Sahaayi - Malayalam Sign Language Recognition Application

Guided by: Assit. Prof Bincy Antony M,
Artha P Satheesh (KNR18CS021), Fadiya Haris (KNR18CS028),
Shahnavas K M (KNR18CS047) and Sukritha K K (KNR18CS055)
Department of Computer Science and Engineering, Government College of Engineering Kannur

***Abstract*—Communication holds a central position in social interactions. In a predominantly aural society, sign language users are often deprived of effective communication. The National Institute of Speech and Hearing(NISH) recently introduced Signs for Malayalam Alphabets. Applications for recognising previously formed sign languages are already available. Here we aim to develop an assisting system that recognizes hand gestures in Malayalam and converts them to textual format. Several solutions have come up but none of them have been portable for them to be used in a standalone device or application. We have alleviated this problem by harnessing the power of the mobile phone and the recent advances in deep learning. Our work emphasizes Dataset creation for this newly formed sign language, training a model using squeezenet to recognize these signs, and integrating the model into a flutter application for ease of access.**

***Index Terms*—Sign Language, Dataset, Machine learning, Squeeqenet, OpenCV, Flutter**

I. Introduction

Communication is very crucial to human beings, as it enables us to express ourselves. Sign language recognition is important for natural and convenient communication between the deaf community and the hearing majority. Currently, most communications between two communities highly rely on human-based translation services. However, this is inconvenient and expensive as human expertise is involved. Therefore, automatic sign language recognition aims to understand the meaning of signs without assistance from experts. Then it can be translated to sound or text based on end users' needs. We believe that sign language recognition is important for providing equal opportunity to every person and improving public welfare.

The lack of a sign language alphabet in Malayalam has been a challenge for teachers in schools for the hearing impaired. English and Hindi have their own sign language. At present, schools mostly rely on lip movement to get the message across. When words have to be written, these are written on students' hands or traced in the air. When tracing in the air, confusion is common.

The National Institute of Speech and Hearing (NISH) has come up with uniform sign language characters in the Malayalam alphabet using finger-spelling on September 29, [1]. Teachers and students at NISH, under the supervision of sign language experts there, have come up with the finger-spelling, which covers vowels and consonants too. This new sign language alphabet in Malayalam will eliminate the challenge

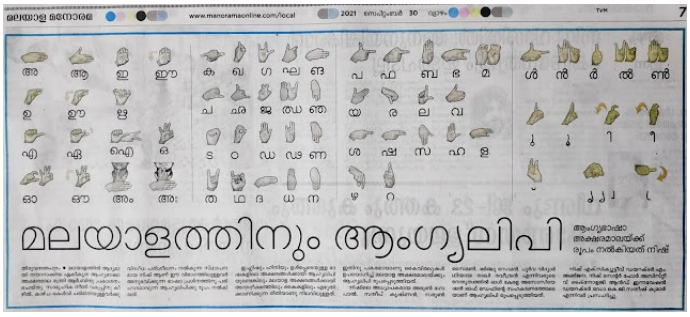


Fig. 1: First uniform sign language alphabet in Malayalam released.

of teachers in conveying messages through lip movement. This would also help bring about a qualitative change in the lives of those who are hearing impaired.

This project aims at introducing a mobile application that recognizes these Malayalam sign alphabets with the help of machine learning technology. Since these Malayalam Sign Alphabets are new, there was no proper dataset for this system. So the creation of a dataset for Malayalam Sign Alphabets is also an objective of our project.

II. Background Information

According to the World Federation of the Deaf, there are approximately 720 million deaf people globally and they use 300 different types of sign languages. As per the 2011 census, about 50 lakh people in India are deaf or mute or have poor hearing abilities. India has been using the 'Sign Language' for years but somehow lacked uniformity as different ways of saying the same thing was in practice. A need for a dictionary was always felt that could act as an intermediate between language and regionalism.

After a lot of research, the Indian Sign Language and Research Training Center created the country's first sign language dictionary. The first edition was launched on March 23, 2018, and around 10,000 words were present in it. To empower the deaf-mute people in the country, sign language has been given the status of an official subject.

III. Literature Review

Sign language recognition is a problem that has been addressed in research for years. The methods used in developing

Sign Language Recognition vary among researchers. Each method has its own strength and weakness compared to other methods. Here we have reviewed four pieces of literature to get an insight into our project.

A. Real-time American Sign Language Recognition with Convolutional Neural Networks [2]

The first literature that we have been into is Real-time American Sign Language Recognition with Convolutional Neural Networks. It presented the development and implementation of an American Sign Language (ASL) fingerspelling translator based on a convolutional neural network. They utilized a pre-trained GoogLeNet architecture trained on the ILSVRC2012 dataset, as well as the Surrey University and Massey University ASL datasets in order to apply transfer learning to this task.

Fig. 2: Datasets used in Real-time ASL recognition with CNN

ASL letter classification is done using a convolutional neural network (CNN or ConvNet). They produced a robust model that consistently classifies letters a-e correctly with first-time users and another that correctly classifies letters a-k in a majority of cases. Given the limitations of the datasets and the encouraging results achieved, they were confident that with further research and more data, they can produce a fully generalizable translator for all ASL letters.

B. Study of vision-based hand gesture recognition using Indian Sign Language [3]

The study of vision-based hand gesture recognition using Indian Sign Language is another piece of literature we have referenced. It states that as compared to other sign languages, ISL interpretation has gotten less attention from the researcher and it discusses some historical background, needs, scope, and concerns of ISL.

Fig. 3: Typical Hand Gesture Recognition System proposed in [3]

It deals with a vision-based approach and gives emphasizes that the study of the hand skeleton model is very essential for developing any hand gesture recognition system. Focuses on various methods/techniques available for vision-based hand gesture recognition. This paper gave us insights on database creation like setting background and dealing with static and dynamic signs. It describes what a typical Hand Gesture Recognition system requires, what the different modules and the common challenges that used to arise. It states the main challenge in developing an accurate system is the lack of availability of a standard dataset.

C. Sign Language Recognition Using Modified Convolutional Neural Network Model [4]

Sign Language Recognition Using Modified Convolutional Neural Network Model is another piece of literature that we have reviewed. The goal of the paper is to implement the i3d inception model for Sign Language Recognition with the transfer learning method. From the test, they got 100 percent accuracy on training with 10 words and 10 signers with 100 classes but the validation accuracy was pretty low.

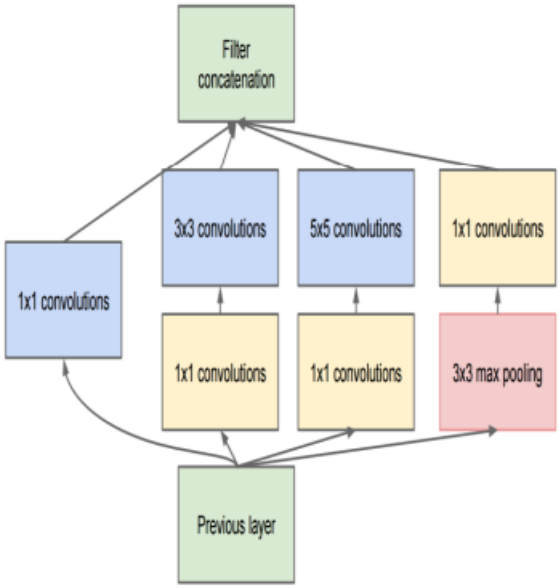


Fig. 4: Details of inception module.

The model was too overfit. They stated the reason for poor validation accuracy may be that the inception 3d model was used without any extra training or modification. It is possible to do a lot more things with this model, like freeze the layers, remove some inception modules, remove the transfer learning, and change the fully connected layer into another deep learning model. Another reason could be the problem of background and lighting conditions in the dataset.

D. Continuous Indian Sign Language Gesture Recognition and Sentence Formation [5]

Continuous Indian Sign Language Gesture Recognition and Sentence Formation was the next paper that we reviewed. In

that paper, they proposed a continuous Indian Sign Language (ISL) gesture recognition system where both hands are used for performing any gesture. As recognizing sign language gestures from continuous gestures is a very challenging research issue. They solved this problem using the gradient-based key frame extraction method. These key frames are helpful for splitting continuous sign language gestures into sequences of signs as well as for removing uninformative frames. After the splitting of gestures, each sign has been treated as an isolated gesture. Then features of pre-processed gestures are extracted using Orientation Histogram (OH) with Principal Component Analysis (PCA) applied for reducing the dimension of features obtained after OH.

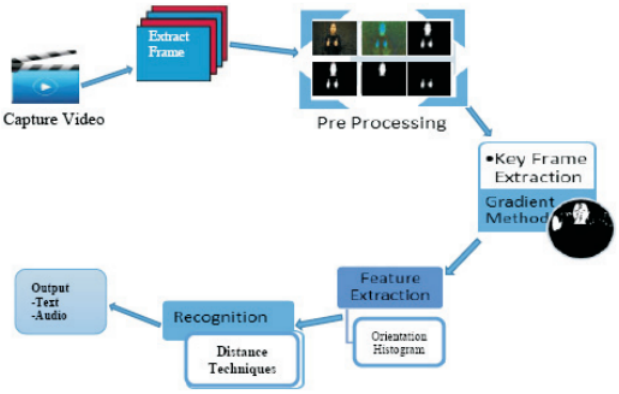


Fig. 5: General diagram of the proposed system in literature[4].

Experiments are performed on their own continuous ISL dataset which is created using a Canon EOS camera in Robotics and Artificial Intelligence Laboratory (IIIT-A). Probes are tested using various types of classifiers like Euclidean distance, Correlation, Manhattan distance, city block distance, etc. A comparative analysis of the proposed scheme was performed with various types of distance classifiers. From this analysis, they found that the results obtained from Correlation and Euclidean distance give better accuracy than other classifiers.

IV. System Architecture

. The system that we have developed is a mobile application named SAHAAYI which recognizes Malayalam signs alphabets and shows the corresponding alphabet in textual format. The system architecture can be modularised mainly into 3 - the training module, the testing module, and the application module.

A pre-trained CNN model called Squeezenet was used for training the model. The model was trained with 12000 labeled images.

The model was tested with the test dataset. The predictions made by the model were compared with the actual label. The parameters were adjusted and the performance of the model was evaluated each time.

In the application module, frames are captured from the camera, and preprocessing is done over these images. The features are extracted from the image and the model classifies the image. The label of the image is displayed in the application.

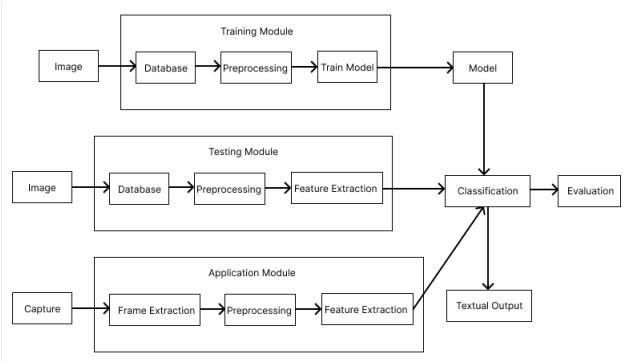


Fig. 6: System Architecture.

A. Dataset Creation

As there was no previous dataset for Malayalam Sign Alphabets, dataset creation was a challenging task for us. National Institute of Speech and Hearing(NISH) introduced around 62 signs for Malayalam letters including symbols in Malayalam [6]. Signs may require using a single hand or both hands or they could be static or dynamic signs.

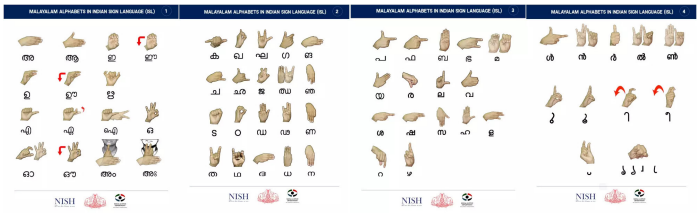


Fig. 7: Malayalam Alphabets in Indian Sign Language

The images were taken using a mobile camera to increase the quality of the image. The mobile camera was connected to the computer using IP Addresses [**?**]. Using OpenCV, frames were created and images were captured from the live camera. For each alphabet, we took 200 images. Thus around 12000 images were captured in the first phase. Initially, the background chosen was a plane wall of ivory color [10]. When using this, the model's accuracy was very low because the model could not distinguish the sign from the background. So the dataset was created again on a black background [11].

(a) Dataset in pale background. (b) Dataset in black background.

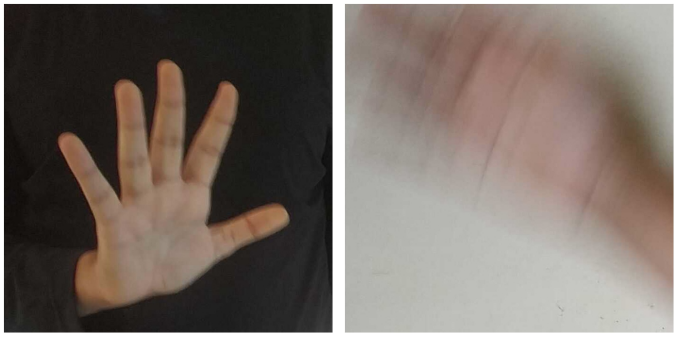
For creating the dataset, it is not important to analyze the entire window. We were only interested in capturing the

image of our hands, for this reason, we have defined a region of interest (ROI) and captured the image only in this area. cv2.imwrite() method in OpenCV is used to save an image to the device. This will save the image according to the specified format in the specified path.

```
if start:
    roi = frame[100:800, 100:800]
    save_path = os.path.join(IMG_CLASS_PATH, '{}.jpg'.format(count + 1))
    cv2.imwrite(save_path, roi)
    count += 1
```

Fig. 9: Dataset creation.

A new symbol other than the ones introduced by NISH was used to indicate **'Space'**. Also a background with no signs was introduced as **'Nothing.'**



(a) Space. (b) Nothing.

Fig. 10: Additional Signs

1) Challenges in Dataset Creation: There were a few challenges during dataset creation.

- **Labeling the data** - Labeling a folder in Malayalam was not supported by the system. So folders were labeled in English. Sometimes symbols along with text were also used for labeling. Example - Da, D_a, Dha, D_ha.
- **Mapping label to malayalam alphabets** - Not all text editors support typing in Malayalam.
- **Dynamic signs** - Since we use image classification, static images were used for training. To include dynamic signs we captured images of the ending position of the sign. We ensured that it does not resemble any other sign.

Fig. 11: Dynamic signs

But in some cases, the end positions were already used. In such cases, we omitted the alphabet.

Fig. 12: Omitted Signs

- **Ambiguity** - For some of the symbols, the National Institute of Speech and Hearing(NISH) itself has not been clarified. A single sign is used for multiple alphabets.

Fig. 13: Single sign for different symbols

B. Model Training

The model is trained with 12000 images captured in the black background.When trained in the local environment, it took a lot of time. So the training was done in Google Colab with the help of GPU. The model was trained over 50 epoches. [8]

1) Squeezenet Model: A pretrained CNN model named Squeezenet was used for training. The Squeezenet architecture comprises several filters. The first level comprises four 1x1 filters that are concatenated at the next layer. The concatenation ensures that the number of parameters is minimal. The primary objective of the Squeezenet architecture is to reduce the number of parameters, and in turn the size of the network. The concatenated layer is fed onto the expand layer and hence the number of interconnections between the squeeze layer and the expand layer is minimal. This ensures that the size of the network is low. The expand layer comprises 3x3 filters along with more 1x1 filters. These are concatenated to attain the result.

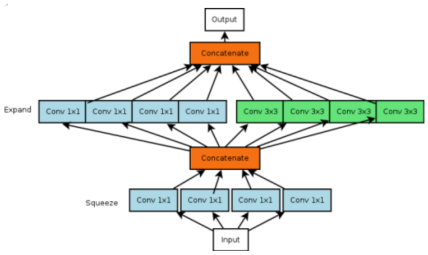


Fig. 14: Squeeqenet Architecture

2) Pre-Processing: Images collected in the dataset are pre-processed in order to fit into the Squeezenet architecture for training.

```
image_path = os.path.join(path,file_name)
# prepare the image
img = cv2.imread(image_path)
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
img = cv2.resize(img, (227, 227))
```

Fig. 15: Pre-processing

- **Converting BGR to RGB.** - OpenCV uses BGR image format. So, when we read an image using cv2.imread() it interprets in BGR format by default. cvtColor() method is used to convert a BGR image to the required RGB format.
- **The images are resized to 227,277.** - Squeezenet expects inputs of size 227x227.
- Each image is mapped to a label.

3) Model Parameters: The parameters passed to the squeezenet model are:

```
def get_model():
    model = Sequential([
        SqueezeNet(input_shape=(227, 227, 3),
        include_top=False),
        Dropout(0.5),
        Convolution2D(NUM_CLASSES, (1, 1),
        padding='valid'),
        Activation('relu'),
        GlobalAveragePooling2D(),
        Activation('softmax')
    ])
    return model
```

Fig. 16: Model Parameters

- **input_shape** - is an image of size 227 x 227 pixels. 3 is for RGB colour range.
- **include_top** - used to select if we want a final dense layer or not. Dense layers are capable of interpreting found patterns in order to classify images. Set to False as we have labeled what each data looks like already.
- **Dropout** - added to reduce overfitting, 0.5 = 50%, so throughout the training process drop 1/2 of what the resultant to keep the model learning new approaches and not become stale.
- **Convolution2D** - controls the size of the layer by the first argument NUM_CLASSES which is 60 in total. It is appended to the already defined neural network of 'SqueezeNet'.
- **Activation (ReLU)** - Rectified Linear Unit turns negative values into 0 and outputs positive values. An Activation function is responsible for taking inputs and assigning weights to output nodes, a model can't coordinate itself well with negative values, it becomes non-uniform in the sense of how we expect the model to perform.
f(x) = max(0, x)

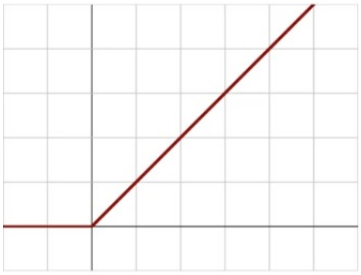


Fig. 17: ReLU function

The output of a ReLU unit is non-negative. Returning x, if less than zero then max returns 0. ReLU has become the default activation function for many types of neural networks because a model that uses ReLU is easier to train and often achieves better performance and good model convergence.

- **GlobalAveragePooling2D** - performs classification and calculates average output of each feature map in previous layer. i.e data reduction layer, prepares model for Activation('softmax').
- **Activation (softmax)** - gives probabilities of each hand sign. We have a 60 image class, 'softmax' handles multi-class, anything more than 2.

4) Training: The model is trained over 50 epochs.

```
model = get_model()
model.compile(
    optimizer=Adam(lr=0.0001),
    loss='categorical_crossentropy',
    metrics=['accuracy']
)
```

Fig. 18: Training

- **Optimizer** - Adam optimization is a stochastic gradient descent method that is based on adaptive estimation of first-order and second-order moments.
Learning_rate: A Tensor, floating-point value, or a schedule ie, tf.keras.optimizers.schedules.LearningRateSchedule, or a callable that takes no arguments and returns the actual value to use, the learning rate. Defaults to 0.001.
- **Loss** - The purpose of loss functions is to compute the quantity that a model should seek to minimize during training.
categorical_crossentropy: The output label is assigned one-hot category encoding value in form of 0s and 1.

The output label, if present in integer form, is converted into categorical encoding using Keras.

- **Metrix** - A metric is a function that is used to judge the performance of the model. Metric functions are similar to lose functions, except that the results from evaluating a metric is not used when training the model.
  Accuracy class - Calculates how often predictions equal labels. This metric creates two local variables, total and count that is used to compute the frequency with which y_pred matches y_true. This frequency is ultimately returned as binary accuracy: an idempotent operation that simply divides total by count. If the sample_weight is None, weights default to 1. Use sample_weight of 0 to mask values.

C. Model Testing

The model was tested using the test dataset. 10 images for each alphabet, summing up to 600 images were taken for testing. All the predictions made by the model were noted. The label is predicted for each image file path. The learning rate and epochs were adjusted based on the predictions to increase accuracy. Of these 600 images, 117 images were correctly predicted. Hence the accuracy of the system is 20%.

Visualisation was done to evaluate Underfitting or Overfitting and to adjust the Hyperparameters.

1) Accuracy VS Epoch Plot: Accuracy is the percentage of classifications that a model gets right during training. It is the measurement used to determine which model is best at identifying relationships and patterns between variables in a dataset based on the input, or training, data. If the model's prediction is perfect, the accuracy is one; otherwise, the accuracy is lower than one.

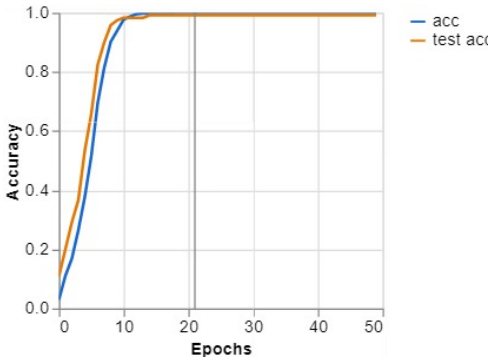


Fig. 19: Accuracy VS Epoch Plot

The gap between training and validation accuracy indicates overfitting. The larger the gap, the higher the overfitting. In our case, the gap between training and validation accuracy is minimum. Hence our model does not have any overfitting.

2) Loss vs Epoch Plot: Loss is a measure for evaluating how well a model has learned to predict the right classifications for a given set of samples. If the model's predictions are perfect, the loss is zero; otherwise, the loss is greater than zero.

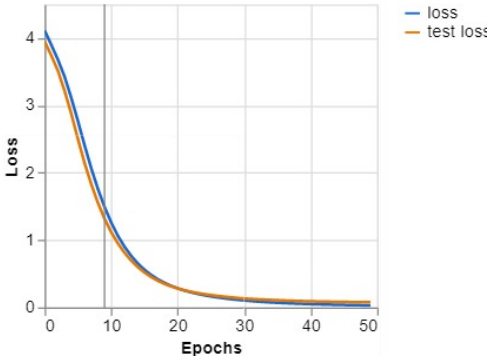
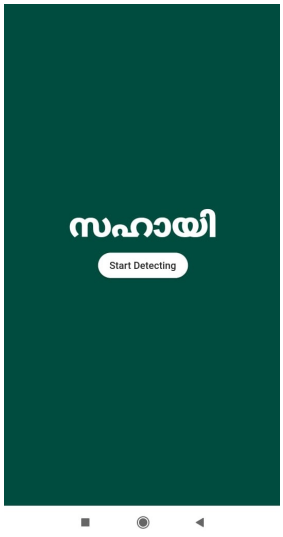


Fig. 20: Loss vs Epoch Plot

The plot of training loss decreased to a point of stability and then maintained 0 loss. Also the plot of validation loss decreased to a point of stability and has a small gap with the training loss. This implies that the model is a good fit.

D. Application Development

To increase the portability of the system, a mobile application was developed using Flutter and have named it "SAHAAYI". [9] Flutter is an open-source software development kit that enables smooth and easy cross-platform mobile app development. We can build high-quality natively compiled apps for iOS and Android quickly, without having to write the code for the two apps separately. The User Interface of the application was designed using Figma. The trained model was integrated into this application using the tflite plugin (tflite: Î1.1.2) which is a Flutter plugin for accessing TensorFlow Lite API [7].



(a) Screen 1.

(b) Screen 2.

Fig. 21: SAHAAYI Application.

V. Detailed Design

A. Flow Chart

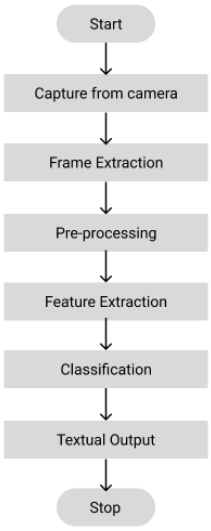


Fig. 22: Flow Chart

B. Use Case Diagram

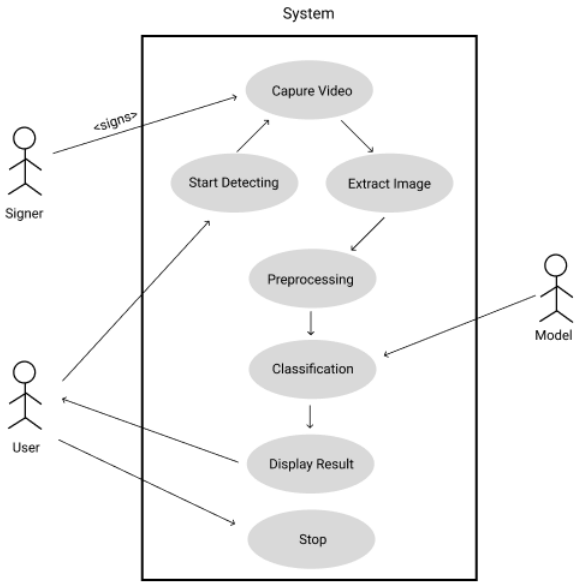


Fig. 23: Use Case Diagram

C. Sequence Diagram

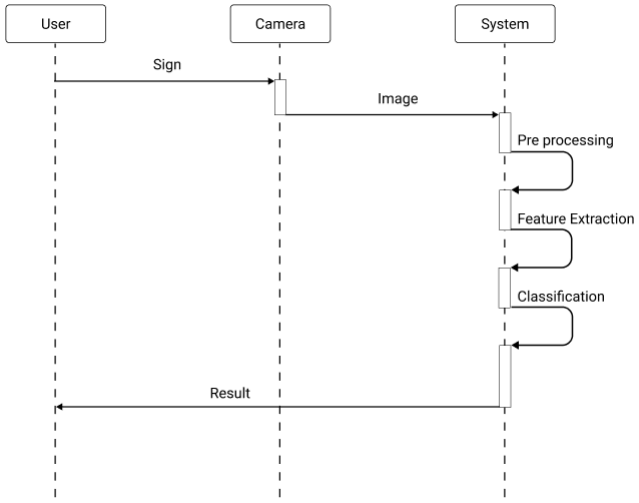


Fig. 24: Sequence Diagram

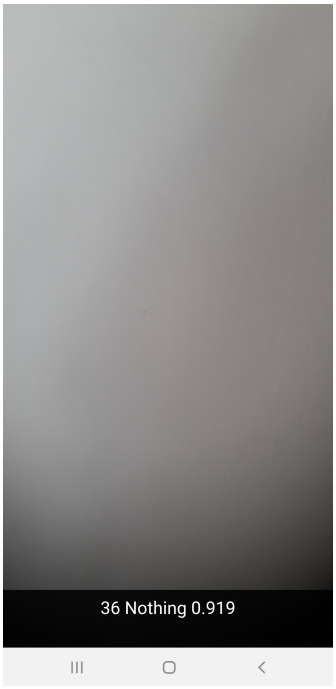
VI. Results and Analysis

A. Achievements

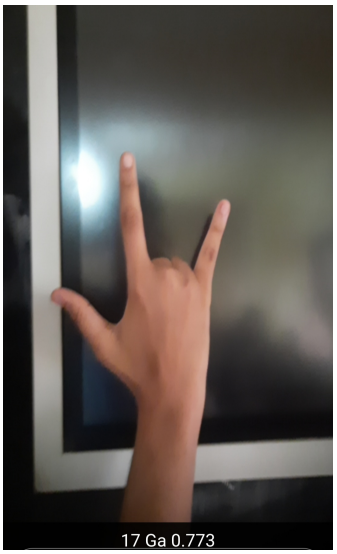
- Created a dataset of around 24000 images.
- Trained the model with satisfiable accuracy.
- Built the application using Flutter.
- Successfully integrated the model with the app.

(a) E

(b) Ee



(a) Nothing

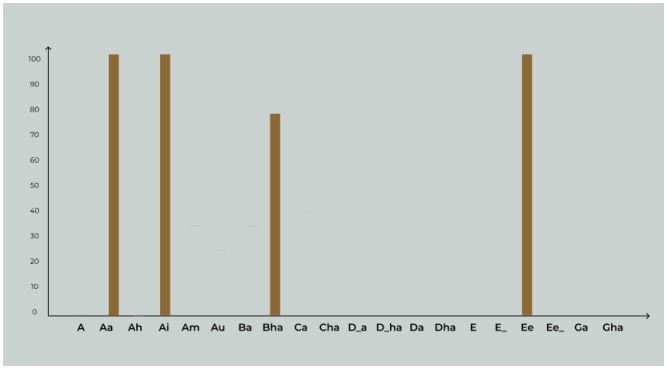


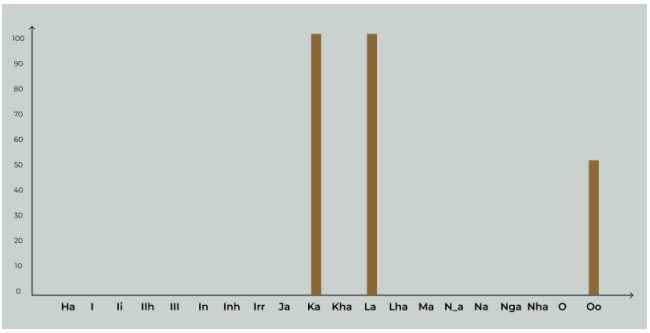
(b) Ga

Fig. 26: Successful results.

B. Test Results

Each letter was tested to check whether it predicts the right letter. Following figures show the result obtained. The x-axis shows the labels and the y-axis shows the percentage of correct prediction. 10 images for each alphabet were taken summing up to 600 images in total. These images were tested and out of these 117 of the images were correctly identified. This yields a total accuracy of 20





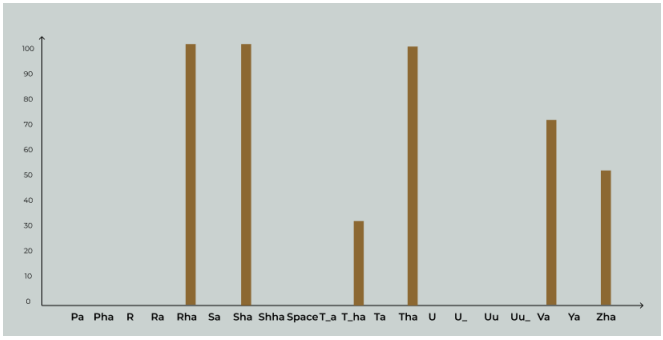


Fig. 27: Test results.

Misclassification - Some signs showed a close resemblance with each other and hence the model was unable to classify properly.

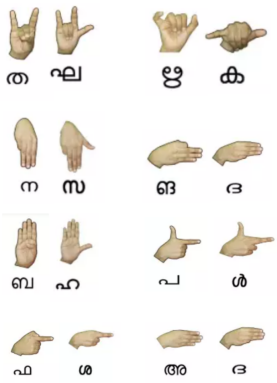


Fig. 28: Similarity in Signs

C. Prediction Table

Some alphabets were predicted correctly all the time while some others were wrongly predicted all the time.

TABLE I: Correct Predictions

Sl. No	True Label	Predicted Label
1	Aa	Aa
2	Ai	Ai
3	Ee	Ee
4	Ka	Ka
5	La	La
6	Nothing	Nothing
7	Rha	Rha
8	Sha	Sha
9	Tha	Tha

TABLE II: Wrong Predictions

Sl. No	True Label	Predicted Label
1	A	Aa
2	Am	U_
3	Ba	U
4	Dha	U_
5	E	Ee_
6	Ee	Ee_
7	Ga	U_
8	Ilh	U_
9	Ma	U
10	Na	Rha

D. Analysis

The model created was successfully integrated into the app, even though the accuracy seems to be not upto the expectation. The labels shown as the textual output fluctuates continuously.

This variation might be because the dataset provided is not sufficient enough for the deep learning model to learn from it. Another reason could be that images are captured in the same background and lighting conditions without trying varieties of background and lighting conditions. The close resemblance of the alphabet was also a challenge for the model.

1) Confusion Matrix: A confusion matrix summarizes how accurate the model's predictions are. It is used to figure out which classes the model gets confused about. The y axis (Class) represents the class of samples. The x-axis (Prediction) represents the class that the model, after learning, guesses those samples belongs to.

Confusion matrices represent counts from predicted and actual values. The output "TN" stands for True Negative which shows the number of negative examples classified accurately. Similarly, "TP" stands for True Positive which indicates the number of positive examples classified accurately. The term "FP" shows False Positive value, i.e., the number of actual negative examples classified as positive; and "FN" means a False Negative value which is the number of actual positive examples classified as negative. One of the most commonly used metrics while performing classification is accuracy.

Sha has the highest acuuracy, and U_ has the least accuracy.

TABLE III: Accuracy of each sign alphabet.

Sl. No	Label	Accuracy
1	A	0.983
2	Aa	0.97
3	Ah	0.981
4	Ai	0.983
5	Am	0.983
6	Au	0.983
7	Ba	0.98
8	Bha	0.996
9	Ca	0.983
10	Cha	0.983
11	Da	0.983
12	Dha	0.983
13	D_a	0.983
14	D_ha	0.983
15	E	0.983
16	Ee	0.983
17	Ee_	0.918
18	E_	0.983
19	Ga	0.983
20	Gha	0.978
21	Ha	0.983
22	I	0.983
23	Ii	0.983
24	Ilh	0.981
25	Ill	0.983
26	In	0.983
27	Irr	0.983
28	Ja	0.983
29	Ka	0.998
30	Kha	0.978
31	La	0.983
32	Lha	0.966
33	Ma	0.983
34	Na	0.983
35	Nga	0.98
36	Nha	0.983
37	Nothing	0.975
38	N_a	0.983
39	O	0.983
40	Oo	0.936
41	Pa	0.983
42	Pha	0.968
43	R	0.983
44	Ra	0.983
45	Rha	0.916
46	Sa	0.98
47	Sha	1
48	Shha	0.983
49	Space	0.975
50	Ta	0.983
51	Tha	0.923
52	T_a	0.933
53	T_ha	0.97
54	U	0.913
55	Uu	0.983
56	Uu_	0.983
57	U_	0.783
58	Va	0.995
59	Ya	0.983
60	Zha	0.938

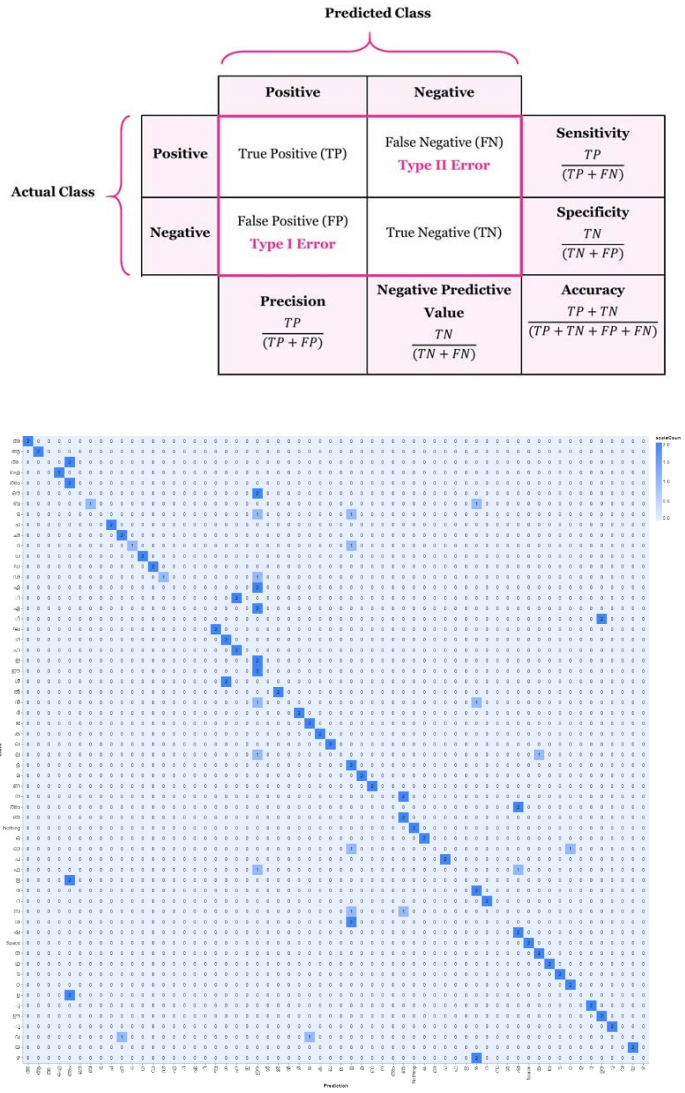


Fig. 29: Confusion Matrix

VII. Conclusion

The model was built successfully and was integrated into the application. The application hardly recognizes the signs since these signs were mostly similar. The accuracy of the system is 20%. Even though the accuracy of the system was less its able to predict some letters. Loss in accuracy may be due to less dataset used in deeplearning for model training. By standardising dataset by adding more images in better lighting may improve the result.

Developing the system into handling dynamic signs will elevate its application to more areas and will smoothen the communication process. The future scope of this project includes integrating this model with video conferencing softwares like Skype, Google Meet etc. It can be extended to incorporating gestures also. In future, the textual output can be modified to deliver as audio.

References

[1] Indian Sign Language Alphabet in Malayalam - Released https://nish.ac.in/index.php/others/news/882-indian-sign-language accessed on march 2022.

[2] Brandon Garcia and Sigberto Alarcon Viesca, Real-time American Sign Language Recognition with Convolutional Neural Networks, *1st International Cognitive Cities Conference (IC3). IEEE*,pp. 202–203,September(2018). pp.68–73.

[3] Ghotkar, Archana and Gajanan, K and Kharate, Gajanan, Study of vision based hand gesture recognition using indian sign language, *International journal on smart sensing and intillegent system*,vol.7,pp.96-115,March(2019).

[4] Suharjito, Suharjito and Gunawan, Herman and Thiracitta, Narada and Nugroho, Ariadi, Sign Language Recognition Using Modified Convolutional Neural Network Model,*13th international conference on computing communication technologies (iccct)*,pp.1-5,September(2018). .

[5] Kumud Tripathi, Neha Baranwal and G. C. Nandi, Continuous Indian Sign Language Gesture Recognition and Sentence Formation, *Procedia Computer Science*,vol.54,pp.523-531,December(2019).

[6] Malayalam sign language alphabets released by the Minister of Higher Education Dr. R Bindu https://www.reporterlive.com/newsroom/malayalam-sign-language-alphabets-released-by-the-minister-of-higher\\-education-dr-r-bindu-60289

[7] A Flutter plugin for accessing TensorFlow Lite API. https://pub.dev/packages/tflite accessed on may 2022

[8] https://github.com/sukrithamukundan/Sahaayi---Model-Creation

[9] https://github.com/sukrithamukundan/Sahaayi---Application

[10] https://github.com/sukrithamukundan/Sahaayi---Model-Creation/tree/master/image_data

[11] https://github.com/sukrithamukundan/Sahaayi---Model-Creation/tree/master/image_data_2

[12] https://play.google.com/store/apps/details?id=com.pas.webcam&hl=en_IN&gl=US