IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
Volume 3, Issue 1, April 2023
Impact Factor: 7.301

Sign Language Detection using Convolutional Neural Network

Anup Kumar Jha[1], Biswajit Dhali[2], Suparana Biswas[3], Antara Ghosal[4], Avali Banerjee[5], Anurima Majumdar[6], Sayan Roy Chaudhuri[7]
Students, Department of Electronics & Communication Engineering[1,2]
Faculty, Department of Electronics & Communication Engineering[3,4,5,6,7]
Guru Nanak Institute of Technology, Kolkata, West Bengal, India

Abstract: *Hand gesture is one of the methods used in sign language for non-verbal communication. It is most commonly used by deaf & dumb people who have hearing or speech problems to communicate among themselves or with normal people. Various sign language systems have been developed by many makers around the world but they are neither flexible nor cost effective for the end users. Hence in this paper introduced software which presents a system prototype that is able to automatically recognize sign language to help deaf and dumb people to communicate more effectively with each other or normal people. Pattern recognition and Gesture recognition are the developing fields of research. Being a significant part in nonverbal communication, hand gestures play a key role in our daily life.*

Keywords: Sign language detection, computer vision, ROI, convolutional neural network

I. INTRODUCTION

Hand Gesture recognition system provides us an innovative, natural, user friendly way of communication with the computer which is more familiar to the human beings. By considering the similarities of human hand shape with four fingers and one thumb, the software aims to present a real time system for recognition of hand gesture on the basis of detection of some shape based features like orientation, Centre of mass centroid, fingers status, thumb in positions of raised or folded fingers of hand**.**

In this field a lot of works already performed. In [1] sensors or motion capturing system has been used.[2] presented a vision based sign language capturing system. In [3] authors reported a sign language recognition system utilizing Machine learning. They have implemented this work using MATLAB and worked with single handed and as well as double handed gestures. Their proposed system achieved the accuracy between 93-96%. For the sign language recognition system different processing methods have been used [4-6]. Such as Hidden Markov Model (HMM) based[7] , Neural Network based [8-12], Naive Bayes Classifier based[13].

In the development of this work, concepts of computer vision, deep learning are broadly used with the availability of their library. For capturing of images to generate train/test data sets, CV2 library is applied. Then a CNN model is generated using packages available in tensorflow library provided by google. Detailed explanation of technical aspects of the model is described exhaustively in coming lines.

The images which we will take, the computer doesn't understand as similar to a human eye. Actually, the images that we see in our phone or computers are generally in pixel format Below is a simple illustration of the grayscale image buffer which stores our image of Abraham Lincoln. Each pixel's brightness is represented by a single 8-bit number, whose range is from 0 (black) to 255 (white):

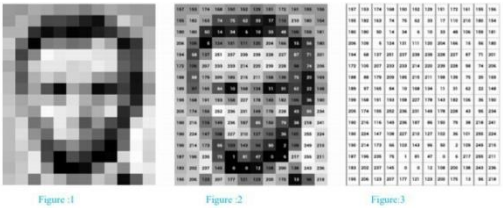


Figure:1 Pixel data diagram. Figure:2 our image of Lincoln; Figure:3 the pixels labeled with numbers from 0–255, representing their brightness.

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
Volume 3, Issue 1, April 2023
Impact Factor: 7.301

In point of fact, pixel values are almost universally stored, at the hardware level, in a one dimensional array. For example, the data from the image above is stored in a manner similar to this long list of unsigned chars.

This way of storing image data may run counter to your expectations, since the data certainly appears to be two-dimensional when it is displayed. Yet, this is the case, since computer memory consists simply of an ever-increasing linear list of address spaces.

Similarly, when we capture every image, images are changed to gray pixel format (because it is helpful in extraction of features for training of models). To do this task we use the OpenCV library. OpenCV is a cross-platform library using which we can develop real-time computer vision applications. It mainly focuses on image processing, video capture and analysis including features like face detection and object detection. In this tutorial, we explain how you can use OpenCV in your applications. Features of OpenCV Library Using OpenCV library, you can –

- Read and write images
- Capture and save videos
- Process images (filter, transform)
- Perform feature detection
- Detect specific objects such as faces, eyes, cars, in the videos or images.
- Analyze the video, i.e., estimate the motion in it, subtract the background, and track objects in it. OpenCV was originally developed in C++. In addition to it, Python and Java bindings were provided. OpenCV runs on various Operating Systems such as Windows, Linux, OSx, FreeBSD, Net BSD, Open BSD, etc.

Creating the dataset for sign language detection: It is fairly possible to get the dataset we need on the internet but in this project, we will be creating the dataset on our own. We will be having a live feed from the video cam and every frame that detects a hand in the ROI (region of interest) created will be saved in a directory (here gesture directory) that contains two folders train and test, each containing 6 folders containing images.

The rest of the paper is organized is as follows: Section II describes the theory of convolution neural network, working principle is described in Section III, Section IV predict the result and finally section V concludes the paper.

II. CONVOLUTIONAL NEURAL NETWORK

In deep learning, a convolutional neural network (CNN/Conv Net) is a class of deep neural networks, most commonly applied to analyze visual imagery. Now when we think of a neural network we think about matrix multiplications but that is not the case with ConvNet. It uses a special technique called Convolution. Now in mathematics convolution is a mathematical operation on two functions that produces a third function that expresses how the shape of one is modified by the other.

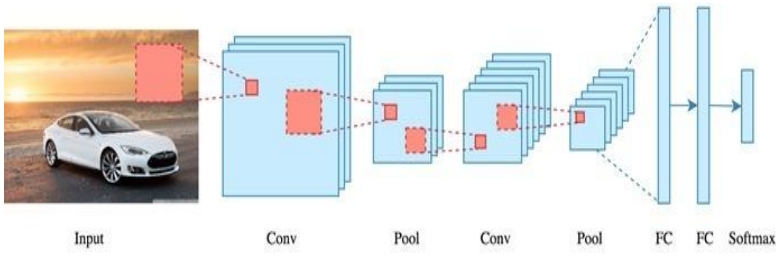


Figure: 4 Architecture of CNN

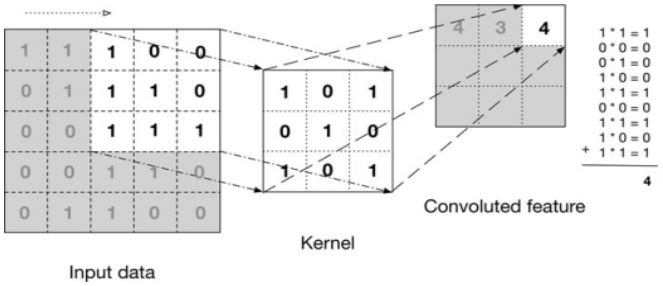


Figure: 5 Convolution operation

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
Volume 3, Issue 1, April 2023
Impact Factor: 7.301

The above image shows what a convolution is. We take a filter/kernel (3×3 matrix) and apply it to the input image to get the convolved feature. This convolved feature is passed on to the next layer.

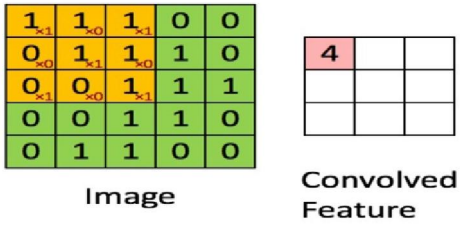


Figure: 6 Convoluted Feature

In the above demonstration, the green section resembles our 5x5x1 input image, I. The element involved in carrying out the convolution operation in the first part of a Convolutional Layer is called the Kernel/Filter, K, represented in the color yellow. We have selected K as a 3x3x1 matrix.

The Kernel shifts 9 times because of Stride Length = 1 (Non-Strided), every time performing a matrix multiplication operation between K and the portion P of the image over which the kernel is hovering.

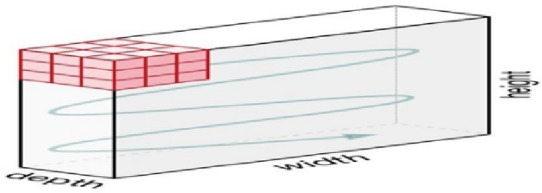


Figure: 7 Movement of the Kernel

The filter moves to the right with a certain Stride Value till it parses the complete width. Moving on, it hops down to the beginning (left) of the image with the same. Stride Value and repeat the process until the entire image is traversed.

Convolutional neural networks are composed of multiple layers of artificial neurons. Artificial neurons, a rough imitation of their biological counterparts, are mathematical functions that calculate the weighted sum of multiple inputs and output an activation value. When you input an image in a Conv Net, each layer generates several activation functions that are passed onto the next layer. The first layer usually extracts basic features such as horizontal or diagonal edges. This output is passed on to the next layer which detects more complex features such as corners or combinational edges. As we move deeper into the network it can identify even more complex features such as objects, faces, etc. Based on the activation map of the final convolution layer, the classification layer outputs a set of confidence scores (values between 0 and 1) that specify how likely the image is to belong to a “class.”In this case the classes are 0,1,2,3,4,5.

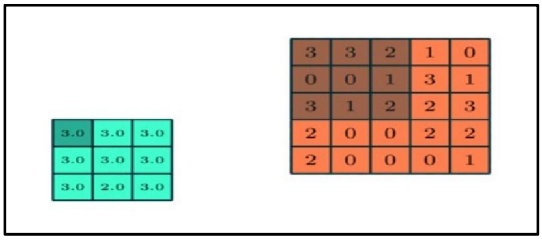


Figure: 8 (3x3) pooling over (5x5) convolved feature

Similar to the Convolutional Layer, the Pooling layer is responsible for reducing the spatial size of the Convolved Feature. This is to decrease the computational power required to process the data through dimensionality reduction. Furthermore, it is useful for extracting dominant features which are rotational and positional invariant, thus maintaining the process of effectively training the model. There are two types of Pooling: Max Pooling and Average Pooling. Max Pooling returns the maximum value from the portion of the image covered by the Kernel. On the other hand, Average Pooling returns the average of all the values from the portion of the image covered by the Kernel. Max Pooling also performs as a Noise Suppressant. It discards the noisy activations altogether and also performs de-noising along with dimensionality reduction. On the other hand, Average Pooling simply performs dimensionality reduction as a noise suppressing mechanism. Hence, we can say that Max Pooling performs a lot better than Average Pooling.

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
Volume 3, Issue 1, April 2023
Impact Factor: 7.301

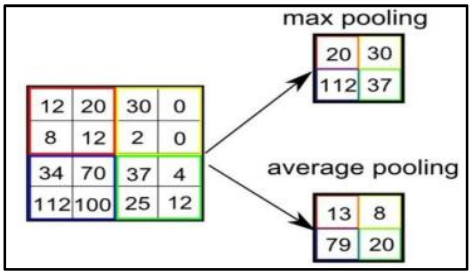


Figure: 9 Types of Pooling

The Convolutional Layer and the Pooling Layer, together form the i-th layer of a Convolutional Neural Network. Depending on the complexities in the images, the number of such layers may be increased for capturing low-level details even further, but at the cost of more computational power. After going through the above process, we have successfully enabled the model to understand the features. Moving on, we are going to flatten the final output and feed it to a regular Neural Network for classification purposes. Classification — Fully Connected Layer (FC Layer).

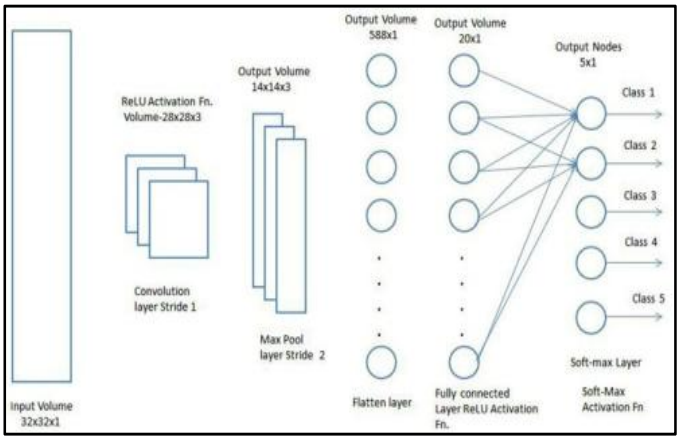


Figure 10 — Fully Connected Layer (FC Layer)

Adding a Fully-Connected layer is a (usually) cheap way of learning non-linear combinations of the high-level features as represented by the output of the convolutional layer. The Fully-Connected layer is learning a possibly non-linear function in that space. Now that we have converted our input image into a suitable form for our Multi-Level Perceptron, we shall flatten the image into a column vector. The flattened output is fed to a feed-forward neural network and backpropagation applied to every iteration of training. Over a series of epochs, the model is able to distinguish between dominating and certain low-level features in images and classify them using the Softmax Classification technique.

III. WORKING PRINCIPLE

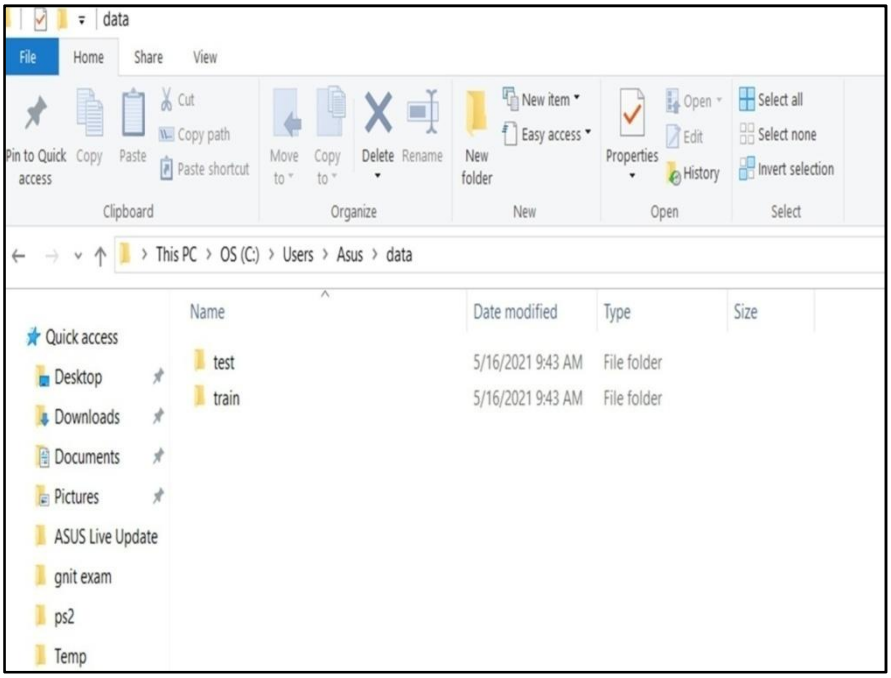
This work consist of 3 important steps these are

1. Creating the dataset
2. Training a CNN on the captured dataset
3. Predicting the data

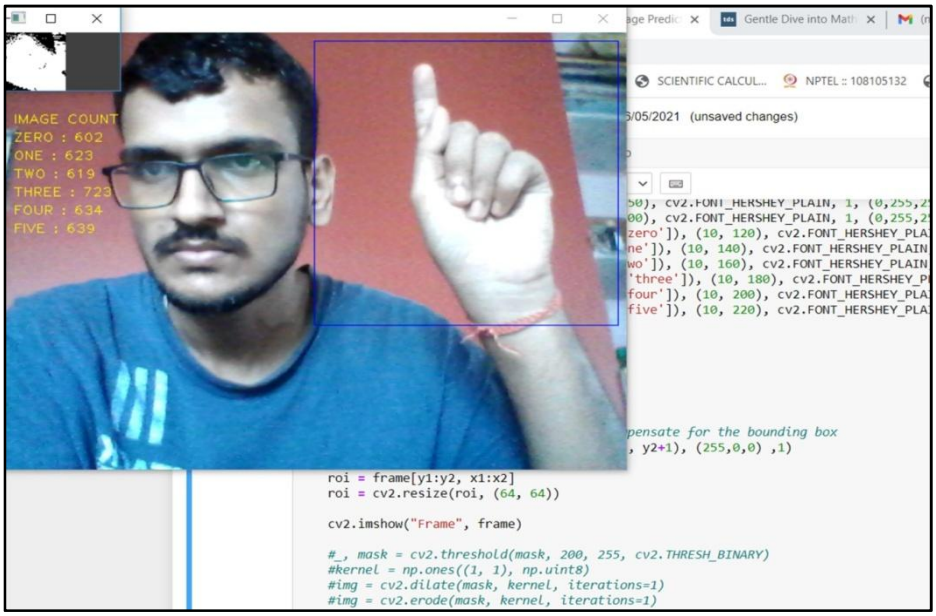
3.1 Creating the Dataset for Sign Language Detection

It is fairly possible to get the dataset we need on the internet but in this project, we will be creating the dataset on our own. We will be having a live feed from the video cam and every frame that detects a hand in the ROI (region of interest) created will be saved in a directory (here gesture directory) that contains two folders train and test, each containing 6 folders containing image.

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
Volume 3, Issue 1, April 2023
Impact Factor: 7.301



Now for creating the dataset we get the live cam feed using OpenCV and create an ROI that is nothing but the part of the frame where we want to detect the hand in for the gestures. The blue box is the ROI and this window is for getting the live cam feed from the webcam.Now we store every data in the file using numerical keys assigned to respective signs like 0 for 'zero' , 1 for 'one' and so on. For the train dataset, we save 600 images for each number to be detected, and for the test dataset, we do the same and create 30 images for each number.



3.2 Training a CNN on the Captured Dataset and Result Graph

Now we design the CNN as follows (or depending upon some trial and error other hyper parameters can be used)

Algorithm:

STEP1: Building the CNN

Initializing the CNN

First convolution layer and pooling

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
Volume 3, Issue 1, April 2023
Impact Factor: 7.301

Second convolution layer and pooling
Flattening the layers
Adding a fully connected layer
Compiling the CNN
STEP 2: Preparing the train/test data and training the model
STEP 3: Saving the model

IV. PREDICTION OF GESTURE AND RESULT

In this, we create a bounding box for detecting the ROI as we did in creating the dataset. This is done for identifying any foreground object. Now we find the max contour and if contour is detected that means a hand is detected so the threshold of the ROI is treated as a test image. We load the previously saved model using keras.models.load_model and feed the threshold image of the ROI consisting of the hand as an input to the model for prediction.

Algorithm for prediction:

1. Loading the model.
2. loading weights into new model
3. Category dictionary
4. Drawing the ROI
5. The increment/decrement by 1 is to compensate for the bounding box
6. Extracting the ROI
7. Resizing the ROI so it can be fed to the model for prediction
8. Sorting based on top prediction
9. Displaying the predictions
10. End

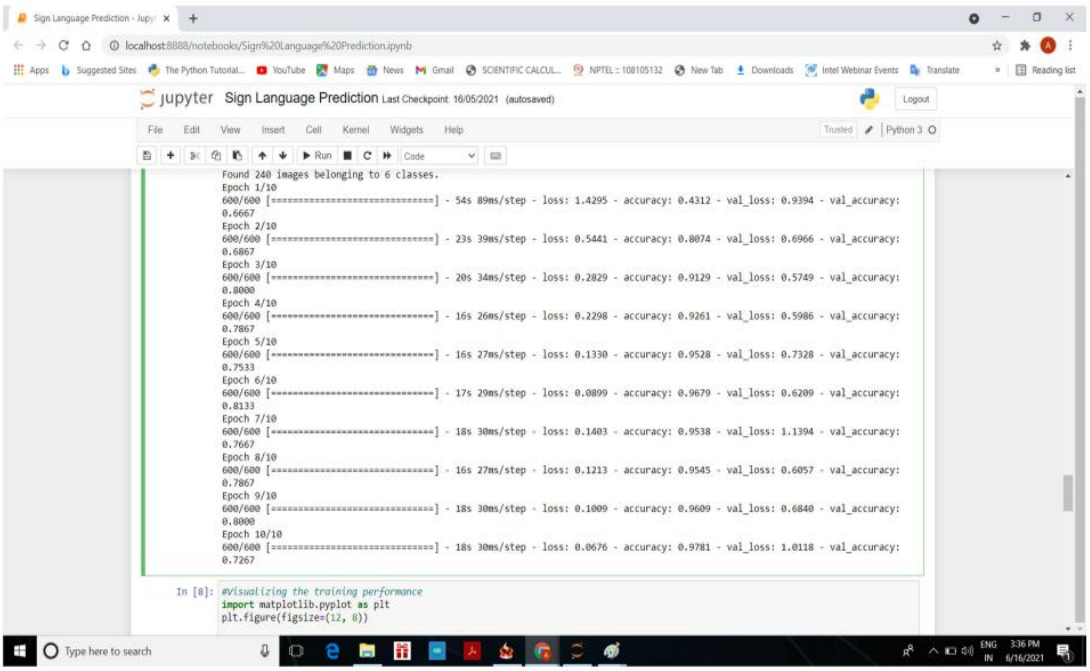


Figure 11. Sign Language prediction using Jupyter notebook

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
Volume 3, Issue 1, April 2023
Impact Factor: 7.301

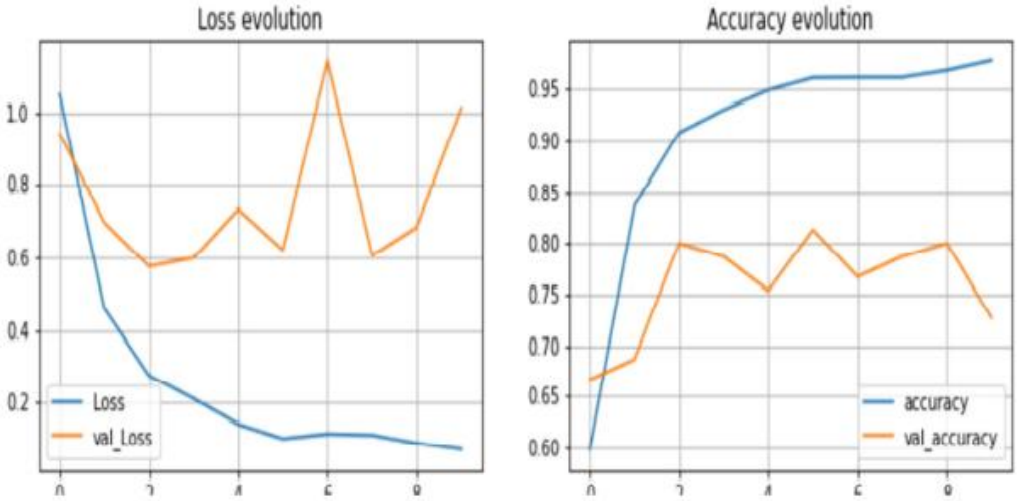


Figure 12. Loss curve and accuracy curve

V. CONCLUSION

Sign language recognition is a hard problem if we consider all the possible combinations of gestures that a system of this kind needs to understand and translate. That being said, probably the best way to solve this problem is to divide it into simpler problems, and the system presented here would correspond to a possible solution to one of them. The system didn't perform too well but it was demonstrated that it can be built a first-person sign language translation system can be built using only cameras and convolutional neural networks.

In our future work more sign languages for alphabets ,symbols etc. can be added and An app can be developed and be used commercially in stores, offices etc. for helping deaf people.

REFERENCES

[1]. Suharjito, Anderson, R., Wiryana, F., Ariesta, M.C., Kusuma, G.P.: Sign Language Recognition Application Systems for Deaf-Mute People: A Review Based on Input-Process-Output. Procedia Comput. Sci. 116, 441–448 (2017). https://doi.org/10.1016/J.PROCS.2017.10.028.

[2]. Konstantinidis, D., Dimitropoulos, K., Daras, P.: Sign language recognition based on hand and body skeletal data. 3DTV-Conference. 2018-June, (2018). https://doi.org/10.1109/3DTV.2018.8478467.

[3]. Dutta, K.K., Bellary, S.A.S.: Machine Learning Techniques for Indian Sign Language Recognition. Int. Conf. Curr. Trends Comput. Electr. Electron. Commun. CTCEEC 2017. 333–336 (2018). https://doi.org/10.1109/CTCEEC.2017.8454988

[4]. Cheok, M.J., Omar, Z., Jaward, M.H.: A review of hand gesture and sign language recognition techniques. Int. J. Mach. Learn. Cybern. 2017 101. 10, 131–153 (2017). https://doi.org/10.1007/S13042-017-0705-5.

[5]. Wadhawan, A., Kumar, P.: Sign Language Recognition Systems: A Decade Systematic Literature Review. Arch. Comput. Methods Eng. 2019 283. 28, 785–813 (2019). https://doi.org/10.1007/S11831-019-09384-2.

[6]. Camgöz, N.C., Koller, O., Hadfield, S., Bowden, R.: Sign language transformers: Joint end-to-end sign language recognition and translation. Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. 10020–10030 (2020). https://doi.org/10.1109/CVPR42600.2020.01004.

[7]. Gaus, Y.F.A., Wong, F.: Hidden Markov Model - Based gesture recognition with overlapping hand-head/hand-hand estimated using Kalman Filter. Proc. - 3rd Int. Conf. Intell. Syst. Model. Simulation, ISMS 2012. 262–267 (2012). https://doi.org/10.1109/ISMS.2012.67.

[8]. Cui, R., Liu, H., Zhang, C.: A Deep Neural Framework for Continuous Sign Language Recognition by Iterative Training. IEEE Trans. Multimed. 21, 1880–1891 (2019). https://doi.org/10.1109/TMM.2018.2889563.

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
Volume 3, Issue 1, April 2023
Impact Factor: 7.301

[9]. Bantupalli, K., Xie, Y.: American Sign Language Recognition using Deep Learning and Computer Vision. Proc. - 2018 IEEE Int. Conf. Big Data, Big Data 2018. 4896–4899 (2019). https://doi.org/10.1109/BIGDATA.2018.8622141.

[10]. Hore, S., Chatterjee, S., Santhi, V., Dey, N., Ashour, A.S., Balas, V.E., Shi, F.: Indian Sign Language Recognition Using Optimized Neural Networks. Adv. Intell. Syst. Comput. 455, 553–563 (2017). https://doi.org/10.1007/978-3-319-38771-0_54.

[11]. Kumar, P., Roy, P.P., Dogra, D.P.: Independent Bayesian classifier combination based sign language recognition using facial expression. Inf. Sci. (Ny). 428, 30–48 (2018). https://doi.org/10.1016/J.INS.2017.10.046.

[12]. Sharma, A., Sharma, N., Saxena, Y., Singh, A., Sadhya, D.: Benchmarking deep neural network approaches for Indian Sign Language recognition. Neural Comput. Appl. 2020 3312. 33, 6685–6696 (2020). https://doi.org/10.1007/S00521-020-05448-8.

[13]. Mohandes, M., Aliyu, S., Deriche, M.: Arabic sign language recognition using the leap motion controller. IEEE Int. Symp. Ind. Electron. 960–965 (2014). https://doi.org/10.1109/ISIE.2014.6864742.