Indian Sign Language converter using Convolutional Neural Networks

Nishi Intwala, Arkav Banerjee, Meenakshi and Nikhil Gala
Mukesh Patel School of Technology Management and Engineering (Mumbai Campus)
SVKM's NMIMS University
Mumbai- 400056, Maharashtra, India
nishiintwala1@gmail.com, arkav1897@gmail.com, minakshisingh116@gmail.com, nikhil.gala@nmims.edu

***Abstract* – People with hearing and speech impairments have to face a lot of difficulties while communicating with the general public. Being a minority, the sign language used by them is not known to a majority of people. In this paper, an Indian sign language converter was developed using a Convolutional Neural Network algorithm with the aim to classify the 26 letters of the Indian Sign Language into their equivalent alphabet letters by capturing a real time image of that sign and converting it to its text equivalent. First a database was created in various backgrounds and various image pre-processing techniques were used to make the database ready for feature extraction. After feature extraction, the images were fed into the CNN using the python software. Several real time images were tested to find the accuracy and efficiency. The results showed a 96% accuracy for the testing images and an accuracy of 87.69% for real time images.**

Index Terms* – *Indian sign language, Convolutional Neural Networks, Transfer learning, GrabCut algorithm.

I. INTRODUCTION

People with speech and hearing impairments are at a disadvantage when it comes to having conversations with people who communicate normally on a daily basis. The main method of communication for people with a speech disability is through gestures. These gestures are compiled together to create a sign language which differs from region to region. The population of India is 132.42 crores out of which 19 lakh people have a speech disability. They communicate via the Indian sign language which was developed in 2001 with 1600 words. It is essential that we develop ways to make communication easy for them and convert their gesture based language to our oral language used on a day to day basis. In this paper, we are presenting an Indian Sign Language alphabet converter which converts the gesture into its equivalent alphabet in the English language. For this project, we have used machine learning which includes an image classifier called MobileNet which uses a convolutional neural network. We have trained this classifier to give us the optimum accuracy.

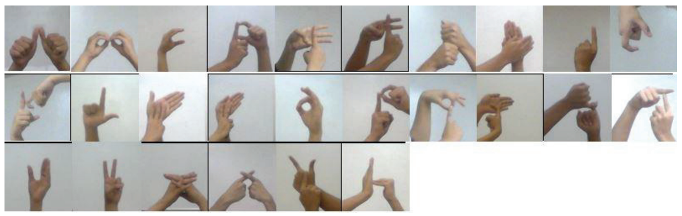


Fig. 1: The Indian sign language letters

II. RELATED WORKS

We have referred to papers which have converted sign languages using different algorithms. One of these papers is by Muttaki Hasan, Tanvir Hossain Sajib, Mrinmoy Dey who have used SVM as their classifier and have used HOG (Histogram of Oriented Gradients) for feature extraction of certain words in the Bengali sign language. In the paper they have converted the expressions into their respective audio outputs [5]. They have first preprocessed the images using binary thresholding, and then segmented the images by cropping the hand gesture with the help of the OpenCV library. The next step taken by them was to train and test the classifier with the testing set containing 64 of the 320 images. Finally, the output from the classifier is converted into audio with the help of the TTS engine in the python library. They achieved an accuracy of 86.53%.

In a paper by Pranali Loke, Juilee Paranjpe, Sayali Bhabal and Ketan Kanere, they have proposed a sign language converter using an artificial neural network on MATLAB. In this paper they haven't implemented but only proposed a method to create an android application for sign language conversion [4]. They have chosen the hand gesture recognition feature to recognize the gestures and have converted these to natural language. They have proposed doing so by utilizing the HSV (hue, saturation and intensity) model for hand tracking and segmentation. For classification they have made use of a neural network. Using an android application, they have captured the images and used them as an input in their neural network. The hand gesture is matched with its respective hand gesture on MATLAB and the resultant converted text is sent back to the user's device, making it a system which will aid the people unknown to sign language.

Another paper that we have referred to is written by Malladi Sai Phani Kumar, Veerapalli Lathasree and S.N. Karishma. In this paper they have used images as their input and have segmented the hand gestures from the background with the help of the GrabCut algorithm, however they have proposed using novel contour as a segmentation process instead of the GrabCut algorithm [9]. They concluded that the novel contour method gave better results in terms of segmenting the image compared to the GrabCut algorithm, which required multiple iterations. They discovered that the average amount of iterations needed to completely segment the image were three. However, the novel contour method did not work well in abstract backgrounds. This paper was mainly focused on comparing two methods of segmentation. In our case, we have used only one iteration for the GrabCut algorithm as we want the process to be automated and also, a single iteration is enough to provide us with good results. Since our backgrounds are abstract, GrabCut algorithm provides us with better segmentation.

III. NEURAL NETWORKS

Convolutional Neural Network is one of the algorithms used in machine learning. They are similar to artificial neural networks, meaning they have nodes or neurons which are connected via weighted links which produce an output in accordance with the given input. The main difference is that convolutional networks are better suited to visual classification such as images. Regular neural networks consist of a hidden layer which is connected to the previous input layer and an output layer where the classification output is given. However regular neural networks cannot handle huge amounts of data. Hence for a large number of images, convolutional neural networks are more efficient [10].

Convolutional neural networks have a 3D shape of neurons meaning that they have height, width and depth. Here, all the neurons of one layer are not only connected to all the other neurons of the adjacent layer, but a small region of neurons are connected to each other. The output layer of the network will convert the image into a single vector along the depth dimension.

In our project we have used transfer learning in which we use a pre-trained convolutional neural network model instead of training a convolutional neural network from scratch. Here the pre-trained model is selected such that its problem statement is similar to the problem statement of the user. The size and similarities between the datasets play an important role which means that in order to classify images, the pre-trained neural network model must be trained on images and not on any other data set.

The pre-trained model called MobileNet is a convolutional neural network. Here we are only retraining the final layer of the model. The previous layer before the final layer is called a bottleneck. First all the bottlenecks for each image are calculated. The bottleneck does the actual classification. Every image is utilized numerous times during the training step and the calculations for each image from the previous layers are cached and stored in the bottlenecks and can be reused. Finally, during the training step, the bottlenecks of the images are found and fed to the final layer. The final layer makes its predictions which are then compared with the actual label of the image and the weights of the final layer are optimized using the backpropagation process.

IV. FRAMEWORK

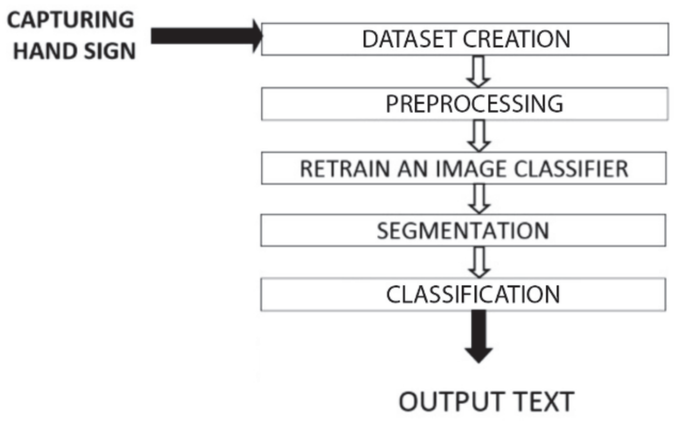


Fig. 2: Implementation Process

A. *Dataset creation*

Data collection is the main aspect of machine learning that makes training of the algorithm possible. Usually, data gathering and processing consumes most of the time involved in the whole machine learning process. The accuracy improves with more the data we gather. For our project we have used a 720p HD webcam for taking pictures of our dataset. Our dataset consists of 26 classes, each representing an English alphabet. We have an Indian sign equivalent to every English alphabet. We have clicked pictures of 2000 such signs for each alphabet. Therefore, we have a total of 52,000 images since there are 26 letters. The images have been taken in different backgrounds. However, since we accumulated abundant data, the computational time also increased. The dataset has been created with the help of MATLAB 2017b software.

B. *Image Cropping*

After the dataset has been created, we have used image cropping. Image cropping is a process where an image is cropped according to the specific area of interest. In our case, the hand gestures were the crux of the image and hence we have used image cropping to pinpoint only those portions of the image. This is necessary to eliminate any effect of the background on the hand gestures while training these images for classification purposes. Cropping an image also reduces the computational time while training the classifier. We have applied image cropping with the help of MATLAB 2017b software.

C. *Image Resizing*

In digital imaging and graphics, image resizing is done to resize a digital image. We can change the total number of pixels using image resizing. We have resized each image to a resolution of 224x224px. We have used python software in order to achieve this. This has been done in order to reduce the computational time as well as to have a uniform dataset to be used for training.

D. *Image Flipping*

A flipped image or reversed image is basically the image that is produced due to the mirror-reversal of the original image across the vertical or horizontal axis. Image flipping is done because the webcam automatically flips the images while capturing it. Hence, flipping across the vertical axis is required once again to regain the original photo.

E. *Training the classifier*

Once the dataset has been pre-processed, it is ready to be put into an image classifier. We have used MobileNet as our classifier which adopts a convolutional neural network in order to classify the images.

F. *Segmentation*

Selecting good features is critical in any object recognition system. Segmentation is one of the feature extraction techniques that is used for images [2]. We have made use of the GrabCut algorithm for segmentation of our real-time images. The working of the algorithm is as follows: -

Initially, a rectangle has to be specified marking the region of interest in the image making the region outside this rectangle the background. Once a rectangle has been specified, the algorithm will try to label the foreground and the background automatically in terms of colour statistics by iterating the image multiple times. The computer labels the foreground and background pixels. We apply a Gaussian Mixture Model(GMM) which tries to cluster unknown pixels in terms of colour statistics which then, either become a part of the foreground or the background. A graph is generated using the pixel distribution with the pixels as nodes and two additional nodes, the Source node and the Sink node respectively are added to this graph [9]. The source node is connected to the foreground pixels while the sink node to the background pixels. Each pixel is connected to the two nodes with the help of edges which have a certain weight associated to them. The weight among two pixels is determined by the edge information or pixel similarity. A large difference in pixel colour will result in a low weight in the edge between two pixels. The graph is cut into two parts separating the source and the sink node with minimum cost function [11]. All the pixels connected to Source node and Sink node become foreground and background after the cut is made. However, there may be some areas which have been classified wrongly in which case, the user has to once again hard label these regions again. Although the segmentation accuracy is not 100% in our case, and some regions are marked wrongly, the result provided by the algorithm is sufficient enough to give us a good result while classifying it. Hard labelling has been avoided in order to give us a quicker result and for the whole process to be conducted automatically as well.

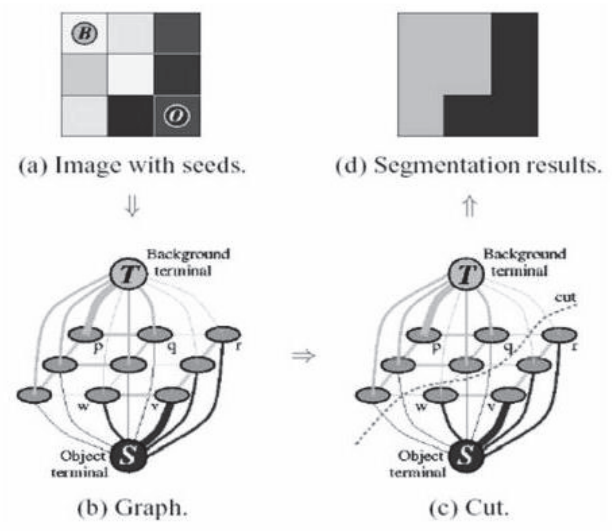


Fig. 3: GrabCut Algorithm

V. RESULTS

Figure 4 depicts the pre-processing applied on the dataset, namely- Resizing, cropping and flipping. As mentioned before, these pre-processing techniques have been applied to only the dataset and not the real time images.

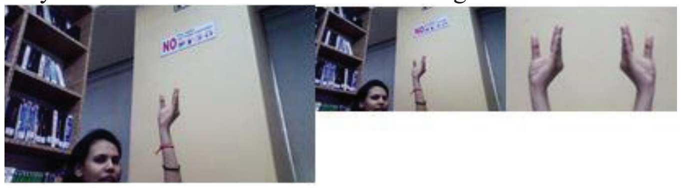


Fig. 4: Pre-processing

For the real time images, we have segmented the images with the help of GrabCut algorithm as shown below. As we can see, the main area of interest, i.e. hand, has not been segmented totally by the algorithm, however it is sufficient enough to provide us with accurate results.

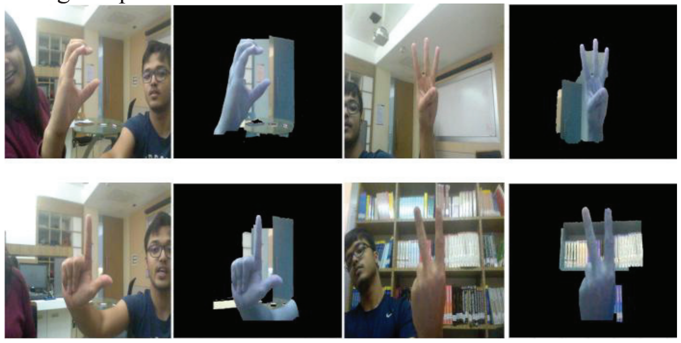
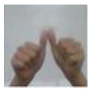
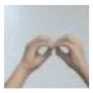
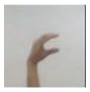
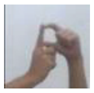
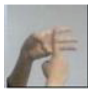
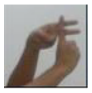
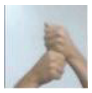
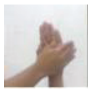
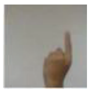
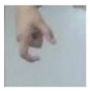
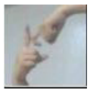
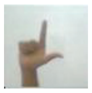
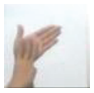
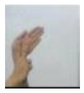
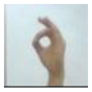
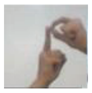
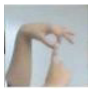
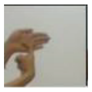
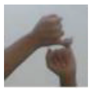
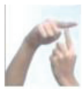
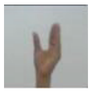
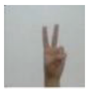
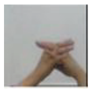
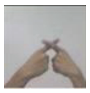
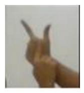
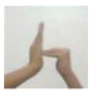


Fig. 5: Segmentation

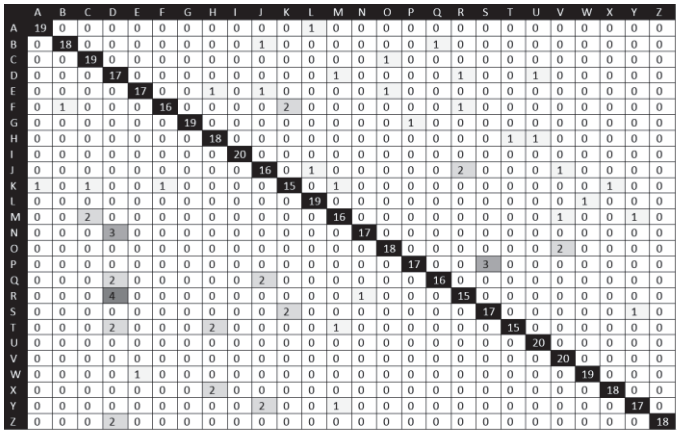
After obtaining a good accuracy of 96% in the testing phase, we further applied the algorithm on 20 real time images for each sign. Since there were a total of 26 signs, the total number of real time images taken into consideration were 20×26=520.

TABLE 1.

Sign	Character	Accuracy	Result
	A	95%	Good
	B	90%	Good
	C	95%	Good
	D	85%	Moderate
	E	85%	Moderate
	F	80%	Moderate
	G	95%	Good
	H	90%	Good
	I	100%	Excellent
	J	80%	Moderate
	K	75%	Poor
	L	95%	Good
	M	80%	Moderate
	N	85%	Moderate
	O	90%	Good
	P	85%	Moderate
	Q	80%	Moderate
	R	75%	Poor
	S	85%	Moderate
	T	75%	Poor
	U	100%	Excellent
	V	100%	Excellent
	W	95%	Good
	X	90%	Good
	Y	85%	Moderate
	Z	90%	Good

In table 1, we considered an accuracy of 100% to be excellent, an accuracy from 90% to 99% to be good, an accuracy from 80% to 89% to be moderate while an accuracy below 80% to be poor. We observed that signs I, U and V had been classified perfectly while signs K and R had been classified poorly. We also observed that out of these 520 images, 456 of them had been classified correctly. Hence we obtained an accuracy of 87.69% for the real time images.

TABLE II. CONFUSION MATRIX FOR THE 26 INDIAN SIGN LETTERS



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	19	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
B	0	18	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
C	0	0	19	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
D	0	0	0	17	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	1	0	0	0	0	0
E	0	0	0	0	17	0	0	1	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
F	0	1	0	0	0	16	0	0	0	0	2	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
G	0	0	0	0	0	0	19	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
H	0	0	0	0	0	0	0	18	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0
I	0	0	0	0	0	0	0	0	20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
J	0	0	0	0	0	0	0	0	0	16	0	1	0	0	0	0	0	2	0	0	0	1	0	0	0	0
K	1	0	1	0	0	1	0	0	0	0	15	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0
L	0	0	0	0	0	0	0	0	0	0	0	19	0	0	0	0	0	0	0	0	0	0	1	0	0	0
M	0	0	2	0	0	0	0	0	0	0	0	0	16	0	0	0	0	0	0	0	0	1	0	0	1	0
N	0	0	0	3	0	0	0	0	0	0	0	0	0	17	0	0	0	0	0	0	0	0	0	0	0	0
O	0	0	0	0	0	0	0	0	0	0	0	0	0	0	18	0	0	0	0	0	0	2	0	0	0	0
P	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	17	0	0	3	0	0	0	0	0	0	0
Q	0	0	0	2	0	0	0	0	0	2	0	0	0	0	0	0	16	0	0	0	0	0	0	0	0	0
R	0	0	0	4	0	0	0	0	0	0	0	0	0	1	0	0	0	15	0	0	0	0	0	0	0	0
S	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	17	0	0	0	0	0	1	0
T	0	0	0	2	0	0	0	2	0	0	0	0	1	0	0	0	0	0	0	15	0	0	0	0	0	0
U	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	20	0	0	0	0	0
V	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	20	0	0	0	0
W	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	19	0	0	0
X	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	18	0	0
Y	0	0	0	0	0	0	0	0	0	2	0	0	1	0	0	0	0	0	0	0	0	0	0	0	17	0
Z	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	18

Table 2 depicts the confusion matrix for the 26 Indian sign letters. The 520 real time images have been considered in the confusion matrix.

V. CONCLUSION AND FUTURE WORK

In this paper, we have proposed a method for the recognition and classification of the 26 Indian sign language alphabets by using CNNs. We observed that MobileNet is an effective approach for the classification of large amounts of data. We considered previous work for the recognition of sign language and came to a conclusion that MobileNet would be efficient to classify the hand signs with a high accuracy. However, letters H and J are dynamic gestures in the Indian Sign Language whereas our method is applicable for static gesture recognition. Hence, these letters have been denoted by a single frame. In the future, we would like to click the real time images with a cellular camera instead of the webcam and get the output on the cell phone itself, since MobileNet is especially suited for cellular operations [10]. Better segmentation approaches can also be used in order to completely segment the hand from the rest of the image.

REFERENCES

[1] R.M. Gurav, P.K. Kadbe, “Real time finger tracking and contour detection for gesture recognition using opencv,” International Conference on Industrial Instrumentation and Control 2015 (ICIC 2015), pp. 974-977, 2015.

[2] Pinaki Pratim Acharjya, Ritaban Das &amp; Dibyendu Ghoshal, “Study and Comparison of Different Edge Detectors for Image Segmentation,” 2012, Global Journal of Computer Science and Technology Graphics &amp; Vision, Volume 12, Issue 13, Version 1.0, Year 2012.

[3] Y.Ramadevi, T.Sridevi, B.Poornima, B.Kalyani, “Segmentation and object recognition using edge detection techniques,” 2010, International Journal of Computer Science &amp; Information Technology (IJCSIT), Vol 2,No 6, December 2010.

[4] Pranali Loke, Juilee Paranjpe, Sayli Bhabal, Ketan Kanere, “Indian Sign Language Converter System Using An Android App,” 2017, International Conference on Electronics, Communication and Aerospace Technology-ICECA, 978-1-5090-5686-6/17.

[5] Muttaki Hasan, Tanvir Hossain Sajib and Mrinmoy Dey, “A Machine Learning Based Approach for the Detection and Recognition of Bangla Sign Language,” IEEE - 978-1-5090-5421-3/16.

[6] Farhad Yasir, P.W.C. Prasad and Abir Alsadoon, “SIFT Based Approach on Bangla Sign Language Recognition,” IEEE 8th International Workshop on Computational Intelligence and Applications, November 6-7, 2015.

[7] S. Karishma, V. Lathasree, “Fusion of skin color detection and background subtraction for hand gesture segmentation,” International Journal of Engineering Research and Technology, vol. 3, no. 2, 2014.

[8] M. K. Ahuja and A. Singh, “Static vision-based hand gesture recognition using principal component analysis,” in IEEE 3rd International Conference on MOOCs, Innovation and Technology in Education 2015(MITE 2015), IEEE, 2015, pp. 402–406.

[9] Malladi Sai Phani Kumar, Veerapalli Lathasree and S.N. Karishma, “Novel Contour Based Detection and GrabCut Segmentation for Sign Language Recognition,” International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET)-IEEE, 978-1-5090-4442-9/17.

[10] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, Hartwig Adam, “MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications,” arXiv:1704.04861v1 [cs.CV] 17 Apr 2017.

[11] C. Rother, V. Kolmogorov, and A. Blake, “Grabcut: Interactive foreground extraction using iterated graph cuts,” in ACM Transactions on Graphics (TOG), vol. 23, ACM, 2004, pp. 309–314.