A deep learning approach for Indian sign language gestures classification with different backgrounds

A deep learning approach for Indian sign language gestures classification with different backgrounds

R. Dhiman[1], G. Joshi[2] and C. Rama Krishna[3]
[1]PG Scholar, Department of Computer Science & Engineering, NITTTR, Chandigarh, INDIA.
[2]Assistant Professor, Department of Electronics & Communication Engineering, UIET, Panjab University, Chandigarh, INDIA
[3]Professor & Head, Department of Computer Science & Engineering, NITTTR, Chandigarh, INDIA.

E-mail: richadhiman41@gmail.com

Abstract. The sign language recognition system recently has drawn the attention of various researchers as there is no universal sign language, moreover, it consists of many patterns and postures**.** Many methods for extracting features and classifying sign language have been proposed in the literature, most of them are based on machine learning techniques. In this article, a deep learning method has been adopted by designing a Convolution Neural Network (CNN) model to extract the sign language features where for classification softmax layer is used. All alphabets in simple as well as complex backgrounds have been considered, where data is collected from 100 subjects in different lighting conditions. The effect of various optimization techniques (Adam, Sgdm, RMSProp), activation functions (ReLU and Leaky ReLU) for generalization ability is also observed. The proposed approach has succeeded in attaining the testing accuracy of 99.10%, 92.69%, and 95.95% on the Indian sign language dataset with simple, complex backgrounds and on a mixed background scenarios, respectively. The model is also tested on NUS dataset-I, NUS dataset-II, their combination, and achieved the accuracy of 100%, 95.95%, and 97.22% respectively.
Keywords: Neural network; sign; language; algorithm; gesture.

1. Introduction
Gestures are the movement of the part of the body to conveying the message or communicating with others. With the support of these movements, anyone can effectively communicate their feelings and thoughts. Literature indicates that humans can communicate non-verbally using gestures [1]. Speech and hearing-impaired individuals use sign language for conversation. Sign language communication incorporates hand, body movement, arms movement, and facial expressions to communicate the feelings of a speaker. Due to a lack of formal training, sign language is not widely understood. Therefore, smooth communication is absent between the hearing-speech impaired and others who don't know sign language. To bridge the communication gap between society and deaf-dumb, a translator is essential. An advancement in human-computer interaction systems motivated scholars to research in the area of an automatic gesture detection system. Automatic sign recognition is a complicated process that has not been fully figured out yet.

The Indian Sign Language (ISL) systems are currently undergrowth, and there is no sign language recognition framework available to interpret gestures in real-time. As compared to single hand sign languages such as American Sign language, gestures in ISL are mostly double hand signs and are complex in shape. Researches with deep learning methods in sign language classification are conducted in other languages. So, there is a need to work on gesture recognition of Indian Sign Language.

To recognize gestures variety of methods have been used over the years which mostly include machine learning approaches but recently paradigm shift toward deep learning methodology has been reported. One of deep learning's main dominance over machine learning algorithms is its efficiency to execute feature engineering on its own. In machine learning techniques, it takes a large time to analyze the data and then extract useful features. These approaches serve a low recognition rate, as they do not automatically derive features. In contrast, deep learning algorithms scan the data to look for associated features and merge them to efficient learning without being expressly instructed to do so [2]. Thus, deep learning-based gesture recognition is explored.

In this paper, various Convolution Neural Network (CNN) parameters have been explored to propose a CNN model with the best accuracy. Moreover, most of the works in ISL as reported in the literature are done with simple background images but in this simple as well as complex background images have been considered. Then two activation functions i.e. Relu, Leaky Relu. Similarly, three optimizers were explored i.e. Adam, Sgdm, RMSprop.

2. Literature survey

There are many works documented in the literature that focus on machine learning-based recognition of sign language. Badhe and Kulkarni designed an algorithm for Indian sign language to text converter. Signs of numbers, alphabets, and some phrases were collected from different signers. The authors collected 130,000 videos for the database; from these 72,000 videos were utilized for the training and the rest of the videos were utilized for testing the system performance. An accuracy of 97.5% was achieved by deploying the algorithm [3]. Uddin and Chowdhury used a machine learning approach to recognize Bangla sign language vowels and consonants collected from 30 signers. For classification, a Support Vector Machine (SVM) was used and the framework achieved a recognition rate of 97.7% [4]. Kaur *et al.* proposed a system that recognized Indian sign language 26 alphabets in uniform background collected from 72 subjects. For deploying the machine learning approach, features were extracted using Krawtchouk Moments (KM). Finally, classification was done with a multilayer perceptron (MLP), SVM, and Extreme Learning Machines (ELM) with the usage of different kernels and compared. Out of MLP, SVM, and ELM, the best accuracy of 97.9% was achieved with SVM [5]. Rao and Kishore proposed a system that recognized 18 dynamic words of Indian sign language collected from 10 signers. They analyzed videos that were captured from the front camera of mobile phone and performed feature extraction with Discrete Cosine Transform (DCT) along with Principle Component Analysis (PCA) technique. They achieved a word matching score of 90% by using an Artificial Neural Network (ANN) [6]. Joshi *et al.* proposed a discriminant correlation analysis-based fusion of shape features and for classification, multiclass SVM was used. For all alphabets taken from 72 signers, accuracy stands at 98.6% for the uniform background using a combination of Zernike Moments (ZM) and KM and 83.8% for the complex background using Dual Hahn Moments DHM and KM [7]. Lahiani and Neji proposed a fusion of two feature extraction techniques, namely, HOG and LBP for NUS dataset-I. They recognized static hand gestures in complex backgrounds with gentle AdaBoost with a detection rate of 91.6% [8] Zhang *et al.* proposed a gesture recognition system where features were extracted from the Histogram of Oriented Gradients (HOG), Local Binary Patterns (LBP) combination. They used the SVM model for classification. Two NUS hand posture dataset-II sub-datasets were included. First having 2000 hand gesture color images and second having 750 hand gesture color images and achieved accuracy of 97.8% and 95.07% respectively [9].

Furthermore, deep learning-based systems are also reported in the literature. Das *et al.* used the inception v3 model with 48 layers to recognize 24 static American sign language alphabets with a simple background. There were on an average 100 images per class and augmentation was done using cropping, scaling, rotating, and flipping. The results showed a consistently high accuracy rate of over 90% [10]. Yang and Zhu proposed a CNN model for the recognition of 40 vocabularies of Chinese Sign Language. Two optimizers were analyzed, namely, Adadelta and Adagrad, and the authors found that Adadelta outperformed Adagrad on their dataset with an accuracy of 99%. [11]. Tushar *et al.*

proposed a Deep CNN model that was used to recognize American sign languages with numerical hand gestures having 500 images in each class. They used batch normalization which was used for fast convergence. Moreover, dropout was used to avoid overfitting. Adadelta optimizer was analyzed and resulted in an accuracy of 98.50% [12]. Bheda and Radpour proposed CNN based model for American sign language recognition which included all alphabets and digits collected from 5 signers. In this CNN architecture, three convolution layers were used and by using stochastic gradient descent optimizer, the accuracy achieved was 82.5% for alphabets and 97% for digits [13]. Sajanraj and Beena developed a system to recognize the numerals (0-9) in Indian sign language. The system was trained using 3000 static images and for testing 100 images of each class were used. Region-based CNN (R-CNN) attained an accuracy of 99.56% for the same subject and the accuracy decreased to 97.26% for low lighting conditions [14]. Rao *et al.* proposed work to classify Indian Sign Language using selfie videos consisting of 200 words collected from 5 different signers and analyzed from 5 different angles. The architecture with 4 convolution layers and 2 stochastic pooling layers achieved an accuracy of 92.88% [15]. Athira *et al.* developed a system to identify Indian sign language fingerspelling alphabets and a few dynamic words collected from seven people. With multiclass SVM, accuracy attained was 91% for fingerspelled gestures and 89% for single-handed complex dynamic words [16]. Wadhawan and Kumar proposed a CNN model to classify 23 Indian sign language alphabets, 0-10 numbers and 67 commonly used words in complex backgrounds collected from few signers and achieved validation accuracy of 98.70% [17]. Kumar *et al.* proposed a system where two color-coded images- the joint topographic descriptor distance and the joint topographic descriptor angle were given to a two-stream CNN architecture. The dataset of 50,000 videos was collected from 10 signers and the authors achieved 92.14% accuracy [18]. Gaikwad *et al.* proposed a CNN architecture using transfer learning having multiple convolution and dense layers. In this, the authors focused on all English alphabets and numbers of American Sign Language. They classified the gestures and converted them into sound [19].

From the literature survey, it is evident that in the earlier work, the area of the hand is first segmented using the skin-color technique, but these methods are prone to variations of illumination. Much of the study in ISL relies on simple backgrounds but for efficient recognition of sign language, there is a need to work with complex backgrounds. Thus, there is a requirement to develop a signer independent vision-based gesture recognition system which is capable of recognizing hand gestures in the variation of background.

Based on the specifications outlined above the purpose of this paper is to recognize static Indian sign language gestures gathered from various subjects by developing a deep learning-based model. To evaluate the performance of the system, accuracy has been taken as the main parameter. Various CNN parameters have been explored to propose an efficient model with the best accuracy. Precision, recall, and F1-score have been analyzed to study the response of the model to each alphabet.

3. Overview of CNN

Convolutional Neural Network (CNN) extract features from the data using convolutions. CNN architectures performed well in image classification problems such as the face, traffic signs, individuals, etc. The architecture of the CNN depends upon the applications. It consists of various layer forms, activation functions, etc. The functionality of some widely used layers is described as follows:

3.1 Convolutional layer:

It is the elementary building block of the CNN Architecture. The input image is first passed through the convolutional layer which produces the feature map after convolving filter or kernel of a particular size. In the convolution process, the kernel values multiply with the input image values of the same size of the filter and produce a single value in the feature map as shown in Figure 1.

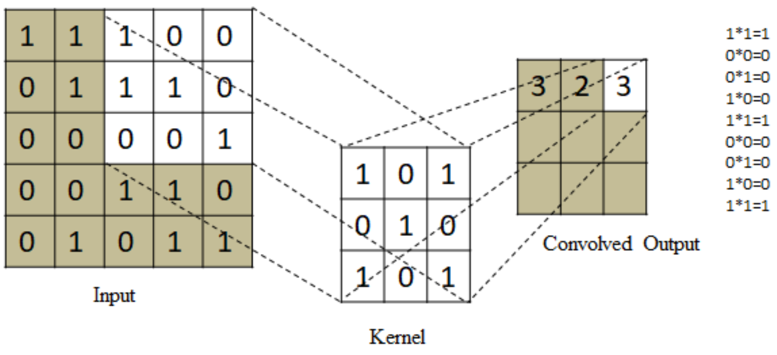

Figure 1. The process of convolution of kernel over the image.

Eq. (1) is the formula to calculate the output size of an image after a convolution layer where *W* denotes the image size, *F* is the filter size considered for convolution and *S* is stride and *P* is padding used if required.

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

For example, suppose image size *W* = 128, Filter size *F* = 5, stride considered *S* = 1 and padding *P* = 1. Then the output according to the formula will be (128 - 5 + 2*1)/1 + 1 = 125.
The performance of the convolution layer is seen by the standardized Eq. (2) as follows:

$$a_n^m = f(\sum_{i \in l_n} y_l^{m-1} * k_{in}^m + b_n^m) \tag{2}$$

where *m* represents the mth layer, a_n^m is the nth output map, *f* is an activation function, * is the convolution operation, the convolutional kernel is represented by k_{in}, y_l^{m-1} means the 1th input map in the (m-1)th layer, b_n represents bias and l_n is used for representing input maps [11].

3.2 Pooling layer:

This layer helps in scale down the dimensions (width, height) or resizing the data with the help of filters and restraint over-fitting. Pooling can be min-pooling where the minimum values are considered from the feature map, max-pooling here maximum values are considered. The key purpose of this phase is to reduce the dimensions of the output of the convolution layer. For example, when an input image of size [128*128*8] and max-pooling is applied with kernel size 2*2 and stride 2, results in [64*64*8] sized output image. This process is denoted by Eq. (3).

$$a_n^m = s(a_i^{m-1}), ¥_i € V_n \tag{3}$$

where *s* is the sub-sampling process and V_n is the nth sampling area in the ith input map [11].

3.3 Activation layer:

After the convolution layer, next comes is the activation layer where ReLU (Rectified Linear Units) activation function is applied. Mathematically, the interpretation is y = max (0, x) means it output the input value if greater than zero else output zero. It is a commonly used activation function, since the network that uses it is easier to train and thus performs better. Some other activation functions are the sigmoid function and the hyperbolic tangent which can also be used to influence the nonlinearity of the network.

ReLU use is favoured because the derivative function makes backpropagation work substantially faster without having some major variation to accuracy. A variant of ReLU known as leaky ReLU is also considered in this work.

3.4 Fully-connected layer:
The CNN cycle starts with convolution and pooling to convert images into the feature set. The product of this process goes to a fully connected neural network structure which derives the final decision on the classification. The final output features map has [1*1*N] features, where *N* represents the number of classes, it is 26 in this case. Every output neuron in the preceding layer is connected to all other neurons with different weights [20]. For example, the convolution layer gives output [64*64*8] where 8 is the number of filters used. Then it will be fed to a fully connected layer and the output volume will be [1*1*32,768].

3.5 Classification:
As this is a multiclass classification problem, for that softmax function is used. The last 26 neurons in the fully connected layer calculate the class score. Finally, the output softmax layer with N neurons is used to measure class ratings. The highest score or rating of a class is considered a classified class.

3.6 Dropout layer:
The proposed system also includes a dropout layer. The dropout layer is used for the regularization purpose where neurons are randomly selected to reduce the overfitting problem. Dropout with 50% probability to randomly drop some neurons to avoid overfitting problem.

4. Proposed gesture recognition system
After discussing the CNN model, this section proposes a gesture recognition system that includes phases such as data acquisition, designing of the CNN model by analyzing various parameters, and training and testing of the CNN model. The flow diagram in Figure 2 depicts the steps undertaken to implement the deep learning-based architecture for the ISL recognition system. First, the dataset of Indian alphabets was acquired. Based on the preliminary study and experience on the machine learning architectures, during the pre-processing step, the input image is resized to 128*128 size. To raise the number of images in the dataset and introduce the variability in the dataset, data augmentation is done. Here, augmentation is done by rotating the image by 20 degrees and translating the image both in the x and y direction by 50,50. So the original 100 images per class are augmented by a factor of three resulting in a total of 7800 images for ISL uniform and complex datasets each. These are combined to create ISL mixed background dataset with a total number of 15600 images after data augmentation. The dataset is partitioned into 10% validation set, 10% test set, and 80% training set and is also shuffled. Then training and testing of the proposed CNN model are done by analyzing various parameters. These steps of the proposed gesture-based recognition system are discussed in the following:

4.1. Sign language datasets:
The proposed model in this paper is focussed primarily on Indian sign language datasets referred to as ISL datasets in the following sections. To estimate the effectiveness of the proposed model, it has been validated on publicly available NUS dataset-I and NUS dataset-II.

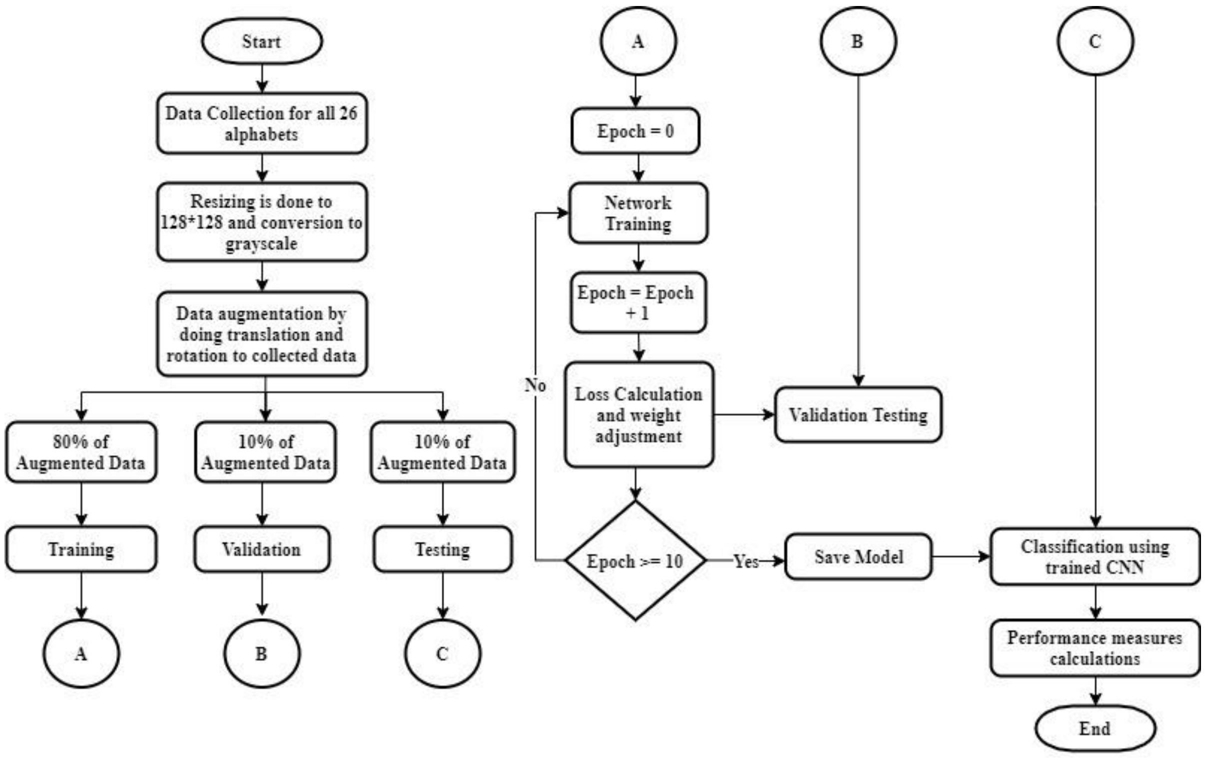
Start
Data Collection for all 26 alphabets
Resizing is done to 128*128 and conversion to grayscale
Data augmentation by doing translation and rotation to collected data
80% of Augmented Data
10% of Augmented Data
10% of Augmented Data
Training
Validation
Testing
A
B
C
A
Epoch = 0
Network Training
Epoch = Epoch + 1
Loss Calculation and weight adjustment
No
Epoch >= 10
Yes
Save Model
B
Validation Testing
C
Classification using trained CNN
Performance measures calculations
End

Figure 2. Flow Diagram of the proposed model for gesture recognition.

i. ISL uniform background dataset: This dataset is composed of Indian sign gestures of 26 alphabets that acquired on a uniform background at Panjab University, Chandigarh, India. It originally consists of 90 subjects, which is extended to 100 subjects in this work resulting in 2600 images that were taken [7]. Sample images are shown in Figure 3. These are augmented to 7800 images.

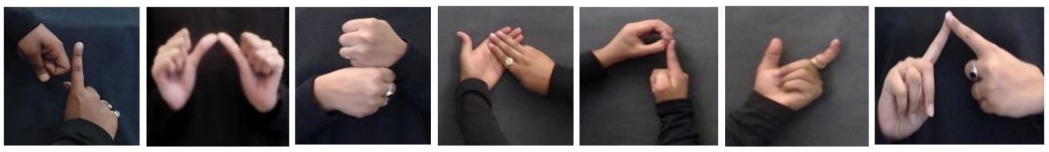

Figure 3. ISL uniform background dataset samples.

ii. ISL Complex Background Dataset: The ISL uniform background dataset was used to develop synthetic images by superimposing uniform background images on five complex backgrounds. As created from the ISL uniform background dataset, it consists of 2600 images, 100 images of each alphabet [7]. Sample images of the dataset are shown in Figure 4. These are augmented to 7800 images.

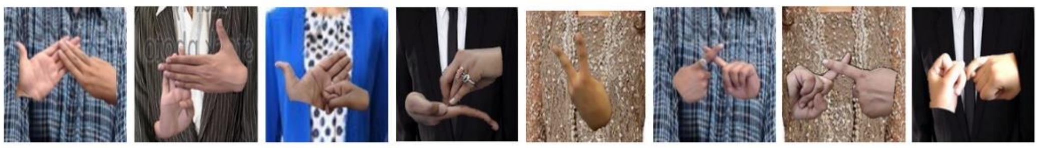

Figure 4. ISL complex background dataset samples.

iii. ISL mixed background dataset: To include sufficient variability in the dataset, the above-mentioned uniform, as well as complex background datasets, were combined to create an ISL mixed background dataset comprising of a total number of 5200 images with 200 images of 26 ISL alphabets each. These are augmented to resulting in 15600 images.

iv. NUS hand posture datasets I and II: It is publicly available which has been created by the National University of Singapore (NUS). NUS dataset-I consists of 10 classes of hand gestures with a uniform background [8]. NUS Dataset-II also consists of 10 hand gestures but with complex backgrounds [9]. Samples of both are shown in Figure 5. and Figure 6.

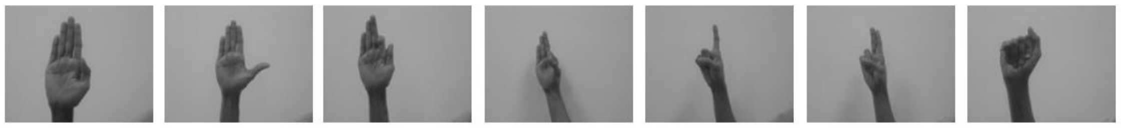
Figure 5. NUS-I uniform background dataset samples

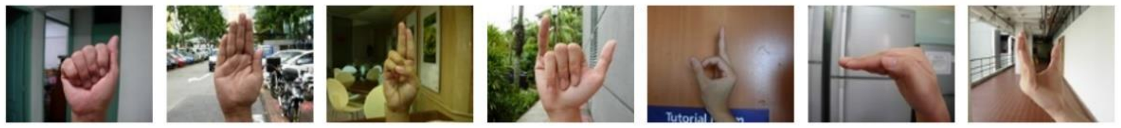
Figure 6. NUS-II complex background dataset samples.

4.2. Proposed CNN design
After collecting the dataset in the previous step, this step implements the proposed CNN model. The CNN model in the proposed work has an input layer, 5 convolution layers, 3 max-pool layers after the first 3 convolution layers, a fully connected layer, and last layer is a softmax layer as shown in Figure 7 The proposed model was selected after changing various parameters such as filter size, number of filters in convolution layers, batch size, epochs, validation frequency, and learning rate. Table 1 shows experimental results by varying various parameters to propose five convolution layers in CNN architecture.

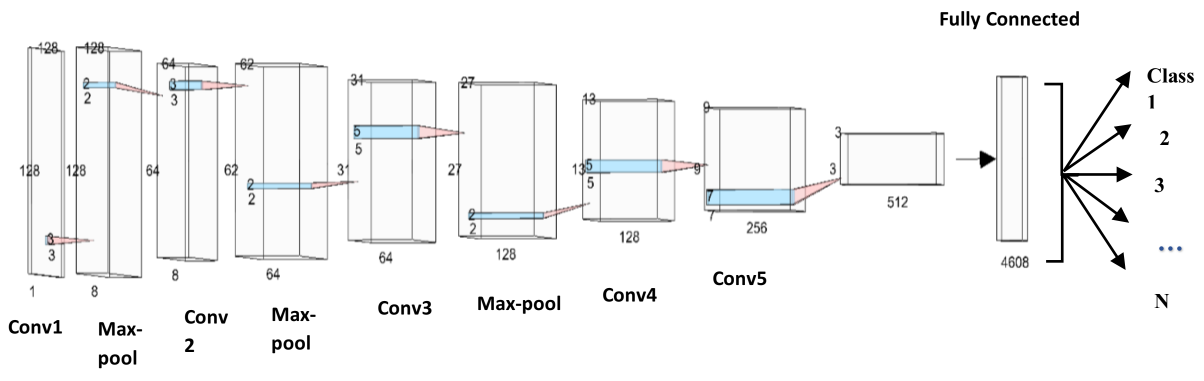

Figure 7. Proposed CNN Model.

CNN architecture is dependent upon input image size. The larger the image size, the larger the architecture will be and hence more time will be taken to train the network. During preliminary experiments, 128*128 and 64*64 image sizes were studied but 128*128 image size provided more appropriate patterns to extract the suitable convolution-based features. Thus, in the work presented in this paper, 128*128 image size is used. Other parameter variations such as filter size, number of filters, strides as listed in Table 1 have also been studied.

Table 1. Experimentation results by varying parameters

Parameters	Values	Training Accuracy %	Validation Accuracy %	Testing Accuracy %
Filter Sizes in 5 Convolutional layers	[3*3,3*3,3*3,3*3,3*3]	99.94	97.18	97.95
	[3*3,3*3,3*3,3*3,5*5]	100	99.23	98.21
	[3*3,3*3,3*3,5*5,7*7]	99.98	99.36	98.72
	[3*3,3*3,5*5,3*3,3*3]	99.95	97.56	97.82
	[3*3,3*3,5*5,5*5,4*4]	100	99.23	98.97
	[3*3,3*3,5*5,5*5,7*7]	**100**	**99.36**	**99.10**
	[3*3,5*5,3*3,5*5,3*3]	100	98.97	98.85
	[3*3,5*5,3*3,5*5,5*5]	99.55	97.18	97.69
No. of filters in each convolutional layer	[3,64,128,256,512]	85.79	84.74	82.95
	[6,32,64,256,512]	100	99.49	98.85
	[8,64,128,256,512]	**100**	**99.36**	**99.10**
Batch Size	100	99.95	98.97	98.46
	128	**100**	**99.36**	**99.10**
	200	100	99.36	98.85
Epochs	8	99.76	98.97	98.46
	10	**100**	**99.36**	**99.10**
	15	100	99.36	99.10
Learning Rate	0.1	77.13	77.56	77.56
	0.01	98.41	96.41	95.77
	0.001	**100**	**99.36**	**99.10**
	0.0001	100	98.21	98.08
Validation Frequency	5	100	98.85	98.46
	10	**100**	**99.36**	**99.10**
	15	100	99.36	98.97

Various kernel sizes have been tried in convolution layers to reach the final selection of [3*3, 3*3, 5*5, 5*5, 7*7] in each layer, respectively. Filter size in subsequent layers is increased since the pattern gets more complex in subsequent layers. The number of filters in each layer are selected as [8, 64, 128, 256, 512], respectively**.** Batch sizes of 100, 128, 200, 300 have been tried and the best result was found with 128. Similarly, the number of epochs 8, 10, 15 are used and 10 found best as after 10 if we were increasing epochs it didn't show any accuracy up-gradation. Different learning rates analyzed are 0.1, 0.01, 0.001, 0.0001, and 0.001 is found to be the best. If the learning rate becomes too high, gradient descent can unintentionally increase rather than decrease the training error. If the learning pace is so poor, learning is not only harder but could be forever left with a high training error. The validation frequency selected is 10 because by using 5, 15 accuracies decreased.

After analyzing these parameters the CNN model was proposed which involves multiple types of layers such as convolutional, pooling layer, dropout layer, etc., different activation functions are shown in Table 2. The padding is used in the first convolution layer. In convolution layers stride considered is 1*1. In between convolution and activation function batch normalization is present. The Batch Normalization layer is used to improve network learning. It normalizes the activation by eliminating the mini-batch mean and then dividing it by the standard deviation of the mini-batch [21]. Further 2 activation functions ReLU and Leaky ReLU are analyzed and found ReLU outperformed Leaky ReLU for this dataset. The ReLU activation function is a simple calculation that directly returns the specified value as input, or the value 0.0 if the input is 0.0 or less. Max pooling is used after the first, second, and third ReLU layer. This reduces the dimensions of the image and so does the feature size. In max-pooling layers, kernel size used is 2*2 and stride is also 2*2.

Table 2. CNN architecture details.

Layer Type	**No. of Filters**	**Feature Size**	**Kernel Size**	**Padding**	**Stride**
Input Image		128*128*1			
Conv Layer 1	8	128*128*8	3*3	1	1*1
Batch Normalisation					
Relu/Leaky Relu					
Max Pooling Layer		64*64*8	2*2		2*2
Conv Layer 2	64	62*62*64	3*3	0	1*1
Batch Normalisation					
Relu/Leaky Relu					
Max Pooling Layer		31*31*64	2*2		2*2
Conv Layer 3	128	27*27*128	5*5	0	1*1
Batch Normalisation					
Relu/Leaky Relu					
Max Pooling Layer		13*13*128	2*2		2*2
Conv Layer 4	256	9*9*256	5*5	0	1*1
Batch Normalisation					
Relu/Leaky Relu					
Conv Layer 5	512	3*3*512	7*7	0	1*1
Batch Normalisation					
Relu/ Leaky Relu					
Dropout (0.5)					
Fully Connected Layer					
Softmax Layer					
Classification Layer		26			

4.3. Optimizers

The five-layer CNN architecture proposed in this paper has been further analyzed for three optimizers: Sgdm (Stochastic gradient descent with momentum), RMSProp (Root mean square propagation), and Adam (Adaptive moment estimation).

i. Stochastic Gradient Descent with Momentum (Sgdm) is an iterative approach for optimizing objective function with sufficient smoothness properties. This may be regarded as a stochastic approximation to gradient descent optimization, as it substitutes the real gradient (calculated from the data collection as a whole) by calculating its randomly chosen data subset. Thus, some samples are selected instead of the whole dataset. The global minima can easily gain in the whole dataset but it becomes problematic in the case of a huge dataset. Mini-batch needs careful selection of learning rates. Small learning rates lead to slow convergence and a high level of convergence can lead to fluctuations in the loss function around the minimum. Adding momentum helps accelerate convergence in the required direction [22].

$$W = W - V_t \tag{4}$$

$$V_t = \beta\, v_{t-1} + \alpha \nabla_w L\ (W, X, y) \tag{5}$$

In mathematical equations Eq. (4), (5) of Sgdm optimization, L is the loss function, ∇ nabla sign is for the gradient concerning weight, and α represents the learning rate and β is the momentum which is usually 0.9.

ii. Root Mean Square Propagation (RMSProp) is a fast and popular optimizer. It is another adaptive learning rate algorithm that is an improvement to Adagrad. RMSProp tries to overcome the dramatically rising learning rate of Adagrad by utilizing the running average of the square gradient. This is focused on the magnitude of the recent gradient descent to stabilize the gradient. This dynamically sets the learning rate and selects a particular learning rate for each parameter. It divides the learning rate by the average of the exponential decay of square gradients. The mathematical equations are as in Eq. (6) and Eq. (7).

$$w = w_{(t-1)} - \frac{\alpha}{\sqrt{s_t + \varepsilon}} * \frac{\partial L}{\partial W} \tag{6}$$

$$S_t = \beta\, s_{t-1} + (1-\beta)\left[\frac{\partial L}{\partial W}\right]^2 \tag{7}$$

where α represents the learning rate, $\frac{\partial L}{\partial W}$ gradient of loss function w.r.t weights. ε is a small positive constant, S_t represents the exponentially weighted average and β is the hyperparameter represents the momentum which is usually 0.9.

iii. Adaptive Moment Estimation (Adam) is a combination of RMSProp and Adagrad. It calculates the single adaptive learning rate for each parameter from the first and second moment estimates of gradients. It is one of the most popular gradient descent optimizer algorithms. Adagrad uses the basic average but Adam uses the exponential moving average of the gradients to scale the learning rate. It holds the average of past gradients declining exponentially. It is very powerful and the memory required is also very less. The mathematical equations are in Eq. (8), (9), (10).

$$w_{t+1} = w_t - \frac{\alpha}{\sqrt{\hat{s}_t + \varepsilon}} * \hat{v}_t \tag{8}$$

$$\hat{V}_t = \beta_1\, v_{t-1} + (1-\beta_1)\frac{\partial L}{\partial w_t} \tag{9}$$

$$\hat{S}_t = \beta_2\, s_{t-1} + (1-\beta_2)\left[\frac{\partial L}{\partial w_t}\right]^2 \tag{10}$$

where α is the learning rate, $\hat{V}_t$ and $\hat{S}_t$ are the bias-corrected estimates of first and second moment respectively. Hyper-parameters $\beta 1$, $\beta 2$ is to regulate the exponential rate of decay of such moving averages. All parameters considered for all optimizers are default.

5. Results and discussion

After implementing the proposed model, this section discusses the obtained results. For testing, 10% of the dataset is considered. The developed ISL recognition model is tested by varying many parameter values. The model is also tested by using different activation functions and three optimizers on simple and complex background images and the mixture of both as well. As dataset shuffling is effective in enumerate random nature to the neural network training process which prevents the network from biasing towards certain parameters. Training has been done with three different optimizers to analyse which one is the best for sign language recognition. All optimizers took different times to train the network. The activation function is also varied to check which one is suitable in this case. Further to ensure the robustness of the proposed model, performance on NUS dataset-I and NUS dataset-II have been analyzed.

5.1. Performance of proposed model on ISL datasets

Table 3 shows the results of the ISL dataset for simple background with two different activation functions (ReLU and leaky ReLU) and three optimizers (Adam, RMSProp, Sgdm). It is found that the ReLU activation function with Adam as an optimizer performed better than other combinations. For *ISL Simple Background Dataset*, training accuracy of 100%, validation accuracy of 99.36%, and test accuracy of 99.10% is noted. For the *ISL Complex Background Dataset*, training accuracy of 99.81%, Validation accuracy of 93.59%, and test accuracy of 92.69% is acquired. In *ISL Mixed Background Dataset*, with a simple background and ISL dataset with a complex background are mixed. The training accuracy of 99.28%, validation accuracy of 96.21%, and test accuracy were 95.95% is reported. In the case of all the scenarios, the combination of ReLU and Adam gave better results. Thus, it is concluded that ReLU and Adam's combination works well in the case of ISL datasets of simple as well as complex background.

Table 3. Results of activation functions and optimizers for ISL datasets.

ISL Simple Background Dataset				
Optimization Techniques	**Activation Functions**	**Train Accuracy %**	**Validation Accuracy %**	**Test Accuracy %**
Adam	**ReLU**	**100**	**99.36**	**99.10**
	Leaky ReLU	100	99.23	99.10
Sgdm	ReLU	99.90	98.97	98.72
	Leaky ReLU	99.97	98.85	98.72
RMSProp	ReLU	99.82	98.72	98.46
	Leaky ReLU	99.98	99.23	98.85
ISL Complex Background Dataset				
Optimization Techniques	**Activation Functions**	**Train Accuracy %**	**Validation Accuracy %**	**Test Accuracy %**
Adam	**ReLU**	**99.81**	**93.59**	**92.69**
	Leaky ReLU	99.89	94.74	91.54
Sgdm	ReLU	99.68	91.54	90.77
	Leaky ReLU	99.68	92.31	92.69
RMSProp	ReLU	99.47	91.15	90.64
	Leaky ReLU	99.54	90.77	90.90
ISL Mixed Background Dataset				
Optimization Techniques	**Activation Functions**	**Train Accuracy %**	**Validation Accuracy %**	**Test Accuracy %**
Adam	**ReLU**	**99.28**	**96.21**	**95.95**
	Leaky ReLU	99.14	95.42	95.95
Sgdm	ReLU	99.50	95.10	95.62
	Leaky ReLU	99.53	95.36	95.88
RMSProp	ReLU	99.22	93.86	94.90
	Leaky ReLU	99.20	93.66	95.03

To analyze the system performance in-depth, the response of the system to each alphabet class is analyzed. For this performance measures such as precision, recall, F1-score are calculated. Precision, recall, F1-Score are basic measures to evaluate the performance of the model for each class. Precision is also known as positive predictive value. It is a fraction of the relevant instances of the instances retrieved. It can be described as in Eq. (11),

$$Precision = \frac{TP}{TP+FP} \tag{11}$$

where TP and FP are the numbers of true and false positives, respectively. The recall is also called sensitivity. It is the fraction of the total amount of relevant instances that were actually retrieved. It can be described as in Eq. (12).

$$Recall = \frac{TP}{TP+FN} \tag{12}$$

where FN is the number of false negatives. To fully evaluate the effectiveness of the model, precision and recall must be examined. So, F1-score is calculated which is the weighted average of Precision and Recall. It can be described as in Eq. (13),

$$F1 - Score = \frac{2*recall*precision}{recall+precision} \tag{13}$$

The outcome for each class of the ISL Mixed Background Dataset is listed in Table 4. Low F1-score (<95%) is observed for alphabets F, I, N, P, Q, S, T, U, V, and X as these alphabets either have complex shapes or similar to other signs.

Table 4. Performance of ISL dataset with a mixed background.

Signs	Precision	Recall	F1_Score	Signs	Precision	Recall	F1_Score
A	0.951	0.967	0.959	**N**	0.981	0.867	0.920
B	0.984	1.000	0.992	**O**	1.000	0.917	0.957
C	1.000	1.000	1.000	**P**	0.897	0.867	0.882
D	1.000	1.000	1.000	**Q**	0.982	0.900	0.939
E	0.952	0.983	0.967	**R**	0.983	0.950	0.966
F	0.908	0.983	0.944	**S**	0.966	0.933	0.949
G	1.000	1.000	1.000	**T**	0.885	0.900	0.892
H	0.952	0.983	0.967	**U**	0.903	0.933	0.918
I	0.965	0.917	0.940	**V**	0.881	0.983	0.929
J	0.984	1.000	0.992	**W**	0.984	1.000	0.992
K	0.938	1.000	0.968	**X**	0.966	0.933	0.949
L	1.000	1.000	1.000	**Y**	0.983	0.967	0.975
M	0.952	0.983	0.967	**Z**	1.000	1.000	1.000

5.2. Performance of proposed model on NUS datasets

Table 5 shows the results of NUS datasets with the proposed model with ReLU and Adam. NUS dataset -I contain 10 gestures with a simple background and NUS dataset-II also consists of 10 gestures but with complex backgrounds. *NUS dataset I* with simple background gave 100% accuracy, with *NUS dataset II* complex background training accuracy is 100%, Validation accuracy is 95%, test accuracy is 95.95%. The mixed dataset model gave 99.91% in training, validation accuracy of 96.53% and test accuracy was 97.22%.

Table 5. Results of NUS Datasets with the proposed model.

Optimization Techniques	Activation Function	NUS Datasets	Training Accuracy %	Validation Accuracy %	Test Accuracy %
Adam	ReLU	NUS-I	100	100	100
		NUS-II	100	95	95.95
		Mixed (NUS-I & NUS-II)	99.91	96.53	97.22

5.3. Performance comparison of proposed model with other sign language recognition systems
After the performance evaluation of the proposed model, this section compares the results achieved with existing techniques which are shown in Table 6. It is found that the research in the sign language recognition domain has been mostly based upon machine learning approaches, with some systems showing a shift towards the deep learning domain. Most of the sign language datasets include a small vocabulary only because databases suitable for sign language recognition with a larger vocabulary are currently not available. A suitable number of training samples in datasets are required to properly train and generalize the deep learning-based architectures. To ensure these datasets must be collected from a large number of subjects rather than video or extensive repetitions. ISL Datasets used in this work is an extension of the dataset reported in [5], [7], and [23]. Thus, as compared to other works dataset has been collected from 100 subjects resulting in variations in terms of skin tone, size of hands, and inter-signer variation in shape made. As compared to single-hand American sign language [10] and Singapore sign language (SgSL) [8] [9] have simple and similar signs. On the other hand, Indian sign language and Bangla sign language [4] mostly have double hand gestures with complex hand shapes. Focussing on CNN parameters the work presented in this paper works well as a sign language recognition system.

Table 6. Comparisons of proposed method with other sign language recognition systems

Year [Ref]	Language	Classes, Dataset	Subjects	Technique	Background	Accuracy (%)
2016 [4]	BSL	vowels consonants	30	SVM	Simple	97.7
2017 [6]	ISL	18 words	10	ANN	Simple	90
2017 [5]	ISL	26, ISL Simple Background	72	SVM	Simple	97.9
2018 [10]	ASL	24 alphabets	-	Inception v3	Simple	90
2018 [15]	ISL	200 signs	5	CNN	Simple	92.88
2018 [7]	ISL	26, ISL Simple Background 26, ISL Complex Background	90	SVM	Simple Complex	98.6 83.8
2018 [8]	SgSL	10, NUS Dataset – I	-	Adaboost	Simple	91.6
2018 [9]	SgSL	10, 2 subsets of NUS Dataset – II	-	SVM	Complex	97.8, 95.07
2019 [16]	ISL	24 static alphabets, 2 dynamic alphabets, 11 dynamic words	7	SVM	Simple	91
2020 [17]	ISL	100 Signs	-	CNN	Complex	98.70
2020 [23]	ISL	26, ISL Complex Background	90	SVM	Complex	92
Proposed CNN Model	ISL	26, ISL Simple Background 26, ISL Complex Background 26, ISL Mixed Background	100	CNN	Simple Complex Mixed	99.10 92.69 95.95

It is apparent from the table that higher recognition levels in the sign language domain are either due to limited vocabulary, limited dataset variations, simple background, or use of a sensor-based wristband. The proposed system outperformed other existing systems considering simple as well as complex background images collected from 100 subjects.

6. Conclusions

In this paper, a deep learning-based five-layer CNN model has been proposed which does not require a hand-crafted feature set. Most of the work in ISL has been done with simple or complex background images because the feature set in the machine learning model is unable to address the complexity in the dataset. In this work, the deep learning approach has been implemented in the proposed architecture which achieved the accuracy of 99.10%, 92.69% and, 95.95% for simple, complex, and mixed backgrounds respectively. Various CNN parameters have been explored to achieve the best results. So, after trying the various set of filter combinations [3*3, 3*3, 5*5, 5*5, 7*7] filter sizes and [8, 64, 128, 256, 512] order of filters have been recommended for each convolutional layer. The pattern gets more complex in subsequent layers thus filters are increased. Accuracy became stable after 10 epochs and 128 batch size. The model learning is most convincing at a 0.001 learning rate and validation frequency of 10. Two activation functions, ReLU and Leaky ReLU have been explored where ReLU performed best. Among the three optimizers deliberated, Adam gave the best results. Thus, with the five-layer of CNN based deep learning architecture, a combination of ReLU and Adam has been proposed for ISL datasets. The model is also tested on NUS dataset-I, NUS dataset-II, and mixed dataset where the accuracy of 100%, 95.95%, and 97.22% respectively has been achieved.

References

[1] Ghotkar A and Kharate G 2012 Hand segmentation techniques to hand gesture recognition for natural human computer interaction *International Journal of Human Computer Interaction* **3** pp 15-25.

[2] Oyedotun O K and Khashman A 2017 Deep learning in vision based static hand gesture recognition *Neural Comput Appl* **28** pp 3941–3951.

[3] Badhe P C and Kulkarni V 2015 Indian sign language translator using gesture recognition algorithm *IEEE International Conference on Computer Graphics, Vision and Information Security* pp 195-200.

[4] Uddin M A and Chowdhury S A 2016 hand sign language recognition for bangla alphabet using support vector machine *IEEE international conference on innovations in science, engineering, and technology (ICISET)* pp 1-4.

[5] Kaur B, Joshi G and Vig R 2017 Indian sign language recognition using krawtchouk moment-based local features *The Imaging Science Journal* **65** pp 171-179.

[6] Rao G A and Kishore P V V 2017 Selfie video based continuous Indian sign language recognition system *Ains Shams Engineering Journal* **9** pp 1929-1939.

[7] Joshi G, Vig R and Singh S 2018 DCA-based unimodal feature-level fusion of orthogonal moments for Indian sign language dataset *IET Computer Vision* **12** pp 570-577.

[8] Lahiani H and Neji M 2018 Hand gesture recognition method based on HOG-LBP features for mobile devices *22nd International Conference on Knowledge-Based and Intelligent Information & Engineering Systems* pp 254-263**.**

[9] Zhang F, Liu Y, Zou C and Wang Y 2018 Hand gesture recognition based on HOG-LBP feature *IEEE International Instrumentation and Measurement Technology Conference* pp 1-6.

[10] Das A, Gawde S, Suratwala K and Kalbande D 2018 Sign Language recognition using deep learning on Custom Processed Static Gesture Images *2018 International Conference on Smart City and Emerging Technology* pp 1-6.

[11] Yang S, Zhu Q 2017 Video-based Chinese sign language recognition using convolutional neural network *IEEE 9th International Conference on Communication Software and Networks* pp 929-934.

[12] Tushar A K, Ashiquzzaman A and Islam M R 2017 Faster convergence and reduction of overfitting in numerical hand sign recognition using DCNN *Humanitarian Technology Conference* pp 638–641.

[13] Bheda V, Radpour D 2017 Using deep convolutional networks for gesture recognition in American sign language arXiv (*Preprint* arXiv:1710.06836).

[14] Sajanraj T D and Beena M V 2018 Indian sign language numeral recognition using region of interest convolutional neural Network *Proceedings of the 2nd International Conference on Inventive Communication and Computational Technologies* pp 636-640.

[15] Rao G A, Syamala K, Kishore P V V and Sastry A S C S 2018 Deep Convolutional neural networks for sign language recognition *Conference on Signal Processing And Communication Engineering Systems* pp 194-197.

[16] Athira P K, Sruthi C J and Lijiya A 2019 A Signer Independent sign language recognition with co articulation elimination from live videos: an Indian scenario *Journal of King Saud University – Computer and Information Sciences* pp 1-11.

[17] Wadhawan A and Kumar P 2020 Deep learning-based sign language recognition system for static signs *Neural*

Computing and Applications **32** pp 7957–7968.
[18] Kumar E K, Kishore P V V, Kiran Kumar M T 2019 3D sign language recognition with joint distance and angular coded color topographical descriptor on a 2-stream CNN *Neurocomputing* **372** pp 40–54.
[19] Gaikwad S, Shetty A, Satam A, Rathod M and Shah P 2019 Recognition of American Sign Language using Image Processing and Machine Learning *International Journal of Computer Science and Mobile Computing* **8** pp 352-357.
[20] Krizhevsky A, Sutskever I and Hinton G E 2012 ImageNet classification with deep convolutional neural networks *In Advances in Neural Information Processing Systems* 1097–1105.
[21] Choudhari G and Mehra R 2019 Iris Recognition using Convolutional Neural Network Design *International Journal of Innovative Technology and Exploring Engineering* **8** pp 672-678.
[22] Poojary R and Pai A 2019 Comparative Study of Model Optimization Techniques in Fine-Tuned CNN Models *International Conference on Electrical and Computing Technologies and Applications* pp 1-4.
[23] Joshi G, Singh S and Vig R 2020 Taguchi-TOPSIS based HOG parameter selection for complex background sign language recognition *Journal of Visual Communication and Image Representation* **70** 1-22