited to only static gestures which are only alphabets and numbers. Dynamic gestures need to be considered to focus on words and expressions rather than alphabets and numbers.
- the datasets used are only focused on hands without any major background. This does not perform well when we try to implement it in real time.
- The features extracted by HOG are low-level features.
- Using I3D to extract the spatiotemporal features has resulted in overfitting. It performs poorly on the test set.
- Haar Cascade Classifier does better in terms of speed since it has lower computational needs but it compromises heavily on precision.
- Majority of the systems do not convert the predicted word in text format into audio form which makes them less user-friendly.
- Complete meaningful sentences have not been formed as only words, letters and alphabets are recognized at a time.

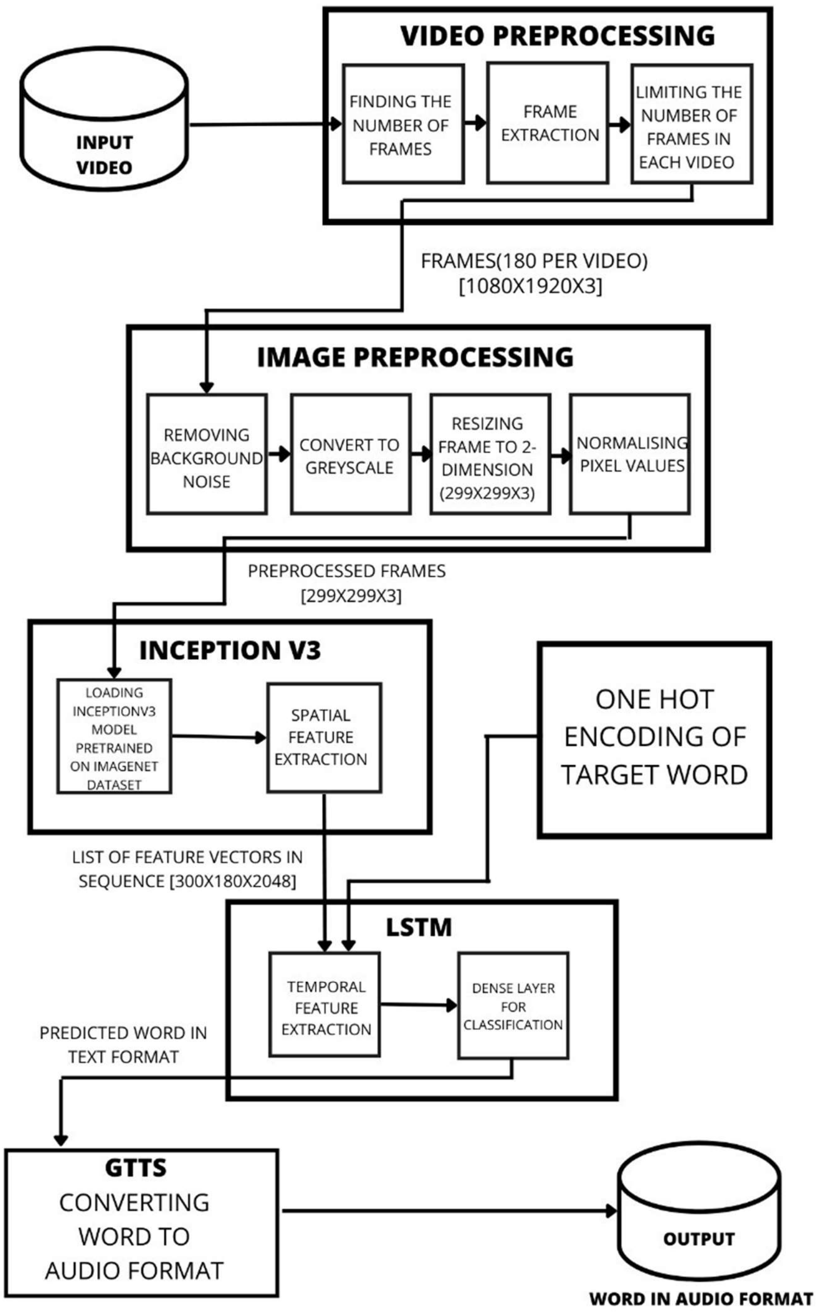
2.3 SOLUTION PROPOSED

- In our proposal we aim to identify Dynamic hand signs which gives way to the use of more complex words that can’t be identified only by the use of static hand gestures.
- We remove the background noise from the extracted frames and focus on the region of interest.
- We are using Inception V3 pre-trained on ImageNet to extract the spatial features of each frame.

- Then we use LSTM to extract the temporal features and thus predict the corresponding label.
- We also convert the identified text to audio output. It is a simple yet pragmatic utility which provides a more convenient way for disabled people to converse with others.

3. SYSTEM DESIGN

3.1 SYSTEM ARCHITECTURE



INPUT VIDEO
VIDEO PREPROCESSING
FINDING THE NUMBER OF FRAMES
FRAME EXTRACTION
LIMITING THE NUMBER OF FRAMES IN EACH VIDEO
FRAMES(180 PER VIDEO) [1080X1920X3]
IMAGE PREPROCESSING
REMOVING BACKGROUND NOISE
CONVERT TO GREYSCALE
RESIZING FRAME TO 2-DIMENSION (299X299X3)
NORMALISING PIXEL VALUES
PREPROCESSED FRAMES [299X299X3]
INCEPTION V3
LOADING INCEPTIONV3 MODEL PRETRAINED ON IMAGENET DATASET
SPATIAL FEATURE EXTRACTION
ONE HOT ENCODING OF TARGET WORD
LIST OF FEATURE VECTORS IN SEQUENCE [300X180X2048]
LSTM
TEMPORAL FEATURE EXTRACTION
DENSE LAYER FOR CLASSIFICATION
PREDICTED WORD IN TEXT FORMAT
GTTS CONVERTING WORD TO AUDIO FORMAT
OUTPUT
WORD IN AUDIO FORMAT

Fig.1 Overall Architecture diagram

3.2 SYSTEM REQUIREMENTS

3.2.1 SOFTWARE REQUIREMENTS

- Python 3.x
- Google Drive Storage (10 Gb)
- Python Libraries
    - OpenCV
    - TensorFlow
    - Keras
    - Numpy
    - Pandas
    - Pickle

3.2.2 HARDWARE REQUIREMENTS

- Intel i3/i5 or Ryzen 3 1200x with minimum 2.4GHZ or equivalent
- 8GB RAM

4. DETAILED ARCHITECTURE

4.1 LIST OF MODULES

4.1.1 VIDEO PREPROCESSING

The input of 375 videos is split into ‘train’ and ‘test’ set in a ratio of 80:20. There are 300 videos in the training set and 75 videos in the test set. We find the average number of frames in the videos in order to maintain uniformity in the number of frames across all the videos to be sent to the model. We fix the number of frames to be 180. In cases where the number of frames in the video is greater than 180, we discard all the frames after the 180th frame (only in case of a few videos the number of frames is greater than 180). In cases where the number of frames in the video is less than 180, we duplicate the last frame of the video till the threshold is reached. Finally, we store the extracted list of frames on the drive. The stored frames are later sent to the image preprocessing stage.

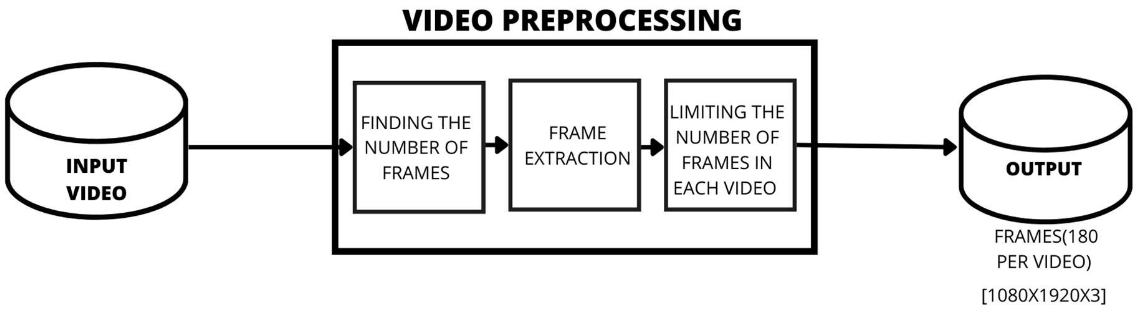


Fig.2 Architecture diagram for Video Preprocessing Cell

INPUT - list of videos (mp3)

OUTPUT - extracted frames [1080 x 1920 x 3] for each

PSEUDOCODE

```
Cap = cv2.VideoCapture(filepath)
while num_of_frames < 180:
    Extract current frame
    Save the frame in jpg format
    If current frame is the last frame, then
        Break
while num_of_frames < 180:
    Duplicate and save the last frame in jpg format
```

4.1.2 IMAGE PREPROCESSING

In this module we deal with preprocessing of images to remove all the unnecessary information and format it to pass it to our CNN (Inception V3) model.

We load the extracted frames from the drive and remove all the background noise from the frame such as the body and the background and consider only the region of interest limited by boundary masks. The region of interest here is the gloved hands of the signer. Then we convert the frames to grayscale format.

We resize the image whose dimensions are 1080 x 1920 x 3 to the dimensions 299 x 299 x 3 which is acceptable by the InceptionV3 model. Finally, we scale the pixel values of the frame between -1 and 1 and save the frames to the drive.

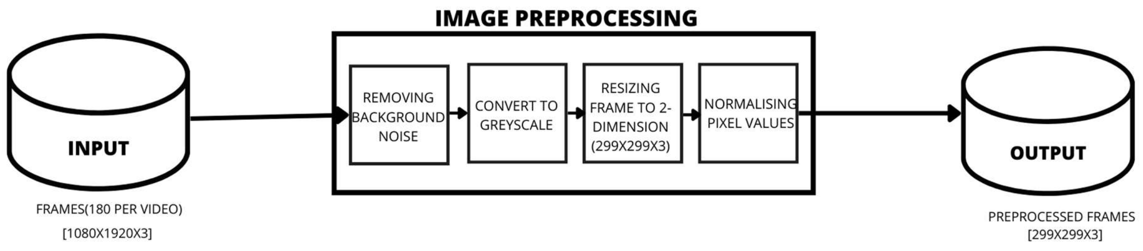


Fig.3 Architecture diagram for Image Preprocessing Cell

INPUT: 180 Frames per video (1080 x 1920 x 3)
OUTPUT: Preprocessed frames (299 x 299 x 3)

PSEUDOCODE

Removal of Background Noise (Focus on hand) and Conversion to Grayscale:

```
Set the boundaries to focus on hands of signer
For each image:
    lower_mask = cv2.inRange(image, lower_boundary1, upper_boundary1)
    upper_mask = cv2.inRange(image, lower_boundary2, upper_boundary2)

    mask = cv2.bitwise_or(lower_mask, upper_mask)
    result = cv2.bitwise_and(result, result, mask=mask)
    Convert to grayscale
```

Resizing and normalizing to meet the requirements of Inception V3:

```
For each image:
    Read the jpg image
    Decode a jpeg encoded image to tensor with channels set to 3
    Resize into 299x299
    The inputs pixel values are scaled between -1 and 1
```

4.1.3 INCEPTION V3

Inception v3 is an image recognition model that has been shown to attain greater than 78.1% accuracy on the ImageNet dataset. In our model we require only the extraction of features so we remove the last fully connected layer which classifies the image. The model will now output the feature vectors of the input frames.

We use transfer learning which is a process where we freeze the early convolutional layers of the network and only train the last few layers which make a prediction. The idea is that convolutional layers extract general, low-level features that are applicable across images — such as edges, patterns, gradients — and the later layers identify specific features within an image such as eyes or wheels (hand signs in our case). We apply transfer learning through InceptionV3 to extract the spatial features of the frames which are stored in numpy file format on drive for later use.

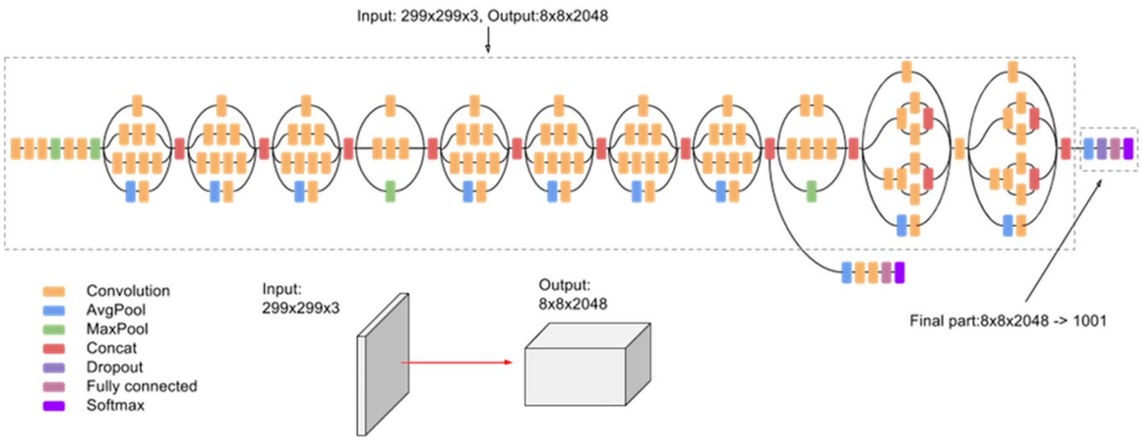


Fig. 4 Architecture diagram for InceptionV3

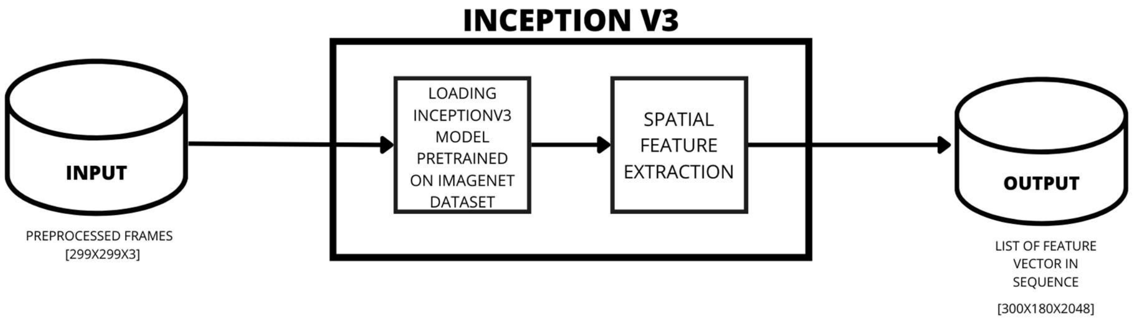

INCEPTION V3
INPUT
PREPROCESSED FRAMES
[299X299X3]
LOADING INCEPTIONV3 MODEL PRETRAINED ON IMAGENET DATASET
SPATIAL FEATURE EXTRACTION
OUTPUT
LIST OF FEATURE VECTOR IN SEQUENCE
[300X180X2048]

Fig.5 Architecture diagram for InceptionV3 module

INPUT: Preprocessed frames (299 x 299 x 3)

OUTPUT: Feature vector of frames (300 x 180 x 2048)

PSEUDOCODE

```
Load the inception V3 model with weights pre trained on ImageNet dataset
Remove the last layer (classification layer) and set pooling to be avg to get 2048x1
dimensional vector
```

```
Create batches of frames where batch size is 64
Pass the batches through the model
Store the output in .npy format
```

4.1.4 LSTM

LSTM stands for long short-term memory networks, used in the field of Deep Learning. It is a variety of recurrent neural networks (RNNs) that are capable of learning long-term dependencies, especially in sequence prediction problems. LSTM has feedback connections and is capable of processing the entire sequence of data, apart from single data points such as images. We use LSTM to extract temporal features. Temporal feature refers to a feature that is associated with or changes over time. It occurs when there is a series of images taken at different times (frames corresponding to a video).

180 feature vectors (corresponding to the frames of each video) are passed into the model in sequence. The model is also given the one hot encoding of the target word for the videos. The final dense layer consists of 25 units corresponding to the number of words (classes) and is used for classification purposes.

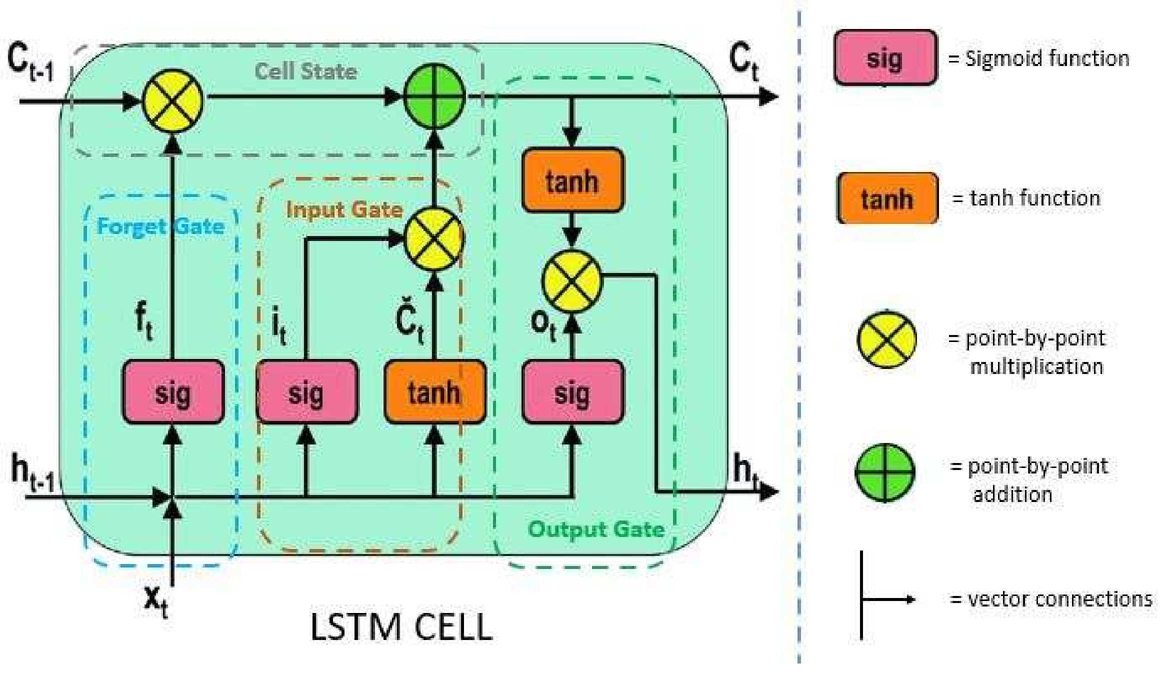


Fig.6 Architecture of a LSTM cell

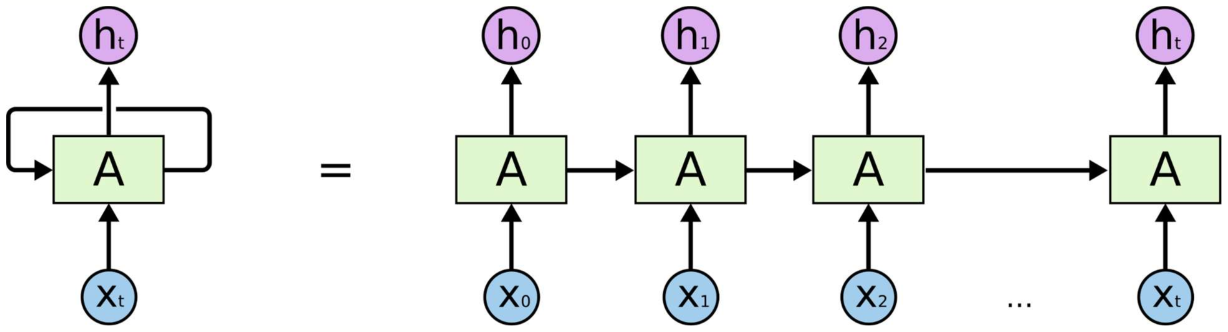


Fig.7 Representation of time sequence of LSTM cells

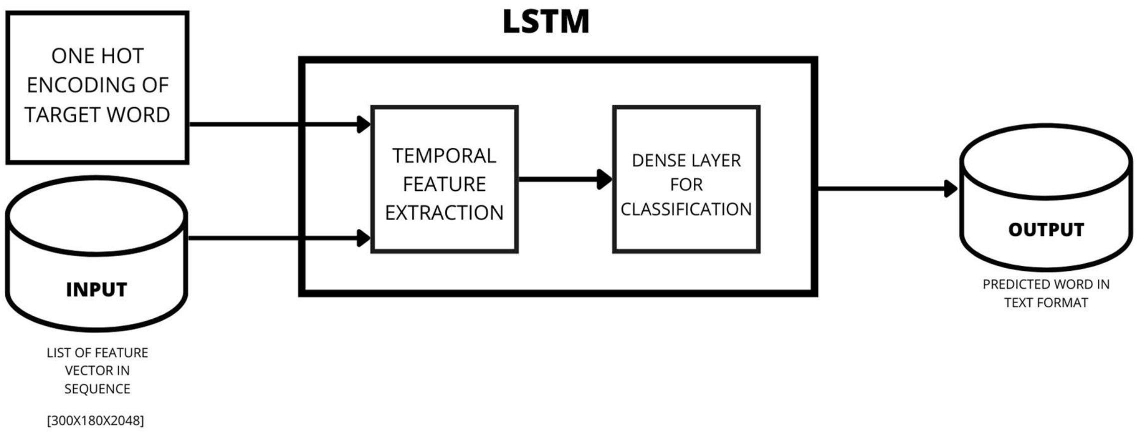

LSTM
ONE HOT ENCODING OF TARGET WORD
INPUT
LIST OF FEATURE VECTOR IN SEQUENCE
[300X180X2048]
TEMPORAL FEATURE EXTRACTION
DENSE LAYER FOR CLASSIFICATION
OUTPUT
PREDICTED WORD IN TEXT FORMAT

Fig.8 Architecture of a LSTM module

INPUT: List of feature vectors in sequence (300 x 180 x 2048) and One Hot encoding of target words.

OUTPUT: Predicted word in text format.

PSEUDOCODE

LSTM model

```
Create a sequential keras model
Add LSTM layer with 256 units
Add a dense layer with 25 (number of words) units and set activation to 'Softmax'
Optimizer is set to 'Adam' and loss is set to 'Categorical Cross entropy'
```

Training the model

```
Create a one-hot encoding of the target words for each video which results in (300, 25) dimensional vector
```

Combine the 180 feature vectors of each video in sequence to form a (300, 180, 2048) dimensional vector
Set batch size to 32 and number of epochs to 20
Fit the model
Plot the loss and accuracy plot
Print the confusion matrix and precision and f1 scores.

4.1.5 gTTS

gTTS (Google Text-to-Speech) is a Python library and CLI tool to interface with Google's text-to-speech API. It converts the text entered (which is the predicted word) into audio which can be saved as a wav file (waveform audio file format). The gTTS API supports several languages including English, Hindi, Tamil, French, German and many more.

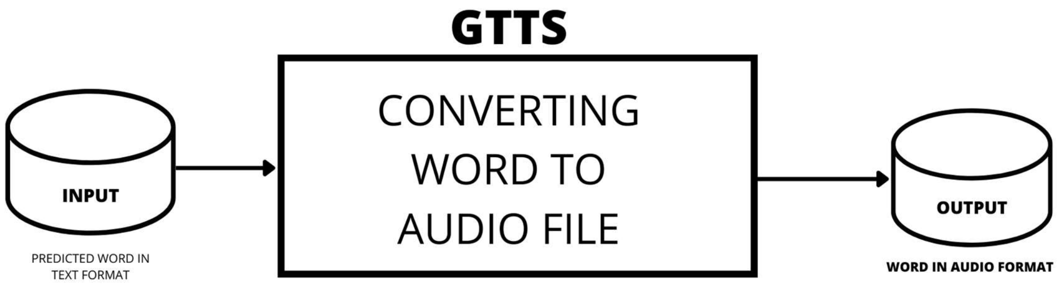


Fig.9 Architecture of gTTs cell

INPUT: Predicted word in text format.
OUTPUT: Predicted word in audio format.

PSEUDOCODE

```
Download gTTS library
Translate predicted text to audio format
Write the audio file into bytestring object
Output the bytestring object
```

5. IMPLEMENTATION

5.1 MODULE OUTPUTS

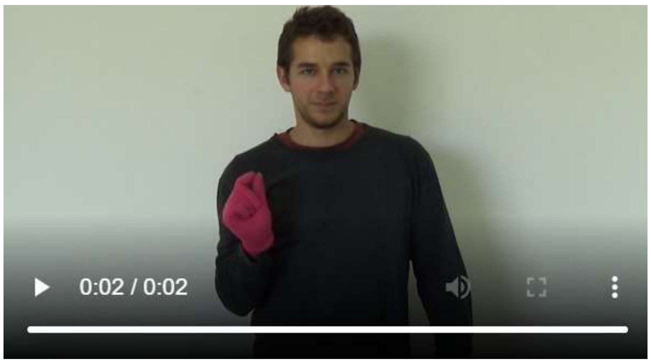
5.1.1 VIDEO PREPROCESSING

Video preprocessing: The input video is broken down into 180 sequences of frames.

INPUT - list of videos (mp3)

OUTPUT - extracted frames [1080 x 1920 x 3]

Table 1 - Video Preprocessing Output

INPUT	**OUTPUT**
 0:02 / 0:02	Frame 1

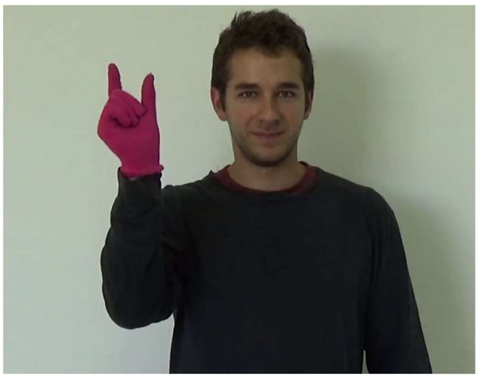
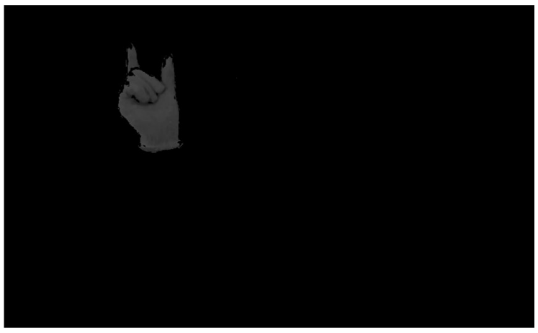
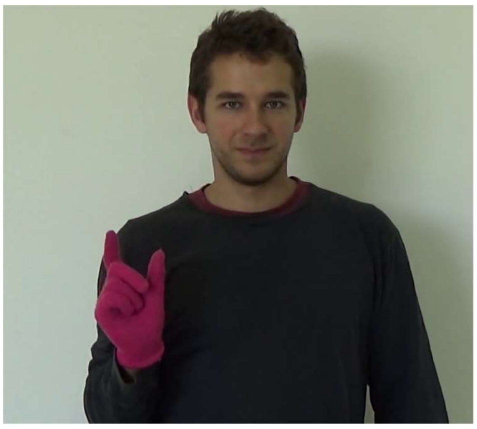
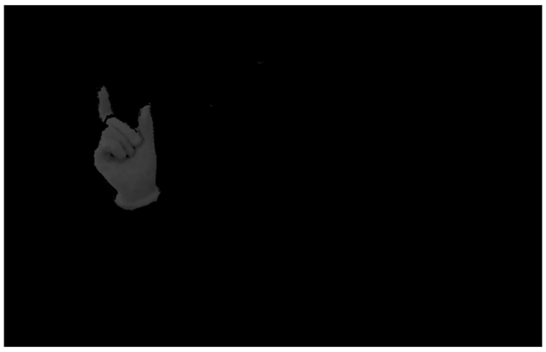
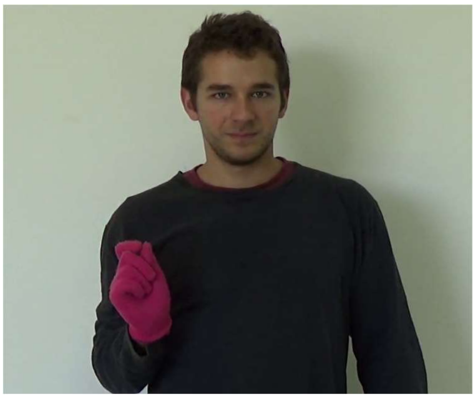
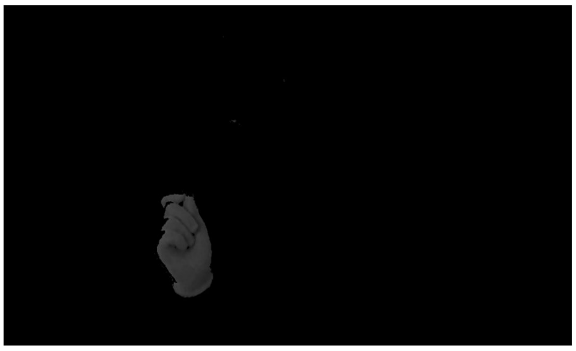
	Frame 130
	Frame 180

5.1.2 IMAGE PREPROCESSING

The extracted frames are preprocessed by removing background noise and considering only ROI which is hand glove. The image is resized into dimensions 299x299x3 and pixel values are scaled between 1 and -1.

INPUT: 180 Frames per video (1080 x 1920 x 3)
OUTPUT: Preprocessed frames (299 x 299 x 3)

Table 2 – Image Preprocessing Output

INPUT	**OUTPUT**
	
	
	

5.1.3 INCEPTIONV3

Using pre-trained inception V3 model to extract spatial features. The feature vector is obtained as output and has dimension 2048x1.

INPUT: Preprocessed frames (299 x 299 x 3)
OUTPUT: Feature vector of frames (300 x 180 x 2048)

Table 3 – Inception V3 Output

INPUT	**OUTPUT**
	array([0.48222047,0.02377112, 0.12764817, ..., 0.4690452 , 0.51383114, 0.15406677], dtype=float32)
	array([0.6491062 , 0.1345267 , 0.32852545, ..., 0.18988268, 0.18445691,0.01487502],dtype=float32)

	array([0.6491062 , 0.1345267 , 0.32852545, ..., 0.18988268, 0.18445691,0.01487502],dtype=float32)

5.1.4 LSTM

INPUT: List of feature vectors in sequence (300 x 180 x 2048) and One Hot encoding of target words.

OUTPUT: Predicted word in text format.

Table 4 – LSTM Output

INPUT	**OUTPUT**
array([0.48222047,0.02377112, 0.12764817, ..., 0.4690452 , 0.51383114,0.15406677],dtype=float32)	COUNTRY
array([0.6491062 , 0.1345267 , 0.32852545, ..., 0.18988268, 0.18445691,0.01487502], dtype=float32)	
array([0.6491062 , 0.1345267 , 0.32852545, ..., 0.18988268, 0.18445691,0.01487502], dtype=float32)	

5.1.5 gTTS

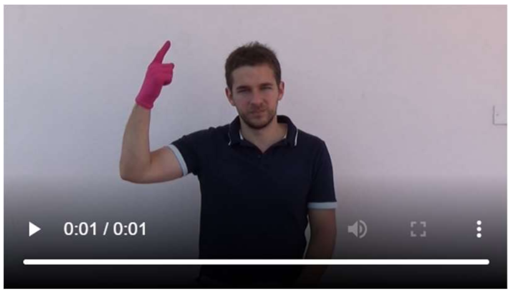
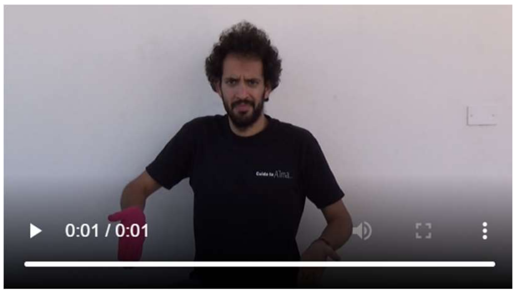
Converting the word in text format into audio format.

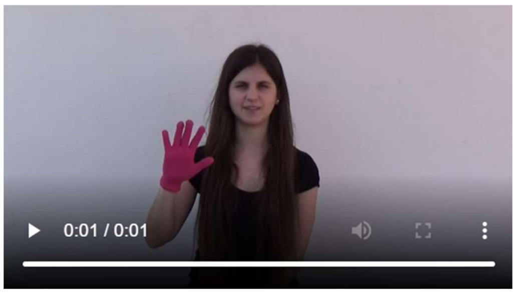
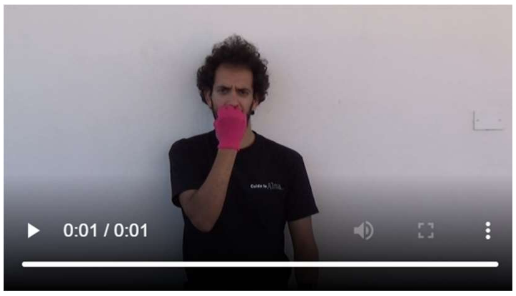
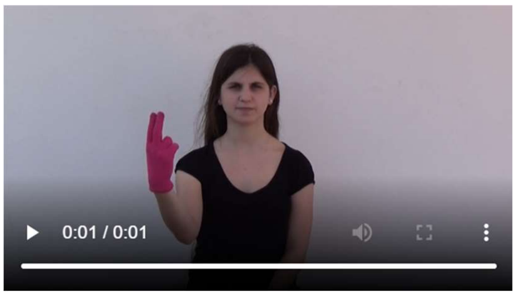
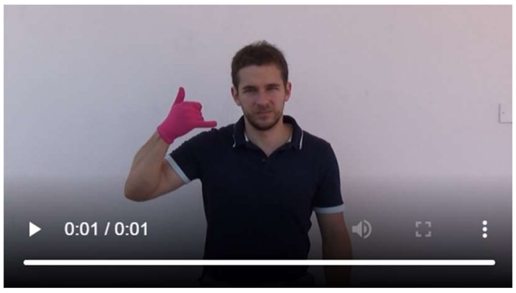
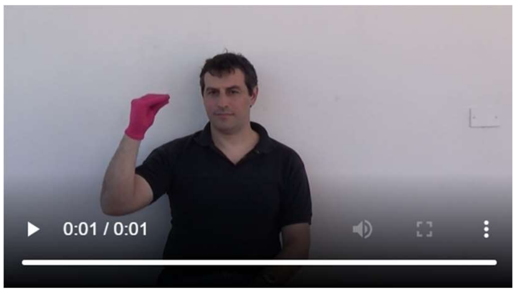
INPUT: Predicted word in text format.

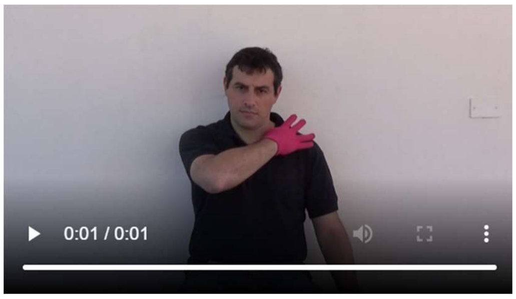
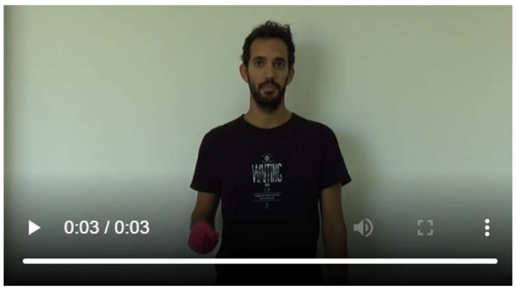
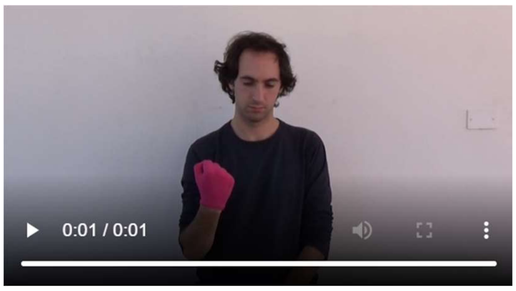
OUTPUT: Predicted word in audio format.

5.2 FINAL OUTPUTS

Table 5 – Final Outputs

S. No.	**Input Video**	**Correct Label**	**Predicted Label**	**Audio File**
1.		Learn	Learn	learn.wav
2.		Born	Born	born.wav

3.		Bright	Bright	bright.wav
4.		Bitter	Bitter	bitter.wav
5.		Son	Son	son.wav
6.		Call	Call	call.wav
7.		Away	Opaque	opaque.wav

8.		Yellow	Yellow	yellow.wav
9.		Country	Country	country.wa v
10.		Drawer	Water	water.wav

6. PERFORMANCE METRICS

6.1 ACCURACY & LOSS

Accuracy - It measures how often the classifier correctly predicts. It is defined as the ratio of the number of correct predictions and the total number of predictions.

$$\text{Accuracy} = \frac{\text{TN+TP}}{\text{TN+FP+FN+TP}}$$

TP	-	True Positive
TN	-	True Negative
FP	-	False Positive
FN	-	False Negative

Accuracy is 89.34% on the test set.

Categorical Cross Entropy Loss - Cross-entropy as a loss function for a multi-class classification task. Cross-entropy builds upon the idea of information theory entropy and measures the difference between two probability distributions for a given random variable/set of events. It is widely used as a loss function when optimizing classification models.

$$L = -\sum_{i=1}^{2} t_i \log(p_i)$$
$$= -\left[t \log(p) + (1-t)\log(1-p)\right]$$

where t_i is the truth value taking a value 0 or 1 and p_i is the Softmax probability for the i^{th} class.

Table 6 - Training Categorical Cross Entropy Loss and Accuracy for each Epoch

EPOCHS	LOSS	ACCURACY
1	3.0207	0.1467
2	1.7843	0.6467
3	1.0460	0.8133
4	0.8520	0.8367
5	0.7970	0.8800
6	0.7270	0.8900
7	0.6169	0.9267
8	0.5202	0.9500
9	0.4910	0.9200
10	0.4482	0.9467
11	0.4115	0.9467
12	0.5090	0.9300
13	0.4967	0.9667

14	0.6189	0.9433
15	0.4401	0.9467
16	0.4102	0.9500
17	0.3259	0.9667
18	0.2598	0.9833
19	0.2258	0.9800
20	0.1989	0.9900

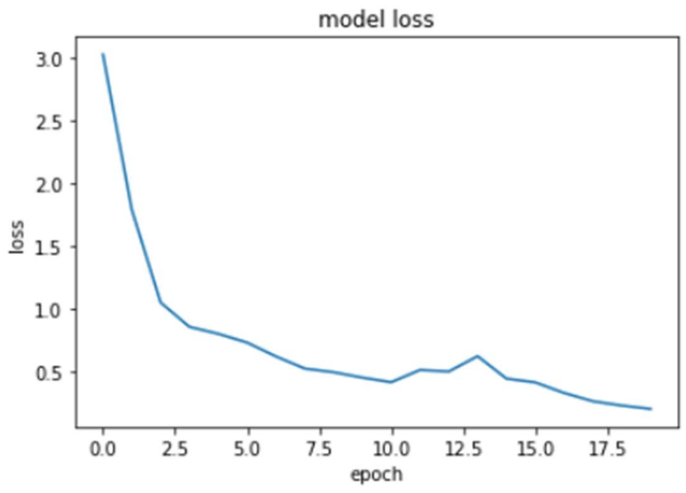


Fig.10 Loss plot

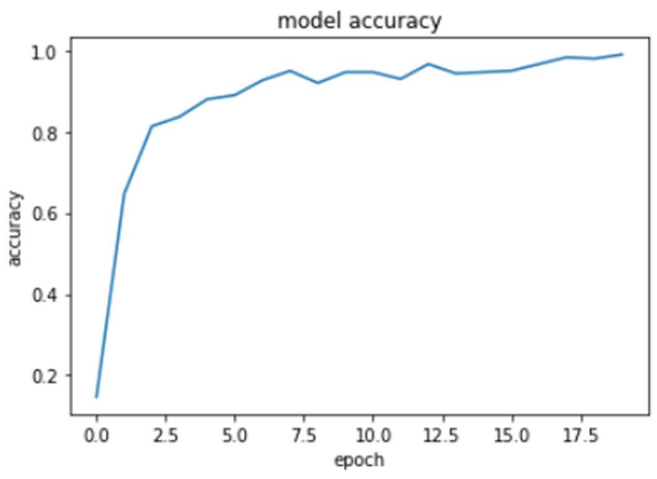


Fig.11 Accuracy plot

6.2 CONFUSION MATRIX

A confusion matrix is used to describe the performance of our classification model (or "classifier") on a set of test data (hand signs) for which the true values (target words) are known. It is extremely useful for measuring the Recall, Precision and Accuracy.

Table 7 – Confusion Matrix

Predicted	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**
Actual																									
0	**3**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	**2**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	**4**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	**3**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

4	0	0	0	0	**3**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	**4**	0	0	0	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	**3**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	**2**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	**1**	0	0	0	0	0	0	**3**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	**1**	0	**4**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	**2**	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	**1**	0	0	0	0	0	0	0	0	0	0	**4**	0	0	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	0	**3**	0	0	0	0	0	0	0	**1**	0	0	**1**	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0	**2**	0	0	0	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**2**	0	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**2**	0	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**5**	0	0	0	0	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**2**	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**3**	0	0	0	0
21	0	0	0	0	0	0	0	0	0	0	0	0	**1**	0	0	0	0	0	0	0	0	**1**	0	0	0
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**	**3**	0	0
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**3**	0
24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**3**

6.3 F1 SCORE, PRECISION AND RECALL

Precision and Recall are metrics that help us evaluate the predictive performance of a classification model on a particular class of interest, also known as the positive class.

Precision - Of all positive predictions, how many are really positive? Precision explains how many of the correctly predicted cases actually turned out to be positive. It can be seen as a measure of quality. Higher precision means that an algorithm returns more relevant results than irrelevant ones.

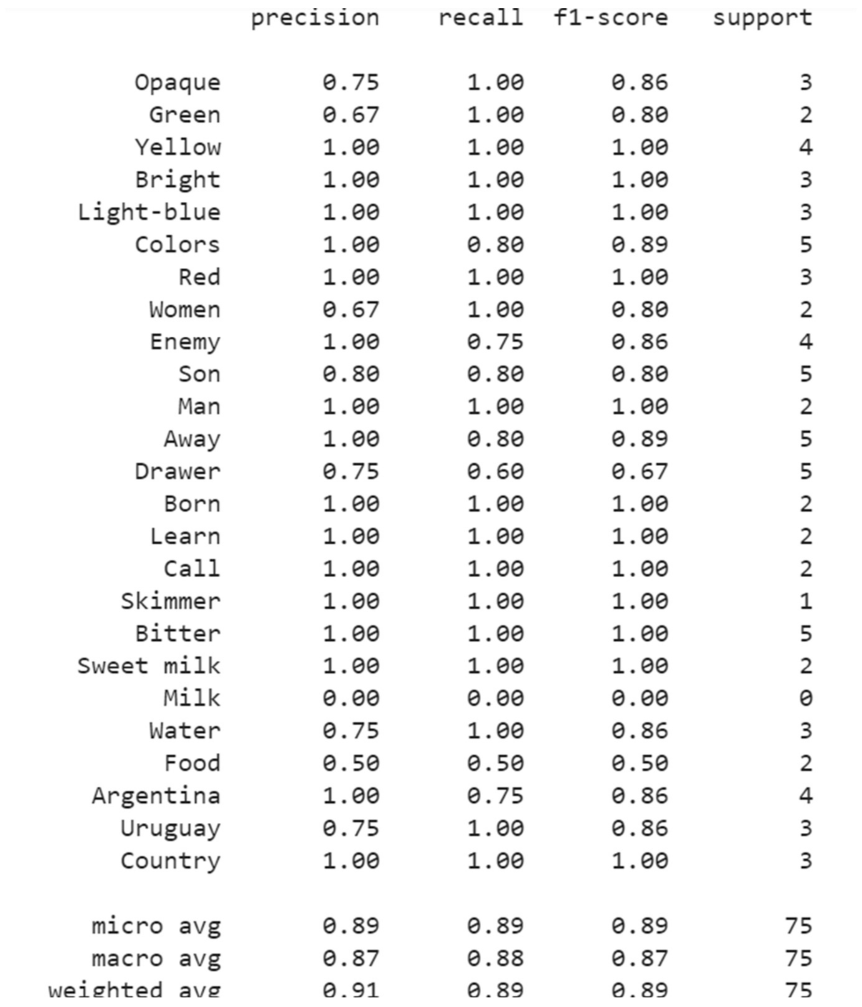
Recall - Of all real positive cases, how many are predicted positive? Recall explains how many of the actual positive cases we were able to predict correctly with our model. It can be seen as a measure of quantity.

$$Precision = \frac{TP}{TP + FP} \qquad Recall = \frac{TP}{TP + FN}$$

Where TP=True Positive, FP= False Positive, FN = False Negative

F1-score - It is the harmonic mean of precision and recall. It combines precision and recall into a single number using the following formula:

$$\text{F1-score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$



```
              precision    recall  f1-score   support

      Opaque       0.75      1.00      0.86         3
       Green       0.67      1.00      0.80         2
      Yellow       1.00      1.00      1.00         4
      Bright       1.00      1.00      1.00         3
  Light-blue       1.00      1.00      1.00         3
      Colors       1.00      0.80      0.89         5
         Red       1.00      1.00      1.00         3
       Women       0.67      1.00      0.80         2
       Enemy       1.00      0.75      0.86         4
         Son       0.80      0.80      0.80         5
         Man       1.00      1.00      1.00         2
        Away       1.00      0.80      0.89         5
      Drawer       0.75      0.60      0.67         5
        Born       1.00      1.00      1.00         2
       Learn       1.00      1.00      1.00         2
        Call       1.00      1.00      1.00         2
     Skimmer       1.00      1.00      1.00         1
      Bitter       1.00      1.00      1.00         5
  Sweet milk       1.00      1.00      1.00         2
        Milk       0.00      0.00      0.00         0
       Water       0.75      1.00      0.86         3
        Food       0.50      0.50      0.50         2
   Argentina       1.00      0.75      0.86         4
     Uruguay       0.75      1.00      0.86         3
     Country       1.00      1.00      1.00         3

   micro avg       0.89      0.89      0.89        75
   macro avg       0.87      0.88      0.87        75
weighted avg       0.91      0.89      0.89        75
```

Fig.12 Precision, Recall & F1-score

7. CONCLUSION

The Sign Language Recognition project is done to help the deaf and dumb community in connecting with the outer world, especially to those who do not understand sign language. The project starts with image processing for detecting the hands of the signer and removing background noise. The dataset is divided into two parts ‘train’ and ‘test’ set in a ratio of 80:20. InceptionV3 (CNN model) is used for extracting spatial features from the corresponding signs. The extracted features are then sent in sequence to LSTM for predicting the corresponding word. The predicted word is then converted to audio format. 89.34% accuracy is obtained on the test set and 99% accuracy on the training set. So, with this model we can continuously predict dynamic hand signs and display the corresponding word on the screen and voice it too.

7.1 FUTURE SCOPE

- The model can be configured to take real time input.
- Our model is limited to detect dynamic hand signs. It can be expanded to detect facial expressions to deduce the mood of the conversation and improve the accuracy.
- Our model is a word level SLR and it can be expanded to form a sentence level SLR where complete meaningful sentences are formed.

8. REFERENCES

1. Aditya Das, Shantanu Gawde, Khyati Suratwala, Dhananjay Kalbandec (2018) 'Sign Language Recognition Using Deep Learning on Custom Processed Static Gesture Images' - International Conference on Smart City and Emerging Technology (ICSCET), DOI: 10.1109/ICSCET.2018.8537248

2. Aishwarya Sharma, Siba Panda, Saurav Verma (2020) 'Sign Language to Speech Translation' - International Conference on Computing and Networking Technology (ICCNT), DOI: 10.1109/ICCCNT49239.2020.9225422

3. Jinalee Jayeshkumar Raval, Ruchi Gajjar (2021) 'Real-time Sign Language Recognition using Computer Vision' - 3rd International Conference on Signal Processing and Communication (ICPSC), doi: 10.1109/ICSPC51351.2021.9451709

4. Kanchan Dabre, kurekha Dholay (2014) 'Machine Learning Model for Sign Language Interpretation using Webcam Images' - International Conference on Circuits, Systems, Communication and Information Technology Applications (CSCITA), DOI: 10.1109/CSCITA.2014.6839279

5. Kohsheen Tiku, Jayshree Maloo, Aishwarya Ramesh, Indra R (2020) 'Real - time Conversion of Sign Language to Text and Speech' - Second International Conference on Inventive Research in Computing Applications (ICIRCA), DOI: 10.1109/ICIRCA48905.2020.9182877

6. Neel Kamal Bhagat, Y. Vishnusai, G. N. Rathna (2019) 'Indian Sign Language Gesture Recognition using Image Processing and Deep Learning' - Digital Image Computing: Techniques and Applications (DICTA), DOI: 10.1109/DICTA47822.2019.8945850

7. Suharjito, Herman Gunawan, Narada Thiracitta, Ariadi Nugroho (2018) ‘Sign Language Recognition Using Modified Convolutional Neural Network Model’ - Indonesian Association for Pattern Recognition International Conference (INAPR), DOI: 10.1109/INAPR.2018.8627014

8. Xinyun Jiang, Wasim Ahmad (2019) ‘Hand Gesture Detection based Real-time American Sign Language Letters Recognition using Support Vector Machine’ - IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/ PiCom/ CBDCom/ CyberSciTech), DOI: 10.1109/DASC/PiCom/CBDCom/CyberSciTech.2019.00078