iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 10 **Issue:** VIII **Month of publication:** August 2022

DOI: https://doi.org/10.22214/ijraset.2022.46098

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 10 Issue VIII Aug 2022- Available at www.ijraset.com

Finger Motion Capture for Sign Language Interpretation

Spurthi S. Bhat[1], Rutuja M. Bhirud[2], Vaishnavi R. Bhokare[3], Ishaan S. Bhutada[4], Aditya S. Chavan[5], Prof. Sandip R. Shinde[6]

[1, 2, 3, 4, 5, 6] *Department of Computer Engineering, Vishwakarma Institute of Technology, Pune*

Abstract: ***Sign language is one of the oldest and most natural form of language and for the deaf and the mute, it is the most convenient form of communication. Though it is a convenient language, many people find it difficult to interpret. Therefore, there is a need of a solution which will make sign language interpretation easy for every common man. This paper presents a novel approach using web camera to capture the person's sign gestures. These sign gestures for American Sign Language are created and translated into corresponding text in real time. A Convolutional Neural Network model is developed for Finger Motion based American Sign Language. The proposed model has 99.3% accuracy for the 26 letters of the English alphabet. The model interprets Sign Language indications into English sentences.***

Keywords: American Sign Language, Convolutional Neural Network, Desktop App, Image Processing, Machine Learning

I. INTRODUCTION

Speech impairment is a disability which affects one's ability to convey their thoughts. Such individuals make use of sign language to communicate with other people. Sign languages are visual based languages for deaf and mute people. [1] However, there is no universal sign language for deaf and mute people. Different countries make use of their own sign language, although there some striking similarities among them. It is yet not clear how many sign languages exist in the world. Some languages have received legal recognition and some have not. India's National Association of Deaf estimates that there are 18 million people in India with hearing impairment. Since most of the common people cannot understand this sign language, sign language translation becomes a necessity to bridge the gap between the common people and deaf people. American Sign Language is a predominant sign language since the only disability deaf and mute people have, is related to communication and they cannot use spoken languages. Hence, the only way for them to strike communication is through sign language. So, the deaf and mute people make use of their hands to express different gestures to express their ideas with other people. Gestures are the non-verbally exchanged messages and these gestures are understood with vision. The language which is used for nonverbal communication of deaf and mute people is called sign language. A model has been developed which can recognize fingerspelling-based hand gestures in order to form a complete word by combining each gesture using Convolutional Neural Networks (CNN). Fig 1 shows the standard American Sign Language (ASL) alphabets for mute and deaf people taken from the paper titled American Sign Language Recognition Using Leap Motion Controller with Machine Learning Approach [2].

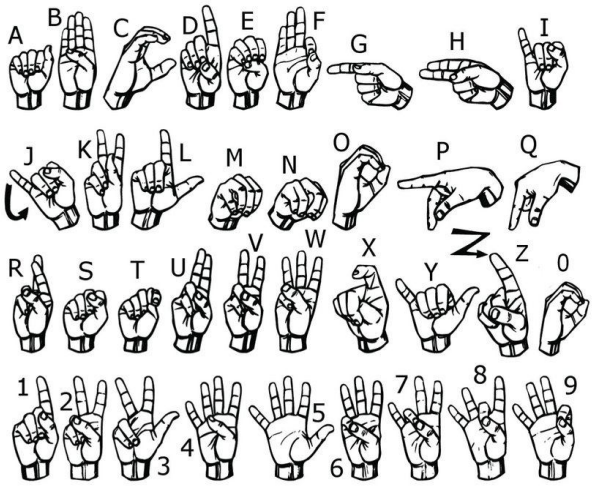


Fig. 1 American Sign Language Alphabets

II. LITERATURE REVIEW

[3] The two explainable adaptations proposed by the authors to the traditional neural SLT models using optimized tokenization-related modules. A frame stream density compression (FSDC) algorithm was introduced for detecting and reducing the redundant similar frames, which effectively shortened the long sign sentences without losing information. Then, the traditional encoder in a Neural Machine Translation (NMT) module was replaced with an improved architecture, which incorporated a temporal convolution (T-Conv) unit and a dynamic hierarchical bidirectional GRU (DH-BiGRU) unit sequentially. [4] The main objective of this paper was to recognize the gestures and display the corresponding word. The first phase involved capturing the gesture using a web camera along with pose estimation library. The web camera captured the image and the image was processed with pose estimation algorithm in tensor-flow utility. The skeleton obtained provided the values for creating the data set; the data set was a collection of the values of the coordinates of the end points of the skeleton. These values were labelled accordingly and were appended to the machine for predicting when the input is taken. [1] The authors have used histograms of oriented gradients (HOG) as a feature descriptor. This approach created a bin, and clubs the histograms of different samples based on magnitude and angle. In the proposed system, the image was segmented first using YCbCr color space and then processed the image through HOG and then provided it to the model. They trained a SVM classifier using 5000 images and developed a model. [5] The authors created an ensemble of two models to recognize gestures in sign language. They have used a custom recorded American Sign Language dataset based off an existing dataset for training the model to recognize gestures. They have proposed to use a CNN (Convolutional Neural Network) named Inception to extract spatial features from the video stream for Sign Language Recognition (SLR). Then by using a LSTM (Long Short-Term Memory), an RNN (Recurrent Neural Network) model, they could extract temporal features from the video sequences. [6] The authors have 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. They produced a robust model that consistently classified letters a-e correctly with first-time users and another that correctly classified letters a-k in a majority of cases.

III. METHODOLOGY

A. Materials/Components/Flowchart/Block Diagram/Theory

The basic technologies used are Machine Learning and Image Processing. Machine learning is used for better translation of sign language for common people to understand with help of capturing images and train convolutional neural networks based on these images. The project has been developed using Python language and using libraries such as Keras, TensorFlow, OpenCV etc.

Fig. 2 shows the flowchart of the process carried out in building the sign language interpreter desktop app.

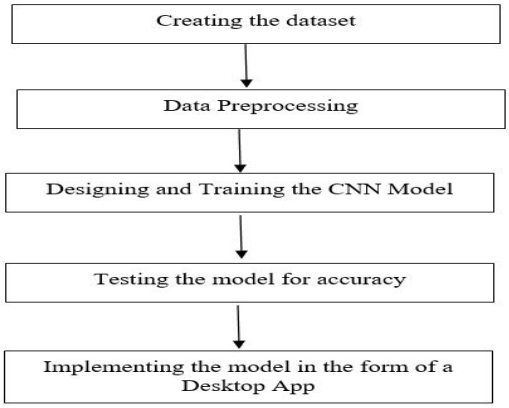


Fig. 2 Flowchart of development of Sign Language Interpreter Desktop App

Firstly, the dataset has been created which consists of images of different hand motion for sign language alphabets. Then the images have been pre-processed to make it suitable for the Convolutional Neural Networks after which the Convolutional Neural Network was designed comprising of two convolutional layers, pooling layers and densely connected layers. After raining the models for all the alphabets, the expected accuracy was not achieved due to similar hand signs for some of the alphabets. To solve this problem, three models for letters T, K, D, I, S, M, N, R and U have been trained to achieve a better accuracy. Then, the desktop application's Graphical User Interface has been designed to show the output of letters, word formation and sentence formation.

Fig. 3 depicts the block diagram of the system architecture of Sign Language Interpreter Desktop App.

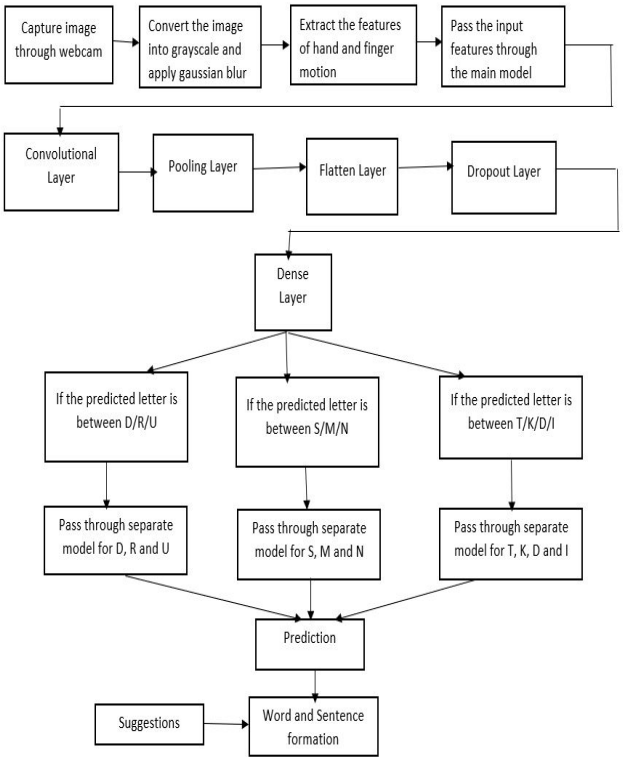


Fig. 3 Block Diagram of the Sign Language Interpreter Desktop App

The system is designed in such a way that the image captured through the web camera is converted to grayscale and then the Gaussian blur is applied on it. After that, the input features from this model are passed through the Convolutional Neural Network of the main model. According to the predictions given by the main model, if it is one of the alphabets whose sign formations are very similar to one other, such as alphabets D, R, U, S etc. then such images are passed through specially trained models and then the predictions is displayed through Graphical User Interface and the related suggestions from the Hunspell library are also displayed.

B. Synthesis/Algorithm/Design/Method

1) Data Set Generation

Data is basic pre-requisite for any Deep Learning model. The images captured in the form of RGB (Red Green Blue) images. Therefore, following steps were followed in order to create the dataset:

a) Open computer vision (OpenCV) library was used in order to produce the dataset Firstly images were captured for each frame shown by the web camera of the machine.
b) In each frame a region of interest (ROI) was defined.
c) Since the image was in Red Green Blue it was converted to grayscale
d) Finally, a Gaussian blur was applied to filter the image so that various features could be extracted from the image.
e) Respective folders were then created for the alphabet wherein the images were stored related to that particular alphabet.

2) Convolutional Neural Network Model

a) *1st Convolution Layer*: The input picture had a resolution of 128x128 pixels. It was first processed in the first convolutional layer using 32 filter weights (3x3 pixels each). This resulted in a 126X126 pixel image, one for each Filter-weights.
b) *2nd Convolution Layer:* These 63 x 63 from the output of the first pooling layer were given as an input to the second convolutional layer. It was processed in the second convolutional layer using 32 filter weights (3x3 pixels each). This resulted in a 60 x 60-pixel image.

c) *2nd Pooling Layer:* The resulting images are down sampled again using max pool of 2x2 and is reduced to 30 x 30 resolution of images.

d) *1st Densely Connected Layer*: These images are further given as an input to a fully connected layer with 128 neurons and the output from the second convolutional layer is reshaped to an array of 30x30x32 =28800 values. The input to this layer consists of an array of 28800 values. The output of this layer is given to the 2nd Densely Connected Layer. A dropout layer of value 0.4 was used to avoid overfitting.

e) *2nd Densely Connected Layer:* The output from the 1st Densely Connected Layer is passed as an input to a fully connected layer with 96 neurons.

f) *Final layer:* The output of the 2nd Densely Connected Layer served as an input for the final layer which would have the number of neurons as the number of classes for classification (alphabets + blank symbol).

The layers added to the model are shown in fig. 4.

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 126, 126, 32)	320
max_pooling2d_1 (MaxPooling2	(None, 63, 63, 32)	0
conv2d_2 (Conv2D)	(None, 61, 61, 32)	9248
max_pooling2d_2 (MaxPooling2	(None, 30, 30, 32)	0
flatten_1 (Flatten)	(None, 28800)	0
dense_1 (Dense)	(None, 128)	3686528
dropout_1 (Dropout)	(None, 128)	0
dense_2 (Dense)	(None, 96)	12384
dropout_2 (Dropout)	(None, 96)	0
dense_3 (Dense)	(None, 64)	6208
dense_4 (Dense)	(None, 27)	1755

Total params: 3,716,443
Trainable params: 3,716,443
Non-trainable params: 0

Fig. 4 Layers in Convolutional Neural Network Architecture of the main model

g) *Activation function:* ReLu (Rectified Linear Unit) was employed in each of the Layers. ReLu calculates max(x,0) for each input pixel. This attributes nonlinearity to the formula and helps to learn more complex features

h) *Pooling layer:* Max pooling was applied to the input image with a pool size of (2, 2) with ReLu activation function. This reduces the amount of parameters thus reducing the computation cost and limits overfitting.

i) *Dropout layers:* The problem of overfitting, where after training, the weights of the network are so tuned to the training examples they are given that the network does not perform well when given new examples. This layer "drops out" a random set of activations in that particular layer by assigning zero values.

j) *Optimizer:*

Adam optimizer was used for updating the model in response to the output of the loss function.

Two layers of algorithms were designed to verify and predict symbols which were more similar to each other so that it would get more accuracy in predicting them. Testing resulted in the following symbols not showing properly and were giving other symbols too:

1. D, R, U
2. S, M, N
3. T, D, K, I

Hence, three separate models were created with same layers but just by providing the corresponding folders of the alphabets as input so that it would correctly identify the alphabets having similar signs.

C. Characterization/Pseudo Code/ Testing

Therefore, in order to test the model, a Graphical User Interface which showed the predicted alphabet using the video feedback along with formation of words using the alphabets and finally displaying the sentence formed using the words. A python library, Hunspell, displays correct alternatives for each (incorrect) input word and a collection of words matching the current word in which the user can select a word to add it to the current sentence. This helps in reducing the mistakes committed in spellings and assists in predicting complex words.

Finger spelling sentence formation Implementation:

1) Whenever the count of a letter detected exceeds a specific value and no other letter is close to it by a threshold, the letter was printed and added to the current string (The value was kept as 20 and the difference threshold as 10).

2) Whenever the blank space was detected and it crossed a certain threshold (40 frames), which indicated that the word had ended, a blank space was added at the end and the word got appended to the sentence.

IV. RESULTS AND DISCUSSIONS

The training loss for the main model is up to 0.25 for the first four iterations. The validation loss is up to 0.02 in the same number of iterations as the training loss. Fig. 5 depicts the graph of training and validation loss.

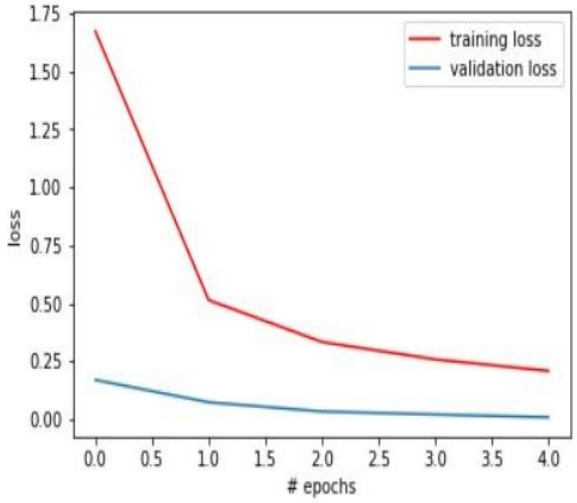


Fig. 5 Graph of training and validation loss of the main model

The training and validation accuracy of the main model is almost 92% and 99% respectively. Fig. 6 depicts the graph of training and validation accuracy.

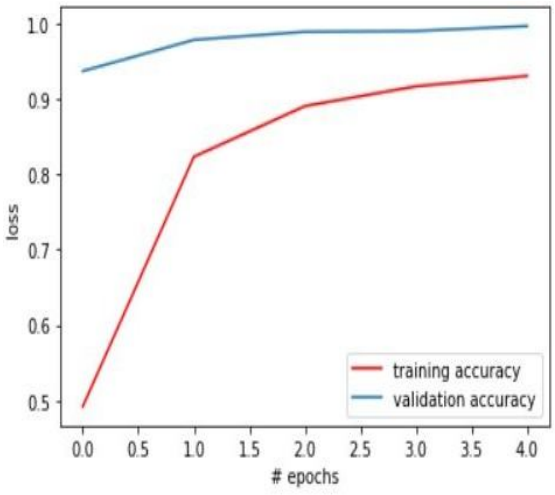


Fig. 6 Graph of Training and Validation Accuracy of the main model

The training loss, for the model trained for letters D, R and U, is up to 0.03 for the first four iterations. The validation loss is up to 0.02 in the same number of iterations as the training loss. Fig. 7 depicts the graph of training and validation loss.

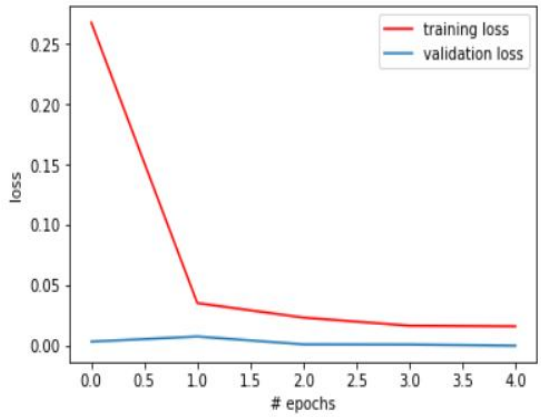


Fig. 7 Graph of training and validation loss of the model trained for alphabets D, R and U

The training and validation accuracy of the model trained for alphabets D, R and U is almost 99% and 99% respectively. This is due to smaller dataset and overfitting. Fig. 8 depicts the graph of training and validation accuracy.

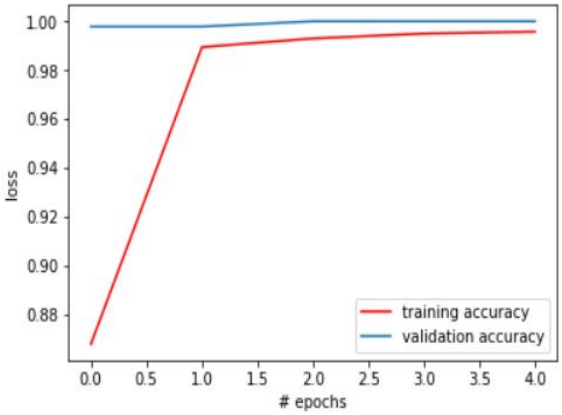


Fig. 8 Graph of Training and Validation Accuracy of the model trained for alphabets D, R and U

The training loss for the model trained for letters S, M and U is up to 0.01 for the first four iterations. The validation loss is up to 0.01 in the same number of iterations as the training loss. Fig. 9 depicts the graph of training and validation loss.

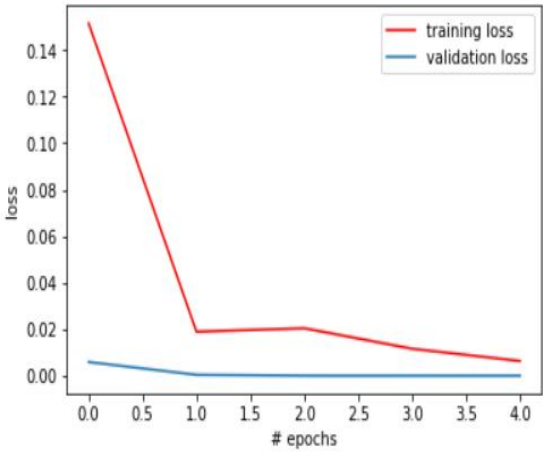


Fig. 9 Graph of training and validation loss of the model trained for alphabets S, M and N

The training and validation accuracy of the model trained for alphabets S, M and N is almost 99% and 99% respectively. This is due to a smaller dataset and overfitting. Fig. 10 depicts the graph of training and validation accuracy.

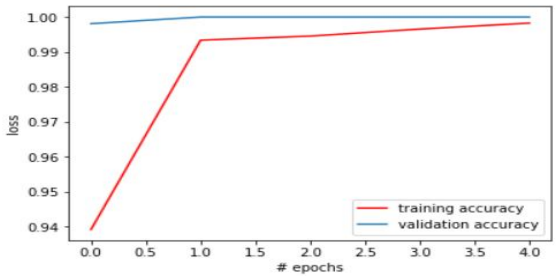


Fig. 10 Graph of Training and Validation Accuracy of the model trained for alphabets S, M and N

The training loss for the model trained for letters T, K, D and I is up to 0.01 in the first four iterations. The validation loss is up to almost 0. Fig. 11 depicts the graph of training and validation loss.

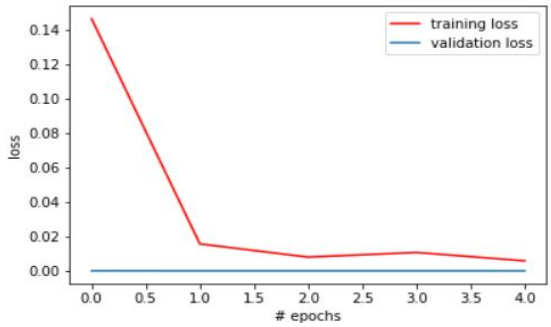


Fig 11. Graph of training and validation loss of the model trained for alphabets T, K, D and I

The training and validation accuracy of the model trained for alphabets T, K, D and I is almost 99% and 99% respectively. Fig. 12 shows the graph of training and validation accuracy.

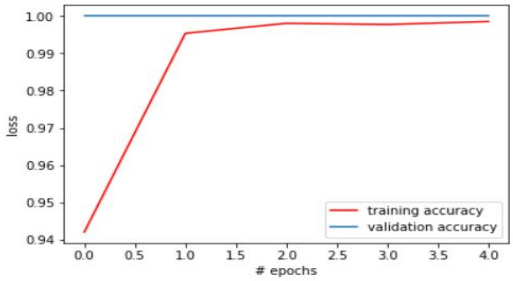


Fig. 12 Graph of Training and Validation Accuracy of the model trained for alphabets T, K, D and I

The user interface is designed according to the user convenience. Fig. 13 shows the user interface of the desktop application after the application is executed and the prediction of the alphabet shown in the web camera as C which is shown through right hand.

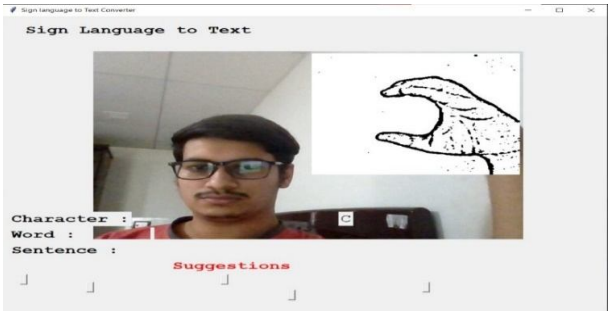


Fig. 13 C Alphabet detected by right hand sign formation

Fig. 14 shows that the model can predict the sign language alphabet irrespective of which hand was used to form the sign.

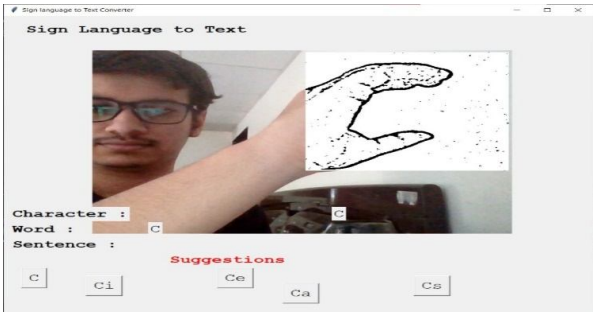


Fig. 14 C alphabet detected by left hand sign formation

Fig. 15 shows the working of the desktop application by giving the prediction of the alphabet along with the sentences formed from suggestions of related words.

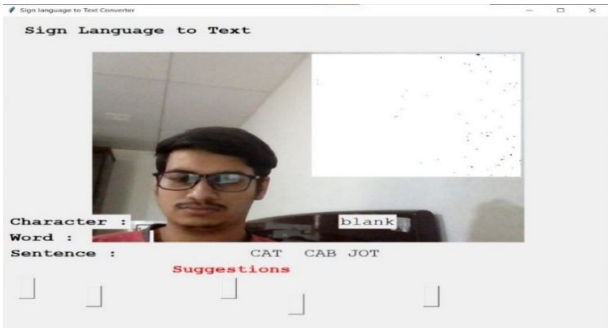


Fig. 15 Working of Desktop App

V. LIMITATIONS

A. The model works well only in good lighting conditions. It may generate errors when tested in dark conditions since the closely related symbols become difficult to differentiate between.
B. Plain background is a must for the model to work efficiently.
C. The user must have a good web camera quality for better performance.
D. Suggestions are not good in some of the cases to form words.

VI. FUTURE SCOPE

The accuracy of the model can be improved to work even in noisy conditions and moderate lighting. The suitable distance from the camera that needs to be maintained while using this application, can be increased for higher efficiency. The suggestion functioning can be developed into an autocorrect system. In future, the model can be made more robust by creating dataset with wide variety of variations with alphabets from other sign languages too. The accuracy of the model can also be increased by using more complex algorithms and by combining this model with natural language processing to make the desktop application more interactive for the users.

VII. CONCLUSIONS

A functional real time vision based American Sign Language Recognition Desktop Application for the deaf and the mute has been successfully developed. The Application is designed to interpret the English Alphabets from A to Z. A dataset was created by capturing images for various letters. The accuracy has been improved by applying various layers of the Convolutional Neural Network as per the requirement, especially for characters that have signs that are similar in appearance. Using this approach, the Application is able to detect almost all signs correctly in noiseless conditions and sufficient lighting.

VIII.ACKNOWLEDGMENT

All the authors would like to thank Prof. Sandip Shinde Sir for his support throughout the project duration and for his valuable suggestions.

REFERENCES

[1] Omkar Vedak, Prasad Zavre, Abhijeet Todkar, Manoj Patil, "Sign Language Interpreter using Image Processing and Machine Learning", International Research Journal of Engineering and Technology (IRJET) Volume: 06 Issue: 04 | Apr 2019

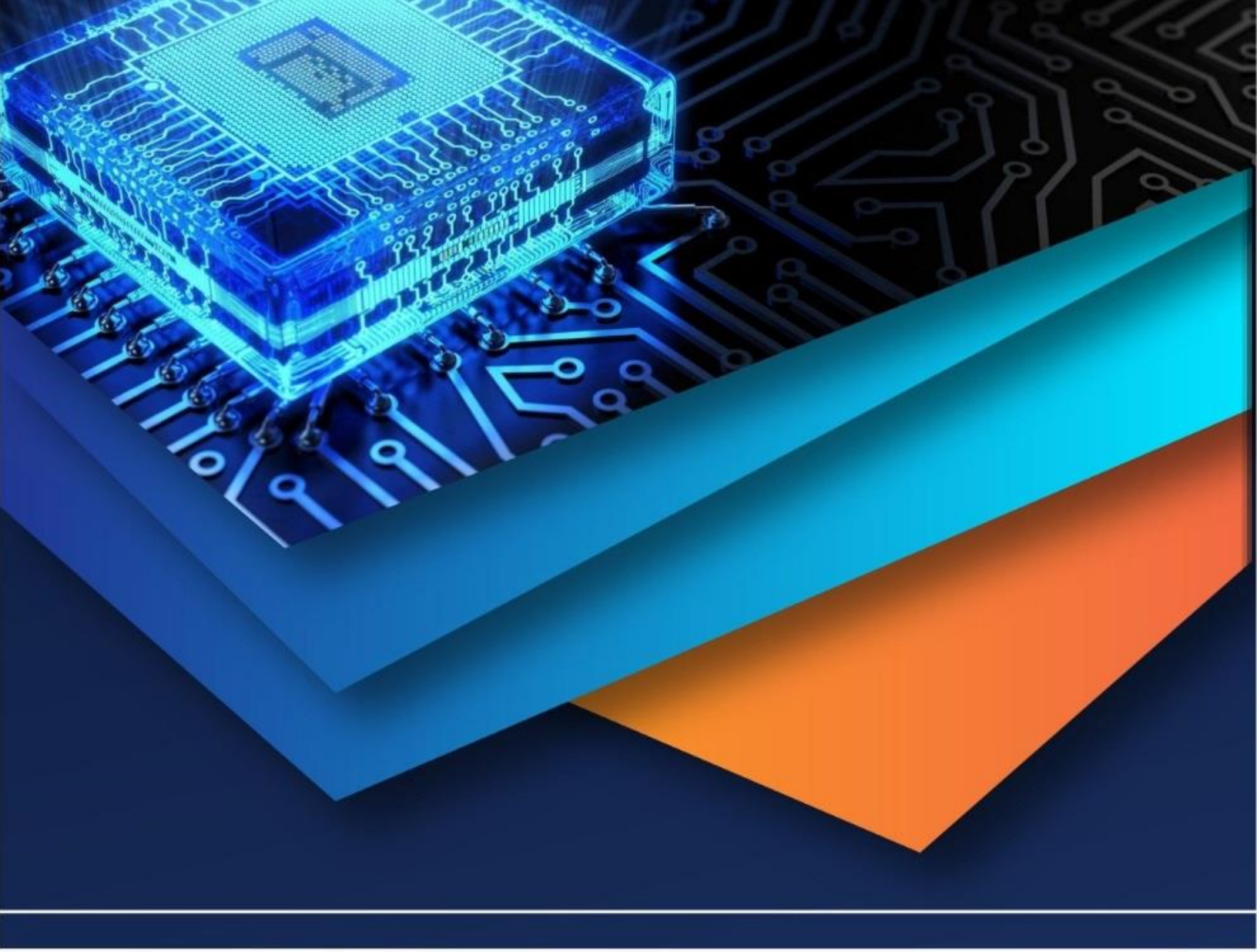
[2] Teak-Wei Chong and Boon-Giin Lee, "American Sign Language Recognition Using LeapMotion Controller with Machine Learning Approach", Department of Electronics Engineering, Keimyung University (19-10-2018)

[3] Jiangbin Zheng, Zheng Zhao, Min Chen, Jing Chen, Chong Wu, Yidong Chen, Xiaodong Shi, Yiqi Tong, "An Improved Sign Language Translation Model with Explainable Adaptations for Processing Long Sign Sentences", Computational Intelligence and Neuroscience, vol. 2020, Article ID 8816125, 11 pages, 2020.

[4] Vishwas S, Hemanth Gowda M, Vivek Chandra H N, Tannvi, "Sign Language Translator Using Machine Learning", International Journal of Applied Engineering Research ISSN 0973-4562 Volume 13, Number 4 (2018)

[5] Kshitij Bantupalli, Ying Xie, "American Sign Language Recognition Using Machine Learning and Computer Vision", Spring 2-18-2019

[6] Brandon Garcia, Sigberto Alarcon Viesca, "Real-time American Sign Language Recognition with Convolutional Neural Networks", http://cs231n.stanford.edu/reports/2016/pdfs/214_Report.pdf



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY