Intelligent Automation & Soft Computing

Tech Science Press

DOI: 10.32604/iasc.2023.035497
*Article*

# Alphabet-Level Indian Sign Language Translation to Text Using Hybrid-AO Thresholding with CNN

**Seema Sabharwal[1,2,*] and Priti Singla[1]**

[1]Department of Computer Science and Engineering, Baba Mastnath University, Rohtak, 124001, India
[2]Department of Computer Science, Government Post Graduate College for Women, Panchkula, 134109, India
*Corresponding Author: Seema Sabharwal. Email: sabharwalseema@gmail.com
Received: 23 August 2022; Accepted: 13 January 2023; Published: 11 September 2023

**Abstract:** Sign language is used as a communication medium in the field of trade, defence, and in deaf-mute communities worldwide. Over the last few decades, research in the domain of translation of sign language has grown and become more challenging. This necessitates the development of a Sign Language Translation System (SLTS) to provide effective communication in different research domains. In this paper, novel Hybrid Adaptive Gaussian Thresholding with Otsu Algorithm (Hybrid-AO) for image segmentation is proposed for the translation of alphabet-level Indian Sign Language (ISLTS) with a 5-layer Convolution Neural Network (CNN). The focus of this paper is to analyze various image segmentation (Canny Edge Detection, Simple Thresholding, and Hybrid-AO), pooling approaches (Max, Average, and Global Average Pooling), and activation functions (ReLU, Leaky ReLU, and ELU). 5-layer CNN with Max pooling, Leaky ReLU activation function, and Hybrid-AO (5MXLR-HAO) have outperformed other frameworks. An open-access dataset of ISL alphabets with approx. 31 K images of 26 classes have been used to train and test the model. The proposed framework has been developed for translating alphabet-level Indian Sign Language into text. The proposed framework attains 98.95% training accuracy, 98.05% validation accuracy, and 0.0721 training loss and 0.1021 validation loss and the performance of the proposed system outperforms other existing systems.

**Keywords:** Sign language translation; CNN; thresholding; Indian sign language

## 1 Introduction

In today's era, specially-abled people are critiqued on the scale of their impairment. Deaf people have difficulty integrating into society due to their communication challenges. Sign language is used as a means of communication by deaf people around the globe apart from its application in various other fields such as defence, trade, sea-diving, etc. Communication with deaf people is hindered majorly by non-deaf people's inability to understand sign language gestures. The syntax and semantics of sign languages vary from one region to another. There are numerous sign languages used across the

world for approximately 70 million deaf people, for instance, Indian Sign Language (ISL), American Sign Language (ASL), Chinese Sign Language (CSL), Japanese Sign Language (JSL), Italian Sign Language, Malaysian Sign Language, etc. ISL is used to communicate with deaf persons in India in addition to other regional sign languages. The building blocks of ISL are 26 alphabets (A–Z), 10 numbers (0–9), and 10,000 words [1].

Due to a huge gap in the vocabulary of the English language and the Indian Sign language, the fingerspelling approach came into existence. This method creates each locution letter by letter via distinct gestures for sign language letters. The gestures of ISL can be classified based on the number of hands, movement of hands, features taken into consideration, and the way they are recognized (Static/Dynamic) [2]. As a consequence of technological innovation, there is a need to craft a sign language processing framework to facilitate deaf people. Currently, three types of systems are available for processing Sign language. Firstly, Sign Language Generation System (SLGS), decodes given text into apt gestures of sign language. Secondly, Sign Language Validation System (SLVS), verifies the legitimacy of a given sign language pose. Finally, Sign Language Translation System (SLTS), construes given sign language pose to typescript. SLTS employs two approaches -Hardware-based SLTS uses gloves (smart glove, data glove, and cyber glove), and sensors (LMC, ACC, EMG) [3].

On the other hand, vision-based SLTS employs a camera and various techniques such as machine learning and deep learning to recognize and construes sign language gestures [4]. The output can be either audio or text (alphabet, word, or sentence). In deep learning, CNN has outperformed several machine learning algorithms in sign language translation [5,6]. It has become a popular choice of researchers because of its architecture, and optimal performance in image classification tasks [7]. Image pre-processing is also one of the important steps in SLTS along with the CNN. The challenging part of an efficient SLTS is to have the finest feature extraction method and an efficient translation mechanism. In recent years, researchers have either focus on premium feature extraction techniques through image segmentation/thresholding or fine-tuning architectural parameters of CNN to have optimal accuracy. This paper aims to explore the role of image thresholding, pooling, and activation functions in an efficient CNN-based sign language translation framework. It focuses on an automatic sign language translation framework for deaf-mute persons. The following are key contributions of this work:

- A novel Hybrid-AO thresholding (based on Adaptive Gaussian thresholding and Otsu Algorithm) approach for segmentation is used on the open access dataset of ISL.
- A 5MXLR-HAO framework is proposed based on Hybrid-AO thresholding and deep learning. It is chosen by comparison among 9 models based on three variations of pooling (Max, Average, Global Average Pooling) and activation functions (ReLU, Leaky ReLU, and ELU).
- Evaluate the performance of the proposed 5MXLR-HAO using Training Accuracy and Validation accuracy metrics.

The paper is organized as follows: Section 2 discusses the related works. Section 3 explains the methodology followed by experimental results and a discussion in Section 4. Finally, the conclusion and future work are presented in Section 5.

## 2 Related Work

Gesture recognition and sign language translation have been well-studied topics in American Sign Language (ASL), but few studies on Indian Sign Language (ISL) have been published. In contrast to

ASL, two-handed signs in ISL have little duality, making them difficult to discern. A deep CNN-based SLRS was developed by [2] to identify the gestures of ISL using a model comprising 8 layers with Stochastic pooling, DiffGrad optimizer, and SoftMax as a classifier. Using transfer learning, a Bengali Sign Language recognition system has been proposed by [8] to classify 3 sets of 37 different symbols to attain a validation accuracy of 84.68%. Alphabet-based Arabic SLTS was developed by [9] which gives a speech as output with an accuracy of 90% employing CNN. Reference [10] proposed an American Sign Language Recognition system using Otsu thresholding for image segmentation and 2 channelled CNN on a self-made dataset with a recognition accuracy of 96.59%.

Reference [11] proposed SLTM for Arabic Sign Language using two datasets using the Synthetic Minority oversampling technique (SMOTE) to attain a test accuracy of 97.29%. The impact of skewed data on system accuracy is also investigated. Seven convolution layers with max pooling, ReLU activation, Batch normalisation and dropout layers are used for classification. Reference [12] proposed a real-time two-way communication system for ISL using the Canny Edge detection technique and CNN to convert sign language gestures to voice and vice versa. Reference [13] developed a hybrid framework called FiST_CNN as a combination of Fast Accelerated Segment Test (FAST), Scale Invariant Feature Transformation (SIFT), and CNN for the recognition of one-hand static ISL alphabets with an accuracy of 95.87%. Reference [14] proposed a model for multilabel classification using wearable sensors sEMG and IMU signals for the recognition of ISL.

Reference [15] proposed real-time translation of ISL using adaptive thresholding and hybridised SIFT with an accuracy of 92.78%. Reference [16] proposed a dynamic model for the recognition of alphanumeric gestures of ISL using Speeded Up Robust Features (SURF) with Support Vector Machine (SVM) and CNN. Canny Edge detection and Gaussian filtering are used for the pre-processing of images. Six Convolution layered CNN is used along with max Pooling, dropout, RELU activation function, and softmax as a classifier. The system uses a self-made dataset of 36000 images of 36 classes from 3 signers. Reference [17] proposed a model for the translation of static ASL gestures of alphanumeric data using fine-tuned CNN. The model comprises 8 layers with max max-pooling, dropout, ReLU activation and softmax as the classifier. Reference [18] used fifth Dimension Technology (5DT) gloves for data acquisition and k-Nearest Neighbour (k-NN) machine learning algorithm for Australian SLR with an accuracy of 97%. A real-time SLTS has been developed to translate the alphabet of Arabic Sign Language using AlexNet architecture [19]. A hybrid sign language recognition model based on CNN and LSTM was proposed by [20] to aid COVID-19 patients. The model was tested on two datasets and obtained a maximum average accuracy of 99.34%. Reference [21] reviewed 72 sensor-based sign language recognition systems and concluded that although they achieve accuracy up to 99.75%, these should be more user-friendly and have minimal circuitry without drawing attention. Further, a sensor-based system works better in experimental setup than in real-world situations. Aparna et al. [22] suggested CNN and stacked long short-term memory (LSTM) models for the recognition of isolated words of ISL. The model achieved a training accuracy of 94% on a self-made video dataset of 6 isolated words. For continuous word recognition in sign language, the different variations of the transformer have been used extensively by researchers to exploit spatial-temporal features [23,24]. Few works have also been done to explore the role of contextual information in sign language recognition with Generative adversarial network (GAN) [25].

The cost and obligation to wear hardware-based SLTS outweigh their accuracy. So, instead of using high-end equipment, we intend to overcome this challenge using cutting-edge computer vision and machine learning methods. It has been analysed that major work of alphabet-level SLTS has been performed in deep learning, which is a reflection of the machine learning (ML) paradigm and expedites learning by simulating human brain activity. Further, computer vision technology has become widely

popular in Sign Language processing as a result of innovations in deep learning. It incorporates the use of numerous layers of neural networks for complicated processing. This study aims to identify alphabets in Indian Sign Language based on the input images of gestures using deep learning. In this research, we have introduced novel hybrid-AO thresholding to translate the alphabets of ISL using state of art CNN framework.

## 3 Methodology

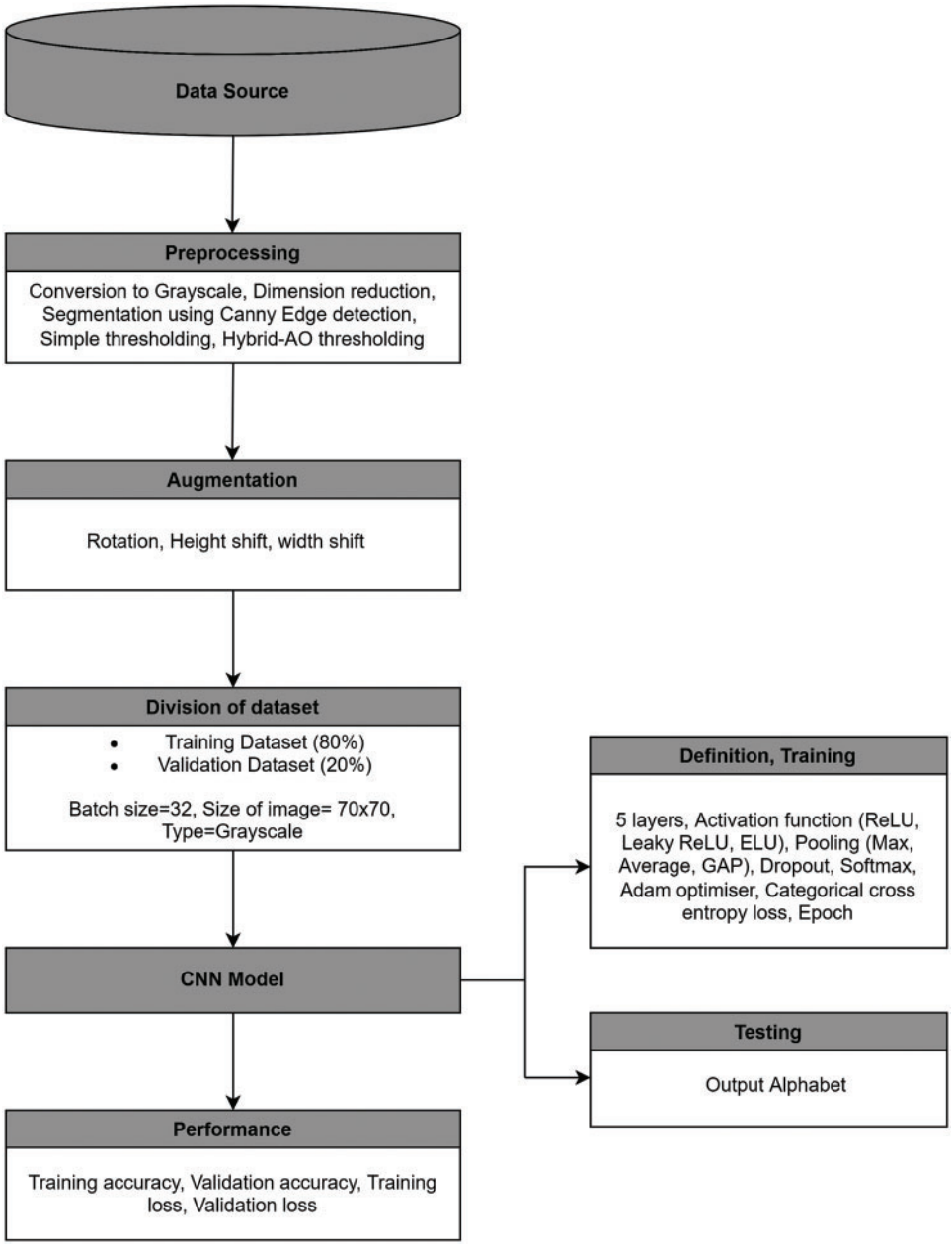


**Figure 1:** Architecture of the proposed ISLTS

The proposed model for the translation of the ISL alphabet to text is rendered using Fig. 1. The SLTS can be divided into the following steps:

**Step 1**-Alphabet level dataset of 26 classes has been taken from open access data source Kaggle [26].

**Step 2**-Data pre-processing has been performed by changing the coloured image to grayscale, dimension has been reduced to 70 × 70. A novel image segmentation called Hybrid-AO thresholding (Section 3.2.3) has been used by applying Adaptive Gaussian thresholding followed by Otsu Algorithm. Canny edge detection algorithm and Simple thresholding have also been performed on the input dataset to check the performance of Hybrid-AO thresholding.

**Step 3**-Three augmentation operations (Rotation, Height shift and Width shift) have been applied to segmented images to increase the number of images and reduce overfitting leading to a total of approx. 31 K grayscale samples.

**Step 4**-Dataset has been further divided into 80:20 for the training and testing phase of the CNN model of deep learning. The value of batch size has been taken as 32.

**Step 5**-CNN model has been applied to the pre-processed images for classification. Nine CNN models have been proposed based on three types of pooling and activation functions. The value of different parameters such as the number of layers, dropout, epoch, number of filters, size, stride, etc., has been defined.

**Step 6**-The performance of these models has been compared using training and validation accuracy.

### *3.1 Dataset*

An open-access alphanumeric dataset for ISL is taken from Kaggle [26] as a data source. The dataset contains 31200 RGB images of 26 classes of alphabets from A to Z numbered from 0 to 25. Each class has 1200 images with 128 × 128 size. Fig. 2 displays the images of the alphabet used for translation in our proposed framework.

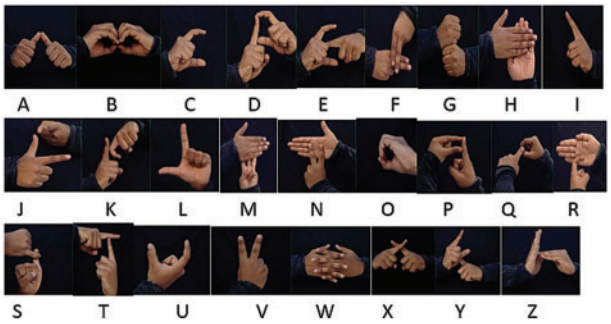


**Figure 2:** Alphabets of ISL dataset

### *3.2 Image Segmentation*

The first and most important step in building SLTS is data pre-processing in which raw input data is prepared for the model. Initially, all the images are labelled and sorted into respective alphabet categories. To facilitate processing, the dimensions of each image in the dataset are reduced to 70 × 70

size and transformed to grayscale. Further, a gaussian filter is used to reduce the noise and smoothen the image.

Hand gesture recognition is an important phase of SLTS, so the foreground of the image must be differentiated from the background. Image segmentation is the mechanism of divvying a digital image into different regions for efficient analysis. The following image segmentation techniques are employed in this research article.

### *3.2.1 Canny Edge Detection Algorithm*

It is a technique of image segmentation used for edge detection in images [27]. In this, the first Gaussian filter G (x, y) is applied to grayscale image f (x, y) for smoothness and removing noise using Eq. (1), where $\sigma$ is the space scale coefficient or standard deviation specifying the amount of smoothing.

$$G(x,y) = \frac{1}{2\pi\sigma^2}\exp\left(-\frac{x^2+y^2}{2\pi\sigma^2}\right) \tag{1}$$

Secondly, gradient magnitude G (x, y) and gradient direction Θ (i, j) is calculated using Eqs. (2) and (3) respectively where $G_x$ and $G_y$ are gradient in two classes.

$$G(x,y) = \sqrt{G_x^{\,2}(i,j) + G_y^{\,2}(i,j)} \tag{2}$$

$$\theta(i,j) = acrctan\left[\frac{G_y(i,j)}{G_x(i,j)}\right] \tag{3}$$

Thirdly, the algorithm aims to look for pixels of maximum edge direction value by traversing all the values of the gradient intensity matrix. This step is called Non-maximum Suppression and the output is binary images with thin edges.

Finally, double thresholding considers only pixels with a high and low threshold value, otherwise, they are discarded leading to final edges. This step is called Hysteresis Thresholding.

### *3.2.2 Simple Thresholding*

Thresholding is another technique of image segmentation that segregates contrasting regions by comparing pixel intensity. Binary Inverse Thresholding is applied to calculate the value of threshold T on a given input image I (x, y) using Eq. (4).

$$I_d(x,y) = \begin{cases} 0, & I_s(x,y) < T \\ max, & otherwise \end{cases} \tag{4}$$

where $I_s$ (x, y) is the source image, $I_d$ (x, y) is the destination image and T is the threshold value.

### *3.2.3 Hybrid-AO Thresholding*

The third technique is the hybrid-AO thresholding method of image segmentation based on a combination of Adaptive Gaussian thresholding and the Otsu algorithm is used. In Adaptive Gaussian Thresholding, local areas of the image are evaluated analytically to determine the optimal threshold value T using Eq. (5).

$$T = mean(I_L - C) \tag{5}$$

where $I_L$ is the local subarea of the image, and C is the constant value to optimize threshold T. There are two ways to calculate mean-Arithmetic and Gaussian Mean. Since we have used the Gaussian mean, the method is used to calculate threshold value in Adaptive thresholding, hence it is called Adaptive Gaussian Thresholding. The weighted sum of neighbouring values in a Gaussian window is the threshold value. The images in the dataset consist of a black background, so Binary Inverse Adaptive Thresholding is used.

Otsu's algorithm automatically evaluates an optimum threshold based on pixel values' observed distribution [28]. It is used to separate the foreground from a background of an image by finding the global thresholding value. It maximizes between class variance values; for this, image f (x, y) is scanned for all the possible values of the threshold. f (x, y) is a grayscale image having threshold value (t) from $(0 \leq t \leq L-1)$.

$\omega_f$ (t), $\omega_b$ (t) are the probabilities of two classes divided by threshold t denoting foreground and background, the value of threshold ranges from 0 to 255 given by Eqs. (6) and (7).

$$\omega_f(t) = \sum_{i=0}^{t-1} p(i) \tag{6}$$

$$\omega_b(t) = \sum_{i=t}^{L-1} p(i) \tag{7}$$

$$p_i = \frac{n_i}{n} \tag{8}$$

where p is the probability of gray levels, t is a threshold, L is bins of histogram and $n_i$ is the number of pixels in the gray level and n is several pixels in the overall image. $\sigma_f$, $\sigma_b$ is the variance of background and foreground and $\sigma$ is the total variance in threshold t, given by Eq. (9).

$$\sigma_b^2 t = \sigma^2 - \sigma_f^2 t = \omega_f(t)\,\omega_b(t)\left[\mu_f(t) - \mu_b(t)\right]^2 \tag{9}$$

Foreground and background Pixel intensity values for two classes $C_1$ (0 to t−1) and $C_2$ (t to L−1) are given by Eqs. (10) and (11).

$$\mu_f = \sum_{i=0}^{t-1} \frac{ip_i}{\omega_f(t)} \tag{10}$$

$$\mu_b = \sum_{i=t}^{L-1} \frac{ip_i}{\omega_b(t)} \tag{11}$$

From the above equations, we can say that sum of probabilities of foreground and background $\omega_f$, $\omega_b$ will be 1 given by Eq. (12).

$$\omega_f + \omega_b = 1 \tag{12}$$

From Eq. (12), we can have Eq. (13).

$$\omega_f \mu_f + \omega_b\, \mu_b = \mu_T \tag{13}$$

where $\mu_T$ which is the average gray mean of the entire image can be given by Eq. (14).

$$\mu_T = \sum_{i=0}^{L-1} ip_i \tag{14}$$

There is the variance of foreground and background class defined by $\sigma_f^2, \sigma_b^2$ in Eqs. (15) and (16).

$$\sigma_f^2 = \sum_{i=0}^{t} \frac{\left(i - \mu_f\right)^2 p_i}{\omega_f} \tag{15}$$

$$\sigma_b^2 = \sum_{i=t+1}^{L-1} \frac{(i - \mu_b)^2 p_i}{\omega_b} \tag{16}$$

There are three types of variances defined for these classes i.e., total variance $\sigma_T^2$ given by Eq. (17), the variance within the class $\sigma_w^2$ given by Eq. (18), and variance between the class $\sigma_{bet}^2$ given by Eq. (19).

$$\sigma_T^2 = \sum_{i=0}^{L-1} (i - \mu_T)^2 p_i \tag{17}$$

$$\sigma_w^2 = \omega_f \sigma_f^2 + \omega_b \sigma_b^2 \tag{18}$$

$$\sigma_{bet}^2 = \omega_f \left(\mu_f - \mu_T\right)^2 + \omega_b \left(\mu_b - \mu_T\right)^2 \tag{19}$$

From Eqs. (17)–(19) we can infer Eq. (20).

$$\sigma_T^2 = \sigma_{bet}^2 + \sigma_w^2 \tag{20}$$

Optimal thresholding after performing discriminant analysis on the gray level can be given using Eq. (21).

$$t = arg_{0 \le t \le L-1} max \left\{\sigma_{otsu}^2 (t)\right\} \tag{21}$$

where $\sigma_{otsu}^2 (t)$ can be given finally using Eq. (22).

$$\sigma_{otsu}^2 (t) = \frac{\left[\mu_T \omega_f (t) - \mu_f\right]^2}{\omega_f (t) \left[1 - \omega_f (t)\right]} \tag{22}$$

Data augmentation includes a set of operations to increase data in the dataset. Rotation, height shift, and width shift operations are performed in data augmentation leading to a total of more than 31 K samples. The dataset is divided into two classes training and validation in the ratio of 80:20. Segmented grayscale images of 70 × 70 size are grouped into a batch size of 32.

### *3.3 Convolution Neural Network*

CNN is popular for its performance in image classification and therefore has a vast span of applications in various fields [29] such as emotion detection [30], the medical field, the agriculture field [2], Natural Language Translation [31–33] and sign language translation. The CNN model is a feedforward artificial neural network used in image recognition tasks because of its optimal performance. In this, connectivity between neurons is similar to the organization of the human visual cortex. A 5-layer CNN model is used for the translation of pre-processed and segmented ISL datasets.

The model consists of three Convolution layers in which convolution operation is performed in 2-dimension using Eq. (23).

$$S (i,j) = (I * K) (i,j) = \sum_m \sum_n I (m,n) K (i - m, j - n) \tag{23}$$

$$Z_i = \sum_j W_{i,j} X_j + B_i \tag{24}$$

$$Y = softmax (Z) \tag{25}$$

$$softmax\,(Z_i) = \frac{\exp\,(Z_i)}{\sum_j \exp\,(Z_j)} \tag{26}$$

where S is the feature map; I is the input image; K is the kernel; m, n are dimensions; i, j are variables;

In Eq. (24), Z is the output of the neuron; W is weight; X is input; Y is output. SoftMax is the activation function used to classify multiple classes in the proposed CNN model, given by Eqs. (25) and (26).

Window-size learnable filters make up the convolution layer. The stride size is taken to be 1. We use a small window size [length $5 * 5$] for the convolution layer that spans the depth of the input matrix. The layer is made up of window-sized learnable filters. We moved the window by a certain amount (usually one stride size) during each iteration to compute the dot product of the filter entries and the input values at a specific location. As we proceed, a 2-Dimensional activation matrix will be created, which will respond to that matrix at every spatial place. In other words, the network will learn filters that turn on when it encounters certain visual features, such as an edge of a certain orientation or a splotch of a certain colour.

The pooling layer is used to reduce the computations and decrease the size of the activation matrix. In Max Pooling, the maximum value of activation is chosen out of window size using the formula [15] shown in Eq. (27).

$$s_j = \max_{i \in R_j} a_i \tag{27}$$

where $S_j$ output function for max pooling and $R_j$ is the set of all activation functions $a_i$.

In Average Pooling average value of activation is chosen out of window size using the formula shown in Eq. (28).

$$s_j = \frac{1}{|R_J|} \sum_{i \in R_j} a_i \tag{28}$$

In Global Average (Max) pooling, the GAP layer is used at the outer connected layer instead of the flattening layer and max pooling is used in the interior layers.

A comparison of three activation functions Rectified Linear Unit (ReLU), Leaky ReLU, and Exponential Linear Unit (ELU) is performed with the above-mentioned types of pooling.

A summary of the proposed work is shown in Fig. 3. It depicts 3 convolution layers of 32 kernels of window size $5 * 5$ with pooling (Max, Global average and average), dropout for regularisation, activation function (ReLU, Leaky ReLU and ELU) and SoftMax activation function as the classifier. The first phase of the model consists of 3 convolution layers with a first layer containing 32 kernels of $3 \times 3$ size and stride value as 1. There is a single hidden dense layer with 128 neurons in our proposed model. In addition to this, early stopping with patience is used to deal with the problem of overfitting. A dropout layer with a probability of 0.25 has been used. Fig. 3a displays the summary of the proposed CNN model with average pooling and further three combinations of the activation function. 4096 neurons have been used in the first input layer of the Artificial Neural Network (ANN). Flatten layer has been used to convert the feature map into a single dimension. Similarly, in Fig. 3b, Global average pooling in the outer layer and max pooling has been used in the inner layers. Instead of flattening the layer, global average pooling has been used for a better representation of the output of the convolution layer. Fig. 3c display the max pooling with three combinations of the activation function. In this, the

first convolution layer has 32 filters, the second and third layer has 64 filters. There are 4096 nodes in the input layer of ANN. There are 26 nodes in the final output layer to classify the alphabets of ISL.

| Layer (type) | Output Shape | Param # |
|---|---|---|
| conv2d_12 (Conv2D) | (None, 68, 68, 32) | 320 |
| average_pooling2d_6 (AveragePooling2D) | (None, 34, 34, 32) | 0 |
| dropout_12 (Dropout) | (None, 34, 34, 32) | 0 |
| conv2d_13 (Conv2D) | (None, 34, 34, 64) | 18496 |
| average_pooling2d_7 (AveragePooling2D) | (None, 17, 17, 64) | 0 |
| dropout_13 (Dropout) | (None, 17, 17, 64) | 0 |
| conv2d_14 (Conv2D) | (None, 17, 17, 64) | 36928 |
| average_pooling2d_8 (AveragePooling2D) | (None, 8, 8, 64) | 0 |
| dropout_14 (Dropout) | (None, 8, 8, 64) | 0 |
| flatten_4 (Flatten) | (None, 4096) | 0 |
| dense_8 (Dense) | (None, 128) | 524416 |
| dense_9 (Dense) | (None, 26) | 3354 |

Total params: 583,514
Trainable params: 583,514
Non-trainable params: 0

(a)

| Layer (type) | Output Shape | Param # |
|---|---|---|
| conv2d_27 (Conv2D) | (None, 68, 68, 32) | 320 |
| max_pooling2d_18 (MaxPooling2D) | (None, 34, 34, 32) | 0 |
| dropout_27 (Dropout) | (None, 34, 34, 32) | 0 |
| conv2d_28 (Conv2D) | (None, 34, 34, 64) | 18496 |
| max_pooling2d_19 (MaxPooling2D) | (None, 17, 17, 64) | 0 |
| dropout_28 (Dropout) | (None, 17, 17, 64) | 0 |
| conv2d_29 (Conv2D) | (None, 17, 17, 64) | 36928 |
| max_pooling2d_20 (MaxPooling2D) | (None, 8, 8, 64) | 0 |
| dropout_29 (Dropout) | (None, 8, 8, 64) | 0 |
| global_average_pooling2d_2 (GlobalAveragePooling2D) | (None, 64) | 0 |
| dense_18 (Dense) | (None, 128) | 8320 |
| dense_19 (Dense) | (None, 26) | 3354 |

Total params: 67,418
Trainable params: 67,418
Non-trainable params: 0

(b)

| Layer (type) | Output Shape | Param # |
|---|---|---|
| conv2d_15 (Conv2D) | (None, 68, 68, 32) | 320 |
| max_pooling2d_6 (MaxPooling2D) | (None, 34, 34, 32) | 0 |
| dropout_15 (Dropout) | (None, 34, 34, 32) | 0 |
| conv2d_16 (Conv2D) | (None, 34, 34, 64) | 18496 |
| max_pooling2d_7 (MaxPooling2D) | (None, 17, 17, 64) | 0 |
| dropout_16 (Dropout) | (None, 17, 17, 64) | 0 |
| conv2d_17 (Conv2D) | (None, 17, 17, 64) | 36928 |
| max_pooling2d_8 (MaxPooling2D) | (None, 8, 8, 64) | 0 |
| dropout_17 (Dropout) | (None, 8, 8, 64) | 0 |
| flatten_5 (Flatten) | (None, 4096) | 0 |
| dense_10 (Dense) | (None, 128) | 524416 |
| dense_11 (Dense) | (None, 26) | 3354 |

Total params: 583,514
Trainable params: 583,514
Non-trainable params: 0

(c)

**Figure 3:** Summary of proposed CNN model (a) Average Pooling with ReLU, Leaky ReLU, ELU (b) GA (Max) Pooling with ReLU, Leaky ReLU, ELU (c) Max Pooling with ReLU, Leaky ReLU, ELU

## 4 Results and Discussion

Python programming language is used to implement this entire experiment on Google Colab Pro with TensorFlow, Keras, NumPy, Matplotlib, OpenCV python packages, categorical cross entropy and Adam optimiser.

In the image segmentation phase, the output image of proposed Hybrid-AO thresholding is compared with Canny Edge detection, and Binary Inverse thresholding and the same is shown in Fig. 4. It has been observed that Hybrid-AO thresholding produces better results than Canny Edge Detection and Binary Inverse Thresholding due to automatic calculation of threshold value. It uses peripheral pixel values to analyze and process the input image and produces minimal noise segmented output image.

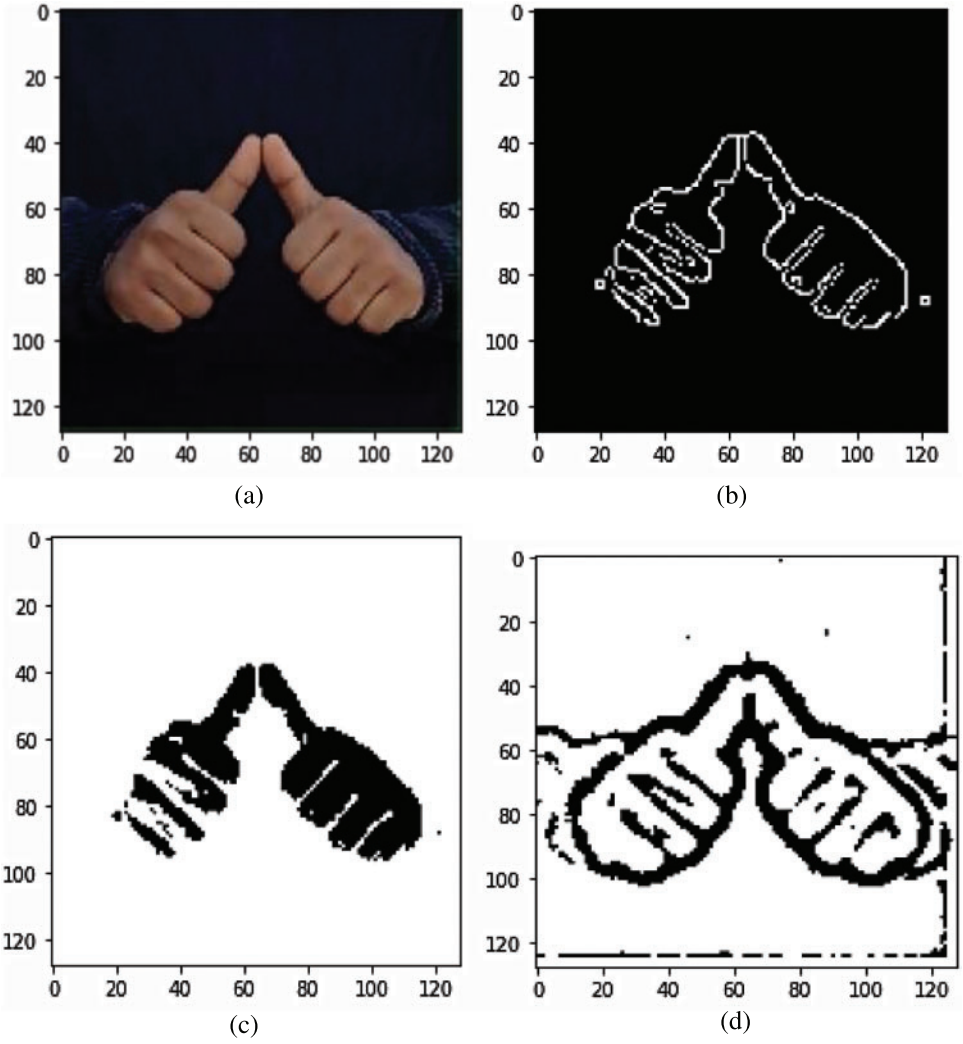


**Figure 4:** Image segmentation results (a) Original RGB image (b) Canny edge detection (c) Simple thresholding (d) Hybrid-AO thresholding

The segmented image is fed to 5 layered CNN sign language translation framework. And to optimise the performance of the proposed framework, three different types of pooling and three

different types of activation functions were considered leading to a total of nine models. The segmentation results obtained from one sign language gesture are shown in Fig. 4d, which verifies the denoising performance of HAO thresholding. In the architectural aspect, 5-layer CNN has been used with leaky ReLU activation function and max pooling. The comparison between three commonly used activation functions and pooling techniques has been performed to obtain the superlative performance of the proposed model.

Table 1 displays the performance metrics of 9 models, i.e., 5AVEL-HAO, 5AVLR-HAO, 5AVRL-HAO, 5GAEL-HAO, 5GALR-HAO, 5GARL-HAO, 5MXEL-HAO, 5MXL-HAO, and 5MXRL-HAO in terms of training accuracy, validation accuracy, training loss and validation loss. The highest and lowest training accuracy has been attained by 5MXLR-HAO and 5GALR-HAO Indian sign language translation frameworks respectively. Similarly, the highest and lowest validation accuracy has been obtained by 5MXLR-HAO and 5GAEL-HAO Indian sign language translation frameworks, respectively.

**Table 1:** Performance metrics of proposed models

| Nomenclature | Pooling | Activation function | Training accuracy | Validation accuracy | Training loss | Validation loss |
|---|---|---|---|---|---|---|
| 5AVEL-HAO | Average | ELU | 92.97 | 92.76 | 0.1911 | 0.2045 |
| 5AVLR-HAO | Average | Leaky ReLU | 94.90 | 92.97 | 0.1152 | 0.1914 |
| 5AVRL-HAO | Average | ReLU | 95.31 | 94.90 | 0.1102 | 0.1189 |
| 5GAEL-HAO | GAP | ELU | 91.78 | 55.47 | 0.2727 | 1.6617 |
| 5GALR-HAO | GAP | Leaky ReLU | 90.62 | 77.34 | 0.2544 | 0.5279 |
| 5GARL-HAO | GAP | ReLU | 92.27 | 90.62 | 0.2358 | 0.4337 |
| 5MXEL-HAO | Max | ELU | 97.66 | 94.90 | 0.0731 | 0.1138 |
| **5MXLR-HAO** | **Max** | **Leaky ReLU** | **98.95** | **98.05** | **0.0725** | **0.1021** |
| 5MXRL-HAO | Max | ReLU | 96.38 | 94.53 | 0.1027 | 0.1490 |

Nine models are proposed based on a comparison between types of pooling and activation functions. The graph of all these nine models is depicted in Fig. 5. The X-axis denotes the number of epochs and the y-axis is used to depict training accuracy, validation accuracy, training loss and validation loss. After 25 epochs, 5MXLR-HAO topped the chart with 98.95% training accuracy and 98.05% validation accuracy and minimum training and validation loss when compared with other proposed ones.

After choosing the best framework for ISLTs, it is important to compare it with existing ones. A comparison of the proposed framework with other existing ones is shown in Table 2. Adaptive thresholding is used in [15] with an accuracy of 92.78% which is lesser than our proposed model. In [34] Modified Canny Edge Detection technique for segmentation is used to achieve a validation accuracy of 95% while [35] accomplishes 90.43% validation accuracy with YOLOv3 with background subtraction and edge detection for translation of Sign language. It has been observed that the accuracy of our proposed framework is better than the three existing systems.

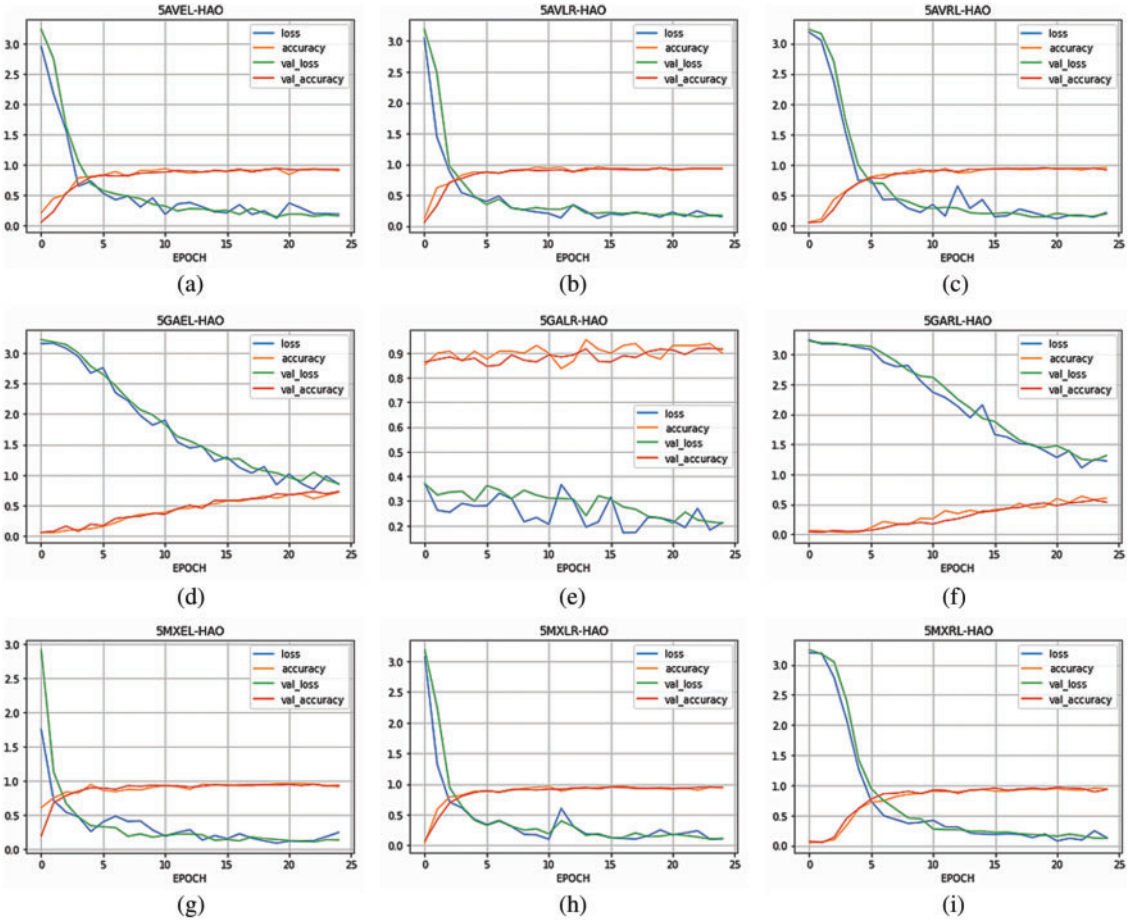


**Figure 5:** Training and validation accuracy and training and validation loss of proposed SLTM (a) 5AVEL-HAO (b) 5AVLR-HAO (c) 5AVRL-HAO (d) 5GAEL-HAO (e) 5GALR-HAO (f) 5GARL-HAO (g) 5MXEL-HAO (h) 5MXLR-HAO (i) 5MXRL-HAO

**Table 2:** Comparison of the proposed ISLTS approach with the existing framework

| Reference | Technique | Validation accuracy (%) |
|---|---|---|
| [15] | Adaptive thresholding | 92.78 |
| [34] | Modified canny edge segmentation | 95 |
| [35] | YOLOv3 with background subtraction and edge Detection | 90.43 |
| **Proposed work** | **5MXLR-HAO** | **98.05** |

In this study, we have presented and applied the proposed 5MXLR-HAO framework for the translation of ISL. The framework has two important phases- firstly segmentation part, and secondly architectural aspect of CNN. In the segmentation part, we have calculated optimal thresholding by applying adaptive Gaussian thresholding, the Otsu algorithm and binary inverse thresholding. Adaptive Gaussian thresholding reduces the noise and increases the sharpness of the image and calculates the regional threshold value. Binary inverse thresholding has been used to separate the background black colour from the foreground gesture. Finally, the Otsu algorithm has been used to calculate the global threshold value of the input image. It has been analyzed that max-pooling gave better results than the average and the global average with max-pooling and leaky ReLU activation function performed better than ReLU and ELU. So, the highest loss in terms of training and validation has been observed in the case of GA pooling and ELU activation functions.

## 5 Conclusion

In this paper, an alphabet-level framework for the translation of Indian Sign Language into text has been proposed based on hybrid-AO thresholding and deep learning. 5-layer CNN framework i.e., 5MXLR-HAO has been selected as the best performing framework among other models based on variations in image segmentation (Canny Edge Detection, Simple thresholding and Hybrid-AO thresholding), pooling (Max, Average, GAP) and activation function (ReLU, Leaky ReLU, ELU). The proposed framework has shown improved performance by attaining a training accuracy of 98.95% and a validation accuracy of 98.05%. Hybrid-AO segmentation can also be applied in various fields such as medical imaging, sign language processing, and other image processing applications. The biggest limitation of our work is the fact that it is tested on a single dataset, so it can be tested on various datasets. Future work would be to contemplate various factors such as variation in the number of layers of CNN, optimization function, and inclusion of non-manual features in the dataset.

**Funding Statement:** The authors received no specific funding for this study.

**Conflicts of Interest:** The authors declare that they have no conflicts of interest to report regarding the present study.

## References

[1] M. Chaitoo, "International Day of Sign Languages," United Nations, 2015. [Online]. Available: https://www.un.org/en/observances/sign-languages-day

[2] U. Nandi, A. Ghorai, M. M. Singh, C. Changdar, S. Bhakta *et al.,* "Indian sign language alphabet recognition system using CNN with diffGrad optimizer and stochastic pooling," *Multimedia Tools and Applications*, pp. 1–22, 2022. https://doi.org/10.1007/s11042-021-11595-48

[3] M. S. Amin, S. T. H. Rizvi and M. M. Hossain, "A comparative review on applications of different sensors for sign language recognition," *Journal of Imaging*, vol. 8, no. 4, pp. 98, 2022.

[4] M. Al-Qurishi, T. Khalid and R. Souissi, "Deep learning for sign language recognition: Current techniques, benchmarks, and open issues," *IEEE Access*, vol. 9, pp. 126917–126951, 2021.

[5] K. Divya Lakshmi and S. R. Balasundaram, "Evaluation of machine learning models for sign language digit recognition," in *Proc. of Int. Conf. on Artificial Intelligence: Advances and Applications*, Jaipur, India, pp. 491–499, 2022.

[6] K. Myagila and H. Kilavo, "A comparative study on performance of SVM and CNN in Tanzania sign language translation using image recognition," *Applied Artificial Intelligence*, vol. 36, no. 1, pp. 2005297, 2022.

[7] R. Rastgoo, K. Kiani and S. Escalera, "Sign language recognition: A deep survey," *Expert Systems with Applications*, vol. 164, pp. 113794, 2021.
[8] M. A. Hossen, A. Govindaiah, S. Sultana and A. Bhuiyan, "Bengali sign language recognition using deep convolutional neural network," in *Proc. of Joint 7th Int. Conf. on Informatics, Electronics & Vision (ICIEV) and 2nd Int. Conf. on Imaging, Vision & Pattern Recognition (icIVPR)*, Kitakyushu, Japan, pp. 369–373, 2018.
[9] M. M. Kamruzzaman, "Arabic sign language recognition and generating arabic speech using convolutional neural network," *Wireless Communications and Mobile Computing*, vol. 2020, pp. 1–9, 2020.
[10] M. A. Rahim, J. Shin and K. S. Yun, "Hand gesture-based sign alphabet recognition and sentence interpretation using a convolutional neural network," *Annals of Emerging Technologies in Computing (AETiC)*, vol. 4, no. 4, pp. 20–27, 2020.
[11] A. A. Alani and G. Cosma, "ArSL-CNN a convolutional neural network for arabic sign language gesture recognition," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 22, no. 2, pp. 1096, 2021.
[12] V. Brahmankar, N. Sharma, S. Agrawal, S. Ansari, P. Borse *et al.,* "Indian sign language recognition using canny edge detection," *International Journal of Advanced Trends in Computer Science & Engineering*, vol. 10, no. 3, pp. 1576–1583, 2021.
[13] A. Tyagi and S. Bansal, "Feature extraction technique for vision-based Indian sign language recognition system: A review," in *Proc. of Computational Methods and Data Engineering, ICMDE 2020*, Singapore, Springer, vol. 1227, pp. 39–53, 2021.
[14] R. Gupta and A. Kumar, "Indian sign language recognition using wearable sensors and multi-label classification," *Computers & Electrical Engineering*, vol. 90, pp. 106898, 2021.
[15] S. Rajarajeswari, N. M. Renji, P. Kumari, M. Keshavamurthy and K. Kruthika, "Real-time translation of Indian sign language to assist the hearing and speech impaired," in *Proc. of Innovations in Computational Intelligence and Computer Vision*, Singapore, Springer, vol. 1424, pp. 303–322, 2022.
[16] S. Katoch, V. Singh and U. S. Tiwary, "Indian sign language recognition system using SURF with SVM and CNN," *Array*, vol. 14, pp. 100141, 2022.
[17] A. Mannan, A. Abbasi, A. R. Javed, A. Ahsan, T. R. Gadekallu *et al.,* "Hypertuned deep convolutional neural network for sign language recognition," *Computational Intelligence and Neuroscience*, vol. 2022, pp. 1–10, 2022.
[18] S. Johnny and S. J. Nirmala, "Sign language translator using machine learning," *SN Computer Science*, vol. 3, no. 1, pp. 36, 2022.
[19] Z. Alsaadi, E. Alshamani, M. Alrehaili, A. A. D. Alrashdi, S. Albelwi *et al.,* "A real-time arabic sign language alphabets (ArSLA) recognition model using deep learning architecture," *Computers*, vol. 11, no. 5, pp. 78, 2022.
[20] A. Venugopalan and R. Reghunadhan, "Applying hybrid deep neural network for the recognition of sign language words used by the deaf COVID-19 patients," *Arabian Journal for Science and Engineering*, pp. 1–14, 2022.
[21] K. Kudrinko, E. Flavin, X. Zhu and Q. Li, "Wearable sensor-based sign language recognition: A comprehensive review," *IEEE Reviews in Biomedical Engineering*, vol. 14, pp. 82–97, 2021.
[22] C. Aparna and M. Geetha, "CNN and stacked LSTM model for Indian sign language recognition," in *Proc. of Symp. on Machine Learning and Metaheuristics Algorithms, and Applications*, Trivandrum, India, vol. 1203, pp. 126–134, 2020.
[23] W. Aditya, T. K. Shih, T. Thaipisuthikul, A. S. Fitriajie, M. Gochoo *et al.,* "Novel spatio-temporal continuous sign language recognition using an attentive multi-feature network," *Sensors*, vol. 22, no. 17, pp. 6452, 2022.
[24] P. Xie, M. Zhao and X. Hu, "PiSLTRc: Position-informed sign language transformer with content-aware convolution," *IEEE Transactions on Multimedia*, vol. 24, pp. 3908–3919, 2022.
[25] I. Papastratis, K. Dimitropoulos and P. Daras, "Continuous sign language recognition through a context-aware generative adversarial network," *Sensors*, vol. 21, no. 7, pp. 2437, 2021.

[26] P. Arikeri, "Indian sign language," 2021. [Online]. Available: https://www.kaggle.com/datasets/prathumarikeri/indian-sign-language-isl
[27] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, 3rd ed., India: Pearson Education, pp. 711–783, 2009.
[28] N. Otsu, "A threshold selection method from gray-level histograms," *IEEE Transactions on Systems Man Cybernetics-Systems*, vol. 9, no. 1, pp. 62–66, 1979.
[29] H. Varun Chand and J. Karthikeyan, "CNN based driver drowsiness detection system using emotion analysis," *Intelligent Automation & Soft Computing*, vol. 31, no. 2, pp. 717–728, 2022.
[30] L. Alzubaidi, J. Zhang, A. J. Humaidi, A. Al-Dujaili, Y. Duan *et al.,* "Review of deep learning: Concepts, CNN architectures, challenges, applications, future directions," *Journal of Big Data*, vol. 8, no. 1, pp. 53–74, 2021.
[31] S. Bawa, "Sanskrit to universal networking language EnConverter system based on deep learning and context-free grammar," *Multimedia Systems*, pp. 1–17, 2020.
[32] S. Bawa, "SANSUNL: A sanskrit to UNL enconverter system," *IETE Journal of Research*, vol. 67, no. 1, pp. 117–128, 2021.
[33] S. Bawa, S. Sitender, M. Kumar and S. Sangeeta, "A comprehensive survey on machine translation for English, Hindi and Sanskrit languages," *Journal of Ambient Intelligence and Humanized Computing*, pp. 1–34, 2021.
[34] N. N. Alleema, S. Babeetha, P. S. Kumar, S. Chandrasekaran, S. Pandiaraj *et al.,* "Recognition of American sign language using modified deep residual CNN with modified canny edge segmentation," *SSRN Journal*, 2022. https://doi.org/10.2139/ssrn.4052252
[35] Y. V. Kuriakose and M. Jangid, "Translation of American sign language to text: Using YOLOv3 with background subtraction and edge detection smart innovation, systems and technologies," in *Proc. of Smart Systems: Innovations in Computing*, Singapore, Springer, vol. 235, pp. 21–30, 2022.