2022 IEEE World Conference on Applied Intelligence and Computing (AIC) | 978-1-6654-7988-2/22/$31.00 ©2022 IEEE | DOI: 10.1109/AIC55036.2022.9848930

Classification Of Indian Sign Language Characters Utilizing Convolutional Neural Networks And Transfer Learning Models With Different Image Processing Techniques

Atharva Dumbre[1], Shrenik Jangada[2], Shreyas Gosavi[3] and Jaya Gupta[4]
Department of Computer Engineering
A.P.Shah Institute of Technology
Mumbai University
Thane-400615, Maharashtra, India
[1]atharva.dumbre1@gmail.com, [2]jangadashrenik@gmail.com, [3]shreyasgosavi2016@gmail.com, [4]jdgupta@apsit.edu.in

***Abstract*— In terms of visual-spatial modality, the sign language is considered to be a natural as well as a full-fledged language. It has all of the linguistic characteristics of spoken language (from phonology to syntax). Sign language is a form of communication in which the hands are used instead of words. It uses a variety of signs to convey thoughts and concepts. For ISL static character recognition, we propose a Convolutional Neural Network (CNN) architecture in this paper. Comparison of different feature extraction techniques tested on CNN architecture is done in this particular paper. We hand-crafted the dataset used to train the CNN model in order to come as near to the real-life scenario in which the model's viability would be assessed as possible. The proposed method was successfully implemented with a 99.90 percent accuracy, which is better than the majority of currently available methods.**

Keywords*— *Indian Sign Language, Convolutional Neural Networks, Transfer learning, K-Means clustering.

I. INTRODUCTION

Sign language is a type of language that uses hand movements, facial expressions and body language to communicate [1]. The deaf community has used it as their primary form of communication for quite some time now. Moreover, it has been developed into a fully developed language with its own grammar, syntax, and vocabulary. There are many challenges that come with being deaf, including limited access to education. We are here to make learning sign language easy and accessible to everyone, everywhere. Sign Language was created as a language to help the Deaf communicate with one another. Hearing impairment (HI) affects around 6.3 percent of Indians, according to WHO (2018) data (63 million people with significant auditory loss). Adult-onset deafness is predicted to affect 7.6 percent of Indians, whereas childhood-onset deafness affects 2% [2].

As previously said, the number of persons afflicted by hearing impairment and adult-onset deafness rises every year, making Indian Sign Language a crucial part of our community. Our study on this issue will assist this demographic in communicating with the broader public and not falling behind in our societal growth. The primary goal of this research is to raise awareness of sign language, since the results will be used in real-world applications such as Indian sign language to English language translation systems and sign language education websites.

Subject, Object, Verb (SOV) is the sentence structure of Sign Language, which is vastly different from the sentence structure of English language, which is – Subject, Verb, and Object (SVO) [3]. For example, if we say "This food is tasty" in everyday speech, we would say "Food tasty" in sign language. Sign language is not the same all across the world; like conventional speech, it has various signs in different places. It features both single-handed and double-handed signals, which adds to the difficulty of creating classifiers for sign language. [4].

In this research, we built a Convolutional neural network from the ground up and compared its performance to that of various transfer learning architectures in order to get the best classifier performance.

II. LITERATURE SURVEY

Till this date a couple of publications with various methodologies of classifying sign language, ranging from PCA analysis to MobileNet designs, have been used in numerous research papers. Some of the papers that we referenced are mentioned below:

In a study on Indian Sign Language converter using Convolutional Neural Networks, the authors Nishi Intwala, Arkav Banerjee, Meenakshi, and Nikhil Gala employed the MobileNet architecture for classification. Grab Cut method and gaussian mixture model was used for feature extraction from images. On real-time photos, they were able to attain an accuracy of 87.69 percent [5].

Mariappan and Gomathi's study on real-time identification of Indian sign language includes hand movements and facial expressions utilized to identify sign language. The region of

978-1-6654-7988-2/22/$31.00 ©2022 IEEE
DOI: 10.1109/aic.2022.70

interest is detected using an OpenCV feature called skin segmentation. The fuzzy c-mean clustering approach is used for sign language training and prediction. Gesture recognition has various uses, including automated homes, gaming control, and sign language interpretation. The technology described in this study is based on real-time sign identification and is beneficial for those with hearing and speech impairments. [6].

The research paper by Kartik Shenoy, Devendra Vyavaharkar, Varun Rao and Tejas Dastane, on recognition of Indian Sign Language (ISL) in real-time demonstrates a system which can recognize Indian-Sign Language with the help of hand signals. The symbols of Sign-language as specified on the Talking Hands website are used in this study article. They use an Android smartphone camera to capture the motions, and the video/frames are subsequently sent to the server for processing. To remove unnecessary sections of the image, a pre-processing technique is used, followed by the extraction of the image of the hands. Object stabilization and Skin Color Segmentation are utilized to detect hands. Grid-based feature extraction is employed for features, while classification is done using K-nearest Neighbours. Using the above technique, the accuracy of 99.7% was achieved for static hand poses and 97.23 for hand gestures [7].

In the research paper on Static Hand Gesture Recognition for American Sign Language using Deep Convolutional Neural Network, American Sign Language is detected using the CNN model. It provides an accuracy of about 94.34%. In the preprocessing step, the image size is reduced to 200 X 200 and further layers are added with varying dimensions for features. They have referred to the AlexNet model by G. E. Hinton as "Imagenet classification with deep convolutional to compare the results with their model. The CNN model has six convolutional layers and uses the SGD algorithm. The validation accuracy was found to be 78.46% and the maximum recognition rate was found to be 94.34% [8].

III. Methodology

In this research, we designed a Convolution Neural Network for a total of 36 classes which include the numbers 0-9 as well as the 26 alphabets present in the Indian Sign language. The dataset was custom created using the signs demonstrated on Indian Sign Language Research and Training Centre (ISLRTC) website which we took as reference and created the dataset with 1000 images per character for Indian sign language. The signs used in this research are demonstrated in Figure 1.

For this research, we will be comparing the performances of the designed CNN architecture using four different types of image pre-processing steps which include grayscale, Gaussian Blur and adaptive threshold, Contour detection and K-means with 7 neighbors as shown in Figure1 along with, comparing the best performing model with two transfer learning models of MobileNetV2 and InceptionV3.

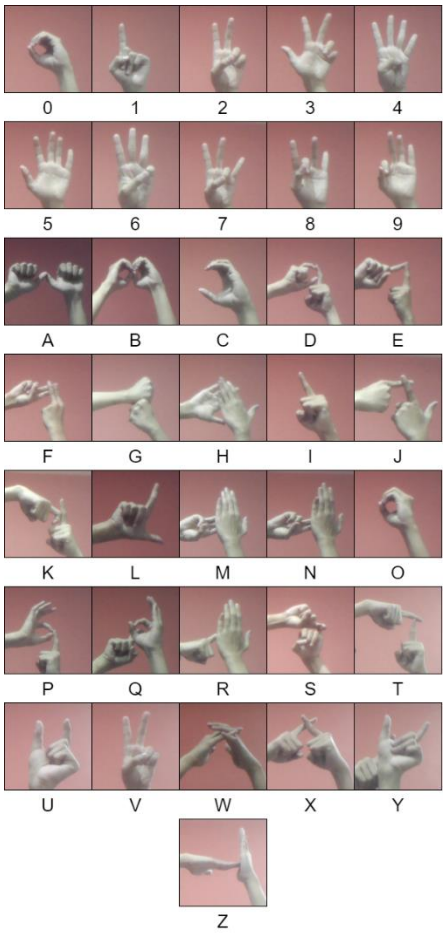

Figure 1. Indian Sign Language for English alphabets & numbers

IV. Data Preparation

For the creation of the dataset used in this research, a laptop camera with a resolution of 720p was used to capture 1000 images of 36 Indian sign language characters namely, alphabets from A-Z and numbers from 0-9. Python's OpenCV library was used to capture the images by cropping the frame into a 250 * 250 frame containing only the sign performed by the subject. The implementation for the same is demonstrated in Figure 2 and Figure 3.

Figure 2 shows the images capturing window with a countdown to provide time to perform the sign and capture images after the countdown. The green box in the top left is our region of interest. Figure 2 shows the images being captured and the number of saved images is also reported in the window. The dataset has been uploaded to Kaggle and can be found on: https://www.kaggle.com/datasets /atharvadumbre/indian-sign-language-islrtc-referred

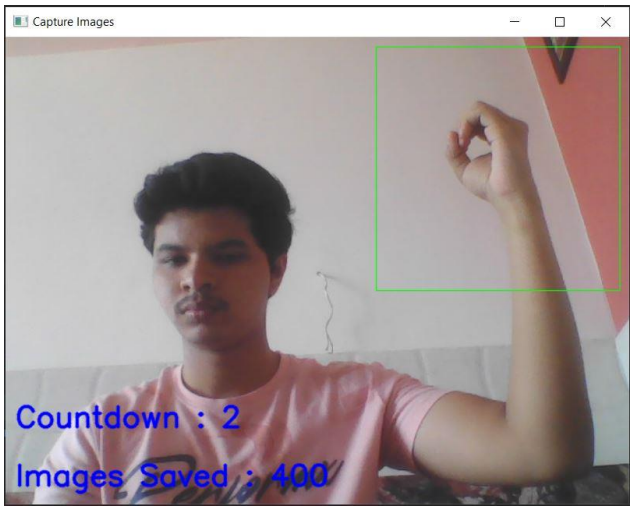

Figure 2. Image capturing with countdown

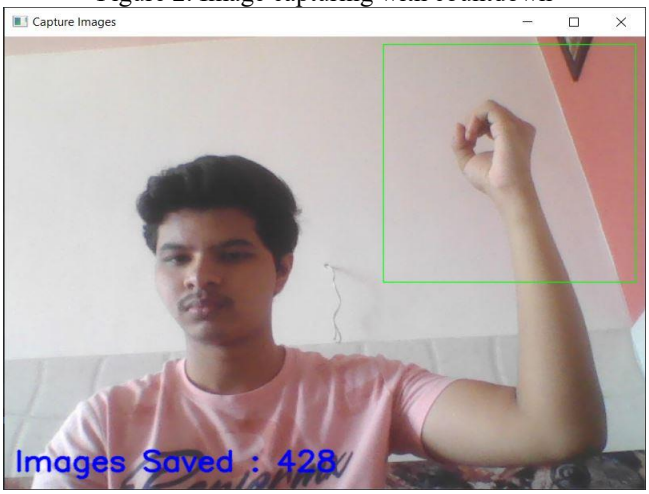

Figure 3. Image capturing in progress

V. PRE-PROCESSING

The dataset includes 1000 images per class which makes the dataset of 36000 images and the following image pre-processing techniques were applied on the dataset. The dimensions of the images are 250*250 with varying backgrounds to make the models more robust.

1. Grayscale:

Gray-scaling is the process of transforming a picture from another color space to shades of grey, such as RGB, CMYK, HSV, and so on. Gray-scaling is the process of reducing the size of a picture. RGB pictures, for example, contain three color channels and three dimensions, whereas grayscale images are one-dimensional. We used OpenCV library to produce the grayscale images by implementing the cv2.COLOR_BGR2GRAY function and it was fed to the CNN model shown in Figure 4 Row 2 and Figure 5.

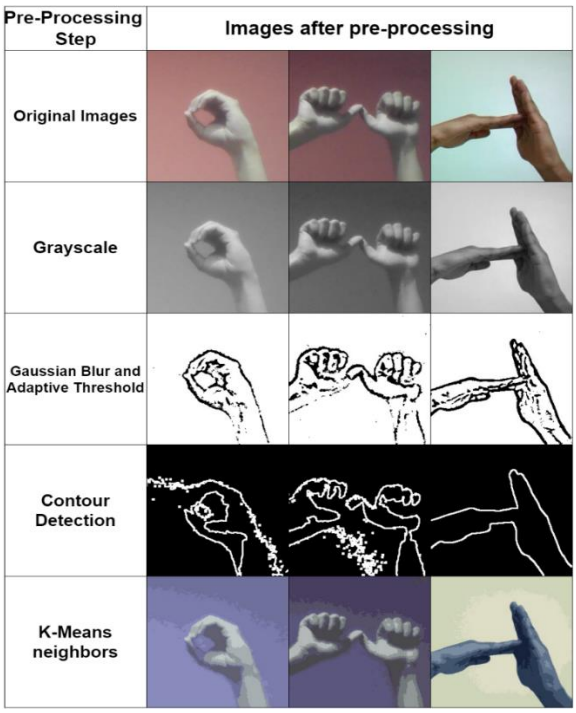

Figure 4. Implemented image after pre-processing techniques

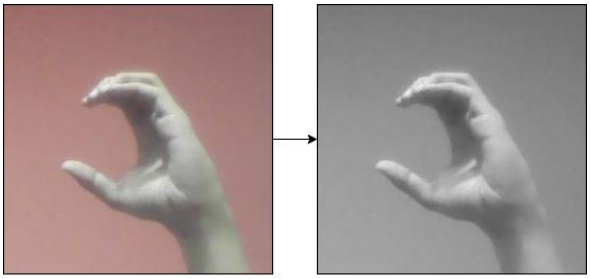
Figure 5. Grayscale Image Pre-Processing

2. Gaussian Blur & Adaptive Threshold:

The Gaussian filter is a spatial filter which operates by convolutionally converging the input image with a kernel. This procedure computes a weighted average of the current pixel's neighborhood, with distant pixels receiving less weight than those in the center. A low-pass filter of this type produces a fuzzy image with better edges than other uniform smoothing methods. A grayscale or color image is passed as input in adaptive thresholding which provides us with a binary image with segmentation. For each pixel in the picture, a threshold must be computed. The foreground value is utilized if the pixel value is less than the threshold; otherwise, the background value is used.

The pre-requisite for this step is to convert the images into grayscale after which we apply Gaussian Blur (cv2.GaussianBlur) as the first step followed by adaptive threshold (cv2.adaptiveThreshold), with cv2.threshold being the final step to complete this pre-processing shown in Figure 4 Row 3 and Figure 6. [9,10]

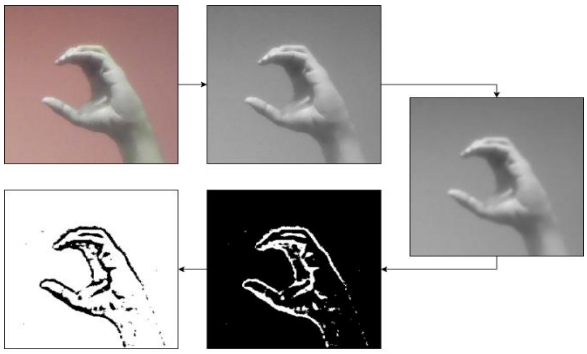

Figure 6. Gaussian Blur & Adaptive Threshold Image Pre-Processing

3. Contour Detection:

A contour is formed by connecting all of the points on an object's boundaries. A particular contour often refers to border pixels that have the same hue and intensity. Using contour detection, we can quickly recognize and pinpoint the edges of objects in an image. Canny edge detection is a method for image processing that detects edges in an image while reducing noise. Contour detection also utilizes grayscale images which are processed by applying threshold (cv2.threshold) and canny edge detection (cv2.Canny) functions Figure 4 Row 4 and Figure 7. [11]

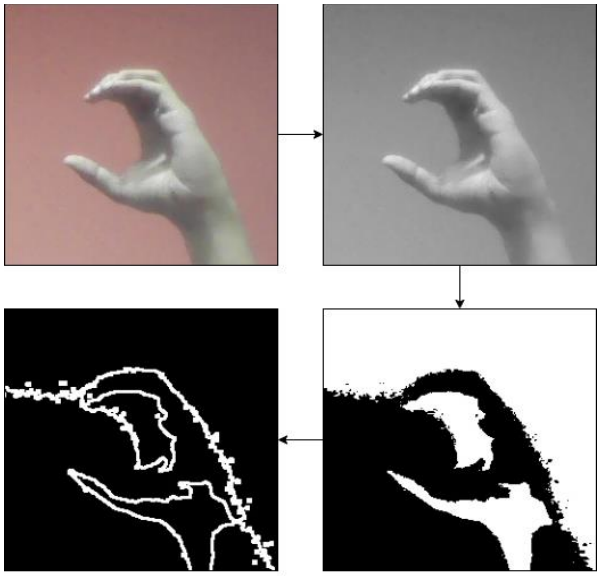

Figure 7. Contour Detection Image Pre-Processing

4. K-Means with 7 neighbors:

K-Means clustering is an unsupervised approach that is used to segregate the region of interest from the rest of the image. It clusters the given data into K-clusters or sections based on the concept of K-centroids. For pre-processing using K-Means algorithm we first converted our images to RGB channel as OpenCV reads images in BGR channel format. We then use cv2.kmeans functions where we provide our value of 'k' as 7 to get the final result Figure 4 Row 5 and Figure 8. [12]

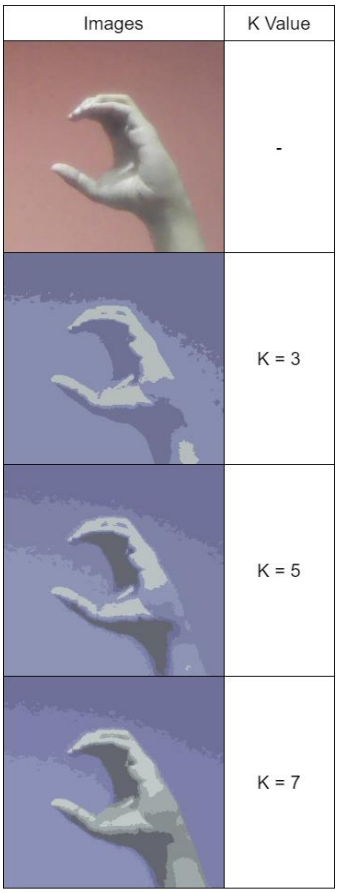

Figure 8. K-Means Image Processing

VI. CNN Architecture

CNNs, or deep learning neural networks, are deep learning neural networks that are meant to analyze ordered arrays of input, such as images. CNN is excellent at detecting design aspects in input images like lines, gradients, circles, and even eyes and faces. This feature is what makes convolutional neural networks so effective for computer vision.

Conv2D layer is responsible for convolution. First, the picture is segmented into perceptron's (algorithm); local fields are formed, and perceptron's are compressed to feature maps as a m x n matrix. The pooling layer is also known as the down sampling layer since it is in charge of lowering the size of activation maps. Same sized filter and stride are applied on the input volume. Because this layer excludes less relevant data, picture recognition is performed in a reduced representation. This layer helps to minimize overfitting. We used 'relu' activation function for all the convolutional layers while the output layer utilizes 'softmax' activation function.

Stochastic Gradient Descent (SGD) is used as an optimizer for our model with learning rate set to 0.01 which

decreases to 20 percent of the previous iteration value every 5 epochs. We evaluated our model in terms of accuracy on all the 36 classes.

After the processing stage is completed, we send our image to the convolutional neural network architecture. We utilized three convolution layers in this CNN design, followed by a max pooling layer after each convolution layer, and finally a dense layer and an output layer with 'softmax' activation. A 3*3 kernel size is used in this particular convolutional neural network for the convolution layers and a 2*2 kernel for the max pooling layers. 'Relu' activation function was used for all the convolution layers. The complete architecture is shown in Figure 9.

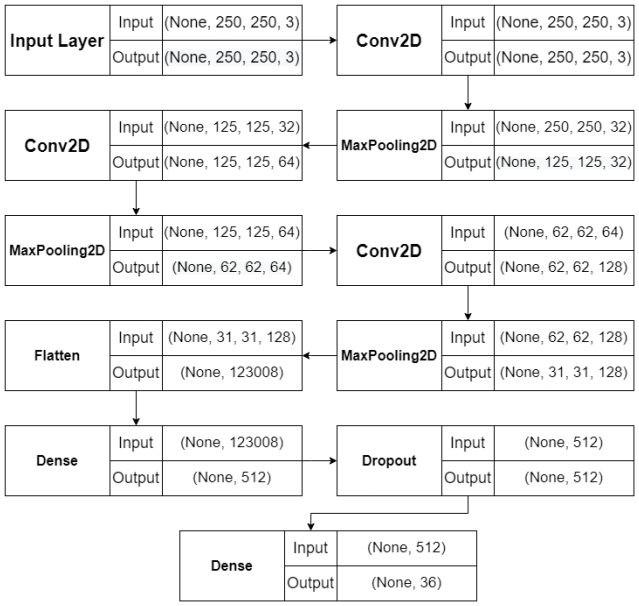

Figure 9. Convolutional Neural Network Architecture

The rectified linear activation function is a straightforward computation that returns the value given as input immediately, or 0.0 if the input is 0.0 or less.

$$RELU(x) = \begin{cases} 0 \; if \; x < 0 \\ x \; if \; x \geq 0 \end{cases}$$

where x is the input to the relu activation function.[13]

The Softmax activation function turns an artificial neuron's input impulses into a probability distribution. Mathematically, the sigmoid function (another activation function) is generalized to achieve this function having the formula as follows –

$$f_j(z) = \frac{e^z j}{\Sigma_k \, e^z k}$$

where z is the vector that serves as the neuron's input and k is the number of classes in the multiclass classification task. [14]

Layers	Output dimensions	Kernel size / Neurons
Convolution 1	250*250*32	(3,3)
Max Pooling 1	125*125*32	(2,2)
Convolution 2	125*125*64	(3,3)
Max Pooling 2	62*62*64	(2,2)
Convolution 3	62*62*128	(3,3)
Max Pooling 3	31*31*128	(2,2)
Flatten	-	-
Dense	-	512
Dropout	-	0.5
Dense	-	36

Table 1. CNN architecture with dimensions & kernel sizes

The kernel size that was used for each layer as well as the output dimensions of every layer is mentioned in Table 1.

We have also used the transfer learning models of InceptionV3 by Googlenet and MobileNetV2 by Google. The Inception V3 is a deep learning model for image categorization that is based on Convolutional Neural Networks. Inception V3 is identical to and includes all of the features of Inception V2 with a few modifications/additions which are:

- RMSprop optimizer is used.
- Batch Normalization in the Auxiliary classifier's fully linked layer.
- 7*7 Convolutions.
- Label Smoothing Regularization is a technique for regularizing the classifier by assessing the influence of label-dropout during training. It stops the classifier from making too confident predictions about a class. The addition of label smoothing improves the error rate by 0.2 percent.

MobileNetV2 is a convolutional neural network model designed for mobile devices. It is built on an inverted residual structure with residual connections between bottleneck levels. As a source of non-linearity, the intermediate expansion layer filters features using lightweight depth wise convolutions. The MobileNetV2 design has an initial fully connected convolution layer with 32 filters which are followed by 19 residual bottleneck layers.[15]

VII. Experimental Results

The results discussed in this section were obtained from a personal computer with 16 GB of RAM, a Ryzen 7 processor with a configuration of 8 cores/16 threads and a 6 GB NVIDIA GPU. For the evaluation of our CNN models, 5-fold accuracy was taken into consideration. For each fold out of the total 36000 images, we have used 28800 images as our training set and the remaining 7200 were used as testing set. We trained different Convolutional neural networks for each variant of image processing as well as two transfer learning models to compare the acquired overall accuracy along with the accuracy for the best fold for each model. Table 2 shows the accuracy that we were able to achieve on our dataset:

Model	Image pre-processing	5-Fold	Best Fold
CNN	Grayscale	99.69%	99.75%
CNN	Gaussian Blur & Adaptive Threshold	99.76%	99.80%
CNN	Contour Detection	99.85%	99.88%
CNN	K-Means (k=7)	99.82%	99.90%
InceptionV3	-	99.27%	99.38%
MobileNetV2	-	99.56%	99.68%

Table 2. Fivefold Accuracy and best fold accuracy

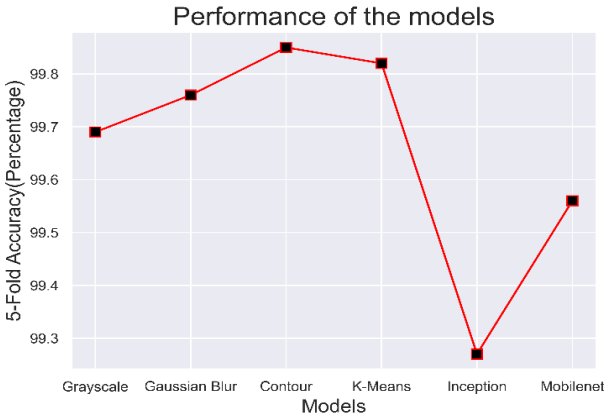

Figure 10. 5-Fold overall accuracy plot for each model

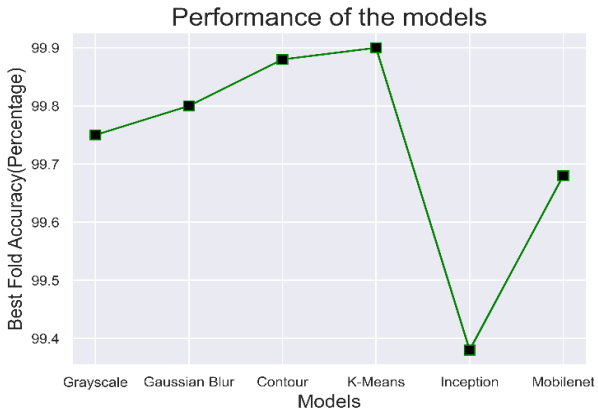

Figure 11. Accuracy plot of the best fold for each model

Table 2 mentions the accuracy achieved using the 5-fold cross validation method and the best performing fold among the 5 folds. We used the proposed CNN model using the image processing techniques of grayscale, gaussian blur & adaptive threshold, contour detection and k-means clustering along with the transfer learning models of InceptionV3 and MobileNetV2 with raw images. As we can see in the Table 2, the best performing fold was of CNN model with K-Means image processing with an accuracy of 99.90% which is closely followed by the best fold of contour detection with 99.88% accuracy. These two were the best performers on our dataset with almost comparable performance. InceptionV3 model has the lowest accuracy among the models that we tried with an accuracy of 99.38%. CNN models with gaussian blur & adaptive threshold and grayscale took the 3rd and 4th place respectively. The MobileNet model was the second worst performing model on our dataset with an accuracy score of 99.68%. Figure 10 and Figure 11 shows the accuracy in the form of line graphs to easily visualize our results.

VIII. Real-time application

The best performing model is used to get predictions in real time application. Through experiments we concluded that a solid background gives the best results. Any interference in the form of lighting or noise in the background starts to impact the performance of the model.

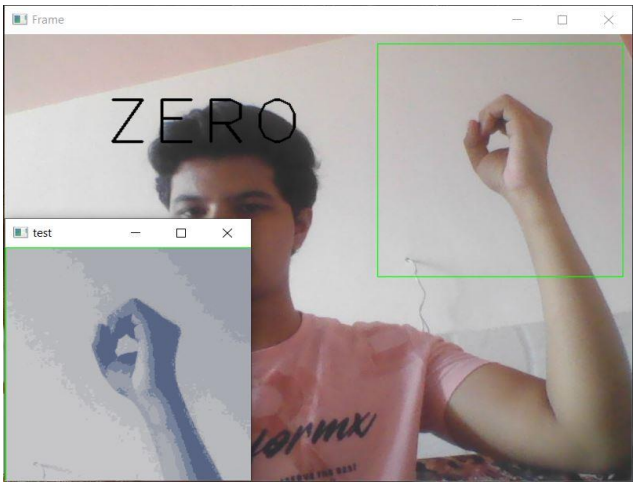

Figure 12. Prediction of '0' from a real-time frame

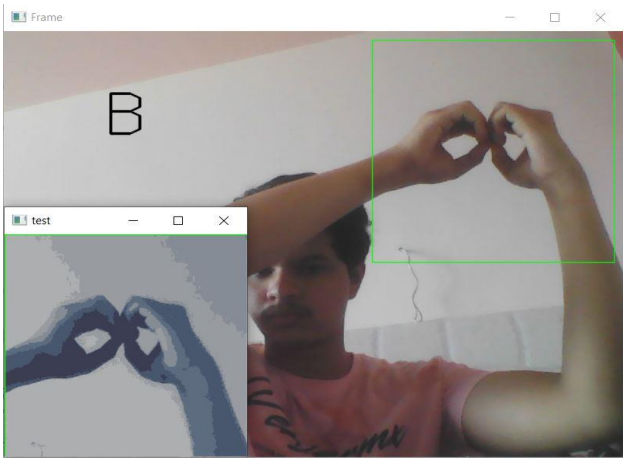

Figure 13. Prediction of 'B' from a real-time frame

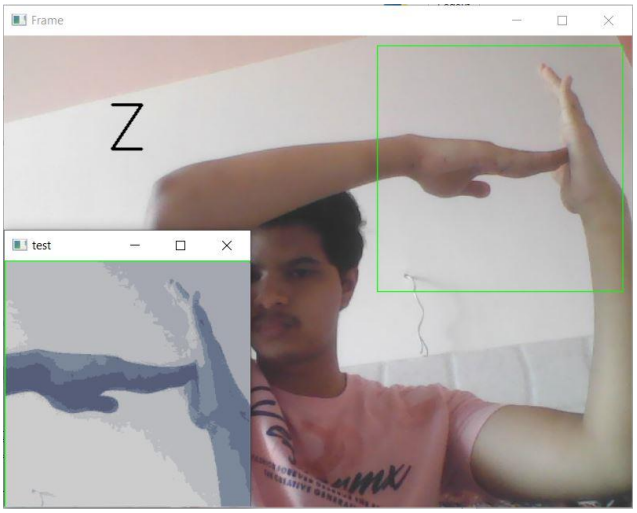

Figure 14. Prediction of 'Z' from a real-time frame

In Figure 12, we passed the sign of number '0' and got the correct prediction from the model which was shown on the screen. Figure 13 and Figure 14 shows the predictions for alphabets of 'B' and 'Z' which were also correct.

IX. Future Scope

Currently this paper aims at only the characters including A-Z alphabets and 0-9 numbers for English language in Indian sign language. This work doesn't include words or sentences although, the same can be created using single characters included in this work. Only two transfer learning models were experimented with in this work. More models can be tried to increase the accuracy even further.

This work can be used to develop educational portals with the aim to educate people in Indian sign language. These portals can deploy the model proposed in this research for testing the user's ability to perform the intended signs. This work can also be used in translation systems for sign language wherein, sign language users can be able to communicate with the people who aren't aware about the sign language.

The scope of this research can be further extended by including the dynamic signs of "H, I and J" wherein we have considered only the final positions for these signs. Moreover, we can include some basic words such as greetings or sentences which may include introductions, etc.

References

[1] Sign Community, Introduction to Sign Language, https://www.signcommunity.org.uk/

[2] Davey S, Maheshwari C, Kumar Raghav S, Singh N, Muzammil K, Pandey P, Impact of Indian Public Health Standards for Rural Health care facilities on National Programme for control of Deafness in India : The result of a Cohort study, 2018,7, 780-786 (Cited-September 2018, Volume-4) http:- www.jfmpc.com/article .asp?issn=2249-4863,year=2018, Volume=7, Issue=4

[3] Divya S, Kiruthika S, Nivin Anton A L, Padmavathi S, "Segmentation, Tracking and Feature Extraction for Indian Sign Language Recognition," International Journal on Computational sciences & Applications, vol.4, no.2, pp.57-72, April 2014.

[4] Kusumika Krori Dutta, Sunny Arokia Swamy Bellary, "Machine Learning Techniques for Indian Sign Language Recognition," International Conference on Current Trends in Computer, Electrical, Electronics and Communication (ICCTCEEC-2017)

[5] Nishi Intwala, Arkav Banerjee, Meenakshi and Nikhil Gala, "Indian Sign Language converter using Convolutional Neural Networks," 2019 5th International Conference for Convergence in Technology (I2CT) Pune, India. Mar 29-31, 2019

[6] H. Muthu Mariappan and V. Gomathi, "Real-Time Recognition of Indian Sign Language," 2019 International Conference on Computational Intelligence in Data Science (ICCIDS), 2019, pp. 1-6, doi: 10.1109/ICCIDS.2019.8862125.

[7] Kartik Shenoy, Tejas Dastane, Varun Rao, Devendra Vyavaharkar - "Real-time Indian Sign Language (ISL) Recognition" – 2018

[8] Prangon Das, Tanvir Ahmed, Md. Firoj Ali - "Static Hand Gesture Recognition for American Sign Language using Deep Convolutional Neural Network" - TENSYMP – 2020

[9] K. Pengsen and Y. Zhenming, "Image blurred region detection based on RGB color space information and local standard deviation," 2017 IEEE 2nd Advanced Information Technology, Electronic and Automation Control Conference (IAEAC), 2017, pp. 2177-2181

[10] H. El Khoukhi, Y. Filali, A. Yahyaouy, M. A. Sabri and A. Aarab, "A hardware Implementation of OTSU Thresholding Method for Skin Cancer Image Segmentation," 2019 International Conference on Wireless Technologies, Embedded and Intelligent Systems (WITS), 2019, pp. 1-5

[11] P. Topno and G. Murmu, "An Improved Edge Detection Method based on Median Filter," 2019 Devices for Integrated Circuit (DevIC), 2019, pp. 378-381

[12] K. Venkatachalam, V. P. Reddy, M. Amudhan, A. Raguraman and E. Mohan, "An Implementation of K-Means Clustering for Efficient Image Segmentation," 2021 10th IEEE International Conference on Communication Systems and Network Technologies (CSNT), 2021, pp. 224-229

[13] "A Gentle Introduction to the Rectified Linear Unit (ReLU)" by Jason Brownlee, https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/

[14] "Decoding Softmax Activation Function for Neural Network with Examples in Numpy, Keras, TensorFlow and PyTorch" by MLK, https://machinelearningknowledge.ai/decoding-softmax-activation-function-for-neural-network-with-examples-in-numpy-keras-tensorflow-and-pytorch/

[15] N. A. Nainan, J. H, R. Jalan, R. S. N, K. V. C and S. P. Shankar, "Real Time Face Mask Detection Using MobileNetV2 and InceptionV3 Models," 2021 IEEE Mysore Sub Section International Conference (MysuruCon), 2021, pp. 341-345