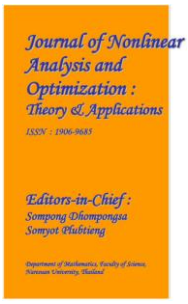
Journal of **N**onlinear **A**nalysis and **O**ptimization
Vol. 15, Issue. 1, No.15 : 2024
ISSN : **1906-9685**



Paper ID: ICRTEM24_173 **ICRTEM-2024 Conference Paper**

SIGN LANGUAGE RECOGNITION USING CONVOLUTIONAL NEURAL NETWORK

[#1]**Dr. T. VEERANNA**, Assoc. Prof., Dept. of CSE,
[#2]**Mr. S. SUNEEL KUMAR**, Asst. Prof., Dept. of CSE(AI&ML),
SAI SPURTHI INSTITUTE OF TECHNOLOGY, SATHUPALLI, KHAMMAM

ABSTRACT: Communicating with a person with a hearing impairment is always difficult. The paper is an effort (extension) to investigate the difficulties associated with character classification in Indian Sign Language (ISL). For people who are deaf or hard of hearing, sign language is not enough for communication. For someone who has never learned this language, the gestures made by people with disabilities are inconsistent or mixed. Both parties should communicate. An Indian Sign Language-based Sign Language recognition is presented in this paper. For the purpose of this analysis, the system must be able to predict and display the name of the captured image, and the user must be able to capture images of hand gestures using a web camera. The captured image is processed through a series of steps that include computer vision techniques like dilation, mask operation, and changing the image to grayscale. Convolutional Brain Organization (CNN) is utilized to prepare our model and recognize the photos. The accuracy of our model is approximately 95%. Key words: Deaf people, Hand Gesture, Convolutional Neural Network (CNN), Sign Language Recognition (SLR), and Indian Sign Language.

1 Introduction

One of the main prerequisites for social endurance is correspondence. Non-deaf and dumb people struggle to comprehend sign language used by deaf and dumb people to communicate with one another. Indian sign language differs significantly from American sign language, despite the extensive research on its recognition. Twenty out of 26 languages use two hands to communicate, whereas ASL uses only one hand. When using both hands, features are frequently obscured as the hands overlap. Moreover, an absence of datasets, joined with the way that sign lan-guage differs relying upon area, has restricted ISL ges-ture location endeavors. The goal of this paper is to make the first step toward using Indian sign language to bridge the communication gap between people who are normal and people who are deaf. The inclusion of words and common phrases in this project will not only make it simpler for people who are deaf or dumb to communicate with the outside world, but it may also aid in the creation of autonomous systems that can assist and understand them.

This paper aims to identify Indian Sign Language alphabets by employing the

corresponding gesture. In American Sign Language, the identification of gestures and sign languages is a well-studied subject, but in Indian Sign Language, it has received little attention. Instead of using high-end technology like gloves or the Kinect to solve this problem, we want to use computer vision and machine learning techniques to extract and classify specific features from photographs that can be accessed through a webcam.

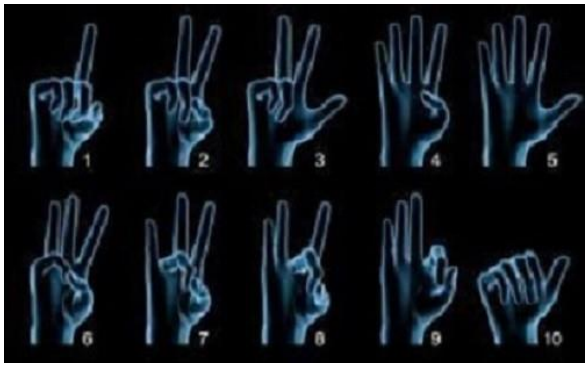


Figure 1. Indian Sign Language Alphabets.

1.1 Problem statement

Figuring out the exact importance of not too sharp peo-ple's emblematic motions and changing over it into comprehend capable language(Text).

2 Literature Survey

Numerous attempts have been made to address sign recognition in videos and images using various algorithms, according to a review of the literature for the proposed framework. The CamShift algorithm was used to detect real-time hand gestures in Jing-hao Sun[1], which separated the human hand from the complex context. The region of hand movements that were observed in real time is then recognized using a convolutional neural network, and ten common digits are identified. There are a total of 1600 images in the proposed system's training dataset, including 4000 images of hand gestures and 400 images for each type. This experi-ment shows precision around 98.3 percent. Hasan[2] utilized scaled standardization to perceive motions utilizing splendid ness factor coordinating. For the purpose of segmenting the input data, thresh-olding methods are applied against a background that is black. Any segmented image's coordinates are shifted to match the hand unit's centroid at the X and Y axis origins. furthermore, the picture's middle not entirely set in stone. Wysoski et al. use a boundary histogram[3]. provided postures that are rotation-invariant. Using a standard contour tracking algorithm, a clustering procedure was used to locate the border line of each category in the pooling image after the input image was captured with a camera and filtered for skin color detection. The image was used to create grids, and the boundaries were normalized. An ASL symbol recognition system was developed by Geethu Nath and Arun C.S. [6] using the ARM CORTEX A8 processor. The Jarvis algorithm and the template matching algorithm are used by the machine to recognize numbers and alphabets. Kumud Tripathi [7] developed a framework for recognizing continuous ISL features by utilizing various distance classifiers and Principal Component Analysis (PCA). Orientation Histogram is used to extract the keyframe features from the own data set and provide them to the device as input. The Modified k-Nearest Neighbor (MKNN) method

was proposed by Noor Tubaiz [8] for the purpose of classifying sequential data. Hand movements can be tracked with data gloves. To enhance the crude information, window-based factual highlights are determined from past crude component vectors and future crude element vectors. The proposed framework was developed using novel techniques based on existing systems (ISL) to recognize ISL terms. (B. Bauer et al.) Describe an approach for a continuous sign language recognition method. A system relies upon consistent secret Markov models pictures (Gee). German sign language (GSL) is used. The device is fed feature vectors that represent manual signs. [9]

Proposed Methodology

3.1 Image Aquistion:

The process of extracting an image from a source, typically a hardware-based source, for the purpose of image processing is referred to as this action. Our project's hardware-based source is WebCamera. It is the most important phase in the work process grouping be-cause no handling should be possible without a picture. The image that is obtained has not undergone any kind of processing.

3.2 Segmentation:

The method of separating objects or signs from the con- text of a captured image is known as segmentation. Con-

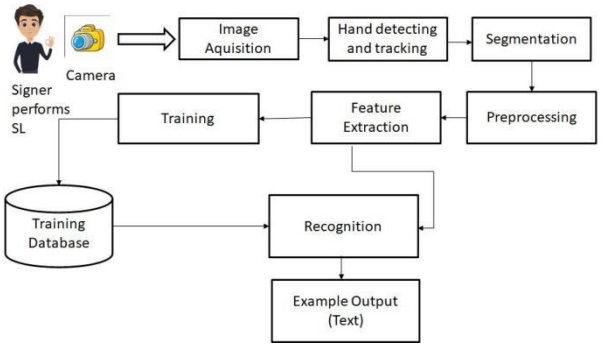


Figure 2. Proposed Methodology.

text subtracting, skin-color detection, and edge detection are all used in the segmentation process. The motion and location of the hand must be detected and segmented in order to recognise gestures.fig:threshold

3.3 Features Extraction:

Predefined characteristics like form, contour, and geometrical characteristics like position, angle, and distance, Color feature, histogram, and other features are taken from the images that have been preprocessed and used later to classify or recognize signs. Highlight extraction is a stage in the dimensionality reduc-tion process that partitions and coordinates a huge assortment of crude data.reduced to more modest, simpler to-oversee classes thus, handling would be easier. The most significant feature is the large number of variables in these massive data sets. A significant amount of computational power is required to process these variables. By selecting and combining variables into functions, function extraction helps to extract the best feature from large data sets, which in turn reduces the size of the data. These features are easy to use while still accurately and uniquely describing the actual data collection.

3.4 Preprocessing:

Each photo placement is preprocessed to dispense with clamor us-ing various channels including disintegration, enlargement, and Gaussian smoothing, among others. When an image is converted from color to grayscale, its size decreases. A typical strategy for diminishing how much information to be handled is to switch a picture over completely to dark scale. The following are the stages of preprocessing:

3.4.1 Morphological Transformation (Morphological Transformation):

A structuring feature on an input image is used in morphological operations to produce an output image of a similar size. To determine the value of each pixel in the output image, it compares the corresponding pixel in the input image to its neighbors. Erosion and dilation are two distinct types of morphological transformations.

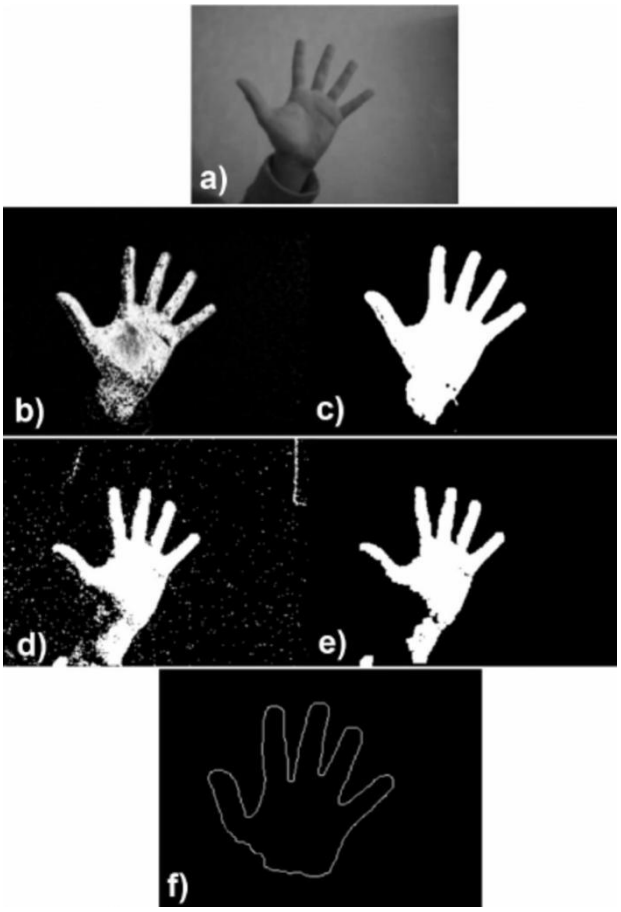


Figure 3. (a) the original hand-drawn image; (b) Hand image after skin color was detected; (c) after morphological tasks and bi-narization; (d) a hand image following background extraction; e) following morphological and binarization processes; f) The hand composed of the concatenation i of images c and e. Dilation: The most extreme worth of all pixels in the area is the worth of the result pixel. If all of a pixel's neighbors in a binary image have the value 1, the pixel is set to 1 Morphological dilation makes artifacts more visible and fills in small gaps.

Erosion: The value of the o/p pixel is the lowest of all the nearby pixels. When all of a pixel's neighbors in a binary image have the value 0, the pixel is set to zero. Morphological erosion removes small artifacts and leaves behind larger objects.

3.4.2Blurring:One example of blurring an image is adding a low-pass filter. In computer vision, the process of removing noise from an image while keeping the rest of the image intact is referred to as a "low-pass filter." Prior to completing other tasks like edge detection, a blur is a straightforward operation.

3.4.3Thresholding:

A type of image segmentation called "thresholding" involves changing the image's pixels to make it easier to inter-pret the image. Thresholding is the process of converting a colour or grayscale image into a binary image, which is simply black and white. We most commonly use thresh-olding to pick areas of interest in a picture while ignoring the sections we are not concerned with.

Recognition:

We'll use classifiers in this case. Classifiers are the meth- ods or algorithms that are used to interpret the signals. Popular classifiers that identify or understand sign lan- guage include the Hidden Markov Model (HMM), K- Nearest Neighbor classifiers, Support Vector Machine (SVM), Artificial Neural Network (ANN), and Principle Component Analysis (PCA), among others. However, in this project, the classifier will be CNN. Because of its high precision, CNNs are used for image classification and recognition. The CNN uses a hierarchical model that builds a network, similar to a funnel, and then outputs a fully-connected layer in which all neurons are connected to each other and the output is processed.

3.4.1 Text output:

Understanding human behaviour and identifying various postures and body movements, as well as translating theminto text.

4 Proposed Algorithm

4.1 Creating the sign language recognitiondataset:

(a) the original hand-drawn image; (b) Hand image after skin color was detected; (c) after morphological tasks and bi-narization; (d) a hand image following background extraction; e) following morphological and binarization processes; f) The hand composed of the concatenation i of images c and e. Dilation: Any frame that detects a hand within the ROI (region of interest) generated can be transferred to a directory that contains two directories, train and look, each containing ten folders containing images captured using the produce gesture knowledge.py perform. The maximum value of all of the pixels in the frame can be transferred to this directory. Now that we have the live camera feed, we will use OpenCV to create

an ROI, which is the only part of the frame where we want to find the hand for the gestures, in order to create the dataset. In order to distinguish the foreground from the background, we must first compute the background's cumulative weighted average and then deduce this value from frames containing an object ahead of the background. By combining the cumulative weight for particular frames and the cumulative average for the context, this can be accomplished. After calculating the background's average, we typically subtract it from each frame we read after sixty frames in order to locate any objects that obscure the background.

4.2 Calculate threshold value:

Using cv.findContours, we now measure the threshold value for each frame and evaluate the contours. Using a function segment, the object's maximum contours, also known as its outermost contours, are returned. Using the contours, we can determine whether a hand or some other foreground object has been identified within the ROI. The ROI image is saved in the train and test sets for the letter or number we're looking for (or a hand is present in the ROI) when the model detects contours. The ROI's thresholded image is displayed in the following window, and the dataset for 1 is generated in the previous example. For each number to be detected, we save 600 images in the train dataset and generate 80 images in the test dataset.

4.3 CNN Layer:

To group the static pictures in our first dataset, we utilized a Convolutional Brain Organization, or CNN, model. Our primary goal when developing a neural network was to define the input layer. In a 28x28 image, each of the 784 pixels has a grayscale value that ranges from 0 (black) to 1 (white). By converting each image into a number sequence, we transform the data into a computer-readable format. Once the input layer has been prepared, the hidden layers of the neural network will process it. The figure depicts the architecture of our neural network: blk2 consists of a number of nodes, each of which receives a weighted sum of the 784 input values. This is the first hidden layer. After that, the inputs are processed by a rectified linear unit, or ReLU, activation function. As can be seen in the graph above, the ReLU will produce 0 when the input is negative, but it will not change the input otherwise. The next hidden-den layer of the network will get its inputs from the outputs of the ReLU.

Table 1. CNN Layers

Layer (type)	Output shape	Param#
conv2d(Conv2D)	(None, 62, 62, 32)	896
max_pooling2d (MaxPooling2D)	(None, 31, 31, 32)	0
conv2d_1 (Conv2D)	(None, 31, 31, 64)	18496
max_pooling2d_1 (MaxPooling2D)	(None, 15, 15, 64)	0
conv2d_2(Conv2D)	(None, 13, 13, 128)	73856
max_pooling2d_2 (MaxPooling2D)	(None, 6, 6, 128)	0
flatten (Flatten)	(None, 4608)	0
dense (Dense)	(None, 64)	294976
dense_1 (Dense)	(None, 128)	8320
dropout_1 (Dropout)	(None, 128)	0
dense_2 (Dense)	(None, 128)	16512
dropout_2 (Dropout)	(None, 128)	0
dense_3 (Dense)	(None, 12	1548

4.4 Training CNN:

A CNN is currently being trained on the newly generated data collection. To begin, we will make use of the keras ImageDataGenerator, which enables us to load the train and test set data by

making use of the flow from directory function. The names of the number folders will serve as the class names for the images that are loaded. All of the callbacks, including reduced LR on plateau and early training, are based on the failure of the validation dataset. After each epoch, the validation dataset is used to measure the accuracy and loss. If the validation loss does not decrease, Re- duceLR is used to reduce the model's LR (Learning Rate) to prevent the model from overshooting the loss.

Epoch No.	Loss		Accuracy	
	Val_Loss		Val_Acc	
1	12.71	0.18	0.81	0.8
2	1.44	0.53	0.30	0.96
3	0.84	0.71	0.15	0.99
4	0.56	0.81	0.02	1.0
5	0.38	0.86	0.01	1.0
6	0.30	0.88	0.0059	1.0
7	0.25	0.89	0.00	1.0
8	0.27	0.90	0.00	1.0
9	0.16	0.94	0.00	1.0
10	0.15	0.95	0.00	1.0

minima. If the accuracy of the validation does not improve after a certain number of epochs, we also employ the early stopping algorithm to stop the training. SGD, also known as stochastic gradient descent, and Adam, also known as a combination of Adagrad and RMSProp, are the two optimization algorithms that are utilized. In other words, the weights are altered at every training instance. We found that the model SGD was more accurate. As can be seen, during training, we achieved 100% training accuracy and 81% validation accuracy.

4.5 Predicting the gesture

Just like we did when we made the dataset, we make a bounding box for finding the ROI and measuring the cumulative average. To identify a foreground entity, this is done. Now that we have found the maximum con- tour, we use the ROI's threshold as a test picture to determine whether or not a hand has been identified. Using keras.models.load_Model, we load the model that has already been saved and then provide the threshold image of the ROI containing the hand to the model as an input for prediction.

Result:

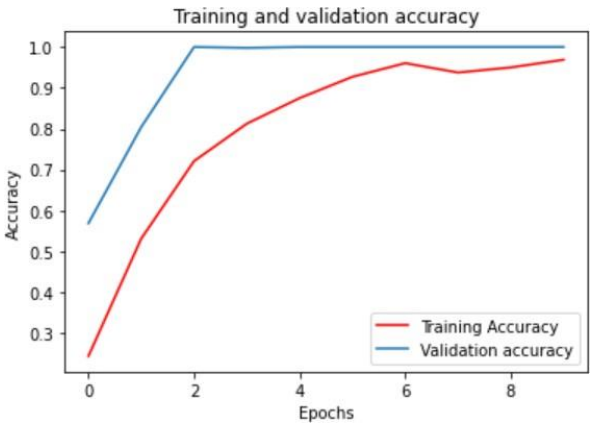


Figure 4. Outputs from the first hidden layer.

1) Accuracy and loss in the model for validation data may vary depending on the situation when we train the model. In normal circumstances, exactness should rise and loss should decrease with each passing epoch. Yet, with approval loss(keras approval misfortune) and validationaccuracy, various cases can be possible like under-neath:

2) 1) The loss of validation begins to rise, and the accuracy of validation begins to fall. This suggests that the model is not learning but rather cramming.

3) 2) As validation loss begins to rise, validation

accuracy will also rise. When soft-max is used in the output layer, this could be an instance of overfitting or different probability values.

4) Validation loss begins to decrease, and validation accuracy begins to rise. This is also fine because it indicates that the manufactured model is learning and coping appropriately. We have plotted the graph of accuracy and loss with respect to epochs after testing our model, and these are the results..

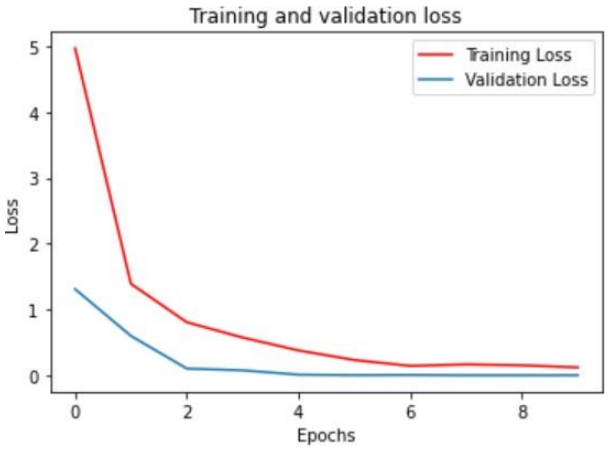


Figure 5. Training and validation loss

Fig 6 shows overall accuracy evolution of model. In whichit has been seen that validation loss is decreasing and vali-dation accuracy is increasing noticeably.

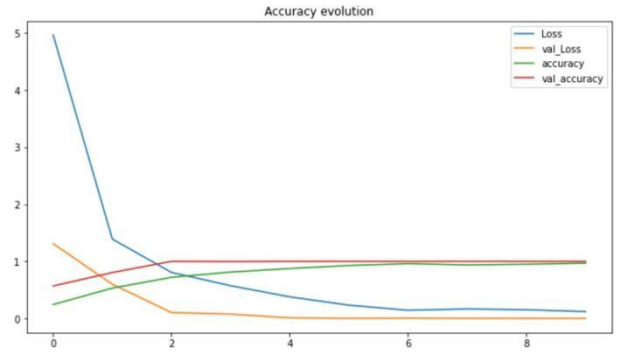


Figure 6. Accuracy Evolution

5 Conclusion and Future Work:

The Gesture based communication Acknowledgment (SLR) framework is a strategy for perceiving an assortment of shaped signs and translat-ing them into text or discourse with the proper setting. The development of productive interactions between humans and machines demonstrates the significance of gesture recognition. In this project, we tried to use a Convolutional Neural Network to build a model. This outcomes in an approval accu-scandalous of around 95%

The Picture Handling segment of future work ought to be en-hanced with the goal that the framework can connect in the two headings,

for example it ought to be equipped for making an interpretation of ordinary language to gesture based communication as well as the other way around.

References

[1] "Research on the Hand Gesture Recognition Based on Deep Learning," presented on February 7, 2019, by Jing-Hao Sun, Ting-Ting Ji, Shu-Bin Zhang, Jia-Kui Yang, and Guang-Rong Ji.

[2] Mokhtar M. Hasan, Pramoud K. Misra. Brightness Factor Matching Using Scaled Normalization for a Gesture Recognition System," International Journal of Computer Science and Information Technology (IJCSIT), Vol. 3(2).

[3] Marcus V. Lamar, Susumu Kuroyanagi, Akira Iwata, and Simei G. Wysoski. Static-Gesture Recognition With Boundary Histograms and Neural Networks: A Rotation-Invariant Approach International Journal of Artificial Intelligence Applications (IJAIA), Vol. 3, No. 4, July 2012

[4] Stergiopoulou, N. Papamarkos. 2009). “ Hand ges-ture acknowledgment utilizing a brain network shape fitting method," Elsevier Designing Utilizations of Ar-tificial Knowledge, vol. 22(8), pp. 1141– 1158

[5]. V. S. Kulkarni and S.D. Lokhande, "Appearance Based Recognition of American Sign Language Using Gesture Segmentation," International Journal on Computer Science and Engineering (IJCSE), Vol. 2(3), pp. 560-565.

[6] "Real Time Sign Language Interpreter" by Geethu G Nath and Arun C S was presented at the 2017 International Conference on Electrical, Instrumentation, and Communication Engineering (ICEICE2017).

[7] "Continuous Indian Sign Language Gesture Recognition and Sentence Formation" by Kumud Tripathi, Neha Baranwal, and G. C. Nandi was presented at the Eleventh International MultiConference on Information Processing-2015 (IMCIP-2015), Procedia Computer 45, NO. 4, September 2015

[9] "Relevant features for video-based continuous sign language recognition," by B. Bauer and H. Hienz, presented at the 2002 IEEE International Conference on Automatic Face and Gesture Recognition.

[10] Pigou, L., S. Dieleman, P.-J. Kindermans, and B. Schrauwen Making use of Convolutional Neural Networks, Sign Language Recognition