Neural Computing and Applications
https://doi.org/10.1007/s00521-019-04691-y

S.I. : HYBRID ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING TECHNOLOGIES

Deep learning-based sign language recognition system for static signs

Ankita Wadhawan[1] · Parteek Kumar[1]

Received: 3 December 2018 / Accepted: 18 December 2019

Abstract
Sign language for communication is efficacious for humans, and vital research is in progress in computer vision systems. The earliest work in Indian Sign Language (ISL) recognition considers the recognition of significant differentiable hand signs and therefore often selecting a few signs from the ISL for recognition. This paper deals with robust modeling of static signs in the context of sign language recognition using deep learning-based convolutional neural networks (CNN). In this research, total 35,000 sign images of 100 static signs are collected from different users. The efficiency of the proposed system is evaluated on approximately 50 CNN models. The results are also evaluated on the basis of different optimizers, and it has been observed that the proposed approach has achieved the highest training accuracy of 99.72% and 99.90% on colored and grayscale images, respectively. The performance of the proposed system has also been evaluated on the basis of precision, recall and *F*-score. The system also demonstrates its effectiveness over the earlier works in which only a few hand signs are considered for recognition.

Keywords Sign language · Data acquisition · Convolutional neural network · Max-pooling · Softmax · Optimizer

1 Introduction

Sign language is a computer vision-based complete convoluted language that engrosses signs shaped by the movements of hands in combination with facial expressions. It is a natural language used by people with low or no hearing sense for communication. A sign language can be used for communication of letters, words or sentences using different signs of the hands. This type of communication makes it easier for hearing-impaired people to express their views and also help in bridging the communication gap between hearing-impaired people and other person.

✉ Ankita Wadhawan
ankita.wadhawan@thapar.edu

Parteek Kumar
parteek.bhatia@thapar.edu

[1] Computer Science and Engineering Department, Thapar Institute of Engineering and Technology, Patiala, Punjab, India

Humans have been adapting to sign language to communicate since ancient times. Hand gestures are as ancient as the human civilization itself [1]. Hand signs are especially useful to express any word or feeling to communicate. Therefore, people around the world use signals from hand constantly to express despite the formulation of writing conventions.

In recent times, much research has been ongoing in developing systems that are able to classify signs of different sign languages into the given class. Such systems have found applications in games, virtual reality environments, robot controls and natural language communications. At present, the Indian Sign Language systems are in the developing stage and no sign language recognition system is available for recognizing signs in real time. So, there is a need to develop a complete recognizer which identifies signs of Indian Sign Language.

The automatic recognition of human signs is a complex multidisciplinary problem that has not yet been completely solved. In the past years, a number of approaches were used which involve the use of machine learning techniques for sign language recognition. Since the advent of deep learning techniques, there have been attempts to recognize human signs. Networks which are based on deep learning paradigms deal with the architectures and learning algorithms that are biologically inspired, in distinction to conventional networks. Generally, the training of deep networks occurs in a layer-wise manner and depends on more distributed features as present in the human visual cortex. In this, the abstract features from the collected signs in the first layer are grouped into primary features in the second layer, which further combined into more defined features present in the next layer. These features are then further combined together into more engrossing features in the following layers, which help in the better recognition of different signs [2].

The sign language presents a huge variability in postures that a hand can have, which makes this discipline a particularly complex problem. To deal with this, a correct generation of the static postures is necessary. In addition, because each region has specific language grammar, it is required to develop the Indian Sign Language database, which has not been available yet.

Most of the research work in sign language recognition based on deep learning technique is performed on sign languages other than Indian Sign Language. Of recent, this area is gaining popularity among research experts. The earliest reported work on sign language recognition is mainly based on machine learning techniques. These methods result in low accuracy as it does not extract features automatically. The main goal of deep learning techniques is automatic feature engineering. The idea behind this is to automatically learn a set of features from raw data that can be useful in sign language recognition. In this manner, it avoids the manual process of handcrafted feature engineering by learning as a set of features automatically.

There exist many reported research systems related to sign language recognition based on deep learning and machine learning techniques. Nagi et al. [3] proposed a max-pooling CNN for vision-based hand gesture recognition. They employed color segmentation to retrieve hand contour and morphological image processing to remove noisy edges. The experiments were performed on 6000 sign images collected from six gesture classes only and achieved an accuracy of 96%.

Rioux-Maldague and Giguere [4] presented a feature extraction technique for the recognition of hand pose using depth images and intensity images that are captured using Kinect. They employed threshold on the maximum hand depth for segmentation, resize the image and use image centralization for preprocessing. The results were evaluated on known users and unseen users using a deep belief network. The recall and precision of 99% were achieved with known users, 77% recall and 79% precision was achieved with unseen users.

Huang et al. [5] presented a Kinect-based sign language recognition system using 3D convolutional neural networks. They used 3D CNN to capture spatial–temporal features from raw data, which help in extracting authentic features to adapt to the large differences of hand gestures. This model is validated on a real dataset collected from 25 signs with a recognition rate of 94.2%. Huang et al. [6] proposed a real-sense-based sign language recognition system. They collected total 65,000 image frames containing 26 alphabet signs, out of which 52,000 were used for training and 13,000 for testing. The deep neural network model was trained and classified using deep belief network and achieved an accuracy of 98.9% with real-sense and 97.8% with Kinect. Pigou et al. [7] contributed their efforts on Microsoft Kinect and CNN-based recognition system. In this system, they used thresholding, background removal and median filtering for preprocessing. They implemented Nesterov's Accelerated Gradient descent (NAG) optimizer and achieved a validation accuracy of 91.7% to recognize Italian gestures. Molchanov et al. [8] presented a multi-sensor system for gesture recognition of the driver's hand. They calibrate the data received from depth, radar and optical sensors, and use CNN to classify ten different gestures. The experimental results showed that the system achieved the best accuracy of 94.1% using a combination of all three sensors. Tang et al. [9] proposed a hand posture recognition system for sign language recognition using the Kinect sensor. They employed hand detection and tracking algorithms for preprocessing of the captured data. The proposed system is trained on 36 different hand postures using LeNet-5 CNN-based model. The testing has been performed using Deep Belief Network (DBN) and CNN, and it has been found that DBN outperformed CNN with the overall average accuracy of 98.12%.

Yang and Zhu [10] presented a video-based Chinese Sign Language (CSL) recognition using CNN. They collected data using 40 daily vocabularies and showed that the developed method simplifies the hand segmentation method and avoids information loss while extracting features. They used Adagrad and Adadelta optimizers for learning CNN and found that Adadelta outperformed Adagrad. Tushar et al. [11] proposed a numerical hand sign recognition method using Deep CNN. They presented a layer-wise optimized architecture in which batch normalization contributes to faster training convergence and the involvement of the dropout technique alleviates data overfitting. The collected American Sign Language (ASL) images were optimized using Adadelta optimizer of CNN and resulted in an accuracy of 98.50%. Oyedotun and khashman [2] developed a vision-based static hand gesture recognition system for recognizing 24 American Sign Language alphabets. The complete hand gestures were obtained from the publicly available Thomas Moeslund's gesture recognition database. They implemented the CNN network and Stacked Denoising Autoencoders (SDAE) network and achieved an accuracy of 91.33% and 92.83% on testing data, respectively. Bheda and Radpour [12] presented an American Sign Language-based recognition system for letters and digits. The proposed CNN-based architecture consists of three groups of convolutional layers followed by a max-pool layer and a dropout layer and two groups of fully connected layers. The collected images were preprocessed using background subtraction technique and achieved an accuracy of 82.5% on alphabets and 97% on digits using stochastic gradient descent optimizer.

Rao et al. [13] developed a selfie-based sign language recognition system using Deep CNN. They created the dataset which performs 200 signs in different angles and under various background environments. They adopted mean pooling, max-pooling and stochastic pooling strategies on CNN, and it has been observed that a stochastic pooling outperformed other pooling strategies with a recognition rate of 92.88%. Koller et al. [14] proposed the hybrid approach that combines the strong discriminative qualities of CNN with the sequence modeling property of Hidden Markov Model (HMM) for recognition of continuous signs. The collected data have been preprocessed by using a dynamic programming-based approach. It has been observed that the hybrid CNN-HMM approach outperforms the other state-of-the-art approaches.

Kumar et al. [15] proposed a two stream CNN architecture, which takes two color-coded images the joint distance topographic descriptor (JDTD) and joint angle topographical descriptor (JATD) as input. They collected and developed the dataset of 50,000 sign videos of Indian Sign Language and achieved an accuracy of 92.14%.

Based on the requirements mentioned above, this paper aims to develop a complete system based on deep learning models to recognize static signs of Indian Sign Language collected from different users. It presents an effective method for the recognition of Indian Sign Language digits, alphabets and words used in day-to-day life. The deep learning-based convolutional neural network (CNN) architecture is constructed using convolutional layers, followed by other layers. A web camera-based dataset of static signs has been created under different environmental conditions. The performance of the proposed system has been evaluated using different deep learning models, optimizers, precision, recall and *F*-score.

The paper is organized as follows. Section 2 describes the generalized CNN architecture used for classification. The proposed system design and architecture are demonstrated in Sect. 3. Section 4 describes the experimental results and analysis. Finally, the research has been concluded in Sect. 5.

2 CNN architecture components

The objective of CNN is to learn the features present in the data with higher order using convolutions. The CNN architecture works well for the recognition of objects which includes images. They can recognize individuals, faces, street signs and other facets of visual data. There exist a number of CNN variations, but each of them is based on the pattern of layers present, as shown in Fig. 1.

CNN architecture consists of different components which include different types of layers and activation functions. The listing describes the purpose and functioning of some commonly used layers which is discussed below.

Convolutional layer The core building blocks of CNN architecture are the convolutional layer. Convolutional layers (Conv) modify the input data with the help of a patch of neurons connected locally from the previous layer. The dot product will be computed by the layer between the region of the neurons present in the input layer and the weights to which they are locally connected present in the output layer.

A convolution is a mathematical operation that describes the rule for merging two sets of information. The convolution operation takes input, applies a convolution filter or kernel, and returns a feature map as an output as shown in Fig. 2. This operation demonstrates the sliding of the kernel across the input data which produces the convoluted output data. At each step, the input data values are multiplied by the kernel within its boundaries and a single value in the output feature map is created.

Let us suppose the frame size of an input image $W \in R^{wXh}$. The convolutional filter with size F is used for convolution with a stride of S and P padding for input image boundary. The size of the output of the convolution layer is presented by Eq. (1).

$$\text{Output} = \frac{W - F + 2P}{S} + 1 \tag{1}$$

For example, there is one neuron with a receptive field size of $F = 3$, the input size is $W = 128$, and there is zero padding of $P = 1$. The neuron stride across the input in stride of $S = 1$, giving output of size $(128 - 3 + 2)/1 + 1 = 128$.

The output of a convolutional layer is denoted with standardized Eq. (2).

$$a_j^n = f\left(\sum_{i \in C_j} y_i^{n-1} * k_{ij}^n + b_j^n\right) \tag{2}$$

where * is the convolution operation, n represents the *n*th layer, a_j^n is the *j*th output map, y_i^{n-1} represents the *i*th input map in the $(n-1)$th layer, the convolutional kernel is

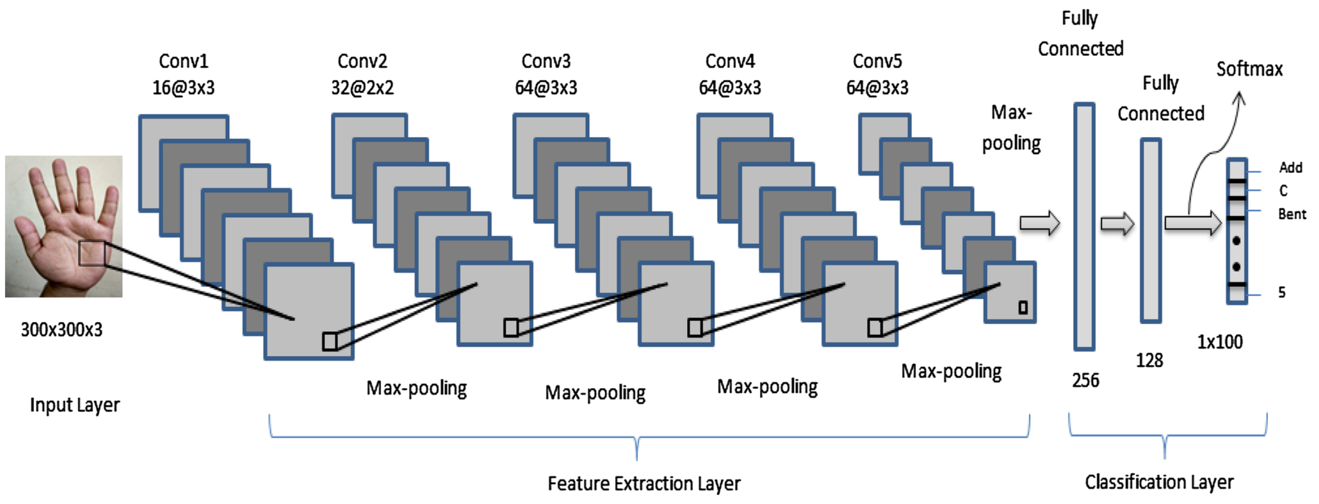

Fig. 1 High-level general CNN architecture

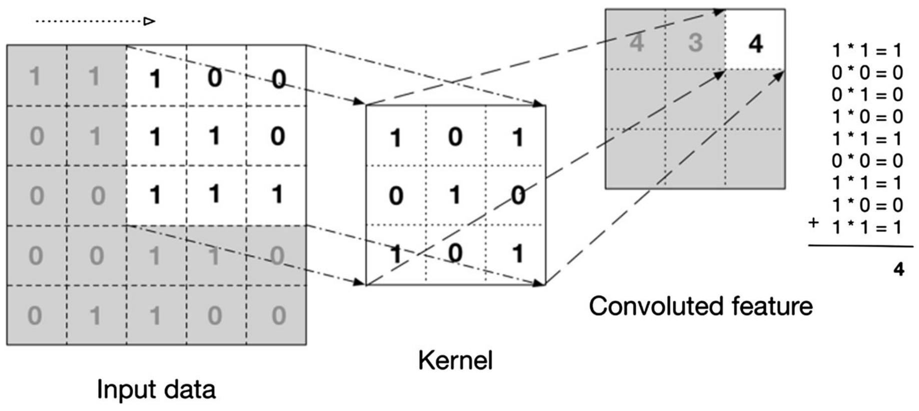

Fig. 2 The convolution operation

represented by k_{ij}, b_j represents bias, C_j is used for representing input maps and f is an activation function [10].

For example, suppose that the input volume has size [128 $\times$ 128 $\times$ 3]. If the filter size is 3 $\times$ 3, then each neuron in the convolution layer will have weights to a [3 $\times$ 3 $\times$ 3] region in the input volume, for a total of 3*3*3 = 27 weights and + 1 bias parameter.

The main objective of other feature extraction layers is to reduce the dimensions of the output generated by convolutional layers. After convolution, the max-method will be used over a region with some specific size for subsampling of feature map. This operation is given by Eq. (3).

$$a_j^n = s\left(a_i^{n-1}\right), \quad \forall i \in V_j \tag{3}$$

where s is the subsampling operation and V_j is the jth region of subsampling in the nth input map [10].

Pooling layer Pooling layers help in reducing the representation of data gradually over the network and control over-fitting. The pooling layer operates in an independent manner on every depth slice of the input. The max () operation used by the pooling layer helps in the resizing of the input data spatially (width, height). This operation is called as *max-pooling*. The down-sampling in this layer has been performed using filters on the input data.

For example, the input volume of size [126 $\times$ 126 $\times$ 16] is pooled with filter size 2, stride 2 into output volume of size [63 $\times$ 63 $\times$ 16].

ReLU layer ReLU stands for Rectified Linear Units. The ReLU layer helps in applying an element-wise activation function over the input data thresholding, for example, $\max(0, x)$ at zero, giving the same dimension output as the input to the layer. The usage of ReLU layers does not affect the receptive field of the convolution layer and at the same time provides nonlinearity to the network. This nonlinear property of the function helps in the better generalization of the classifier. The nonlinear function $f(x)$ used in the ReLU layer is shown in Eq. (4).

$$f(x) = \max(0, x) \tag{4}$$

The sigmoid function and hyperbolic tangent are some other activation functions that can also be used to influence nonlinearity in the network. The usage of ReLU is preferred because the derivative of the function helps back-propagation work considerably faster without making any noticeable difference to generalization accuracy [16].

Fully connected layer/output layer Fully connected layer is used to compute scores of different features for classification. The dimensions of the output volume are $[1 \times 1 \times N]$, where N represents the number of output classes to be evaluated. Each output neuron is connected with all other neurons in the previous layer with different sets of weights. Furthermore, the fully connected layer is a set of convolutions in which each feature map is connected with every field of the consecutive layer and filters consist of the same size as that of the input image [16].

For example, a fully connected layer with $[63 \times 63 \times 16]$ volume and a convolutional layer use filter size 16, giving output volume $[1 \times 1 \times 63{,}504]$.

The final and last layer is the classification layer. As this sign language recognition is a multi-classification problem, softmax function is used in the output layer for classification. Finally, the last fully connected layer with 1000 neurons is used that computes the class scores. Here, 1000 represents the total number of classes in the dataset.

Generally, the CNN architecture consists of four main layers that are a convolutional layer, the pooling layer, the ReLU layer and the fully connected or output layer. The proposed sign language recognition system has been tested on approximately 50 models of CNN by making variations in the hyperparameters such as filter size, stride and padding as presented in Sect. 3. The system has also been tested by changing the number of convolutional and pooling layers. To enhance the effectiveness of the results, one more layer, i.e., dropout layer, is also added in the proposed approach, which is a regularization technique used to ignore randomly selected neurons at the time of training and it helps in reducing the chances of over-fitting.

3 System design and rationale

The proposed sign language recognition system includes four major phases that are data acquisition, image preprocessing, training and testing of the CNN classifier. Figure 3 describes the data flow diagram depicting the working model of the system. The first phase is the data acquisition phase, in which the RGB data of static signs get collected using a camera. The collected sign images are then preprocessed using image resizing and normalization. These normalized images are stored in the data store for future use. In the next phase, the proposed system gets trained using CNN classifier and then the trained model is used to perform testing. The last phase is the testing phase in which the CNN architecture parameters are fine-tuned until the results match the desired accuracy.

3.1 Data acquisition

The three-channel image frames (RGB) are retrieved from the camera, and then these images are passed to the image preprocessing module. The dataset consists of the collection of the RGB images for different static signs. The dataset comprises 35,000 images which include 350 images for each of the static signs. There are 100 distinct sign classes that include 23 alphabets of English, 0–10 digits and 67 commonly used words (e.g., bowl, water, stand, hand, fever, etc.). The dataset consists of static sign images with various sizes, colors and taken under different environmental conditions to assist in the better generalization of the classifier. A few examples from the dataset are shown in Fig. 4.

3.2 Data preprocessing

The data preprocessing is the application of different morphological operations that are used to remove noise from the data. In this phase, the sign images are preprocessed using two methods that are image resizing and normalization. In image resizing, the image is resized to 128×128. These images are then normalized to change the range of pixel intensity values which results in mean 0 and variance 1.

3.3 Model training

The model training is based upon convolutional neural networks. The proposed model is trained using the Tesla K80 Graphical Processing Unit (GPU), 12 GB memory, 64 GB Random Access Memory (RAM) and 100 GB Solid State Drive (SSD). The classifier takes the preprocessed sign images and classifies it into the corresponding category. The classifier is trained on the dataset of different ISL signs. The dataset is shuffled and divided into training and validation set with the size of training set being 80% of the whole dataset. Shuffling the dataset is very significant in terms of adding randomness to the process of neural network training which prevents the network from being biased toward certain parameters. The configuration of the CNN architecture used in the proposed system is described in Table 1.

3.4 Testing

The developed sign language recognition system has been tested on approximately 50 convolutional neural network models. The algorithms with different optimizers are used

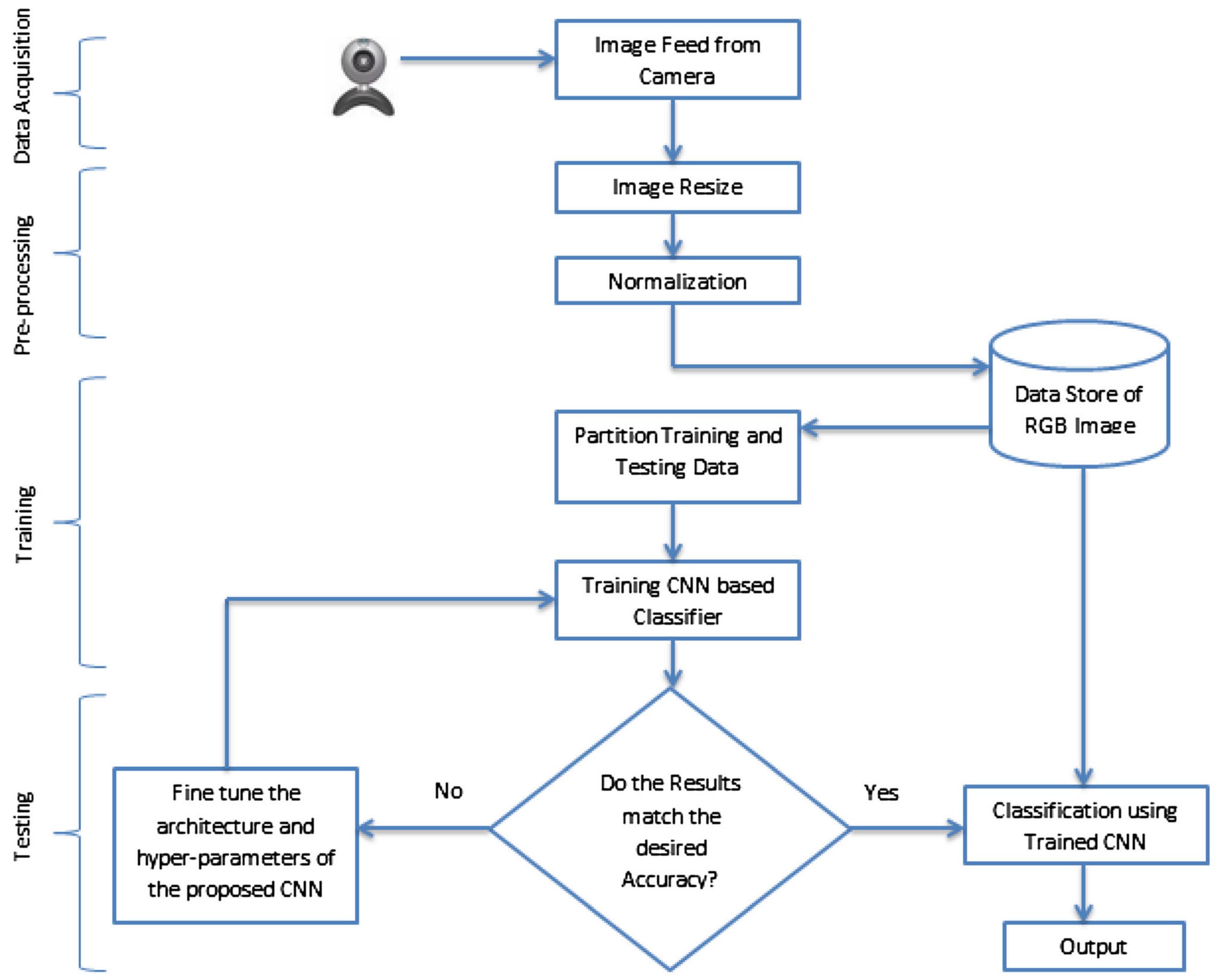
Data Acquisition
Pre-processing
Training
Testing
Image Feed from Camera
Image Resize
Normalization
Data Store of RGB Image
Partition Training and Testing Data
Training CNN based Classifier
Do the Results match the desired Accuracy?
No
Yes
Fine tune the architecture and hyper-parameters of the proposed CNN
Classification using Trained CNN
Output

Fig. 3 System flowchart

to train the network for a maximum of 100 epochs with the loss function as categorical cross-entropy. Some of the other parameters which were used to fine-tune the network architecture based upon the preliminary results and after applying some heuristics to increase the accuracy and find an optimal CPU/GPU computing usage are described in Table 2.

It can be observed from Table 2 that the accuracy of the proposed model gets increased as we limit the number of layers in CNN architecture. The training and validation accuracy get increased to 99.17% and 98.80% by reducing the number of layers from 8 to 4, respectively. On the other hand, the accuracy gets decreased as we alter the number of filters from 16 filters to 32 filters and then to 64 filters with 20 epochs. It has been observed that the recognition rate is high with only 20 epochs.

The optimizers are used to tweak the parameters or weights of the model which helps in minimizing the loss function and predict results as accurate as possible. In this paper, the proposed model is tested on different optimizers such as Adaptive Moment Estimation (Adam), Adagrad, Adadelta, RMSprop and Stochastic Gradient Descent (SGD). The model is trained using AlexNet and Adam as an optimizer and achieved training and validation accuracy of 10% and 5%, respectively. It took a total 4 h to train our model, and it has been observed that the model obtained is highly under-fitted. In the next step, we have reduced the number of layers from 8 to 5 and it has been found that the training and validation accuracy get increased to 42% and 26%, respectively, using Adam as an optimizer and 16 filters. The proposed model achieved the best result with training and validation accuracy of 99.17% and 98.80%, respectively, using total 4 layers, 16 filters and Adam as an optimizer.

The proposed model is tested using different optimizers. Experimental results with respect to optimizers and colored

Fig. 4 Sample dataset

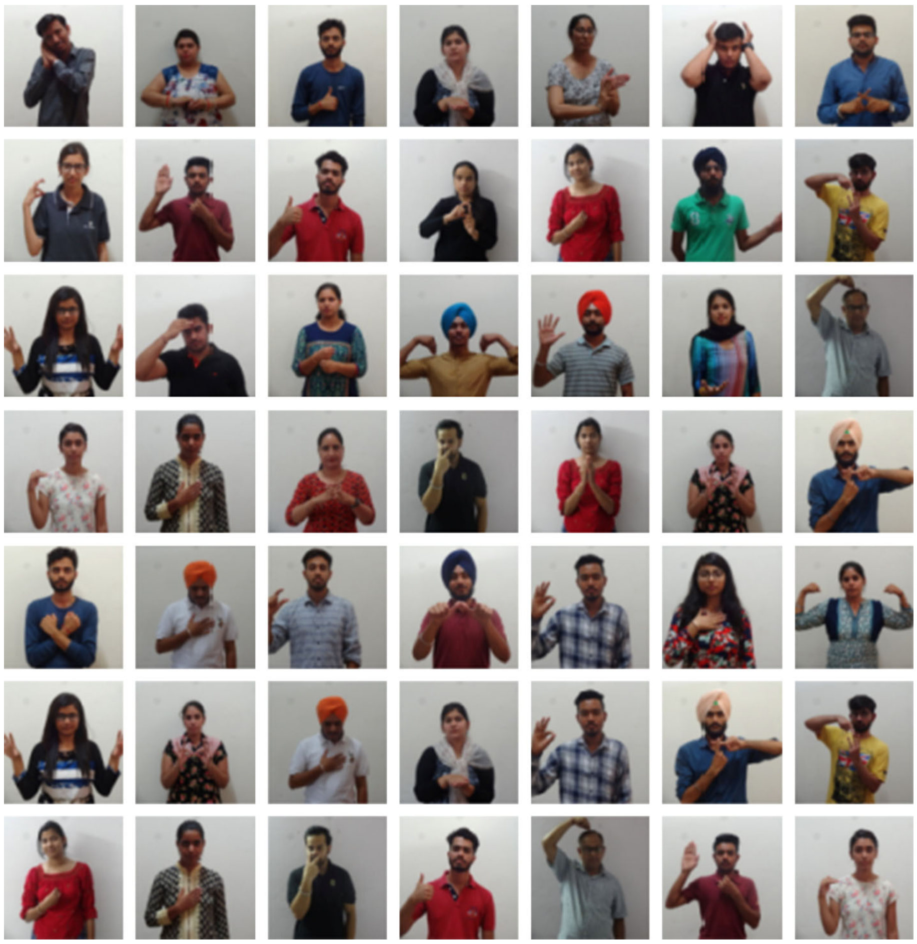

Table 1 Proposed system architecture

Layer type	Output size	Parameters
Input	128 × 128 × 3	–
Conv2d_1	(128, 128, 16)	448
Conv2d_2	(126, 126, 16)	2320
Maxpooling2d_1	(63, 63, 16)	0
Dropout	(63, 63, 16)	0
Flatten	63,504	0
Dense_1 (FC1)	64	4,064,272
Dense_2 (FC2)	100	6500
Total parameters	4,073,540	
Trainable parameters	4,073,540	
Non-trainable parameters	0	

image datasets are represented in Table 3. It has been observed that the SGD outperformed RMSProp, Adam and other optimizers with 16 filters and 4 layers. The proposed model obtained the training and validation accuracy of 99.72% and 98.56% using SGD optimizer, respectively. However, it is the distinct advantage of SGD that it does faster calculations and performs updates more frequently on massive datasets.

The proposed model is also tested on grayscale data. The results obtained with respect to different optimizers, 16 filters, 4 layers and grayscale image datasets are given in Table 4. It has been observed that the model achieved the training and validation accuracy of 99.24% and 98.85%, respectively, using Adam optimizer. The system achieved training and validation accuracy of 99.76% and 98.35%, respectively, using RMSProp and it has been found that the SGD optimizer outperformed Adam, RMSProp and other optimizers with training and validation accuracy of 99.90% and 98.70%, respectively, on grayscale image dataset.

4 Experimentation and results

The performance of the Indian Sign Language recognition system is evaluated on the basis of two different experiments. Firstly, the parameters used in training the model are fine-tuned in which the number of layers, number of filters and optimizers have been changed. In the second experiment, the performance of the trained model is

Table 2 Experimental results with respect to parameters

Number of layers	Number of filters	Training accuracy (%)	Validation accuracy (%)	Number of epochs
8 (5 CL, 3FC)	16	10	5	100
5 (3 CL, 2FC)	16	42	26	100
4 (2 CL, 2FC)	16	99.17	98.80	20
4 (2 CL, 2FC)	32	98.82	98.53	20
4 (2 CL, 2FC)	64	99.05	98.76	20

Table 3 Experimental results with respect to optimizer and colored images

Model	Training accuracy (%)	Training loss	Validation accuracy (%)	Validation loss	Optimizer
I	99.17	0.0280	98.80	0.0684	Adam
II	99.59	0.0378	98.27	0.1940	RMSProp
III	99.72	0.0126	98.56	0.0759	SGD

Table 4 Experimental results with respect to optimizer and grayscale images

Model	Training accuracy (%)	Training loss	Validation accuracy (%)	Validation loss	Optimizer
I	99.24	0.0280	98.85	0.0684	Adam
II	99.76	0.0378	98.35	0.1940	RMSProp
III	99.90	0.0126	98.70	0.0759	SGD

evaluated on color as well as on the grayscale image dataset. The average precision, recall, F_1-score and accuracy of the ISL recognition system have also been computed.

Precision is defined as,

$$\mathrm{TP}/(\mathrm{TP}+\mathrm{FP}) \tag{5}$$

where TP and FP are the numbers of true and false positives, respectively.

The Recall is defined as,

$$\mathrm{TP}/(\mathrm{TP}+\mathrm{FN}) \tag{6}$$

where FN is the number of false negatives

The F_1-score is defined as,

$$2 * \mathrm{Precision} * \mathrm{Recall}/(\mathrm{Precision}+\mathrm{Recall}) \tag{7}$$

The classification performance for some of the grayscale sign samples showing precision, recall and F_1 score is shown in Table 5. The complete description of results for all the signs is given in "Appendix."

The compilation accuracy and loss range from about 12% and 3.623 after the third epoch to 99.90% and 0.012 after the 20th epoch on training data, whereas the validation accuracy and validation loss range from 14 and 3.458 to 98.70% and 0.023 during the first 20 epochs as described in Fig. 5. The early stopping mechanism is also applied in case the validation accuracy stops improving before the completion of maximum of 30 epochs to avoid over-fitting. The training concluded after the 20th epoch due to stagnation in the improvement in validation loss.

4.1 Comparison with existing systems

The comparative analysis of the proposed Indian Sign Language recognition system with other classifiers using our own dataset is shown in Table 6. It has been found that the authors of the existing systems have used machine learning-based techniques for classification, whereas in our methodology we have proposed an Indian Sign Language recognition system using a deep learning-based CNN technique. It has been observed that the proposed Indian Sign Language recognition system outperformed all the other existing ISL systems with an accuracy of 99.90%. It has been also concluded that the CNN convolute structure in large datasets by using the algorithm of backpropagation which indicates how a machine could change its parameters that are used to evaluate the representation in each layer from the representation in the previous layer.

The results of the proposed CNN-based sign language recognition system are best when experimentation was performed with different number of layers in CNN architecture. The rigorous experimentation was also performed to find the optimal parameter values (number of layers, kernel size) for the implementation of the algorithm.

Table 5 Classification performance

Sign	Precision	Recall	F_1-score	Sign	Precision	Recall	F_1-score
A	1.00	0.96	0.98	Me	1.00	1.00	1.00
Afraid	0.97	0.97	0.97	Nose	0.98	1.00	0.99
B	1.00	1.00	1.00	Oath	1.00	1.00	1.00
Bent	0.97	1.00	0.99	Open	1.00	0.97	0.98
Coolie	0.97	0.94	0.96	P	1.00	0.97	0.98
Claw	1.00	1.00	1.00	Pray	1.00	1.00	1.00
D	0.79	0.97	0.87	Q	0.97	1.00	0.99
Doctor	0.98	1.00	0.99	S	0.95	1.00	0.97
Eight	0.96	0.90	0.93	Sick	1.00	1.00	1.00
Eye	1.00	1.00	1.00	Strong	0.97	1.00	0.98
Fever	0.95	1.00	0.97	T	0.99	1.00	0.99
Fist	0.97	0.98	0.97	Tongue	0.99	1.00	0.99
Gun	0.97	1.00	0.99	Trouble	1.00	0.95	0.97
H	1.00	1.00	1.00	U	1.00	0.99	0.99
Hand	0.97	1.00	0.98	V	1.00	1.00	1.00
I	1.00	1.00	1.00	West	1.00	0.93	0.96
Jain	0.99	1.00	0.99	Water	0.93	0.98	0.95

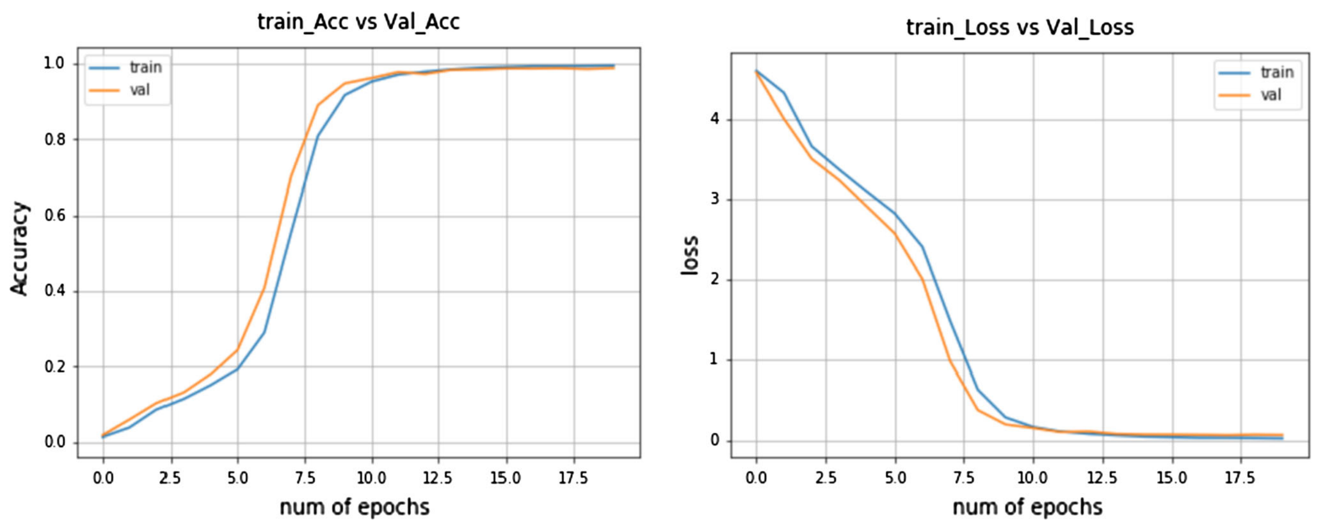

Fig. 5 Accuracy and loss curves for training and validation datasets

Table 6 Comparative analysis of the proposed ISL system with other classifiers

Author	Technique used	Recognition rate (%)
Rahaman et al. [17]	*K*-nearest neighbors	95.95
Uddin and Chowdhury [18]	Support vector machine	97.90
Rao and Kishore [19]	Artificial neural network	98
Proposed system	CNN	99.90

5 Conclusion and future scope

In this research, an effective method for the recognition of ISL digits, alphabets and words used in daily routine is presented. The proposed CNN architecture is designed with convolutional layers, followed by ReLU and max-pooling layers. Each convolutional layer consists of different filtering window sizes which help in improving the speed and accuracy of recognition. A web camera-based dataset of 35,000 images from 100 static signs has been generated under different environmental conditions. The proposed architecture has been tested on approximately 50 deep learning models using different optimizers. The system results in the highest training and validation accuracy of 99.17% and 98.80%, respectively, with respect to change in parameters such as the number of layers and number of filters. The proposed system is also tested using different optimizers, and it has been found that SGD outperformed Adam and RMSProp optimizers with training and validation accuracy of 99.90% and 98.70%, respectively, on the grayscale image dataset. The results of the proposed system have also been evaluated on the basis of precision, recall and *F*-score. It has been found that the system outperformed other existing systems even with less number of epochs.

The major source of challenge in sign language recognition is the capability of sign recognition systems to adequately process a large number of different manual signs while executing with low error rates. For this condition, it has been shown that the proposed system is robust enough to learn 100 different static manual signs with lower error rates, as in contrast to other recognition systems described in other works in which few hand signs are considered for recognition.

For future work, there is a need to collect more datasets to refine the recognition method. Furthermore, the experimentation is ongoing on the trained CNN model to recognize signs in real time. In addition, the system will be extended to recognize dynamic signs which require the collection and development of a video-based dataset and the system is tested using CNN architecture by dividing the videos into frames. A video sequence contains temporal as well as spatial features. Firstly, a hand object is focused to reduce the time and space complexity of network. After that, the spatial features are extracted from the video frames and the temporal features are extracted by relating the video frames in the meantime. The frames of the training set will be given to the CNN model for training process. Finally, the trained model will be used as future reference to make predictions of the training and test data. The work will also be extended to develop a mobile-based application for the recognition of different signs in real time.

Acknowledgements This publication is an outcome of R&D work undertaken in the project under the Visvesvaraya PhD Scheme of Ministry of Electronics and Information Technology, Government of India, being implemented by Digital India Corporation (formerly Media Lab Asia). We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan XP GPU used for this research.

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

Appendix

Grayscale sign samples showing precision, recall and F_1 score

S no.	Sign	Precision	Recall	F_1-score	S no.	Sign	Precision	Recall	F_1-score
1	A	1.00	0.96	0.98	51	Leprosy	1.00	1.00	1.00
2	Afraid	0.97	0.97	0.97	52	M	0.97	0.79	0.87
3	Add	1.00	1.00	1.00	53	Me	1.00	1.00	1.00
4	B	1.00	1.00	1.00	54	N	1.00	1.00	1.00
5	Bottle	1.00	0.97	0.99	55	Nose	0.98	1.00	0.99
6	Bud	1.00	1.00	1.00	56	Nine	1.00	1.00	1.00
7	Bent	0.97	1.00	0.99	57	Nurse	0.99	0.98	0.98
8	Between	1.00	0.99	0.99	58	O	0.96	0.97	0.96
9	Blind	1.00	1.00	1.00	59	Oath	1.00	1.00	1.00
10	Bowl	0.97	1.00	0.98	60	One	1.00	0.99	0.99
11	Brain	0.99	0.99	0.99	61	Open	1.00	0.97	0.98
12	C	1.00	1.00	1.00	62	Owl	1.00	1.00	1.00
13	Coolie	0.97	0.94	0.96	63	P	1.00	0.97	0.98
14	Cough	1.00	1.00	1.00	64	Policy	0.98	1.00	0.99
15	Cow	0.99	0.97	0.98	65	Pray	1.00	1.00	1.00
16	Chest	1.00	1.00	1.00	66	Promise	1.00	1.00	1.00
17	Claw	1.00	1.00	1.00	67	Q	0.97	1.00	0.99
18	D	0.79	0.97	0.87	68	R	0.98	1.00	0.99
19	Devil	0.98	0.94	0.96	69	S	0.95	1.00	0.97

S no.	Sign	Precision	Recall	F_1-score	S no.	Sign	Precision	Recall	F_1-score
20	Doctor	0.98	1.00	0.99	70	Seven	0.96	1.00	0.98
21	E	1.00	1.00	1.00	71	Soldier	1.00	1.00	1.00
22	East	1.00	1.00	1.00	72	Shirt	1.00	1.00	1.00
23	Eight	0.96	0.90	0.93	73	Six	1.00	0.97	0.99
24	Evening	1.00	0.96	0.98	74	Sick	1.00	1.00	1.00
25	Elbow	0.95	1.00	0.98	75	Skin	1.00	1.00	1.00
26	Eye	1.00	1.00	1.00	76	Shoulder	1.00	0.98	0.99
27	F	0.97	0.97	0.97	77	Stand	1.00	1.00	1.00
28	Fat	1.00	1.00	1.00	78	Strong	0.97	1.00	0.98
29	Faith	0.98	1.00	0.99	79	Sleep	1.00	0.95	0.98
30	Fever	0.95	1.00	0.97	80	Sunday	1.00	1.00	1.00
31	Feel	0.97	1.00	0.98	81	T	0.99	1.00	0.99
32	Few	1.00	1.00	1.00	82	Ten	1.00	0.98	0.99
33	Food	0.98	0.86	0.92	83	Telephone	1.00	1.00	1.00
34	Four	1.00	0.96	0.98	84	Tongue	0.99	1.00	0.99
35	Fist	0.97	0.98	0.97	85	Thorn	0.96	1.00	0.98
36	Five	1.00	1.00	1.00	86	Three	0.97	1.00	0.99
37	G	0.97	0.97	0.97	87	Trouble	1.00	0.95	0.97
38	Gun	0.97	1.00	0.99	88	Two	1.00	1.00	1.00
39	Good	1.00	1.00	1.00	89	U	1.00	0.99	0.99
40	H	1.00	1.00	1.00	90	V	1.00	1.00	1.00
41	Hair	1.00	1.00	1.00	91	W	1.00	1.00	1.00
42	Hand	0.97	1.00	0.98	92	West	1.00	0.93	0.96
43	Head	0.90	0.99	0.94	93	Wedding	0.99	0.99	0.99
44	Hear	0.97	1.00	0.98	94	Water	0.93	0.98	0.95
45	I	1.00	1.00	1.00	95	White	1.00	0.98	0.99
46	Jain	0.99	1.00	0.99	96	X	1.00	1.00	1.00
47	K	0.98	0.98	0.98	97	Y	1.00	1.00	1.00
48	King	1.00	0.95	0.97	98	You	0.97	1.00	0.99
49	L	1.00	1.00	1.00	99	Z	0.99	1.00	0.99
50	Love	1.00	0.98	0.99	100	Zero	1.00	1.00	1.00

References

1. Corballis MC (2003) From mouth to hand: gesture, speech and the evolution of right-handedness. Behav Brain Sci 26(2):199–208
2. Oyedotun OK, Khashman A (2017) Deep learning in vision-based static hand gesture recognition. Neural Comput Appl 28(12):3941–3951
3. Nagi J, Ducatelle F, Di Caro GA, Cireşan D, Meier U, Giusti A, Gambardella LM (2011) Max-pooling convolutional neural networks for vision-based hand gesture recognition. In: IEEE international conference on signal and image processing applications (ICSIPA), pp 342–347
4. Rioux-Maldague L, Giguere P (2014) Sign language fingerspelling classification from depth and color images using a deep belief network. In: IEEE Canadian conference on computer and robot vision (CRV), pp 92–97
5. Huang J, Zhou W, Li H, Li W (2015) Sign language recognition using 3D convolutional neural networks. In: IEEE international conference on multimedia and expo (ICME), pp 1–6
6. Huang J, Zhou W, Li H, Li W (2015) Sign language recognition using real-sense. In: IEEE China summit and international conference on signal and information processing (ChinaSIP), pp 166–170
7. Pigou L, Dieleman S, Kindermans PJ, Schrauwen B (2014) Sign language recognition using convolutional neural networks. In: Workshop at the European conference on computer vision. Springer, Cham, pp 572–578
8. Molchanov P, Gupta S, Kim K, Pulli K (2015) Multi-sensor system for driver's hand-gesture recognition. In: 11th IEEE international conference and workshops on automatic face and gesture recognition (FG), vol 1, pp 1–8
9. Tang A, Lu K, Wang Y, Huang J, Li H (2015) A real-time hand posture recognition system using deep neural networks. ACM Trans Intell Syst Technol (TIST) 6(2):21

10. Yang S, Zhu Q (2017) Video-based Chinese sign language recognition using convolutional neural network. In: IEEE 9th international conference on communication software and networks (ICCSN), pp 929–934
11. Tushar AK, Ashiquzzaman A, Islam MR (2017) Faster convergence and reduction of overfitting in numerical hand sign recognition using DCNN. In: Humanitarian technology conference (R10-HTC), IEEE Region 10, pp 638–641
12. Bheda V, Radpour D (2017) Using deep convolutional networks for gesture recognition in American sign language. arXiv preprint arXiv:1710.06836
13. Rao GA, Syamala K, Kishore PVV, Sastry ASCS (2018) Deep convolutional neural networks for sign language recognition. In: IEEE conference on signal processing and communication engineering systems (SPACES), pp 194–197
14. Koller O, Zargaran S, Ney H, Bowden R (2018) Deep sign: enabling robust statistical continuous sign language recognition via hybrid CNN-HMMs. Int J Comput Vis 126(12):1311–1325
15. Kumar EK, Kishore PVV, Kiran Kumar MT (2019) 3D sign language recognition with joint distance and angular coded color topographical descriptor on a 2—stream CNN. Neurocomput 372:40–54
16. Prabhu R (2018) Understanding of convolutional neural network (CNN) — deep learning. https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148. Accessed 4 Mar 2018
17. Rahaman MA, Jasim M, Ali MH, Hasanuzzaman M (2014) Real-time computer vision-based Bengali Sign Language recognition. In: 17th IEEE international conference on computer and information technology (ICCIT), pp 192–197
18. Uddin MA, Chowdhury SA (2016) Hand sign language recognition for Bangla alphabet using support vector machine. In: IEEE international conference on innovations in science, engineering and technology (ICISET), pp 1–4
19. Rao GA, Kishore PVV (2017) Selfie video based continuous Indian sign language recognition system. Ain Shams Eng J 9(4):1929–1939