Multimedia Tools and Applications (2023) 82:9627–9648
https://doi.org/10.1007/s11042-021-11595-4

1222: INTELLIGENT MULTIMEDIA DATA ANALYTICS AND COMPUTING

Indian sign language alphabet recognition system using CNN with diffGrad optimizer and stochastic pooling

Utpal Nandi[1] · **Anudyuti Ghorai**[2] · **Moirangthem Marjit Singh**[3] · **Chiranjit Changdar**[4] · **Shubhankar Bhakta**[5] · **Rajat Kumar Pal**[6]

Received: 17 February 2021 / Revised: 13 August 2021 / Accepted: 22 September 2021 / Published online: 8 January 2022

Abstract
India has the largest deaf population in the world and sign language is the principal medium for such persons to share information with normal people and among themselves. Yet, normal people do not have any knowledge of such language. As a result, there is a huge communication barrier between normal and deaf-dumb persons. Again, sign language interpreters are not easily available and it is a very costly solution for a long period. The sign language recognition system reduces the communication gaps between normal and deaf-dumb persons. The methodologies to recognize Indian sign language are recently in the developing stage and there is no approach to recognize signs in real-time. Here, we have proposed a fingerspelling recognition system of static signs for the Indian sign language alphabet using convolutional neural networks combined with data augmentation, batch normalization, dropout, stochastic pooling, and diffGrad optimizer. To continue the research, a total of 62,400 images of 26 static signs have been taken from various users. The proposed method achieves the highest training and validation accuracy of 99.76% and 99.64%, respectively , that outperforms other examined systems.

Keywords Sign language · Data augmentation · Pooling · Optimizer · Dropout · Convolution · Batch normalization

✉ Utpal Nandi

1 Dept. of Computer Science, Vidyasagar University, West Bengal, India

2 Dept. of Computer Science and BCA, Kharagpur College, West Bengal, India

3 Dept. of Computer Science and Engineering, North Eastern Regional Institute of Science and Technology, Arunachal Pradesh, India

4 Dept. of Computer Science, Belda College, West Bengal, India

5 Dept. of Computer Science, Mahishadal Girl's College, West Bengal, India

6 Dept. of Computer Science and Engineering, University of Calcutta, West Bengal, India

1 Introduction

India has the largest deaf population in the world where "one of every five people who are deaf in the world, lives in India". Sign language (SL) [44] is the principal medium for deaf-dumb persons to share information that is composed of different gestures of hand shapes, its movements, location, orientations, and also facial expressions. There are two types of SL symbols: single-handed and double-handed. Both types are either static or dynamic. The types of SL with corresponding abbreviations are shown in Fig. 1. The single-handed symbols are represented by the dominant hand only. The dynamic double-handed symbols are grouped into type 0 (both the hands are active) and type 1 (dominant hand is more active) signs. Type 0 signs are produced by using two hands whereas type 1 requires more participation of the dominant hand.

The SL symbols either are expressed by only hands known as manual signs or by body postures, mouth gestures, and facial expressions, also known as non-manual signs. However, normal people do not have any knowledge of such a language. As a result, there is a huge communication barrier between normal and deaf-dumb persons. A solution to this problem is to take the help of an SL interpreter, but interpreters are not easily available and this requires a very costly solution for a long period. Therefore, research works are continuing to design and implement a system that can almost automatically recognize the gestures of SL to reduce the communication gap between two groups of people in society.

Recently, several research works have been continued to develop recognition systems of different SLs not only for the deaf population but also for applications in robot controls, video games, and virtual reality environments [36, 38]. A lot of researches have been done

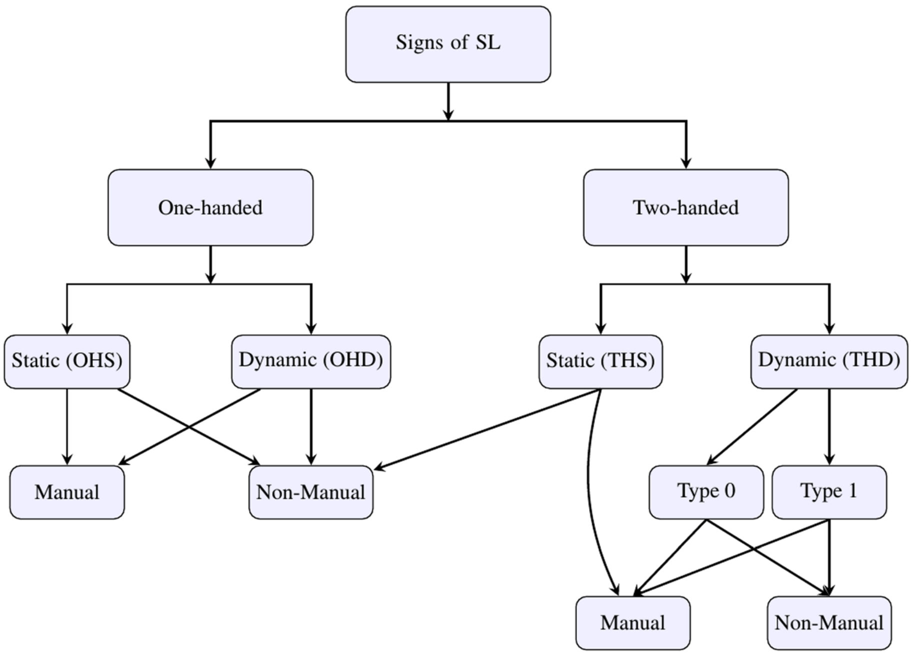

Fig. 1 Variation of different types of signs available in Indian sign language

to develop SL Recognizer (SLR) for other countries. Those are at an advanced stage but Indian SL (ISL) recognition methodologies are recently in a growing stage and there is no approach to recognize signs in real-time. Therefore, it is necessary to continue the research toward the development of a complete ISL recognition system.

The ISL has different types of posture variability that turned recognition of a complex problem and therefore, recordings of perfect postures are required for framing the dataset. Again, it is necessary to create the ISL database since no standard ISL dataset is available yet. The research works on sign language recognition reported in the past years were based on machine learning that offers poor accuracy for the absence of automatic extraction of features. In recent time, deep neural network-based approaches have reached demand to solve problems in many fields [11] and outperform the traditional techniques in natural language processing [6, 12], computer vision [21, 30, 46, 47], robotics [46], signal processing [34, 59], image processing [8, 62], and other various fields of artificial intelligence. Deep learning based SLRs have also been designed. However, the majority of works in deep learning based SLR are done on sign languages other than ISL where automatic feature extraction is possible.

Based on the necessity stated above, we have proposed a fingerspelling recognition system of static signs for the ISL alphabet using a convolutional neural network (CNN) that is built by applying six convolutional (Conv.) layers with stochastic pooling, batch normalization and dropout, followed by the applications of two fully connected (FC) layers and diffGrad optimization method. Data augmentation is also done to achieve better performance by populating the dataset. The training and validation accuracies and losses of the presented approach have been obtained for four distinct optimizers and three types of pooling methods. The other performance measures like precision, recall (or sensitivity), and F1-score have also been presented. The proposed approach offers better results than the remaining examined systems.

The paper is organized as follows. Section 2 describes the related works of other SL and ISL while Section 3 explains a basic CNN architecture as an SL symbol classifier. The proposed system architecture has been demonstrated in Section 4. Section 5 presents the results of our experiments. At last, the conclusion has been drawn with possible future scopes in Section 6.

2 Related works

There is a huge variety of SLs worldwide. Some of them are American SL (ASL), Arabic SL (ArSL), Chinese SL (CSL), Indian SL (ISL), Persian SL (PSL), Irish SL ((IrSL)), and so on. ASL is distributed across the states of America, part of Canada, and Africa. Sun et al. [49] introduced an approach to recognize ASL that consists of 73 signs where mi-SVM and Adaboost were used to train the model and classify data, respectively. Sun et al. [50] also proposed a Latent SVM-based SLR for ASL that yields 82.9% and 86% accuracy for 63 sentences and 73 signs, respectively, prepared using the Kinect sensor. Moreover, the accuracy became 96.67% for fusing the data glove and camera.

Kim et al. [18] presented an ASL recognizer by the impulse radio sensor. The CNN architecture was applied to classify signs and achieved more than 90% accuracy. Oyedotun and Khashman [39] introduced an ASL recognition technique for 2040 alphabet signs. The segmentation of these signs was done by applying a median filter. The system achieved 91.33% accuracy for ConvNet.

ArSL is used in Mideast and North Africa regions. Al-Rousan et al. [3] designed an SLR for 30 THD type signs of ArSL that extracts features using Discrete Cosine Transform (DCT) and zonal coding and Hidden Markov Model (HMM) for classification. The system offered 93.8% and 90.6% accuracy for signer-dependent and signer-independent mode, respectively. Shanableh and Assaleh [45] proposed an ArSL recognizer for 3450 signs that applied K-nearest neighbor (KNN) to achieve 87% accuracy. Mohandes et al. [35] proposed a signer-independent ArSL recognition method for THS type signs by using a region growing scheme. Dahmani and Larabi [7] presented a framework for recognizing ArSL based on shape descriptors and classification done using KNN with Support Vector Machine (SVM). A user-dependent ArSL recognizer was presented by Tubaiz et al. [53] for dynamic continuous signs where modified KNN was applied for classification.

CSL has spread across the counties of China, Malaysia, and Taiwan. Yang et al. [57] introduced a continuous CSL recognizer for few sentences where the model used LB-HMM (Level Building HMM) and its variant LB-Fast-HMM for classification to decrease computational complexity. Guo et al. [13] presented a CSL recognizer. The Histograms of Oriented Gradients (HOG) and Principal Component Analysis (PCA) were used for feature extraction and offered 67.34% accuracy. Jiang et al. [15] proposed a CNN-based CSL recognizer with stochastic pooling, batch normalization, and dropout that yields a maximum accuracy of 90.91%.

PSL is used by deaf people in Iran. Karami et al. [16] presented a discrete wavelet transform-based PSL recognizer for SHS type alphabets. IrSL has spread in part of Ireland. Kelly et al. [17] introduced a person-independent recognition system for IrSL. ISL is distributed across South Asia with approximately 2,700,000 users [31]. However, ISL recognition systems are recently in a growing period. Tripathi et al. [51] proposed a continuous ISL gesture recognizer for two-hand gestures using gradient-based key frame extraction method. Mehrotra et al. [33] introduced a recognition method for 37 two-hand signs of ISL captured by Microsoft Kinect using multi-class SVM that obtained approximately 86% accuracy. Tripathi et al. [52] presented an approach for recognizing ISL sentences where the HMM were applied for the classification of signs. It achieved 91% accuracy. Kishore et al. [20] introduced an approach for sentence recognition of ISL and offered approximately 90% accuracy.

Naglot et al. [37] applied an ANN to classify OHD type signs of ISL using Leap Motion Controller (LMC). Kumar et al. [22] proposed a continuous ISL recognizer where a mobile front camera was used to collect signs. The system reached 90% accuracy. Ahmed et al. [2] presented an SLR for 24 double-handed dynamic signs where 90% accuracy was achieved. Kumar et al. [23] proposed a sensor-dependent ISL recognizer using leap motion equipment for 50 sign words. The model applied both HMM (95.60% accuracy) and Bidirectional Long Short Term Memory Neural Network (BLSTM-NN) (84.57% accuracy) for classification. Kumar et al. [24] developed an ISL recognition method depending on HMM for 25 OHD type signs applying Kinect with leap motion. Rao and Kishore [43] presented an SLR for selfie video of ISL where DCT was utilized to extract features and offered 90% average accuracy.

Kumar et al. [28] developed an ISL recognition model for 2240 OHS type signs applying leap motion that had 63.57% accuracy for SVM and BLSTM-NN. Kumar et al. [25] introduced an approach for ISL recognition applying Kinect for static sign words. The model achieved approximately 83.77% accuracy. Wadhawan and Kumar [55] presented a systematic literature review of SL recognizers between the last decades. Wadhawan and Kumar [56] presented an ISL recognizer for 100 static signs using deep learning-based CNN that achieved above 99% accuracy on both colored and grayscale images. Kumar et al. [26] developed a

CNN based ISL recognizer for 50,000 sign videos that achieved 92.14% accuracy. Raghuveera et al. [40] introduced an algorithm to translate ISL into English text and speech for the dataset of 4600 images captured by Microsoft Kinect. Mali et al. [32] proposed a system with a computer human interface for ISL using SVM classifier. The Soft Computing technique was also applied in ISL recognition by Sahoo et al. [4].

3 CNN architecture for ISL alphabet recognition system

As an example, a common CNN architecture is depicted in Fig. 2, where Conv. layers are applied for feature extraction and two FC layers are kept for classification. Methodologies of CNN architecture for ISL recognition system are explained in the following sub-sections.

3.1 Convolutional layer

The CNN model is specially designed for 2D images, although it may be applied with 1D and 3D data. Central to this net is the Conv. layer that does convolution operation on data. In CNN, this linear operation involves the multiplication of many weights with the input. For a 2D image, the multiplication is done between the input image and a 2D array of weights, termed as a filter. The same filter used in a systematic way across the entire input, left to right and top to bottom, is an effective idea to find a specific feature anywhere in the input. The output of convolution operation is a 2D array termed as 'feature map'. The convolution operation explained in the context of CNN is a 'cross-correlation' operation. The Conv. layer output size for input image with size n and filter with size f is presented by Eq. (1) where s and p represent stride and padding, respectively.

$$Output\_size = \lfloor \frac{(n + 2p - f)}{s} + 1 \rfloor \quad (1)$$

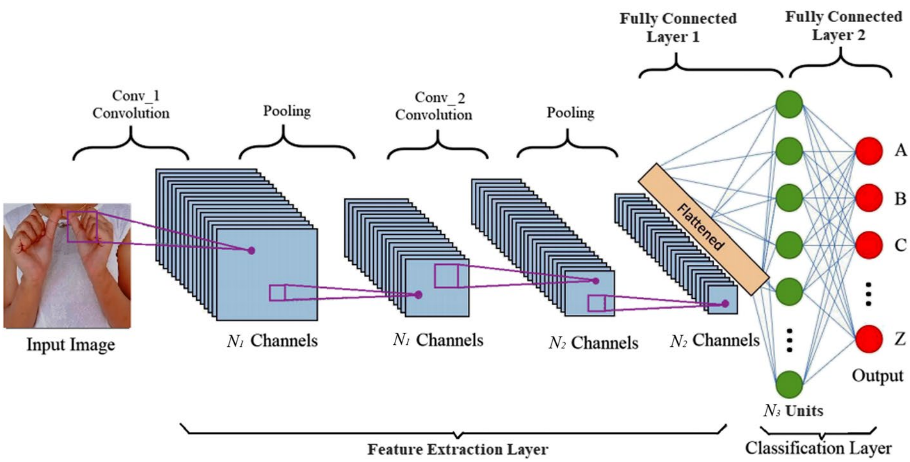

Fig. 2 An example of CNN architecture for ISL recognition system

Figure 3 demonstrates an example of convolution operation for stride $s = 1$ and padding $p = 0$, where a filter with size 3 is applied to a 6×6 2D input image to produce a feature map. The size of the feature map is $\lfloor \frac{(n+2p-f)}{s} + 1 \rfloor = \lfloor \frac{(6-3)}{1} + 1 \rfloor = 4$.

3.2 Pooling layer

The pooling layer is placed after the Conv. layer within a CNN that processes each feature map individually to produce the same number of pooled new feature maps. Pooling chooses a pooling operation (e.g., a filter) whose dimension is less than the dimension of the feature map. The pooling operation with 2×2 pixels and a stride of 2 pixels reduces feature map size by half. There are mainly two common schemes of pooling: max pooling and average pooling. Former keeps patch-wise highest values and the later keeps averages of the same. For the activation set Z in a pooling region R defined in Eq. (2), the max and average pooling can be obtained by Eq. (3) and Eq. (4) respectively, where $|Z_R|$ is the length of Z.

$$Z = [z_i | i \in R] \tag{2}$$

$$Max\ pooling(P_M) = Max(Z_R) \tag{3}$$

$$Average\ pooling(P_A) = \frac{\sum Z_R}{|Z_R|} \tag{4}$$

An example of both pooling operations is illustrated for 2×2 filter and stride of 2 in Fig. 4. For input with multiple channels, pooling reduces the dimension but keeps the number of channels unchanged.

The pooled feature maps are a summary of the features found in the input image where minor variations in the feature position will output the same position in the pooled feature map. This capability is termed the "model's invariance to local translation".

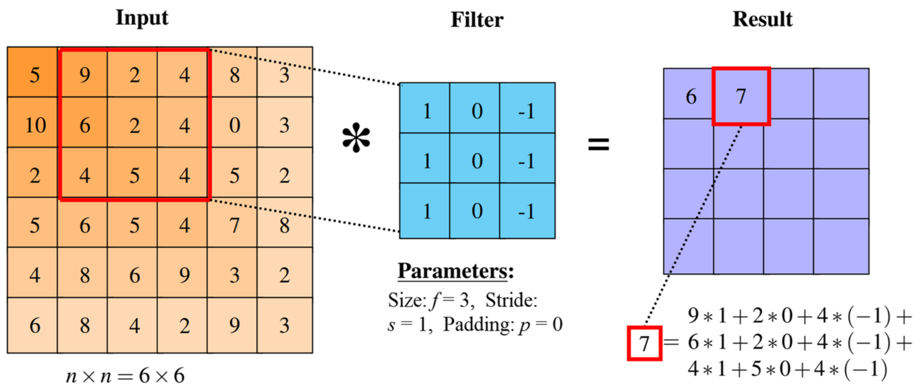

Fig. 3 Example of a filter applied to a 2D image to create a feature map

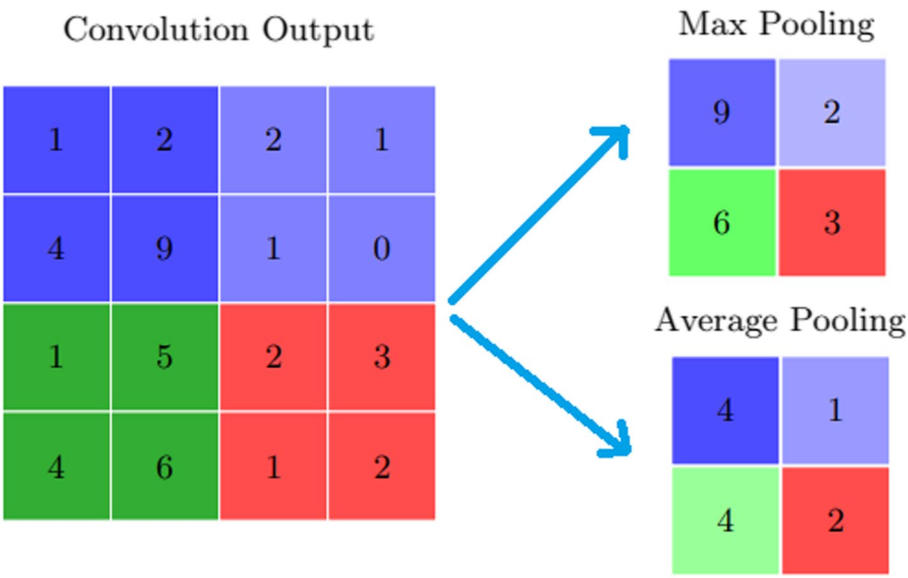

Fig. 4 Example of Max Pooling and Average Pooling

3.3 Batch normalization

Training deep networks are very challenging where a layer's inputs' distribution is often influenced by the change of the parameters of the previous layers. This may slow down the training process. The Batch Normalization (BN) [14] is applied in training deep networks to standardize input to a layer. The technique dramatically reduces the training epochs needed for training the network. The BN layer is placed after the Conv. layer or an FC layer (see Fig. 5). The equations of BN with learning parameters γ and β for input set $[a_i]$ and mini-batch output $[b_i]$ are defined in Eqs. (5), (6),(7), and (8), respectively.

$$Mini\text{−}batch\ mean(\mu) = \frac{1}{n}\sum_{i=1}^{n} a_i \tag{5}$$

$$Mini\text{−}batch\ variance(\sigma_\beta^2) = \frac{1}{n}\sum_{i=1}^{n} (a_i - \mu)^2 \tag{6}$$

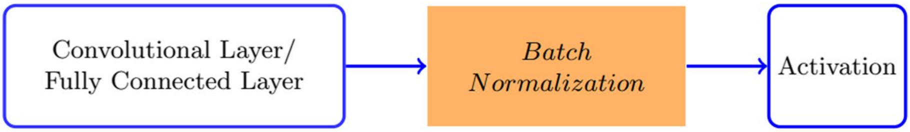

Fig. 5 Illustration of batch normalization

$$Normalized\ value(\hat{a}_i) = \frac{(a_i - \mu)}{\sqrt{\sigma^2 + \epsilon}} \tag{7}$$

$$b_i = \gamma \hat{a}_i + \beta \equiv BN_{\gamma,\beta}(a_i) \tag{8}$$

3.4 Dropout

Deep neural nets may overfit a training dataset quickly and is a serious problem. Dropout technique [48] can deal with this problem. The technique temporarily removes units from the neural network with its incoming and outgoing connections randomly from the neural net at the time of training that produces a thinned network containing units that survived dropout. It not only reduces overfitting but also offers significant improvements over other regularization schemes. Fig. 6(a) and (b) show an example of standard neural network and its corresponding thinned neural network, respectively. It is noticed that some neural units are discarded from each layer of a thinned neural network at a specific dropout rate.

3.5 Optimizer

Optimizers are applied in the neural network to tune attributes like weights and learning rate to reduce the losses and to offer possible accurate results. Stochastic Gradient Descent (SGD) [5] is an efficient popular optimization method where the gradient determines the path where a function has the sharp rate of change and the model parameters are updated

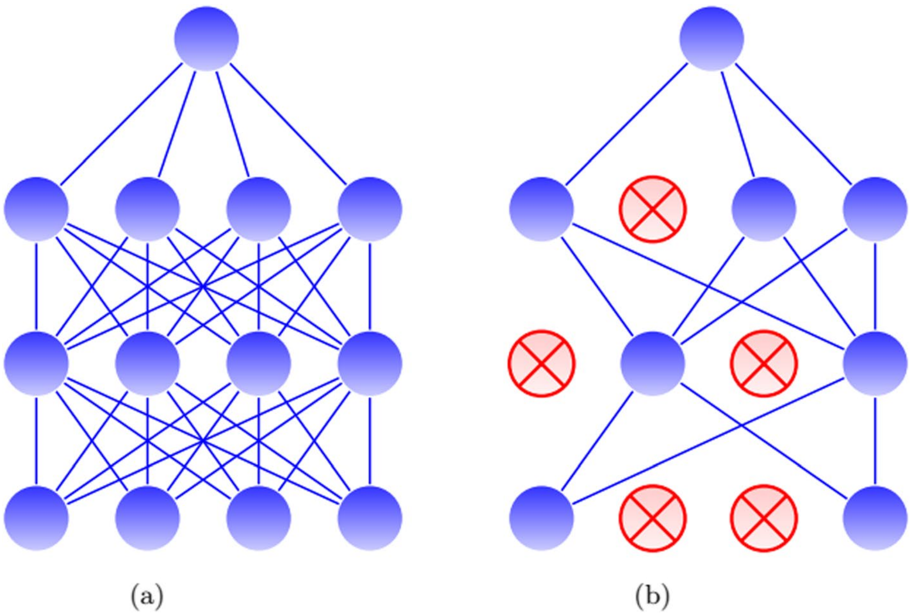

Fig. 6 Example of dropout Neural Network: (a) A standard Neural Network; (b) A thinned Neural Network produced after applying dropout

after calculation of loss on each training example instead of single time as in Gradient Descent. However, the major limitation of SGD is that it updates all parameters in equal-sized steps. Again, it has a high variance in parameters and may shoot even after reaching global minima. These steps can be made adaptive in size for each parameter to improve SGD further. Many optimization methods like AdaGrad [10], AdaDelta [60], and adaptive moment estimation (Adam) [19] were proposed for the same.

4 Proposed ISL alphabet recognition system

Most of the existing recognition systems use either max pooling or average pooling as discussed in Section 3.2. However, both pooling has few limitations. The max pooling retains more texture information whereas the average pooling retains more background information of the image. The proposed system applied probabilistic stochastic pooling [61] to reduce these limitations. The method picks the activation within each pooling region randomly based on multinomial distribution and it is free of hyperparameters. The stochastic pooling computes first the probabilities P for each region j via activation a_i by Eq. (9). It then picks a location l within the region randomly based on P. The pooled activation is then simply a_l as given in Eq. (10), where $l \sim P(p_1, ...p_{|R_j|})$. An example of a stochastic pooling procedure is illustrated in Fig. 7. It is observed that the chosen element is not the largest for the pooling region.

$$P_i = \frac{a_i}{\sum_{k \in R_j} a_k} \tag{9}$$

$$S_j = a_l \tag{10}$$

Again, the optimizers mentioned in Section 3.5 cannot take the benefit of local change in gradients since they depend on past gradients. Therefore, the diffGrad optimization

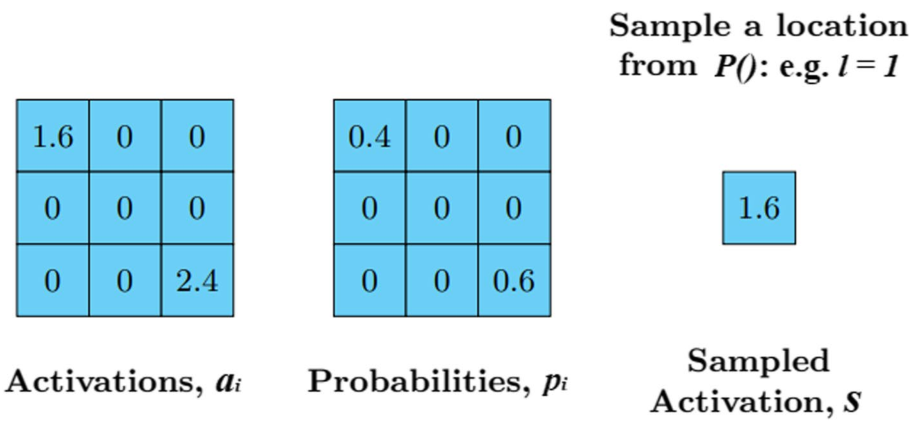

Fig. 7 Example of Stochastic Pooling

method [9] is chosen for our work that depends on the difference from the present gradient to the past gradient. The optimizer has a high step length for fast gradient updating parameters and a small step length for low gradient updating parameters.

The proposed ISL alphabet recognizer is designed to recognize 26 static signs of the ISL alphabet. It has mainly five phases: data collection, resizing and normalization, training data augmentation, classifier training, and testing (see Fig. 8). These are explained as follows.

4.1 Data collection

The dataset is prepared under various lighting conditions and backgrounds using a camera that includes 26 static signs of ISL alphabets each having 2400 images. Therefore, there are total 24,00 × 26= 62,400 images.

4.2 Data resizing and normalization

Image resizing and normalization are applied for preprocessing. Since the neural network model takes the same size input, all collected images require to be resized to a fixed size before feeding those to the network. The larger the image size, the less shrinking is needed that means less deformation of patterns and features of data. Again, normalizing the pixel values of images is a good practice where each pixel value has a value between 0 and 1. It helps to speed up the learning process and leads to faster convergence. Therefore, the collected images are resized into 256 × 256 shapes and then the pixel values of images are normalized by dividing 255 since the highest pixel value is 255.

4.3 Training data augmentation

Data augmentation generates new variations of images from existing training images artificially. It significantly increases the amount and diversity of images for training. In this way, the dataset in a model becomes rich and sufficient, and the model offers better performance. These augmentation techniques are applied in our dataset for this reason. The model gets trained using 49,920 images (80% of the total 62,400 images) and tested using 20%, i.e., 12,480 images. Training images are augmented using zoom (scaling images by a factor), rotation (rotating images by a degree), shear (shear angle in a counter-clockwise direction in degrees), width and height shift, and ZCA whitening (shift the color values mostly present in images like PCA but preserve the spatial arrangement of pixels important for CNN) operations. As a result, each image generates 90 new images and there are 49,920 × 91= 4,542,720 images in augmented training data.

4.4 Model training

The system gets trained using CNN classifier using images of 26 signs since ISL has 26 English alphabets. The training process tunes the parameters of the network until the accuracy is satisfied. To add randomness to the training of CNN, the dataset is shuffled to prevent bias on parameters. The shuffled data helps to break symmetry and offers better performance where the weights are set randomly close to zero and every neuron performs no longer the similar computation.

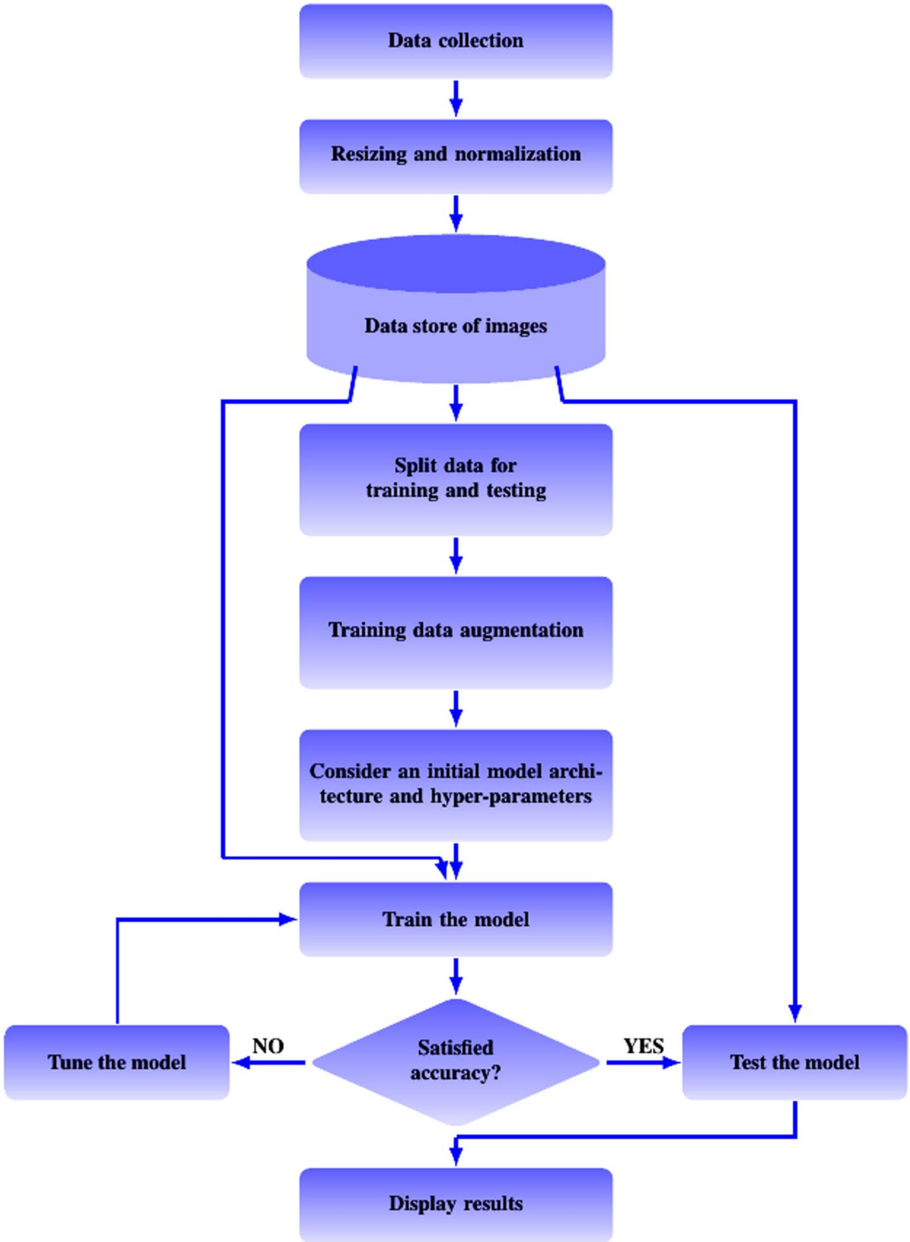

Fig. 8 The schematic diagram of the proposed ISL alphabet recognition system

4.5 Testing

The model is tested using 20%, i.e., 12,480 images. The trained model is used to predict the images of the test dataset and the results in terms of classification performance are measured using metrics like validation accuracy, validation loss, precision, recall, and F1-score.

5 Experimental results

The ISL dataset for our experiment has been created by different volunteers using a camera. Part of samples of the ISL alphabet dataset used for our experiment is shown in Fig. 9. There are 26 classes of the dataset each having 2400 color images of size 256×256.

Fig. 9 Sample images of our ISL alphabet dataset

5.1 Data augmentation results

The augmentation methods like zooming, rotation, shear, width and height shift, and so on are applied to the dataset to significantly expand the dataset beneficial for deep learning. It also improves accuracy by reducing over-fitting. The sample augmented images of our ISL alphabet dataset are depicted in Fig. 10.

5.2 Structure of the proposed CNN

The CNN architecture has been fine-tuned and finalized as an eight-layer CNN with six Conv. layers followed by two FC (FC_1 and FC_2) layers as given in Table 1, where Block_2 (64 $5 \times 5 \times 32$ Conv, BN, ReLU, SP) represents Conv. layer with 64 filters of size 5×5, 32 channels, and applied batch normalization (BN), ReLU activation and stochastic pooling (SP). The 30% dropout has been done between the sixth Conv. layer and the first FC layer.

Fig. 10 Sample augmented images of our ISL alphabet dataset

Table 1 The details of the proposed CNN architecture

Layer	Activations	Weights	Bias
Input	$256 \times 256 \times 3$		
Block_1 (32 $7 \times 7 \times 3$ Conv, BN, ReLU, SP)	$125 \times 125 \times 32$	$7 \times 7 \times 3 \times 32$	$1 \times 1 \times 32$
Block_2 (64 $5 \times 5 \times 32$ Conv, BN, ReLU, SP)	$60 \times 60 \times 64$	$5 \times 5 \times 32 \times 64$	$1 \times 1 \times 64$
Block_3 (128 $5 \times 5 \times 64$ Conv, BN, ReLU, SP)	$23 \times 23 \times 128$	$5 \times 5 \times 64 \times 128$	$1 \times 1 \times 128$
Block_4 (128 $5 \times 5 \times 128$ Conv, BN, ReLU, SP)	$11 \times 11 \times 128$	$3 \times 3 \times 128 \times 128$	$1 \times 1 \times 128$
Block_5 (256 $5 \times 5 \times 128$ Conv, BN, ReLU, SP)	$4 \times 4 \times 256$	$3 \times 3 \times 128 \times 256$	$1 \times 1 \times 256$
Block_6 (256 $5 \times 5 \times 128$ Conv, BN, ReLU, SP)	$1 \times 1 \times 256$	$3 \times 3 \times 256 \times 256$	$1 \times 1 \times 256$
FC_1 (100)	$1 \times 1 \times 100$	100×256	100×1
FC_2 (26)	$1 \times 1 \times 26$	26×100	26×1

5.3 Classification performance

After tuning parameters of the model, the training accuracy, validation accuracy, training loss, and validation loss of the model are obtained using DiffGrad optimization for 10 runs and kept in Table 2. The highest and lowest training accuracies are 99.76% and 98.96%, respectively, and the highest and lowest validation accuracies are 99.64% and 98.98%, respectively. The overall training and validation accuracies reach up to 99.52± 0.2334% and 99.36± 0.2595%, respectively. The accuracy and loss curves for the first run are illustrated in Figs. 11(a)

Table 2 The training and validation accuracies of the proposed model

Run	Proposed Method			
	Training Accuracy	Training Loss	Validation Accuracy	Validation Loss
1	99.68	0.013	99.64	0.0641
2	99.46	0.0135	99.17	0.0687
3	99.49	0.0109	99.45	0.0675
4	99.71	0.0131	99.51	0.0662
5	99.76	0.0108	99.64	0.0644
6	99.38	0.0105	99.06	0.0715
7	98.96	0.0109	98.98	0.0715
8	99.49	0.012	99.55	0.0661
9	99.58	0.0119	99.07	0.0695
10	99.69	0.0114	99.53	0.0671
Average	99.52	0.0118	99.36	0.0676
SD	0.2334	0.0011	0.2597	0.0025

and 12(b), respectively. The model is trained up to the 30th epoch due to improvement stagnation in the accuracy.

The precision (*Pr*), recall (*Rc*), and F1-score (*Fs*) defined in Eqs. (11), (12), and (13), respectively, have also been evaluated for the first run using the numbers of true positives (*TP*), false positives (*FP*), and false negatives (*FN*). The *Pr*, *Rc*, and *Fs* of all 26 signs are given in Table 3.

$$Pr = \frac{TP}{(TP + FP)} \tag{11}$$

$$Rc = \frac{TP}{(TP + FN)} \tag{12}$$

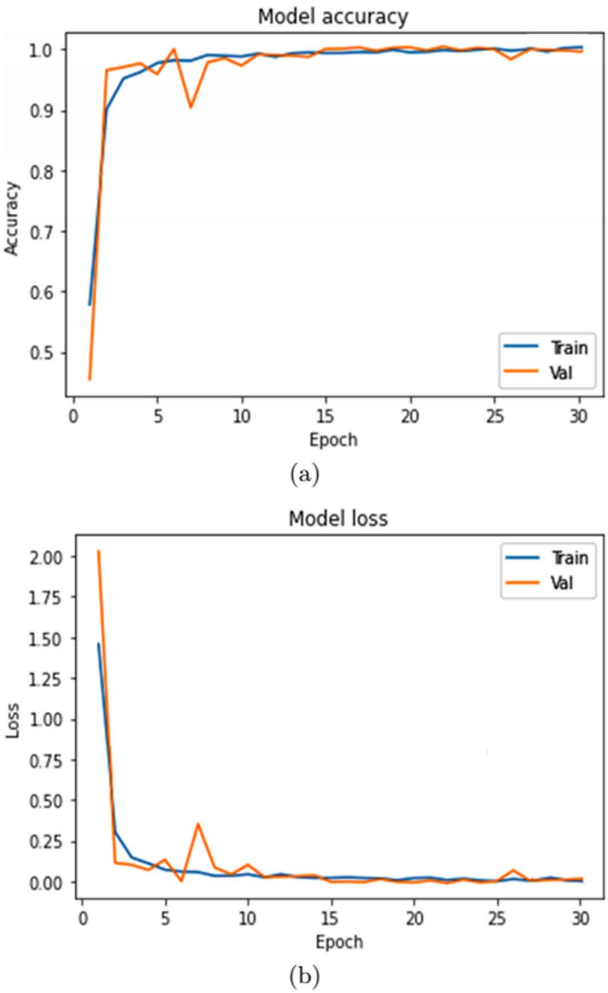

Fig. 11 Accuracy and loss curves for **a** training and **b** validation

Table 3 Classification performance

Sign	Precision	Recall	F1-score	Sign	Precision	Recall	F1-score
A	1.00	1.00	1.00	N	1.00	1.00	1.00
B	1.00	1.00	1.00	O	0.99	1.00	1.00
C	1.00	1.00	1.00	P	0.95	1.00	0.98
D	1.00	0.99	1.00	Q	1.00	0.95	0.97
E	0.99	1.00	1.00	R	1.00	0.98	0.99
F	1.00	1.00	1.00	S	1.00	1.00	1.00
G	0.99	1.00	1.00	T	1.00	1.00	1.00
H	1.00	1.00	1.00	U	1.00	1.00	1.00
I	1.00	0.98	0.99	V	0.99	1.00	1.00
J	1.00	1.00	1.00	W	1.00	1.00	1.00
K	0.99	1.00	1.00	X	1.00	1.00	1.00
L	1.00	1.00	1.00	Y	1.00	0.99	1.00
M	1.00	1.00	1.00	Z	0.99	1.00	1.00

$$Fs = \frac{2 * Pr * Rc}{(Pr + Rc)} \tag{13}$$

5.4 Pooling method comparison

In this study, the stochastic pooling has been compared with average and max pooling as depicted in Table 4. It is noticed that stochastic pooling achieves better average training and validation accuracies of 99.52± 0.2334% and 99.36± 0.2595%, respectively, than the other two. However, the maximum pooling offers the maximum training and validation accuracies of 99.84% and 99.73%, respectively, among these three pooling methods for the fourth run.

5.5 Dropout rate

The average training and validation accuracies with varying dropout rate (10 runs for each) are obtained and the graphical representations are depicted in Fig. 12(a) and (b), respectively. Both the accuracies increase with an increase of the dropout rate started from 0% and maximum training and validation accuracies 99.52±0.2334% and 99.36±0.2597%, respectively, are achieved for 30% dropout rate. Further increase of dropout rate reduces accuracy gradually. Hence, the dropout rate is chosen as 30%.

5.6 Experimental results with respect to optimizer

The presented model has been tested using four optimization algorithms for 10 runs. The average accuracy and loss of these optimizers are kept in Table 5. It is noticed that the validation accuracy of DiffGard outperformed RMSProp, Adam, and SGD optimizers. The

Table 4 Pooling method comparison

Run	Pooling Methods					
	Average Pooling		Max Pooling		Stochatic Pooling	
	Training Accuracy	Validation Accuracy	Training Accuracy	Validation Accuracy	Training Accuracy	Validation Accuracy
1	98.72	98.41	99.4	99.25	99.68	99.64
2	98.94	98.62	99.29	99.05	99.46	99.17
3	99.21	99.1	99.45	99.23	99.49	99.45
4	98.13	98.02	99.84	99.73	99.71	99.51
5	98.66	98.54	98.95	99.03	99.76	99.64
6	98.2	97.86	99.28	98.91	99.38	99.06
7	98.31	98.02	99.26	99.12	98.96	98.98
8	99.3	99.15	99.58	99.37	99.49	99.55
9	98.08	98.07	99.31	98.91	99.58	99.07
10	98.26	98.1	99.37	99.18	99.69	99.53
Average	98.58	98.39	99.37	99.18	99.52	99.36
SD	0.4525	0.4593	0.2307	0.2438	0.2334	0.2597

proposed model offers the average training and validation accuracy of 99.52% and 99.36% using DiffGard optimizer, respectively.

5.7 Comparison to the state-of-the-art approaches

The comparative result using the recognition rate of the presented ISL recognition approach with others is given in Table 6. It has been noticed that the presented approach outperforms other listed models since the former uses a deep learning-based CNN with multiple technologies like data augmentation to expand the dataset, batch normalization to standardize the inputs to a layer, dropout to handle overfitting, stochastic pooling to resolve the overfitting and down-weight issue and diffGard optimizer.

5.8 Performance comparison of the proposed model architecture with standard architectures

The performance of the proposed model architecture is also compared with Inception V3, ResNet18, and ResNet50 standard architectures in Table 7. It is noticed that the average training and validation accuracies of the proposed model architecture are very close to these standard architectures.

5.9 Prediction time of test set

The network takes only 0.728 seconds to predict 12,480 images of the test set and therefore is very useful for real-time output of SL interpretation.

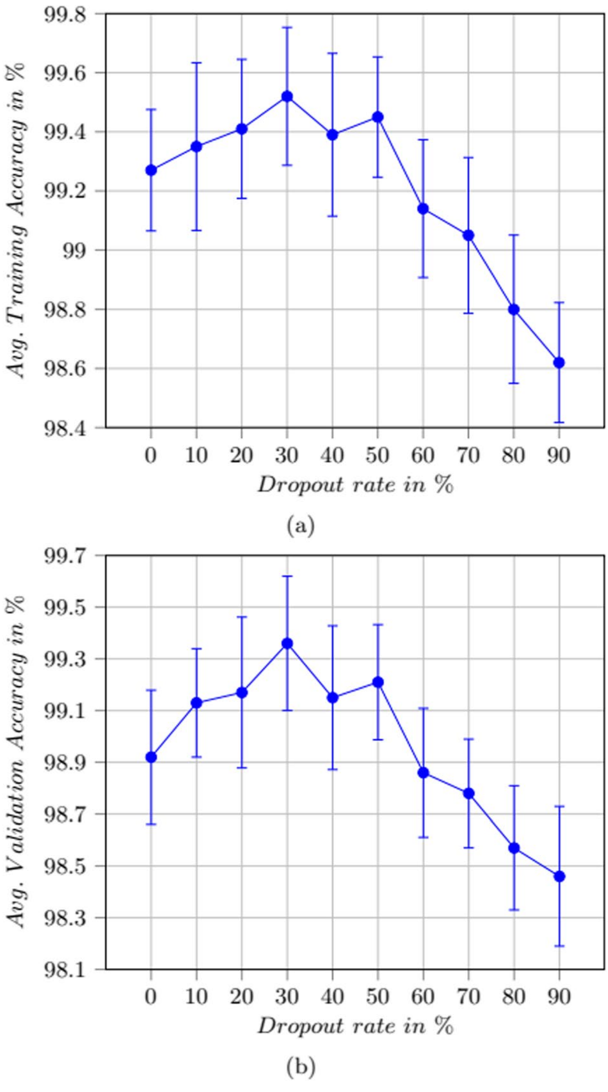

Fig. 12 (a) Error bar of average training accuracy against dropout rate and (b) Error bar of average validation accuracy against dropout rate

Table 5 Comparison of different optimizers

Optimizer	Avg. Training Accuracy	Avg. Training Loss	Avg. Validation Accuracy	Avg. Validation Loss
SGD	99.56	0.0126	99.27	0.0759
RMSProp	99.54	0.0378	99.18	0.1940
Adam	99.27	0.0280	99.14	0.2168
DiffGard	99.52	0.0118	99.36	0.0676

Table 6 Comparison of the proposed system with other classifiers for static single/double handed signs

Author	Single/ double handed	Techniques used	Acquisition mode	Recognition rate in %
Agrawal et al. [1]	Double	Multi-class SVM	Camera	93%
Rahaman et al. [41]	Double	KNN	Camera	98.17% (vowels) and 94.75% (consonants)
Yasir et al. [58]	Double	SVM	Camera	96%
Uddin and Chowdhury [54]	Double	SVM	Camera	97.70%
Rao and Kishore [42]	Single	ANN	Camera	90.00%
Kumar et al. [27]	Single	SVM and BLSTM-NN	Leap motion	63.57%
Kumar et al. [29]	Single	HMM	Kinect	83.77%
Proposed	Single	CNN	Camera	99.36%

Table 7 Performance comparison of the proposed model architecture with standard architecture

Network Architecture	Avg. Training Accuracy	Avg. Validation Accuracy
Inception V3	99.52	99.48
ResNet18	99.56	99.45
ResNet50	99.59	99.47
Proposed	99.52	99.36

6 Conclusion and future scope

This paper has proposed an ISL alphabet recognition system using optimized CNN with data augmentation, stochastic pooling, batch normalization, dropout, and diffGrad optimizer. The model offers the maximum training accuracy and validation accuracy of 99.76% and 99.64%, respectively. The comparison of accuracies of the model for using stochastic, average, and max pooling methods is done where the former reduces overfitting significantly. We have also tested the system using four optimizers and noticed that diffGrad outperformed Adam and SGD optimizers. The proposed system can also be eventually updated to learn all static signs of ISL.

However, still the proposed system is failed to recognize dynamic signs and real-time sign words.There could be assumed as open problems for the time being. In the future, the experiment should be continued to design a system to recognize dynamic signs and signs in real-time of ISL.

Acknowledgements We'd like to thank to the Dept. of Computer Science, Vidyasagar University for providing infrastructure and reviewers for criticisms and suggestions.

Declarations

Conflicts of interest The authors have no potential conflict of interests.

References

1. Agrawal SC, Jalal AS, Bhatnagar C (2012) Recognition of indian sign language using feature fusion. 2012 4th International Conference on Intelligent Human Computer Interaction (IHCI), pp 1–5
2. Ahmed W, Chanda K, Mitra S (2016) Vision based hand gesture recognition using dynamic time warping for indian sign language. In: 2016 international conference on information science (ICIS). IEEE, pp 120–125
3. Al-Rousan M, Assaleh K, Tala'a A (2009) Video-based signer-independent Arabic sign language recognition using hidden markov models. Appl Soft Comput 9(3):990–999
4. Ashok Kumar Sahoo, Pradeepta Kumar Sarangi, Parul Goyal (2020) Indian Sign Language Recognition Using Soft Computing Techniques. Book Editor(s):Muthukumaran Malarvel, Soumya Ranjan Nayak, Surya Narayan Panda, Prasant Kumar Pattnaik, Nittaya Muangnak, https://doi.org/10.1002/9781119682042.ch2
5. Bottou L (2010) Large-scale machine learning with stochastic gradient descent. In: Lechevallier Y, Saporta G (eds) Proceedings of COMPSTAT'2010, Physica-Verlag HD, Heidelberg, pp 177–186
6. Collobert R, Weston J (2008) A unified architecture for natural language processing: deep neural networks with multitask learning. In: Proceedings of the 25th international conference on machine learning, pp 160–167
7. Dahmani D, Larabi S (2014) User-independent system for sign language finger spelling recognition. J Vis Commun Image Representation 25(5):1240–1250
8. Dong C, Loy CC, He K, Tang X (2015) Image super-resolution using deep convolutional networks. IEEE Trans Pattern Analysis Mach Intell 38(2):295–307
9. Dubey SR, Chakraborty S, Roy SK, Mukherjee S, Singh SK, Chaudhuri BB (2019) diffgrad: An optimization method for convolutional neural networks. IEEE Transactions on Neural Networks and Learning Systems 33(11):1–12
10. Duchi J, Hazan E, Singer Y (2011) Adaptive subgradient methods for online learning and stochastic optimization. Journal of Machine Learning Research 12(61):2121–2159, http://jmlr.org/papers/v12/duchi11a.html
11. Goodfellow I, Bengio Y, Courville A (2016) Deep learning. MIT Press
12. Greff K, Srivastava RK, Koutník J, Steunebrink BR, Schmidhuber J (2016) Lstm: a search space odyssey. IEEE Trans Neural Netw Learning Syst 28(10):2222–2232
13. Guo D, Zhou W, Li H, Wang M (2017) Online early-late fusion based on adaptive hmm for sign language recognition. ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM) 14(1):1–18
14. Ioffe S, Szegedy C (2015) Batch normalization: accelerating deep network training by reducing internal covariate shift. In: ICML'15: proceedings of the 32nd international conference on machine learning, pp 448–456
15. Jiang X, Lu M, Wang SH (2019) An eight-layer convolutional neural network with stochastic pooling, batch normalization and dropout for fingerspelling recognition of chinese sign language. Multimedia Tools and Applications 79:1–19
16. Karami A, Zanj B, Sarkaleh AK (2011) Persian sign language (psl) recognition using wavelet transform and neural networks. Expert Systems with Applications 38(3):2661–2667
17. Kelly D, McDonald J, Markham C (2010) A person independent system for recognition of hand postures used in sign language. Pattern Recogn Lett 31(11):1359–1368
18. Kim SY, Han HG, Kim JW, Lee S, Kim TW (2017) A hand gesture recognition sensor using reflected impulses. IEEE Sensors Journal 17(10):2975–2976
19. Kingma DP, Ba J (2014) Adam: a method for stochastic optimization. 1412.6980
20. Kishore P, Prasad M, Kumar DA, Sastry A (2016) Optical flow hand tracking and active contour hand shape features for continuous sign language recognition with artificial neural networks. In: 2016 IEEE 6th international conference on advanced computing (IACC). IEEE, pp 346–351
21. Krizhevsky A, Sutskever I, Hinton GE (2012) Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems, pp 1097–1105
22. Kumar DA, Kishore P, Sastry A, Swamy PRG (2016) Selfie continuous sign language recognition using neural network. In: 2016 IEEE Annual India Conference (INDICON). IEEE, pp 1–6

23. Kumar P, Gauba H, Roy PP, Dogra DP (2017) A multimodal framework for sensor based sign language recognition. Neurocomputing 259:21–38
24. Kumar P, Gauba H, Roy PP, Dogra DP (2017) Coupled hmm-based multi-sensor data fusion for sign language recognition. Pattern Recogn Lett 86:1–8
25. Kumar P, Saini R, Roy PP, Dogra DP (2018) A position and rotation invariant framework for sign language recognition (slr) using kinect. Multimed Tools Applic 77(7):8823–8846
26. Kumar EK, Kishore P, Kumar MTK, Kumar DA (2020) 3d sign language recognition with joint distance and angular coded color topographical descriptor on a 2-stream cnn. Neurocomputing 372:40–54
27. Kumar P, Saini R, Behera SK, Dogra DP, Roy PP (2017) Real-time recognition of sign language gestures and air-writing using leap motion. In: 2017 Fifteenth IAPR International Conference on Machine Vision Applications (MVA), pp 157–160
28. Kumar P, Saini R, Behera SK, Dogra DP, Roy PP (2017) Real-time recognition of sign language gestures and air-writing using leap motion. In: 2017 fifteenth IAPR international conference on machine vision applications (MVA). IEEE, pp 157–160
29. Kumar P, Saini R, Roy PP, Dogra DP (2018) A position and rotation invariant framework for sign language recognition (slr) using kinect. Multimedia Tools and Applications 77(7):8823–8846, https://doi.org/10.1007/s11042-017-4776-9
30. Lan X, Zhang S, Yuen PC, Chellappa R (2017) Learning common and feature-specific patterns: a novel multiple-sparse-representation-based tracker. IEEE Trans Image Process 27(4):2022–2037
31. Lewis M, Simons G, Fennig C (2014) Ethnologue: languages of the world. 7th edn. SIL International
32. Mali, Deepali and Limkar, Nitin and Mali, Satish (2019) Indian sign language recognition using SVM classifier. Proceedings of International Conference on Communication and Information Processing (ICCIP), https://ssrn.com/abstract=3421567
33. Mehrotra K, Godbole A, Belhe S (2015) Indian sign language recognition using kinect sensor. In: International conference image analysis and recognition. Springer, pp 528–535
34. Mnih V, Kavukcuoglu K, Silver D, Rusu AA, Veness J, Bellemare MG, Graves A, Riedmiller M, Fidjeland AK, Ostrovski G et al (2015) Human-level control through deep reinforcement learning. Nature 518(7540):529–533
35. Mohandes M, Deriche M, Johar U, Ilyas S (2012) A signer-independent Arabic sign language recognition system using face detection, geometric features, and a hidden markov model. Comput Electric Eng 38(2):422–433
36. Nagi J, Ducatelle F, Di Caro GA, Cireşan D, Meier U, Giusti A, Nagi F, Schmidhuber J, Gambardella LM (2011) Max-pooling convolutional neural networks for vision-based hand gesture recognition. In: 2011 IEEE International Conference on Signal and Image Processing Applications (ICSIPA). IEEE, pp 342–347
37. Naglot D, Kulkarni M (2016) Ann based indian sign language numerals recognition using the leap motion controller. In: 2016 international conference on inventive computation technologies (ICICT). IEEE, vol 2, pp 1–6
38. Nguyen TN, Huynh HH, Meunier J (2013) Static hand gesture recognition using artificial neural network. Journal of Image and Graphics 1(1):34–38
39. Oyedotun OK, Khashman A (2017) Deep learning in vision-based static hand gesture recognition. Neural Comput Applic 28(12):3941–3951
40. Raghuveera T, Deepthi R, Mangalashri R, et al (2020) A depth-based Indian Sign Language recognition using Microsoft Kinect. Sadhana 45 (34). https://doi.org/10.1007/s12046-019-1250-6
41. Rahaman MA, Jasim M, Ali MH, Hasanuzzaman M (2014) Real-time computer vision-based bengali sign language recognition. 2014 17th International Conference on Computer and Information Technology (ICCIT), pp 192–197
42. Rao GA, Kishore P (2018) Selfie video based continuous indian sign language recognition system. Ain Shams Engineering Journal 9(4):1929 – 1939, https://doi.org/10.1016/j.asej.2016.10.013, http://www.sciencedirect.com/science/article/pii/S2090447917300217
43. Rao GA, Kishore P (2018) Selfie video based continuous indian sign language recognition system. Ain Shams Engineering Journal 9(4):1929–1939
44. Sandler W, Lillo-Martin D (2006) Sign language and linguistic universals. Cambridge University Press. https://doi.org/10.1017/CBO9781139163910
45. Shanableh T, Assaleh K (2011) User-independent recognition of Arabic sign language for facilitating communication with the deaf community. Digital Signal Processing 21(4):535–542
46. Shao R, Lan X, Yuen PC (2018) Joint discriminative learning of deep dynamic textures for 3d mask face anti-spoofing. IEEE Trans Inform Forensics Secur 14(4):923–938
47. Simonyan K, Zisserman A (2014) Very deep convolutional networks for large-scale image recognition. arXiv preprint 14091556

48. Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R (2014) Dropout: a simple way to prevent neural networks from overfitting. Journal of Machine Learning Research 15:1929–1958
49. Sun C, Zhang T, Bao BK, Xu C, Mei T (2013) Discriminative exemplar coding for sign language recognition with kinect. IEEE Trans Cybern 43(5):1418–1428
50. Sun C, Zhang T, Xu C (2015) Latent support vector machine modeling for sign language recognition with kinect. ACM Transactions on Intelligent Systems and Technology (TIST) 6(2):1–20
51. Tripathi K, Nandi NBG (2015) Continuous indian sign language gesture recognition and sentence formation. Procedia Comput Sci 54:523–531
52. Tripathi K, Baranwal N, Nandi GC (2015) Continuous dynamic indian sign language gesture recognition with invariant backgrounds. 2015 International Conference on Advances in Computing. Communications and Informatics (ICACCI). IEEE, pp 2211–2216
53. Tubaiz N, Shanableh T, Assaleh K (2015) Glove-based continuous Arabic sign language recognition in user-dependent mode. IEEE Transactions on Human-Machine Systems 45(4):526–533
54. Uddin MA, Chowdhury SA (2016) Hand sign language recognition for bangla alphabet using support vector machine. 2016 International Conference on Innovations in Science, Engineering and Technology (ICISET), pp 1–4
55. Wadhawan A, Kumar P (2019) Sign language recognition systems: a decade systematic literature review. Archives of Computational Methods in Engineering 28:1–29
56. Wadhawan A, Kumar P (2020) Deep learning-based sign language recognition system for static signs. Neural Computing and Applications 32:1–12
57. Yang W, Tao J, Ye Z (2016) Continuous sign language recognition using level building based on fast hidden Markov model. Pattern Recogn Lett 78:28–35
58. Yasir F, Prasad P, Alsadoon A, Elchouemi A (2016) Sift based approach on bangla sign language recognition. In: 2015 IEEE 8th International Workshop on Computational Intelligence and Applications, IWCIA 2015 - Proceedings, IEEE, Institute of Electrical and Electronics Engineers, United States, pp 35–39, https://doi.org/10.1109/IWCIA.2015.7449458
59. Yu D, Deng L (2010) Deep learning and its applications to signal and information processing [exploratory dsp]. IEEE Signal Processing Magazine 28(1):145–154
60. Zeiler MD (2012) Adadelta: an adaptive learning rate method. 1212.5701
61. Zeiler M, Fergus R (2013) Stochastic pooling for regularization of deep convolutional neural networks. In: ICLR, pp 1–9
62. Zhang XL, Wu J (2012) Deep belief networks based voice activity detection. IEEE Transactions on Audio, Speech, and Language Processing 21(4):697–710