Sign Language Recognition using Deep Neural Network

Aakanksha Rukmana Rangdal
Electronics and Computer Engineering
Sreenidhi Institute of Science and Technology
Hyderabad, India
arrangdal@gmail.com

Sheethal Bandari
Electronics and Computer Engineering
Sreenidhi Institute of Science and Technology
Hyderabad, India
sheethalbandari409@gmail.com

Budhi Manisha
Electronics and Computer Engineering
Sreenidhi Institute of Science and Technology
Hyderabad, India
manishab034@gmail.com

Dr. A. Venkata Ramana
Associate Professor
Electronics and Computer Engineering
Sreenidhi Institute of Science and Technology
Hyderabad, India
venkataramana@sreenidhi.edu.in

***Abstract*—Sign language recognition is an attractive research field with a wide range of applications including telesurgery techniques. Another important application of hand gesture recognition is the translation of sign language, which is a complicated structured form of hand gestures. In sign language, the fingers' configuration, the hand's orientation, and the hand's relative position to the body are the basis of structured expressions. The importance of hand gesture recognition has increased due to the prevalence of touchless applications and the rapid growth of the hearing-impaired population. However, developing an efficient recognition system needs to overcome the challenges of hand segmentation, local hand shape representation, global body configuration representation, and gesture sequence modeling. In this paper, a novel system is proposed for dynamic hand gesture recognition using multiple deep learning architectures for hand segmentation, local and global feature representations, and sequence feature globalization and recognition. The proposed system is evaluated on a very challenging dataset, which consists of dynamic hand gestures performed by subjects in an uncontrolled environment. The results show that the proposed system outperforms state-of-the-art approaches, demonstrating its effectiveness.**

I. Introduction

There are many different sign languages like British, Indian, and American Sign Language. A functioning signing recognition system could provide a chance for the inattentive to communicate with non-signing people without the necessity for an interpreter. It might be wont to generate speech or text: making the deaf more independent. Sign language recognition has been established as a major research area in computer vision with the aim to develop device capable of real-time communication. This paper proposes a novel solution for sign language recognition by adopting convolutional neural networks (CNNs) with deep learning to solve the problem of physically challenged people who are deaf and dumb particularly. Sign Language Recognition can be briefly said as the process of converting the signs and gestures shown by the user to a particular letter that the sign represents. This importantly helps to fill the gap between people who cannot speak and the general public. Image Processing has nowadays become a rapidly improving technology with various algorithms and here, we are using Image Processing algorithms with neural networks that map the gesture to appropriate letter in the training data. In this project, we aim to develop a system that classify signing accurately. As the deaf and speech impaired people need a proper channel to communicate, there is a need for this system. Not everyone can understand the sign language of these challenged people[1].

II. Literature Survey

A model named ASL Recognition Based on Coupling Between HMMs and 3D Analysis was proposed, in this the authors Vogler, C. and Metaxas, D. 2001 developed a framework for recognizing American Sign Language (ASL) from three-dimensional data obtained with computer vision techniques. They showed how to collect three-dimensional data from computer vision and use them as input to Hidden Markov Models. By using context-dependent modelling, improved recognition performance was achieved with the hierarchical back propagation algorithm [5]. The system on Dynamic-static unsupervised sequentiality, statistical subunits and lexicon for sign language recognition by Stavros Theodorakis, Vassilis Pitsikalis, Petros Maragos, 2014 Volume32,Issue8 work focuses on the development of algorithms that support automatic recognition of signs using non-parametric deep neural networks. We introduce a deep neural network based on dynamic-static statistical subunits and provide sequentiality in an unsupervised manner, without prior linguistic information. Subunit "sequentiality" refers to the decomposition of signs into two types of parts, varying and non-varying, that are sequentially stacked across time. Sequences of the generated subunits are used as sign pronunciations in a data-driven lexicon. Based on this lexicon and the corresponding segmentation, each subunit is statistically represented and trained on multimodal sign data as a hidden Markov model[6].

Adithya, V., Vinod, P. R., Gopalakrishnan introduced Artificial neural network-based method for Indian sign language recognition. This method was used to analyze the digital image of a written word, which was derived from video recordings of sign language. The authors implemented a convolutional neural network for recognizing fingerspelling in Indian Sign Language. The proposed method was designed for measuring character recognition error, which is indicative of how well the system recognizes different signs and the occurrence of other types of errors[10]. Gesture recognition is a challenging task due to the large number of possible hand segmentations and motions. Traditional methods of gesture recognition currently rely on visual information that is often inaccurate due to illumination changes. Vision Based Hand Gesture Spotting and Recognition Using CRF and SVM by Fayed F. M. Ghaleb, Ebrahim, A. Youness and Mahmoud Elmezain proposed a novel gesture spotting and recognition technique based on Conditional Random Fields in conjunction with Support Vector Machine (SVM). To alleviate the impact of illumination changes in real-time, the model also proposed a feature extraction method that uses random sliders to extract hand segmentations from continuous hand motion[1].

K-nearest correlated neighbor classification for Indian sign language gesture recognition using feature fusion. This proposes a model for recognizing signed alphabets in the Indian Sign Language. The model first categorizes them as single-handed or double-handed. For both categories, two kinds of features, namely HOG and SIFT, are extracted for a set of training images and are combined into a single matrix. Correlation is computed between these matrices, which results in different values depending on the sign given by the user. A K-Nearest Neighbor Classifier is then used to classify test images [8].

Sandrine Tornay, Oya Aran, Mathew Magimai-Doss proposed an HMM Approach with Inherent Model Selection for Sign Language and Gesture Recognition. This project introduces an efficient and fast algorithm for the identification of the number of fingers opened in a gesture representing an alphabet of the Binary Sign Language. The system does not require the hand to be perfectly aligned to the camera. The primary objective of this project is to develop a computer-based intelligent system that will enable dumb people significantly to communicate with all other people using their natural hand gestures. The system is we are implementing for Binary sign language but can detect any sign language with prior image processing [5].

III. Proposed Methodology

The foundational point of our research is to fill the gap between voice impaired and deaf people. In our proposed system, we are using Convolutional Neural Networks. The proposed system is a deep learning-based solution for American Sign Language (ASL) recognition. The first step of this system is to collect data. Our system uses the web camera to take photos of signs and after a series of processing operations a background region is eliminated using color extraction. Segmentation is then performed on images to detect skin tones, using an algorithm that involves morphological operations and dilation and erosion. With open CV, the images obtained undergo algorithms including de-noising and resizing so there won't be any difference in size between different gestures. We have 2,000 ASL images that we use for training; 800 are for testing purposes, with 80% being for training and 20% for testing. It's in the ratio 80:20. This project is intended to address the issue of recognition of signs by analyzing the sequences of images and extracting the codes that correspond to each symbol. The vision system is implemented using TensorFlow framework and a convolutional neural network (CNN) that uses CVX-11 loss function for training. The convolutional neural networks are a type of artificial neural network (ANN) developed by playing around with mathematical equations. They have been used in many applications to recognize and identify objects, animals, colors and many other things. There are several ways that we can implement these networks on top of any machine learning task. The first step in implementing them is by training the network on data that is suitable for it as a way for us to get good results out of it. The use of feature extraction also helps in scaling algorithms that can handle large volumes of data faster than before.

In this project, we will show you how to use CNN to recognize hand gestures. This model is a multilayered feedforward neural network mostly used in image recognition. The architecture of CNN consists of some convolution layers, each comprising of a pooling layer, activation function, and batch normalization which is optional. It also has a set of fully connected layers. As one of the images moves across the network, it gets reduced in size. This happens as a result of max pooling. The last layer gives us the prediction of the class probabilities.

To address the challenges of data engineering, we propose a sign language recognition system using convolutional neural networks (CNN). The proposed model is designed to process large volume of sign language data streams to achieve robust training and testing results. It consists of two parts - sequential model for segmenting a set of hand pixels; and recurrent network for recognizing hand gestures by comparing captured frames with handwritten text labels.

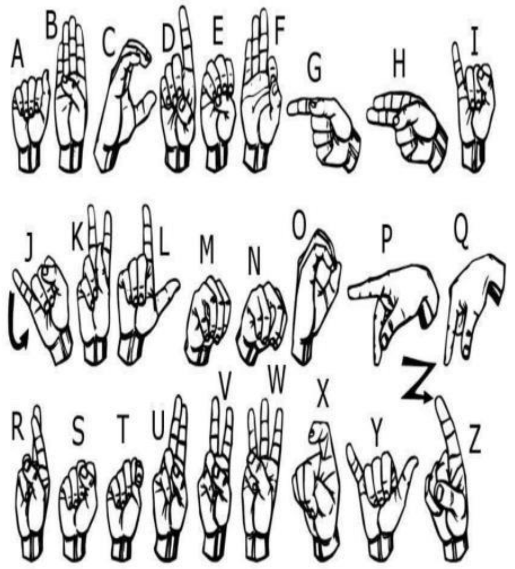


Fig3.1 – American Sign Language

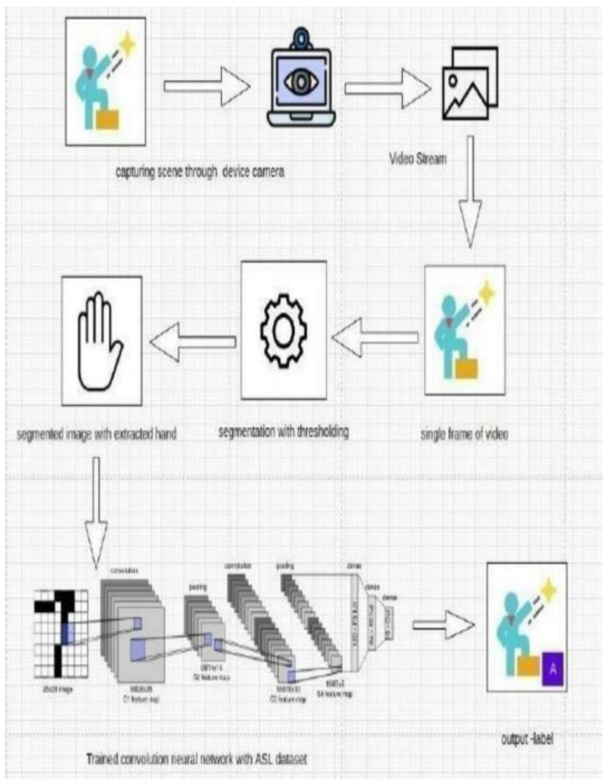

capturing scene through device camera
Video Stream
segmented image with extracted hand
segmentation with thresholding
single frame of video
Trained convolution neural network with ASL dataset
output -label

Fig. 1. Block Diagram of sign language recognition

IV. Functional Requirements

The functional requirements are to be necessarily specified. A function can be specified by identifying the state at which data is to be input to the system, its input data domain, the output domain, and the type of processing to be carried on the input data to obtain the output data. Functional requirement define specific behavior or function of an application.

Following are the functional requirements of this project:
FR-01 Face Detection: This software shall utilize a face detection system to filter out faces from the video capturing device. The face recognition system will be implemented with the help of OpenCV. To improve face detection accuracy and to detect multiple faces simultaneously, thus reducing the amount of calculation needed to perform hand detection.
FR-02 Skin Detection Module: This software shall perform skin color detection and filter out all objects that do not contain the color of skin. By filtering objects of non-skin color, the system can then use its remaining resources and focus on hand detection and gesture recognition. This also allows the system to pinpoint possible locations of the user's hands. The skin detection module can be achieved by using the Marvin Image Processing Framework. FR-03 Filtered Object Detection: After the recognition process, the network is able to classify each blob as either skin or non-skin colored object. For example, if an image has two blobs with different intensities of color but they are both skin colored, then they can be classified as a single blob with the same intensity of color. Once the program has filtered out most of the unwanted parts of the picture after using the skin detection module, it shall recognize groups of "clusters" skin- colored objects also known as "blobs" and extract their sizes. FR-04 Object Location: Upon detecting, the system will be now able to compute the exact location of the object using simple trigonometric math.
FR-05 Hand Calibration: Depending on the user's preferences, the system shall perform adjustments according to the user's dominant hand. This means that if the user is right- handed, the mouse control gesture mode should be recognized near the right side of the face instead of the whole field of view. This is achieved through trigonometry math conversions.
FR-06 Mouse Movement Gesture Control Mode: After obtaining the location of the hand, the software shall use the detected location as the mouse cursor point. As the user moves his/her hands, the mouse should follow promptly on the screen. FR-07 Browsing Gesture Control Mode: The software shall allow the user to use the "Browsing Gesture Mode". In this mode, the user's hand gesture will only be recognized for commands including the previous page, next page, scroll up and scroll down.

V. Non-Functional Requirements

A non-functional requirement is a requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behaviours. Non-functional requirements may be defined as constraints which are not essential to the performance of the solution. Often these are specifications on how much information should be collected, when and for what purpose, what computational resources must be available, or who will use it. Following are the non-functional requirements of the project: NFR-01 Efficiency in Computation: This software willminimize the use of Central Processing Unit (CPU) and memory resources on the operating system. When HGR is executing the software shall utilize less than 80% of the system's CPU resource and less than 100 megabytes of system memory.

NFR-02 Extensibility: The software will be extensible to support future developments and add-ons to the HGR software. The gesture control module of HGR shall be at least 50% extensible to allow new gesture recognition features to be added to the system.

NFR-03 Portability: The HGR software is 100% portable to all operating platforms that support Java Runtime Environment (JRE). Therefore, this software should not depend on different operating systems.

NFR-04 Performance: This software will minimize the number of calculations needed to perform image processing and hand gesture detection. Each captured video frame will be processed within 350 milliseconds to achieve 3 frames per second performance.

NFR-05 Reliability: The HGR software will be operable in all lighting conditions. Regardless of the brightness levels in user's operating environment, the program will always detect user's hands.

NFR-06 Usability This software shall be easy to use for all users with minimal instructions. 100% of the languages on the graphical user interface (GUI) shall be intuitive and understandable by non- technical users.

VI. Technologies Used

- Software Requirements

Programming Language: Python

IDE	:	pycharm/Jupyter notebook

- Hardware Requirements

Processor	:	Intel i3 and above
RAM	:	4GB and Higher
Hard Disk	:	500GB Minimum

VII. Implementation

Supervised machine learning involves creating a model of the data that is being analyzed, then training that model using input data and expected output data. To create such a model, it is necessary to go through four phases:

- Model construction
- Model Training
- Model testing
- Model evaluation

Beginning with Model Construction, it depends on machine learning algorithms. In this projects case, it was convolutional neural networks. Such an algorithm looks like:

1. Begin with its object: model = Sequential()

2.Then consist of layers with their types: model.add(type_of_layer())

3. After adding a sufficient number of layers the model is compiled. At this moment Keras communicates with TensorFlow for construction of the model. During model compilation it is important to write a loss function and an optimizer algorithm. It looks like: model.compile(loss='name_of_loss_function',optimizer='name_of_opimazer_alg') The loss function shows the accuracy of each prediction made by the model before training begins.

After model construction, training is required to learn how the model works. In this phase, the model is trained using training data and expected output for this data. It looks as follows: model.fit(training_data, expected_output). Progress is visible on the console when the script runs. At the end it will report the final accuracy of the model. In Model testing phase, a second set of data is loaded. This second set of data has never been seen by the model, and therefore its true accuracy will be verified. After training is complete, it is understood that the model shows the right result, saving it as a .h5 file (Hierarchical Data Format) using model.save("name_of_file.h5"). Finally, the saved model can be used in the real world. The name of this phase is evaluation; this means that it can be used to evaluate new data. Data Preprocessing and Image classification a are then continued. Training and testing data sets are used that comprises of images of hands gestures that represent particular signs

VIII. Algotihms Used

1) Histogram calculation:

Histograms collect counts of data that are organized into a set of predefined bins.

When we say data, we are not restricting it to be an intensity value. The data collected can be whatever, a feature you find useful to describe your image. For example, imagine that a Matrix contains information of an image (i.e. intensity in the range 0−255):

What happens if we want to count this data in an organized way? Since we know that the range of information value for this case is 256 values, we can segment our range in subparts (called bins) like:

[0,255]=[0,15]U[16,31]U...U[240,255]range=bin1Ubin2Ubinn=15

And we can keep count of the number of pixels that fall in the range of each bini

2) Back Propagation: Back-propagation is the essence of neural net training. It is the method of fine-tuning the weights of a neural net based on the error rate obtained in the previous epoch (i.e., iteration). Proper tuning of the weights allows you to reduce error rates and to make the model reliable by increasing its generalization.

Backpropagation is a short form for "backward propagation of errors." It is a standard method of training artificial neural networks. This method helps to calculate the gradient of a loss function with respect to all the weights in the network.

3) Optimizer (Adam): Adam can be looked at as a combination of RMSprop and Stochastic Gradient Descent with momentum. It uses the squared gradients to scale the learning rate like RMSprop and it takes advantage of momentum by using moving average of the gradient instead of gradient itself like SGD with momentum. Adam is an adaptive learning rate method, which means, it computes individual learning rates for different parameters. Its name is derived from adaptive moment estimation, and the reason it's called that is because Adam uses estimations of first and second moments of gradient to adapt the learning rate for each weight of the neural network. N-th moment of a random variable is defined as the expected value of that variable to the power of n.

Loss Function (categorical cross entropy): Categorical cross entropy is a loss function that is used for single label categorization. This is when only one category is applicable for each data point. In other words, an example can belong to one class only.

4) SEGMENTATION

Image segmentation is the process of partitioning a digital image into multiple segments (sets of pixels, also known as image objects). The goal of segmentation is to simplify and/or change the representation of an image into something that is more meaningful and easier to analyze. Modern image segmentation techniques are powered by deep learning technology.

Image Segmentation working: Image Segmentation involves converting an image into a collection of regions of pixels that are represented by a mask or a labeled image. By dividing an image into segments, you can process only the important segments of the image instead of processing the entire image. A common technique is to look for abrupt discontinuities in pixel values, which typically indicate edges that define a region. Another common approach is to detect similarities in the regions of an image. Some techniques that follow this approach are region growing, clustering, and thresholding. A variety of other approaches to perform image segmentation have been developed over the years using domain-specific knowledge to effectively solve segmentation problems in specific application areas [5].

5) CLASSIFICATION: CONVOLUTIONAL NEURAL NETWORK

Image classification is the process of taking an input (like a Picture) and outputting its class or probability that the input is a particular class. Neural networks are applied in the following steps:

1) One hot encoding of the data: A one-hot encoding can be applied to the integer representation. This is where the integer encoded variable is removed and a new binary variable is added for each unique integer value.

2) Define the model: A model said in a very simplified form is nothing but a function that is used to take in certain input, perform certain operation to its best on the given input (learning and then predicting/classifying) and produce the suitable output.

3) Compile the model: The optimizer controls the learning rate. We will be using 'adam' as our optimizer. Adam is generally a good optimizer to use for many cases. The adam optimizer adjusts the learning rate throughout training. The learning rate determines how fast the optimal weights for the model are calculated. A smaller learning rate may lead to more accurate weights (up to a certain point), but the time it takes to compute the weights will be longer.

4) Train the model: Training a model simply means learning (determining) good values for all the weights and the bias from labeled examples. In supervised learning, a machine learning algorithm builds a model by examining many examples and attempting to find a model that minimizes loss; this process is called empirical risk minimization.

5)Test the model: A convolutional neural network convolves learned featured with input data and uses 2D convolution layers [9].

Convolution Operation:

In purely mathematical terms, convolution is a function derived from two given functions by integration which expresses how the shape of one is modified by the other.

Convolution formula: Here are the three elements that enter into the convolution operation:

• Input image

• Feature detector

• Feature map

Steps to apply convolution layer:

• You place it over the input image beginning from the top-left corner within the borders you see demarcated above, and then you count the number of cells in which the feature detector matches the input image.

• The number of matching cells is then inserted in the top-left cell of the feature map

You then move the feature detector one cell to the right and do the same thing. This movement is called a and since we are moving the feature detector one cell at time that would be called a stride of one pixel.

• What you will find in this example is that the feature detector's middle-left cell with the number 1 inside it matches the cell that it is standing over inside the input image. That's the only matching cell, and so you write “1” in the next cell in the feature map, and so on and so forth.
• After you have gone through the whole first row, you can then move it over to the next row and go through the same process. There are several uses that we gain from deriving a feature map. These are the most important of them: Reducing the size of the input image, and you should know that the larger your strides (the movements across pixels), the smaller your feature map[10].

ReLu Layer:
Rectified linear unit is used to scale the parameters to non-negative values. We get pixel values as negative values too. In this layer we make them as 0’s. The purpose of applying the rectifier function is to increase the non-linearity in our images. The reason we want to do that is that images are naturally non-linear. The rectifier serves to break up the linearity even further in order to make up for the linearity that we might impose an image when we put it through the convolution operation. What the rectifier function does to an image like this is remove all the black elements from it, keeping only those carrying a positive value (the grey and white colors).
The essential difference between the non-rectified version of the image and the rectified one is the progression of colors. After we rectify the image, you will find the colors changing more abruptly. The gradual change is no longer there. That indicates that the linearity has been disposed of[11].

Pooling Layer:
The pooling (POOL) layer reduces the height and width of the input. It helps reduce computation, as well as helps make feature detectors more invariant to its position in the input This process is what provides the convolutional neural network with the “spatial variance” capability. In addition to that, pooling serves to minimize the size of the images as well as the number of parameters which, in turn, prevents an issue of “overfitting” from coming up. Overfitting in a nutshell is when you create an excessively complex model in order to account for the idiosyncrasies we just mentioned. The result of using a pooling layer and creating down sampled or pooled feature maps is a summarized version of the features detected in the input. They are useful as small changes in the location of the feature in the input detected by the convolutional layer will result in a pooled feature map with the feature in the same location. This capability added by pooling is called the model’s invariance to local translation.

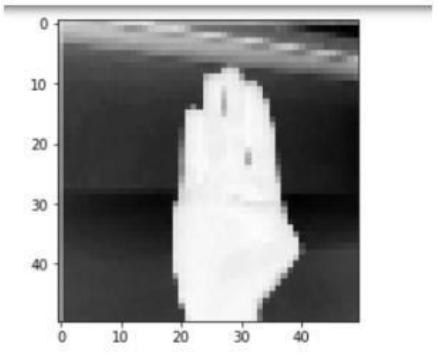
Fully Connected Layer:
The role of the artificial neural network is to take this data and combine the features into a wider variety of attributes that make the convolutional network more capable of classifying images, which is the whole purpose from creating a convolutional neural network. It has neurons linked to each other, and activates if it identifies patterns and sends signals to output layer.
The output layer gives output class based on weight values, for now, all you need to know is that the loss function informs us of how accurate our network is, which we then use in optimizing our network in order to increase its effectiveness. That requires certain things to be altered in our network. These include the weights (the blue lines connecting the neurons, which are basically the synapses), and the feature detector since the network often turns out to be looking for the wrong features and has to be reviewed multiple times for the sake of optimization. This full connection process practically works as follows:

• The neuron in the fully-connected layer detects a certain feature; say, a nose.

• It preserves its value.
• It communicates this value to the classes trained images

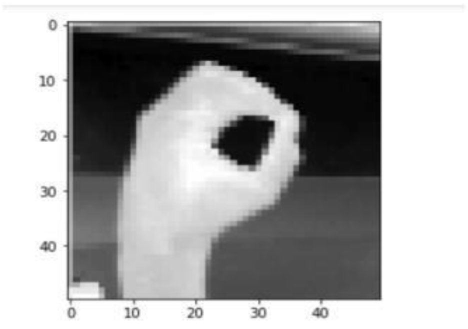
IX. Results

The hand gesture of American sign language in sample result 1 is the letter B and letter O in sample result2. Our sign language recognition predicted the output based on the given input as test image and using the trained data. The input is passed to the model and trained using Convolutional neural network and the output predicted is letter B and letter O.



```
Original Labels: B
Model prediction: B
```

Fig 9.1 result – 1



```
Original Labels: O
Model prediction: O
```

Fig 9.2 result-2

Model Accuracy

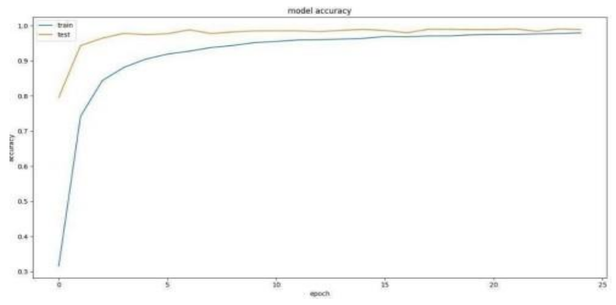


Fig 9.3 Model Accuracy graph

X. CONCLUSION

Nowadays, applications need several kinds of images as sources of information for explanation and analysis. Several features are to be extracted so as to perform various applications. When an image is transformed from one form to another such as digitizing, scanning, and communicating, storing, etc. degradation occurs. Therefore, the output image has to undertake a process called image enhancement, which contains of a group of methods that seek to develop the visual presence of an image. Image enhancement is fundamentally enlightening the interpretability or awareness of information in images for human listeners and providing better input for other automatic image processing systems. Image then undergoes feature extraction using various methods to make the image more readable by the computer. Sign language recognition system is a powerful tool to prepare an expert knowledge, edge detect and the combination of inaccurate information from different sources, the intend of convolution neural network is to get the appropriate classification

The proposed sign language recognition system used to recognize sign language letters can be further extended to recognize gestures facial expressions. Instead of displaying letter labels it will be more appropriate to display sentences as more appropriate translation of language. This also increases readability. The scope of different sign languages can be increased. More training data can be added to detect the letter with more accuracy. This project can further be extended to convert the signs to speech.

REFERENCES

[1] Fayed F. M. Ghaleb, F., Youness, E., Elmezain, M. and Dewdar, F. “Vision-Based Hand Gesture Spotting and Recognition Using CRF and SVM” 2015, Journal of Software Engineering and Applications, 8, 313-323.
[2] Kumar, P., Gauba, H., Roy, P. P., and Dogra, D. P,” Coupled HMM-based multi-sensor data fusion for sign language recognition. Pattern Recognition Letters”,2017, 86:1–8.
[3] B. Liao, J. Li, Z. Ju and G. Ouyang, "Hand Gesture Recognition with Generalized Hough Transform and DC-CNN Using Real sense," 2018 Eighth International Conference on Information Science and Technology (ICIST), Cordoba, 2018, pp. 84-90.
[4] https://www.researchgate.net/publication/327529562_Gesture_and_Sign_Language_Recognition_with_Deep_Learning.
[5] Vogler, C. and Metaxas,” A framework for recognizing the simultaneous aspects of American signlanguage. Computer Vision and Image Understanding”, 2001,81(3):358–384.
[6] Stavros Theodorakis, Vassilis Pitsikalis,, Petros Maragos, “Dynamic-static unsupervised sequentiality, statistical subunits and lexicon for sign language recognition”, 2014Volume- 32 Isuue-8
[7] Ong, E.-J., Cooper, H., Pugeault, N., and Bowden, “Sign language recognition using sequential pattern trees. In Computer Vision and Pattern Recognition (CVPR)”, 2012 IEEE Conference on, pages 2200–2207. IEEE
[8] B. Gupta, P. Shukla and A. Mittal, "K-nearest correlated neighbor classification for Indian sign language gesture recognition using feature fusion," 2016 International Conference on Computer Communication and Informatics (ICCCI), 2016, pp. 1-5
[9] S. Ji, W. Xu, M. Yang, and K. Yu, ‘‘3D convolutional neural networks for human action recognition,’’ IEEE Trans. Pattern Anal. Mach. Intell., vol. 35, no. 1, pp. 221–231, Jan. 2013.
[10] V. Adithya, P. R. Vinod and U. Gopalakrishnan, "Artificial neural network based method for Indian sign language recognition," 2013 IEEE Conference on Information & Communication Technologies, 2013, pp. 1080-1085.