2021 2nd Global Conference for Advancement in Technology (GCAT)
Bangalore, India. Oct 1-3, 2021

SIGNify – A mobile solution for Indian sign language using MobileNet architecture

Mohammed Siddiq S
Department of Computer Science and Engineering
Dayananda Sagar University
Bengaluru, India
mdsiddiq.as@gmail.com

Roopashree S
Department of *Computer Science and Engineering*
(AI&ML) and (Cybersecurity)
Faculty of Engineering and Technology
Jain (Deemed-to-be-University)
Bengaluru, India
roopashaily@gmail.com

MusfirahSuha
Department of Computer Science and Engineering
Dayananda Sagar University
Bengaluru, India
musfirah.suha@gmail.com

MRS Ruthvik
Department of Computer Science and Engineering
Dayananda Sagar University
Bengaluru, India
rsruthvikgtl@gmail.com

KuruvaDivyasree
Department of Computer Science and Engineering
Dayananda Sagar University
Bengaluru, India
divyasree6898@gmail.com

2021 2nd Global Conference for Advancement in Technology (GCAT) | 978-1-6654-1836-2/21/$31.00 ©2021 IEEE | DOI: 10.1109/GCAT52182.2021.9587639

***Abstract* — Sign language assists in visual communication for the vocal and hearing-impaired population. It benefits people with other disabilities such as autism, down syndrome etc. The work recommends an intelligent system specifically to recognize the Indian Sign Language (ISL). It is an interesting and challenging problem, as the solution brings a leap in both social and technological aspects. A signer independent methodology based on different techniques of deep learning, is a real-time recognition system for Indian sign language developed to avoid the isolation of disabling groups from the rest of the society. In our work, we build and compare two pre-trained CNN models, MobileNet and InceptionV3 architectures. The suggested approach using the MobileNet model showed an accuracy of 99% on a custom-built dataset. The working CNN model can perform real-time recognition of ISL alphabets, numbers and a few selected gestures integrated into an Android mobile platform called SIGNify using React-Native, for better accessibility and user-friendly access. The study highlights a small step involved in human-computer interaction.**

Keywords— Android, Indian Sign Language, InceptionV3, MobileNet, React native, Sign language recognition system

I. I. INTRODUCTION

The main disadvantage of the people with disabilities is lack of access to good education, proper health and social interactions. Approximately 70 million of the population across the globe utilize sign language. Hearing or speech disable are mainly dependent on sign language as a kind of interaction [1]. Developing a novel interface using machine learning for both one-hand and two-hand signs and few gestures for effective communication is proposed in the work.

In India, the chief mode of communication for the hearing and vocal impaired is Indian Sign Language (ISL). Indian sign language consists of two-hand gestures. The ISL comprises of facilitating the hand movements, facial expressions and body language. The paper recognizes the static signs and few gestures of ISL alphabet, numerals and gestures [2]. The signs include 26 English alphabet letters, 0-9 numerals and some gestures such as thick, thin, together etc. Recommending a real-time system for recognition of ISL using the mobile platform will surely achieve better accessibility and user friendliness. A deep learning model using convolution neural networks built by implementing the transfer learning approach has proved to be an efficient solution for ISL.

II. RELATED WORKS

A 35,000 Indian sign image collection comprises of around 100 static signs gathered from various users, is used to develop an architecture based on CNN which consist of 3 convolutional layers followed by maxpool, dropout and two fully connected layers. Each convolutional layer includes many filtering window sizes which assist to improve the recognition speed and accuracy [2]. The accuracy of training is 99.17% and validation is 98.80% using 4 layers with 16 filters and Adam optimizer.

The hand gesture recognition [3], a real time system captured data using the Microsoft Kinect RGBD camera. The Convolutional Neural Networks (CNNs) are trained for 36 static gestures of ISL numbers and alphabets. The obtained training accuracy of the model is 98.81% from 45,000 RGB and depth images. The model showed good flexibility to gestures of American Sign Language (ASL) giving 97.71% accuracy.

The paper [4] intends to benchmark on machine learning algorithms for Two Hand Indian Sign Language (THISL) dataset made up of 26 gestures of English alphabets. This dataset has total of 9100 images with each image of size 50x50 and 350 images for each gesture. The THISL dataset validated on different machine learning classification models showcasing an accuracy of 91.72%.

Binary silhouette hand region is used in the work [5] as input to the CNN. A custom dataset of 2500 images are built using 7different signers. The CNN architecture withsix hidden layers including dropout and output layer resulting with an accuracy of 98.64%. However, it predicts the saved images and not from external camera. A new system proposed [6] show success in recognizing the American Sign Language (ASL) which can interpret ASL alphabets in a real time environment. The dataset has 27,455 images including 24 ASL alphabet

signs. The model is a mobile application that shows95.03% accuracy and 2.42 seconds of recognition time.

The above survey reveals that no major contributions are found on ISL using transfer learning approach in deep learning. As transfer learning is an efficient method when the dataset is of limited size. Our projected work proposes a new system to classify the real-time ISL signs with a good accuracy.

A. Deep Learning

The sub-domain of machine learning is deep learning where a model performs image classification including sound or text classification. The deep learning models have the capability to exceed the performance of humans. The models are trained on large labelled dataset and neural network architecture consisting of many layers. Many fields in computer vision area including recognition of speech, machine vision and Natural Language Processing (NLP) showcase comparable results surpassing the performance of human experts. The deep belief network, deep neural network, CNN and recurrent neural network are some of the implementations of deep learning architecture [7].

B. Transfer Learning (TL)

To build any CNN from scratch requires a large, high-quality dataset for any domain problem. Transfer learning in deep learning provides a solution to the problem where the CNN can be built for a limited training dataset. In deep transfer learning, there is no necessity to build CNN from scratch, but rather use as source model the deep pre-trained CNN models such as VGG19, VGG16, Inception V3, MobileNet to learn the features and classify the domain specific dataset using the target model. Fig. 1 shows the basic flow of transfer learning. Hence, transfer learning approach provides a positive response to domains with insufficient train data but yet produces good accuracy in classification [6].

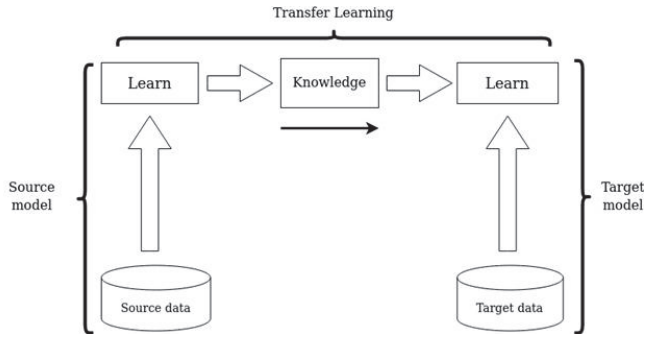

Fig. 1. Basic flow of transfer learning

C. Convolution Neural Networks (CNN)

The CNN is one of the deep learning algorithms which accepts the input images and differentiate one from another. It reduces the images to assist for easy process, without eliminating any important features required for prediction [8]. CNN used for classification of image, consists of one input layer, many hidden layers and an output layer as depicted in Fig. 2. The hidden layers consist of convolutional and pooling layers along with ReLU (activation function), normalization and fully connected layers. CNN showcases good results with limited preprocessing of the raw data and increased training set [9].

The edges are the high-level features extracted from any input image is the main objective of the convolution layer. The spatial size (convolved feature) is reduced by the pooling layer which decreases the computing power needed to process data by reducing the dimension. The two categories of pooling are max and average pooling. Maxpooling reveals the highest value of image portion covered by the kernel and average pooling returns average values of image portion covered by kernel [8]. Each neuron (in every layer) is connected to all the neurons in other layer in fully connected layers. Softmax is one of the best activation layers that is used to build models using CNN along with ReLU activation function [10]. The Softmax activation function transforms all inputs in the range of 0 to 1 to interpret the values in terms of probabilities. The ReLU activation function converts all negative inputs to zero. The best optimizer-*Adam* is used to minimize errors which uses an adaptive learning rate stochastic optimization algorithm.

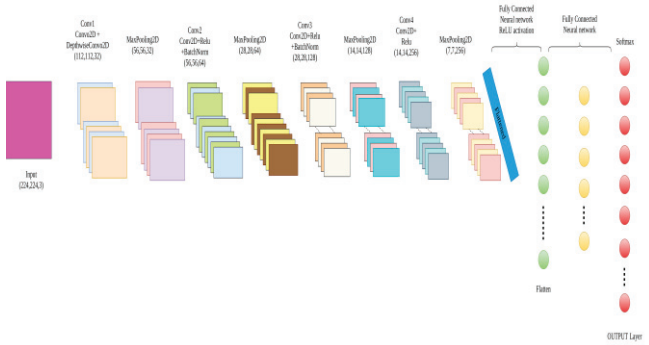

Fig. 2. General architecture of CNN

D. Inception V3

A model generally used for image recognition is Inception V3 architecture with 42 layers of deep neural network shows good accuracy rate on the ImageNet dataset. It has asymmetric and symmetric blocks involving convolutions, fully connected layers, average pooling, max pooling layers, and dropouts. The Inception V3 architecture is exemplified in Fig. 3 [11].

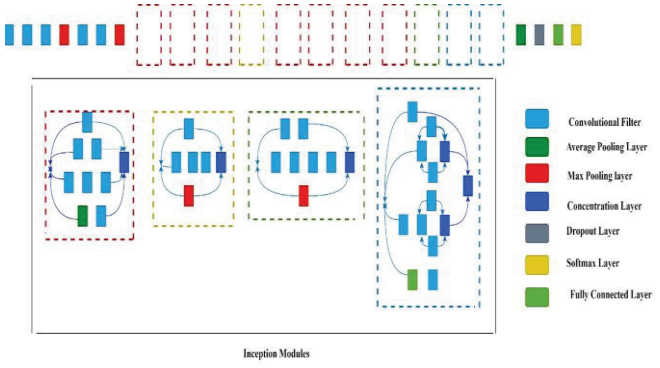

Fig. 3. General Inception-v3 architecture

E. MobileNet

MobileNet incorporates the usage of depth-wise separable convolutions in order to build lightweight deep networks. The specialty of MobileNet includes less computing power to implement transfer learning method. Hence, it can be good fit into embedded systems, mobile devices and any computers with low computational efficiency or GPU without compromising on the accuracy.

Depth wise separable convolutions is the core layer of MobileNet, a factorized convolution where standard convolution is factorized into depth-wise and pointwise convolution. A filter applied to every channel of input using depth-wise convolution followed by pointwise convolution [12].

ReLU and batch-norm follows the MobileNet layers and finally for classification into a Softmax layer. Fig. 4 shows convolutions, batch-norm and ReLU layer with pointwise convolution and depth-wise convolution. Stride convolution in depth-wise convolution handles down sampling. Spatial resolution is reduced to 1 by final average pooling before the fully connected layer. MobileNet consists of 28 layers keeping the convolutions of depth-wise and pointwise as separate layers [12].

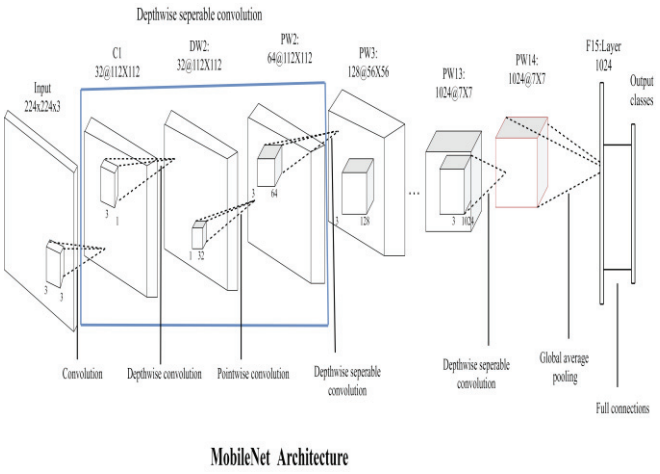

Fig. 4. General MobileNet architecture

F. React-Native

React-Native is a mobile framework by Facebook. It is an open-source and by using this framework, we can develop applications for Android, IOS, mac-OS, Android TV etc. React native permits the users to write the code in languages such as Java for android and Object-C or swift for IOS. React Native uses the JavaScript library Tensorflow.js (TF.js) developed by google for training and using ML models in the mobile application. In Tensorflow.js, the trained model is used by converting it to JSON file [13].

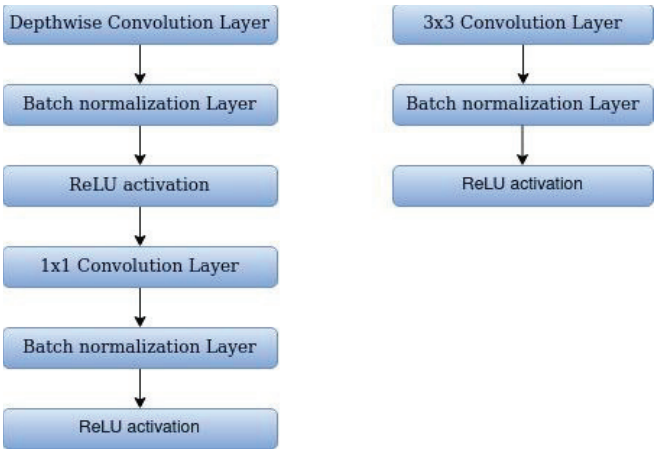

Fig. 5. Left: Depth-wise separable convolutions followed by batch normalization and activation function i.e., ReLU. Right: Standard convolutional layer along with batch normalization layer and ReLU

G. TensorFlow and Keras

TensorFlow is an open-source in Machine Learning (ML) platform which mainly focuses on training and interference of deep neural networks. TensorFlow is a differential programming that has tools, libraries and community resources. ML powered applications can be built and deployed easily. The architecture is so flexible to deploy across various platforms such as TPUs, GPUs and CPUs, from desktops to cluster of servers, mobile app and edge devices.

Keras is the top most running machine learning platform written in Python. Keras is an API of deep learning which focuses on experimenting at a faster rate. Keras is the most highly productive interface and an approachable high-level API of TensorFlow for solving machine leaning problems. Machine learning solutions are provided by Keras as it acts as a building block for developing and shipping ML solutions [14].

III. METHODOLOGY

A. Dataset

We built a new custom dataset for training and testing the proposed model, as no standard ISL dataset is available. The proposed dataset uses both hands to represent every alphabet, numbers and some hand gestures. The signed images of 26 alphabets [A-Z], 10 numbers [0-9] and few gestures are taken with a black background. Each sign image is captured using the hands of different persons. A total of 60 images for each classis procured, the dataset contains 2,820 images of 47 classes. Sampling of the dataset is shown in Fig. 6. These gestures are captured in front of a black background using a black cloth to distinguish between foreground and background. The training set consists of 2126 images and testing set with 707 images as shown in Table I.

Fig. 6. Samples of the proposed dataset used for building the model

TABLE I. THE TRAINING AND TESTING DATASET SIZE

Description	Number of images
No. of images in training set	2126
No. of images in testing set	707

B. Proposed System

We propose a mobile application that recognizes Indian sign language (ISL) in real time which acts as a communication medium for communicating with hearing/vocally impaired. The model is trained with Indian sign language and then loaded to the application. Once the trained model is loaded, it prompts the user to either upload the image directly from the gallery/camera roll or to capture the sign language in real time or recognition of Indian sign language. To upload an image from the gallery, the user should grant permission to access photo, media and files else if the user wishes to capture the sign language in real time, the proposed application asks permission to open the camera. The input image is resized to 224 x 224 x 3 and fed for further pre-processing and converted to image tensors. These image tensors are tested against the trained model for prediction. Finally, the class of the image is depicted on the screen as shown in Fig. 7.

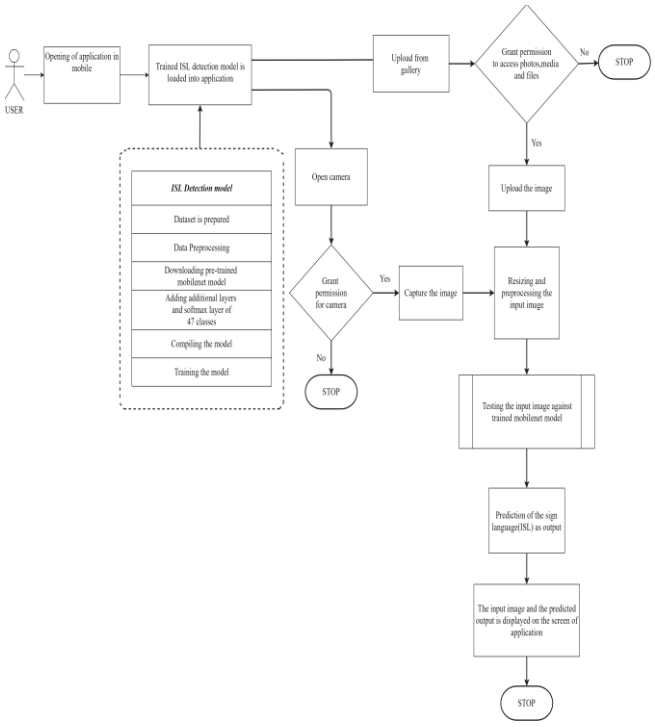

Fig. 7. Flow chart for the proposed system

1) Proposed Architecture

The proposed model shown in Fig. 8 has 4 phases, namely preprocessing, model creation, training and deployment. Initially, the input image is read by the model from the user. The image is then resized to 224 X 224 and normalized. All the images in the dataset are split into 75:25 ratio to train and test. In creating the model phase, the input data is loaded and the model is built by adding fully connected layers, ReLu and Softmax to the final layer for classification. In the train phase, the model is compiled using Adam optimizer and the model is finally saved for testing. The saved model is deployed into the android application which preprocess and labels the query sign images deployed in SIGNify app.

2) Data Preprocessing

First step in all vision-based sign language recognition systems is data pre-processing, the sign images are pre-processed and the image is resized to 224 x 224 x 3. We use preprocess input, which is an argument of the ImageDataGenerator () function used to pre-process a tensor or NumPy array encoding a batch of images and is meant to adequate images to the format the model requires. A powerful tool generally used to augment and feed the images into the model is ImageDataGenerator() [15]. Therefore, the dataset gets split into training data and validation data, in our model the validation split is 0.25. The ImageDataGenerator is used to set up the input feed data into a directory structure.

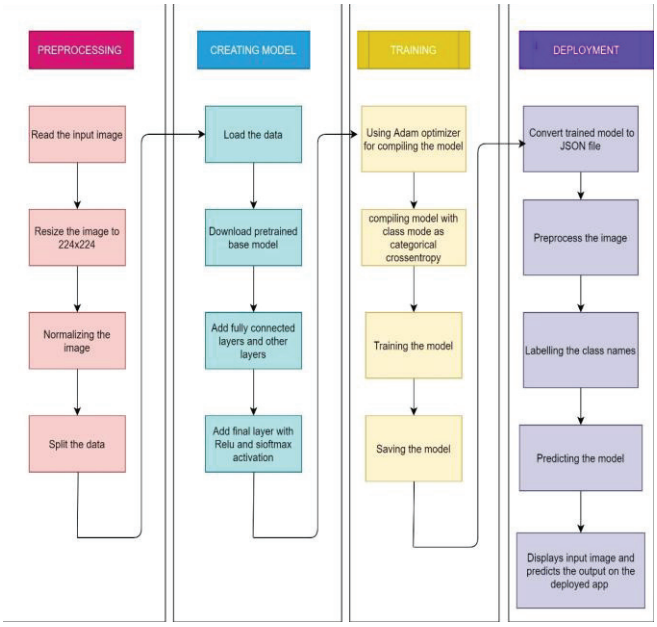

Fig. 8. General architecture of proposed system

3) Loading Pre-trained Model

The pre-trained model is loaded using a transfer learning approach which can lower generalization error, for a neural network model it has the benefit of decreasing the training time. The pre-trained model is trained by huge data sets. Hence, resource utilization improves the recognition and related computer vision tasks by reducing the training costs [16]. The trained model is loaded using TensorFlow as a backend. Combination of ImageDataGenerator and the approach of transfer learning resulted in drastic reduce of the data required to train and also the time required to set up.

MobileNet model is easy and efficient to deploy in embedded and mobile devices. Hence, we propose MobileNet model for classification of Indian sign language using transfer learning method. The work compares the Inception V3 and MobileNetmodel.

4) Adding additional layers to pre-trained models

A unique approach is adopted in this work by adding layers, global average pooling and Softmax activation layer, hence this approach is based on Global Average Pooling (GAP).

Global average pooling enforces the correspondence between categories and feature map. The GAP ensures easy interpretation of feature maps as categories of confidence maps. Overfitting is avoided in the global average pooling. Furthermore, GAP is robust to input spatial translations, as it sums out the spatial information [17].

Therefore, first we give input size (224x224) to the MobileNet, as a base additionally the GAP layer 2D (2X2) filter is performed for spatial data and reduce the dimension of data, two dropout layers randomly select neurons with a regularization rate of 0.25 during the training, and finally dense layer is FC (Fully Connected) layer for all neurons connected to the next output nodes, therefore in our approach we used 3 dense layers as shown in Fig. 9.

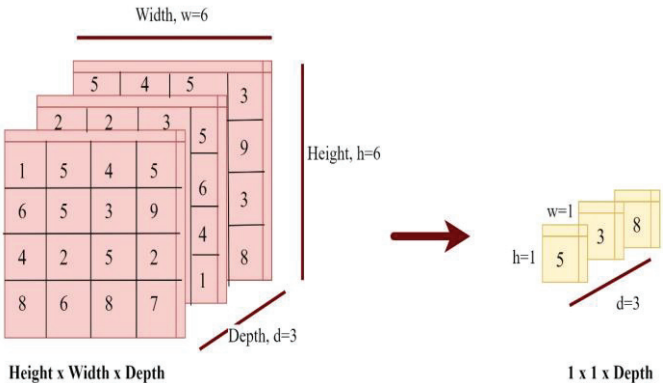

Fig. 9.A 2D Global average pooling operation. Here 'Depth' = 'Filters'

Next, the learnable parameters such as activation function (ReLU, Softmax) are applied to obtain feature map and loss function, optimization algorithms are deployed to classify images. After training the model, the layers have been created as in Fig. 10.

Anew model is proposed performing under different hyper-parameters namely filter size, number of kernels in all input channels, number weights, pooling. Activation function translates the input range and calculates the loss function with the help of forward propagation on the training images. Batch normalization and ReLU are applied after every convolution channel, for preventing the overfitting problems epochs and data augmentation techniques are implemented (ImageDataGenerator). GAP layers act as more depth type of reduced dimensionality as shown in Eq. 1.

$$H\text{imp} * W\text{imp} * D\text{imp} \quad (1)$$

Where Himp is height, Wimp weight and D is dimension. By handling all HW average values, GAP layers decrease by a single number ($1\times1\times D$) of each H*W feature map as shown in Fig. 9. 1024 neurons dropout with a regularization rate of 0.25 during the training. The convolution layer features are extracted from the dense layers and down sampled by the GAP layer. In a fully connected layer (dense layer) every input node is connected to the output node [18].

5) Training the model

The categorical cross-entropy loss function is minimized, as it is a multi-class classification problem. We instantiated an optimizer before passing it to model.compile(). Here the model is compiled with an optimizer known as Adam optimizer with the default learning rate [19].

The model proposed is trained using a GPU with 8 GB RAM for the dataset of Indian sign language. The model accepts a sign image which is preprocessed and classifies to its corresponding label. In this training phase, the data splits into two components, the training and testing set. The train set contains the images used for training and the test set for testing.

The model.fit() function takes the train and test datasets as arguments and returns the fit model and training history. Therefore, we have trained our model with batch_size = 64, 10 epochs and 33 steps per epochs. Then the model is saved for deploying using model.save().

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 224, 224, 3)]	0
conv1 (Conv2D)	(None, 112, 112, 32)	864
conv1_bn (BatchNormalization	(None, 112, 112, 32)	128
conv1_relu (ReLU)	(None, 112, 112, 32)	0
conv_dw_1 (DepthwiseConv2D)	(None, 112, 112, 32)	288
conv_dw_1_bn (BatchNormaliza	(None, 112, 112, 32)	128
conv_dw_1_relu (ReLU)	(None, 112, 112, 32)	0
conv_pw_1 (Conv2D)	(None, 112, 112, 64)	2048
conv_pw_1_bn (BatchNormaliza	(None, 112, 112, 64)	256
conv_pw_1_relu (ReLU)	(None, 112, 112, 64)	0
conv_pad_2 (ZeroPadding2D)	(None, 113, 113, 64)	0
conv_dw_2 (DepthwiseConv2D)	(None, 56, 56, 64)	576
conv_dw_2_bn (BatchNormaliza	(None, 56, 56, 64)	256
conv_dw_2_relu (ReLU)	(None, 56, 56, 64)	0
conv_pw_2 (Conv2D)	(None, 56, 56, 128)	8192
global_average_pooling2d (Gl	(None, 1024)	0
dense (Dense)	(None, 1024)	1049600
dropout (Dropout)	(None, 1024)	0
dense_1 (Dense)	(None, 512)	524800
dropout_1 (Dropout)	(None, 512)	0
dense_2 (Dense)	(None, 47)	24111

Total params: 4,827,375
Trainable params: 4,805,487
Non-trainable params: 21,888

Fig. 10. Output of the Proposed Use case trained model architecture

6) Deploying model to mobile application (SIGNify)

We deploy the model to the mobile platform, aiming that the project will be user friendly. The mobile application is developed using ReactNative framework. React Native uses the JavaScript library Tensorflow.js (TF.js) for machine learning (ML) models. The trained model has to be converted to a JSON file for deploying it to Tensorflow.js. The initial step to start TensorFlow in React Native is to integrate TF.js in an EXPO app and platform adapter. The platform adapter module is tfjs-react-native which supports loading tfjs models and provides GPU support.

Testing to ensure the successful load of tfjs into the app before rendering of the app. This is accomplished by an asynchronous function known as tf.ready()[20].

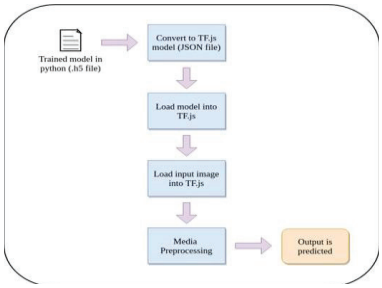

Fig. 11. Setting up the model for mobile environment

Fig.11 shows the converted TF.js model loaded well before providing the input image. The input image is retrieved by capturing the sign language in real time or uploading from the gallery/camera roll. Once the image gets uploaded or captured, it is resized to 224 x 224 x 3 and fed for feature extraction. Now, the raw image data is converted to image tensors. The image tensors are normalized by changing the pixel intensity range values [15]. The image is then displayed on screen, later when the user clicks on *predict* button, the image gets tested against the trained model and the sign language is predicted and displayed on the screen.

IV. DISCUSSION OF RESULTS AND ITS SIGNIFICANCE

A. Comparison of results

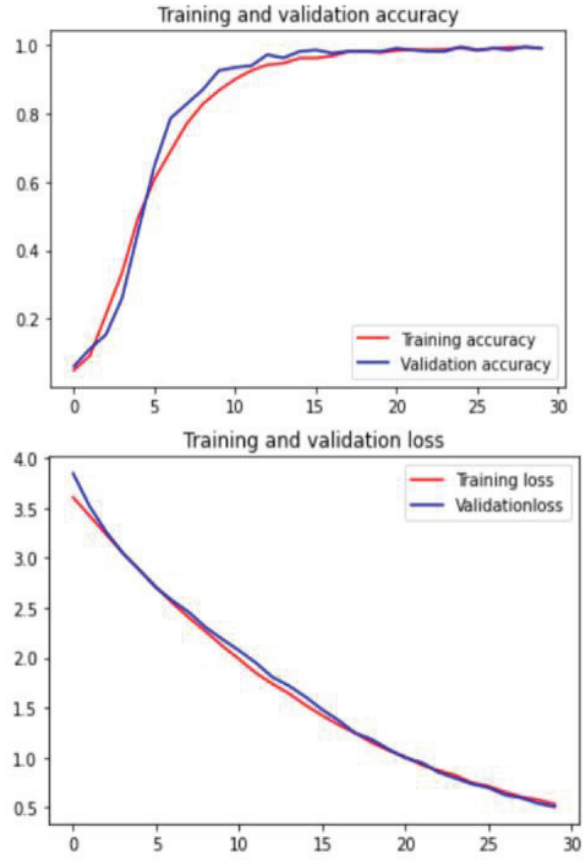

Fig. 12. Training and Validation graph of InceptionV3

Fig. 12 represents the training and validation graph of the InceptionV3 model. On training the model, it depicts the training accuracy and validation accuracy which is 98% and 99% respectively. Fig. 13 represents the training and validation graph of the MobileNet model. The model depicts 99% accuracy for both training and validation.

Table II shows that the MobileNet model performs better than Inception V3. The training of the models is performed in Google Colab with the following configurations i.e., Intel GPU, 8GB RAM on laptop hardware.

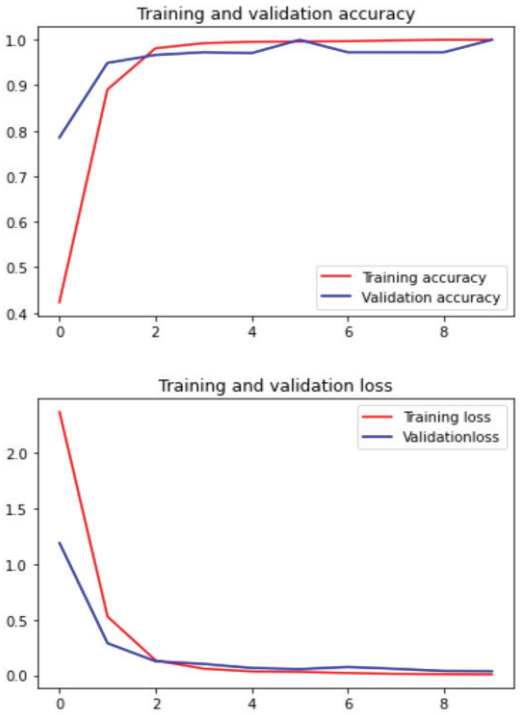

Fig. 13. Training and Validation graph of MobileNet

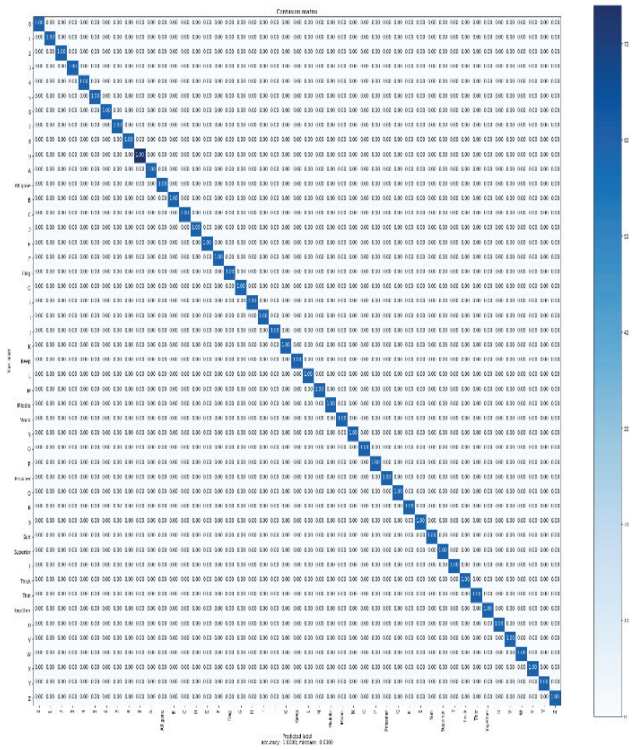

Fig. 14. Represents the confusion matrix of MobileNet

TABLE II. MODEL PERFORMANCE METRICS - ACCURACY AND TIME

Models-Accuracy/Time	Inception V3	MobileNet
Training accuracy	98%	99%
Testing accuracy	99%	99%
Training time (seconds)	6630	3000
Testing time for 707 images (Validation testing) (seconds)	25	20
Testing Time for one image (Real time testing) (seconds)	4	2

B. *Performance measures*

A classification report is used to measure the quality of predictions from a classification algorithm. The classification report states that the weighted average of F1 score is better in MobileNet than inception v3, i.e., 1.00 in MobileNet and 0.02 in inception V3 as shown in Fig.15 and Fig. 16. Therefore, as not difficult to satisfy, the weighted average of F1 should be used to analyze the classifier models, not global accuracy. Hence MobileNet outperforms inception v3.

Classes	precision	recall	f1-score	support
0	0.00	0.00	0.00	15
1	0.07	0.07	0.07	15
2	0.00	0.00	0.00	15
3	0.00	0.00	0.00	15
4	0.00	0.00	0.00	15
5	0.00	0.00	0.00	15
6	0.00	0.00	0.00	15
7	0.07	0.07	0.07	15
8	0.00	0.00	0.00	15
9	0.00	0.00	0.00	18
A	0.00	0.00	0.00	15
All gone	0.00	0.00	0.00	15
B	0.00	0.00	0.00	15
C	0.00	0.00	0.00	15
D	0.00	0.00	0.00	15
E	0.07	0.07	0.07	15
F	0.00	0.00	0.00	15
Flag	0.00	0.00	0.00	15
G	0.07	0.07	0.07	14
H	0.00	0.00	0.00	15
I	0.00	0.00	0.00	15
J	0.07	0.07	0.07	15
K	0.00	0.00	0.00	15
Keep	0.00	0.00	0.00	15
L	0.00	0.00	0.00	15
M	0.00	0.00	0.00	15
Middle	0.00	0.00	0.00	15
Moon	0.00	0.00	0.00	15
N	0.00	0.00	0.00	15
O	0.00	0.00	0.00	15
P	0.00	0.00	0.00	15
Prisoner	0.07	0.07	0.07	15
Q	0.00	0.00	0.00	15
R	0.00	0.00	0.00	15
S	0.07	0.07	0.07	15
Sun	0.00	0.00	0.00	15
Superior	0.07	0.07	0.07	15
T	0.07	0.07	0.07	15
Thick	0.00	0.00	0.00	15
Thin	0.07	0.07	0.07	15
Together	0.07	0.07	0.07	15
U	0.00	0.00	0.00	15
V	0.00	0.00	0.00	15
W	0.00	0.00	0.00	15
X	0.07	0.07	0.07	15
Y	0.00	0.00	0.00	15
Z	0.00	0.00	0.00	15
accuracy			0.02	707
macro avg	0.02	0.02	0.02	707
weighted avg	0.02	0.02	0.02	707

Fig. 15. The classification report of InceptionV3 architecture

Classes	precision	recall	f1-score	support
0	1.00	1.00	1.00	60
1	1.00	1.00	1.00	60
2	1.00	1.00	1.00	60
3	1.00	1.00	1.00	60
4	1.00	1.00	1.00	60
5	1.00	1.00	1.00	60
6	1.00	1.00	1.00	60
7	1.00	1.00	1.00	60
8	1.00	1.00	1.00	60
9	1.00	1.00	1.00	74
A	1.00	1.00	1.00	60
All gone	1.00	1.00	1.00	60
B	1.00	1.00	1.00	60
C	1.00	1.00	1.00	60
D	1.00	1.00	1.00	60
E	1.00	1.00	1.00	60
F	1.00	1.00	1.00	60
Flag	1.00	1.00	1.00	60
G	1.00	1.00	1.00	59
H	1.00	1.00	1.00	60
I	1.00	1.00	1.00	60
J	1.00	1.00	1.00	60
K	1.00	1.00	1.00	60
Keep	1.00	1.00	1.00	60
L	1.00	1.00	1.00	60
M	1.00	1.00	1.00	60
Middle	1.00	1.00	1.00	60
Moon	1.00	1.00	1.00	60
N	1.00	1.00	1.00	60
O	1.00	1.00	1.00	60
P	1.00	1.00	1.00	60
Prisoner	1.00	1.00	1.00	60
Q	1.00	1.00	1.00	60
R	1.00	1.00	1.00	60
S	1.00	1.00	1.00	60
Sun	1.00	1.00	1.00	60
Superior	1.00	1.00	1.00	60
T	1.00	1.00	1.00	60
Thick	1.00	1.00	1.00	60
Thin	1.00	1.00	1.00	60
Together	1.00	1.00	1.00	60
U	1.00	1.00	1.00	60
V	1.00	1.00	1.00	60
W	1.00	1.00	1.00	60
X	1.00	1.00	1.00	60
Y	1.00	1.00	1.00	60
Z	1.00	1.00	1.00	60
accuracy			1.00	2833
macro avg	1.00	1.00	1.00	2833
weighted avg	1.00	1.00	1.00	2833

Fig. 16. Represents the classification report of MobileNet

C. *Real time implementation and output*

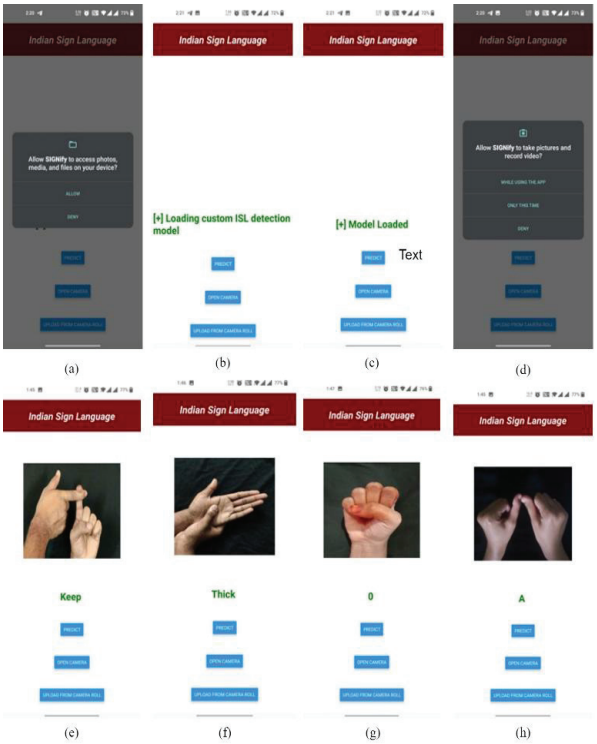

Fig. 17. Screenshots of the SIGNify application

The Fig.17 shows the screenshots of the proposed SIGNify application on a mobile platform. Once the application is launched, it prompts for the user permission to access photos, media and files from the mobile device. If the permission is granted, the trained ISL detection model is loaded. The SIGNify app also provides an option to the user either to capture the image in real time or access images from the device. Once the image is fed, the app predicts the sign language and displays the label of it.

V. CONCLUSION AND FUTURE WORK

An intelligent system integrated into a mobile platform as SIGNifyuses deep learning techniques for Indian sign language. The system can interpret any of the Indian alphabets, numbers and basic word signs. It acts as visual communication for the vocal and hearing impaired and helps to communicate technically and socially. The custom dataset built exclusively for ISL used to implement InceptionV3 and MobileNet deep learning architecture to predict the real-time images of the sign language. Both the deep models are trained

successfully on 26 ISL static alphabets, 0 to 9 numbers and 11 gestures such as all gone, flag, keep, middle, moon, prisoner, sun, superior, thick and thin with an accuracy of 85% for inceptionV3, and more than 99% accuracy for MobileNet. MobileNet model outperforms inceptionv3 in a real environment using a mobile application. Hence, the findings prove that the MobileNet model suggested can be integrated into any mobile device as it is lightweight architecture and predicts accurately with few epochs.

The mobile application proposed can further be extended to include more features to automate the process of recognizing the Indian sign language. The new features to be incorporated infuture include facial expression, context analysis and speech translator for voice output as a powerful assistive technology.

REFERENCES

[1] Wadhawan, Ankita, and Parteek Kumar. "Deep learning-based sign language recognition system for static signs." *Neural Computing and Applications* 32, no. 12 (2020): 7957-7968.

[2] Sharma, Madhuri, Ranjna Pal, and Ashok Kumar Sahoo. "Indian sign language recognition using neural networks and KNN classifiers." ARPN Journal of Engineering and Applied Sciences 9, no. 8 (2014): 1255-1259.

[3] Bhagat, Neel Kamal, Y. Vishnusai, and G. N. Rathna. "Indian Sign Language Gesture Recognition using Image Processing and Deep Learning." In *2019 Digital Image Computing: Techniques and Applications (DICTA)*, pp. 1-8. IEEE, 2019.

[4] Mangamuri, Leela Surya Teja, Lakshay Jain, and Abhishek Sharmay. "Two hand Indian sign language dataset for benchmarking classification models of machine learning." In *2019 International Conference on Issues and Challenges in Intelligent Computing Techniques (ICICT)*, vol. 1, pp. 1-5. IEEE, 2019.

[5] Sruthi, C. J., and A. Lijiya. "Signet: A deep learning based indian sign language recognition system." In *2019 International Conference on Communication and Signal Processing (ICCSP)*, pp. 0596-0600. IEEE, 2019.

[6] Rathi, Dhruv. "Optimization of Transfer Learning for Sign Language Recognition Targeting Mobile Platform." arXiv preprint arXiv:1805.06618 (2018).

[7] Zhu, Ling, Zhenbo Li, Chen Li, Jing Wu, and Jun Yue. "High performance vegetable classification from images based on alexnet deep

learning model." International Journal of Agricultural and Biological Engineering 11, no. 4 (2018): 217-223.

[8] Zhu, Runjie, XinhuiTu, and Jimmy Xiangji Huang. "Deep learning on information retrieval and its applications." In *Deep Learning for Data Analytics*, pp. 125-153. Academic Press, 2020.

[9] Albawi, Saad, Tareq Abed Mohammed, and Saad Al-Zawi. "Understanding of a convolutional neural network." In *2017 International Conference on Engineering and Technology (ICET)*, pp. 1-6. Ieee, 2017.

[10] O'Shea, Keiron, and Ryan Nash. "An introduction to convolutional neural networks." arXiv preprint arXiv:1511.08458 (2015).

[11] Demir, Ahmet, Feyza Yilmaz, and OnurKose. "Early detection of skin cancer using deep learning architectures: Resnet-101 and inception-v3." In *2019 Medical Technologies Congress (TIPTEKNO)*, pp. 1-4. IEEE, 2019.

[12] Thangaraj, Rajasekaran, D. Dinesh, S. Hariharan, SivaramakrishnanRajendar, D. Gokul, and T. R. Hariskarthi. "Automatic Recognition of Avocado Fruit Diseases using Modified Deep Convolutional Neural Network." *International Journal of Grid and Distributed Computing* 13, no. 1 (2020): 1550-1559.

[13] Eisenman, Bonnie. *Learning react native: Building native mobile apps with JavaScript*. " O'Reilly Media, Inc.", 2015.

[14] Brownlee, Jason. *Deep learning with Python: develop deep learning models on Theano and TensorFlow using Keras*. Machine Learning Mastery, 2016.

[15] Chollet, Francois. "Building powerful image classification models using very little data." Keras Blog (2016).

[16] Liu, Xuefeng, ZhenqingJia, XiaokeHou, Min Fu, Li Ma, and Qiaoqiao Sun. "Real-time marine animal images classification by embedded system based on mobilenet and transfer learning." In OCEANS 2019-Marseille, pp. 1-5. IEEE, 2019.

[17] Hsiao, Ting-Yun, Yung-Chang Chang, Hsin-Hung Chou, and Ching-Te Chiu. "Filter-based deep-compression with global average pooling for convolutional networks." Journal of Systems Architecture 95 (2019): 9-18.

[18] Sivasamy, Jeevitha, and T. Subashini. "Classification and predictions of lung diseases from chest x-rays using MobileNet." International Journal of Analytical and Experimental Modal Analysis 12, no. 3 (2020): 665-672.

[19] Jais, Imran Khan Mohd, Amelia Ritahani Ismail, and Syed QamrunNisa. "Adam optimization algorithm for wide and deep neural network." Knowl. Eng. Data Sci 2, no. 1 (2019): 41-46.

[20] Ozarkar, Saket, Raj Chetwani, Sugam Devare, Sumeet Haryani, and Nupur Giri. "AI for Accessibility: Virtual Assistant for Hearing Impaired." In 2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT), pp. 1-7. IEEE, 2020.