A hybrid approach for Bangla sign language recognition using deep transfer learning model with random forest classifier


Published in:
Expert Systems with Applications

Document Version:
Publisher's PDF, also known as Version of record

Queen’s University Belfast - Research Portal:
Link to publication record in Queen’s University Belfast Research Portal

**Publisher rights**
Copyright 2022 the authors. This is an open access article published under a Creative Commons Attribution-NonCommercial-NoDerivs License (https://creativecommons.org/licenses/by-nc-nd/4.0/), which permits distribution and reproduction for non-commercial purposes, provided the author and source are cited.

**General rights**
Copyright for the publications made accessible via the Queen's University Belfast Research Portal is retained by the author(s) and / or other copyright owners and it is a condition of accessing these publications that users recognise and abide by the legal requirements associated with these rights.

**Take down policy**
The Research Portal is Queen’s institutional repository that provides access to Queen’s research output. Every effort has been made to ensure that content in the Research Portal does not infringe any person’s rights, or applicable UK laws. If you discover content in the Research Portal that you believe breaches copyright or violates any law, please contact openaccess@qub.ac.uk.

**Open Access**
This research has been made openly available by Queen’s academics and its Open Research team. We would love to hear how access to this research benefits you. – Share your feedback with us: http://go.qub.ac.uk/oa-feedback
A hybrid approach for Bangla sign language recognition using deep transfer learning model with random forest classifier

Sunanda Das a, Md. Samir Imtiaz a, Nieb Hasan Neom a, Nazmul Siddique b,c, Hui Wang c

a Department of Computer Science and Engineering, Khulna University of Engineering & Technology, Khulna 9203, Bangladesh
b School of Computing, Engineering and Intelligent Systems, Ulster University, Londonderry BT48 7JL, UK
c School of Electronics, Electrical Engineering and Computer Science, Queen’s University Belfast, Belfast, BT9 5BN, UK

A R T I C L E I N F O

Keywords:
Bangla sign language
Deep learning
Convolutional neural network (CNN)
Transfer learning
Character recognition
Digit recognition

A B S T R A C T

Sign language is the comprehensive medium of mass communication for hearing and speaking impaired individuals. As they cannot speak or hear, they are not able to use sound or vocal signals as an information medium for their communication. Rather, they are bound to exchange visual signals to express their feeling in their day-to-day life. For this, they use various body language mainly hand gestures as sign language. Sign language fundamentals can be largely divided into two parts namely digits (numerals) and characters (alphabetical). In this paper, we proposed a hybrid model consisting of a deep transfer learning-based convolutional neural network with a random forest classifier for the automatic recognition of Bangla Sign Language (numerals and alphabets). The overall performance of the presented system is verified on ‘Ishara-Bochon’ and ‘Ishara-Lipi’ datasets. ‘Ishara-Bochon’ and ‘Ishara-Lipi’ are datasets of isolated numerals and alphabets respectively which are the first complete multipurpose open-access dataset for Bangla Sign Language (BSL). Besides, we also proposed a background elimination algorithm that removes unnecessary features from the sign images. Along with the proposed background elimination technique, the system is able to achieve accuracy, precision, recall, f1-score values of 91.67%, 93.64%, 91.67%, 91.47% for character recognition and 97.33%, 97.89%, 97.33%, 97.37% for digit recognition respectively. The detailed experimental analysis assures the feasibility and effectiveness of the proposed system for BSL recognition.

1. Introduction

Sign language has been developed for speaking and hearing-impaired people to communicate effectively. A language is a system of communication that comprises a collection of sounds and written symbols used by a population. But, hearing and speaking impaired people cannot use that language to communicate, rather they use sign language. There is no standardized universal form of sign languages and thus sign languages from different countries or regions are not mutually intelligible with each other.

According to World Health Organization (WHO), about 466 million people (432 million adults and 34 million children) globally are suffering from some kind of deafness and hearing disability that makes up to 6.1% of the world’s population and by the year 2050, the number is estimated to go over 900 million which is about one in every ten people (W.H.O., 2021). These numbers show the importance of sign language and machine-based sign language translation systems. A lot of research work has been done on the recognition of sign languages.

In the recent years, Bengali Sign Language (BSL) recognition has also received a lot of attention. Machine learning and lately deep learning have achieved great improvement in the field of recognition and classification (Anand, Urolagin, & Mishra, 2021; Machiraju, Urolagin, Mishra, & Sharma, 2021; Singh, Mishra, Urolagin, & Sharma, 2021). With the invention and development of powerful Convolutional Neural Network (CNN) models, it has reached new heights of success. Applying CNN models in the automatic recognition of sign language is a popular choice among researchers nowadays. A sign language recognition and translation system can break down the communication barrier between normal people and hearing-impaired people by creating a communication bridge. Allowing hearing-impaired people to participate in social, economic, and political activities can help them become valuable assets to the society.

Despite the fact that Bangla is the world’s seventh most spoken language, with 265 million speakers (Berlitz, 2021), there are limited studies and prospects for working in the automatic detection of Bengali
sign language. To make a contribution to this field, we have developed a deep learning-based BSL recognition system that can accurately categorize BSL characters and digits from sign images. Many of the previous research (Haque, Das, & Kaspy, 2019; Rahman, Abdullah, Mondal, et al., 2012; Uddin, Arko, Tabassum, Trisha, & Ahmed, 2017) used a limited amount of data due to the unavailability of large-scale open-source datasets in sign language recognition and performed very inadequately. We employed data augmentation and several pre-trained models to overcome the lack of limited samples in datasets, as pre-trained models can effectively deal with minimal datasets and increase model performance (Das, Fime, Siddique, & Hashem, 2021; Kieffer, Babale, Kalra, & Tizhoosh, 2017). The significant contributions of the research work are:

- Employment of the proposed background elimination algorithm with the help of different morphological operations and adaptive Gaussian Threshold technique which dynamically decides threshold value for a pixel by considering certain small neighborhoods around it so that an optimum threshold value can be determined for different regions depending on local background pixels.
- Employment of transfer learning with the help of weight transferring from different backbone networks (VGG16, VGG19, InceptionV3, Xception, ResNet50) pre-trained on ImageNet to handle small-sized datasets and hybridization of transfer learning and Random Forest (RF) classifier. Because the RF only considers a small subset of features at each split of the tree, it can deal with data having higher dimensions. If a single decision tree contains high variance, the RF bagging or bootstrap aggregating method combines a lot of weak learners into strong learners, which averages out the variance.
- Utilization of grid search algorithm on different batch sizes for each model separately to come up with a fine-tuned batch size which is necessary to generate optimum results from the proposed hybrid architecture. The proposed architecture is trained with the mini-batch technique which offers a regularizing effect, updates the network’s weights after each propagation, and reduces the generalization error while providing stable convergence and satisfactory test performance which makes the batch size an important hyper-parameter.
- Improvement of evaluation parameters such as accuracy, precision, recall, and f1-score compared to other existing recognition methods.

The rest of the paper is arranged as follows: A very extensive overview and analysis of previous research in this domain are provided in Section 2. Section 3 describes the overall system methodology along with the proposed background elimination module. In Section 4, the evaluation metrics are presented. After that, Section 5 illustrates the detailed experimental analysis of this study. Finally, in Section 6, the study’s conclusions are drawn, along with the plan for further research.

2. Literature review

Sign language, like any natural language, has a wide range of expressions in different parts of the world. Over the years, some sign languages, such as American Sign Language (ASL), and Indian Sign Language (ISL) have gained prominence over others. To have a better understanding of the state of the art, we looked at several of these sign language recognition systems as well as certain Bengali Sign Language (BSL) recognition systems.

2.1. American Sign Language (ASL)

American Sign Language (ASL) is the primary sign language of hearing-impaired people in the United States and parts of Anglophone Canada. Fok, Cheng, and Ganganath (2015) proposed a novel real-time ASL digit recognition system that senses data of hand gesture using two Leap Motion sensors (depth-sensors). Data of the two sensors are integrated using Kalman filters. The Hidden Markov Model is used for the recognition of hand gestures. The system performed better with higher evaluation metric values than other single depth-sensor based systems.

Zaki and Shaheen (2011) proposed a sign language recognition system that uses a novel combination of vision based features. Kurtosis position, principle component analysis and motion chain code features are combined to be mapped to four components of sign languages such as hand shape, place of articulation, hand orientation, and hand movement. Using HMM (Hidden Markov Model) as classifier, they achieved 89.1% recognition accuracy.

Kasukurthi, Rokad, Bidani, Dennisin, et al. (2019) introduced a neural network-based model for ASL alphabet identification from color images that can run on mobile devices. Prior to the conventional CNN architecture they used squeezeenet architecture (Iandola et al., 2016) that uses 1×1 filters for convolution operation which minimizes the number of trainable parameters and keeps the model size small to be deployable in mobile devices or cell phones. With their proposed model, they were able to achieve an accuracy of 83.29%.

Masood, Thuwal, and Srivastava (2018) used VGG16 model to classify 36 different hand gestures (26 alphabets and 10 numerals) of ASL from the dataset given by Barczak, Reyes, Abastillas, Piccio, and Susnjak (2011). They initialized the parameters of their model by transferring weights from the VGG16 network pretrained on ImageNet dataset which consists of more than a million images of 1000 categories. They modified the softmax layer of the VGG16 network according to their need as they had only 36 categories and trained the model using the ASL dataset. With this technique they obtained an impressive accuracy of 96%.

In Bilgin and Mutfulodog (2019), Bilgin et al. used Lenet and capsule networks separately for ASL character recognition. Then, they compared the outcome of capsule network with the outcome of LeNet-5 (LeCun, Bottou, Bengio, & Haffner, 1998) network architecture. They achieved 88% accuracy using capsule networks, where LeNet-5 was able to achieve 82% accuracy. Conventional CNN models like LeNet-5 evaluates features detected in the past layers as weighted sum in the following layers without preserving spatial information i.e. position and orientation related feature information about an image. Capsule networks successfully overcomes these limitations by treating features in the images as vectors in exchange for scalars which helps it preserve spatial information.

2.2. Indian Sign Language (ISL)

Despite popular belief, Indian Sign Language (ISL) is not a manual representation of spoken English or Hindi, and it has its own syntax. Sharma, Tomar, Mishra, and Chariar (2021) presented a transfer learning-based deep CNN architecture where they used the MobileNetV2 model pre-trained on ImageNet to recognize 35 classes of ISL symbols. They updated the weights of the unfrozen layers of the model by training them with the ISL dataset and achieved satisfactory performance on the test samples.

Sruthi and Lijiya (2019) proposed a vision-based recognition system for ISL static alphabets i.e. those alphabets that does not need continues movement of hand or fingers. They used a deep learning-based approach to classify 24 sign categories and achieved a validation accuracy of 98.64%. Before training the model, they segmented the image dataset accordingly. For this, first they detected the face in the image with a face detection algorithm and removed it by replacing the corresponding pixels with black pixels. After that, they applied hand region segmentation using color segmentation algorithm and largest connected component algorithm. Then, the modified image dataset was fed into their ‘signet model’ for training. A real time system for numeral recognition of ISL was proposed by Sajanraj and Beena...
work for classification purpose. They achieved maximum performance in the segmented images. Then cropping was applied to keep only the region of interest. Filtering and morphological operations were performed to remove noise.

A complete system for BSL alphabet recognition was proposed by Rahman et al. in Rahman et al. (2012). It can take static hand gesture images as input and classify the gesture out of 36 BSL alphabet gestures with an average accuracy of 80.90% with high variance. As the classifier, they used standard feed-forward Artificial Neural Network (ANN) with sigmoid as activation function and mean square error as loss function. Uddin et al. (2017) introduced a novel fingertip finder algorithm to identify one-handed static sign gestures of Bengali Sign Language including vowels, consonants and numerals from sign images of static background. After collecting necessary image data, in the preprocessing stage, they applied otSU's thresholding to come up with well segmented binary image. Median filtering and morphological operations were performed to remove noise in the segmented images. Then cropping was applied to keep only the region of interest. They used a multilayered feed forward neural network for classification purpose. They achieved maximum performance with 27-nodes in the hidden layer with an average recognition rate of 88.69%.

Hossen et al. presented a CNN-based model for BSL recognition in Hossen, Govindaiyah, Sultana, and Bhuiyan (2018). They worked with a diverse and natural color image dataset (Aziz, Wadud, Sultana, Hussain, & Bhuiyan, 2017) with varied background, color and lighting condition making it usable in real time systems. On a comparatively small sized diverse dataset consisting of 1147 images of 37 different signs, they achieved a validation set accuracy of 84.68%. To get good results with this small dataset, they used transfer learning and fed their data to a VGG16 network pre-trained with ImageNet dataset. A large diverge dataset along with a deep learning-based recognition model was proposed in Rafi et al. (2019). They considered 38 BSL alphabets, used the pre-trained VGG19 transfer learning model for classification, and were able to achieve 89.60% test accuracy on their dataset.

Abedin, Prottoy, Moshрубa, and Hakim (2021) proposed a novel approach to solve the problem of BSL recognition which they called as ‘Concatenated BSL Network’. The specialty of the network is that it combines a CNN along with a pose estimation network to enhance overall performance. In their model, ‘Openpose’, a pretrained hand pose estimation algorithm, is used to extract positional features from the images. In Tasmere, Ahmed, and Hasan (2020), Tasmere et al. proposed a real time method that can classify BSL digits from video data. The training data was built by capturing images from real-time video frames. The moving objects from video data was detected using Gaussian Mixture Model (GMM). After that, the images were further pre-processed with blurring, thresholding, normalization before being fed to a deep CNN model. The system is 97.63% accurate in classifying sign digits.

Hasan and Ahsan (2019) proposed a BSL digits recognition method that uses Histogram of Oriented Gradient (HOG) feature based multi-class Support Vector Machine (SVM). HOG features were calculated from binary image data. These features were used to train a ONE VS ONE multi-class SVM for classification. The system performs with 94.74% accuracy while requiring less time for sign recognition. Basnin, Nahar, and Hossain (2021) presented a CNN–LSTM-based model for Bangla Sign Language recognition. A large dataset consisting of 13,400 images of 36 classes was used to train and validate the system. The images were thoroughly pre-processed using background subtraction, gray scale conversion, morphological erosion, and median filtering. The images were then fed to a customized CNN network where a time-distributed flattening layer separates the CNN model from the LSTM layer. The model achieved an accuracy of 88.5% with no overfitting.

Tabassum et al. (2020) proposed a HOG feature based single-handed Bengali Sign Language recognition system. They prepared a dataset of 1400 images and pre-processed them accordingly with histogram equalization, and lightness smoothing to improve the image quality. Hand portion of the images were segmented using YCbCr color space. Then, the images were converted to gray scale and HOG features were calculated. These features were used to train a K-Nearest Neighbor classifier. The model is 91.1% accurate in classifying Bengali Sign images.

Ahmed et al. (2019) proposed a complete computer vision-based system that can recognize Bangla Sign digits and translate them to Bangla speech. Deep CNN was chosen for the classification purpose. The maximum accuracy achieved by the system is 92%. Haque et al. (2019) proposed a model to classify the BSL alphabets. They used a self-build dataset consisting of 130 samples and achieved 77.88% recognition accuracy. The image data was pre-processed with size normalization, gray scale conversion, and histogram equalization. The principal component analysis (PCA) was used to extract features and reduce dimensionality in the data. A K-Nearest Neighbor classifier was trained using these features for classification.

Shanta, Anwar, and Kabir (2018) presented a Bangla Sign Language detection system using Scale Invariant Feature Transformation (SIFT)
and CNN. In the pre-processing step, the skin masking technique was used to segment the image to keep only the region of interest. An elliptical kernel was used to remove noise by performing erosion and dilations. Then, the edges of the images were detected using canny edge detection. Then, SIFT was applied to calculate scale (rotation, scaling, elliptical kernel was used to remove noise by performing erosion and dilations. Then, the edges of the images were detected using canny edge detection. Then, SIFT was applied to calculate scale (rotation, scaling, etc.) invariant features. These features were further clustered using k-means clustering before being fed to a CNN model for classification.

Khan, Joy, Asaduzzaman, and Hossain (2019) implemented a real-time, low-cost, fast Sign Language Translator prototype integrated into a Raspberry Pi device. It can detect 5 signs in real-time from video data coming through a webcam. At first, self build image dataset was created, augmented, and pre-processed accordingly. Then, the images were fed to a CNN classifier consisting of 3 convolutional, 3 normalizations, and 2 fully connected layers. After that, the model was integrated into a Raspberry Pi 3B. The accuracy while classifying signs from video data was approximately 94%.

2.4. Other sign languages

Apart from these, we have also studied Chinese Sign Language (Chai et al., 2013), German Sign Language (Dreuw, Deselaers, Keyser, & Ney, 2006), Korean Sign Language (Shin, Kim, & Jang, 2019), Japanese Sign Language (Sako & Kitamura, 2013), etc., to build a robust system that can recognize sign language accurately.

3. Methodology

Fig. 1 depicts the procedures involved in our sign language recognition system. The Ishara-Lipi datasets (Islam, Mousumi, Rabby et al., 2018; Islam, Mousumi, Rabby, Hossain et al., 2018) were used as the starting point. The images’ backgrounds were eliminated in order to focus solely on the objects of interest. If the image dataset already consists of binary images then this step is skipped as the images are already well segmented. Then, necessary data augmentation and re-scaling were done on the data. The data was then fed into the proposed model, which employed backbone networks as feature extractors and fed the extracted features to a random forest classifier. All of these steps resulted in a model that can successfully classify BSL characters and digit signs.

Algorithm 1: Background Elimination Algorithm

3.1. Data preprocessing

3.1.1. Background elimination

Background elimination is applied to remove unnecessary features from the image data which does not contribute to the classification process. It detects the background from an image and replaces all the pixels of the background with black pixels, leaving only the object of interest in the image. The proposed approach for background elimination is presented in Algorithm 1 and the graphical illustration of the algorithm is shown in Fig. 2. The traditional technique for threshold calculation based on histogram, which occasionally creates some undesirable region, is also presented in this figure to demonstrate the superiority of the proposed algorithm. It is evident that the proposed algorithm’s several stages are particularly effective at reducing various sorts of noise.

At first, we create a copy of the grayscale version of the original image. Secondly, we apply thresholding to the grayscale image. Thresholding is a technique in which the pixel value is compared with the threshold value, and the pixel value is then set based on the threshold value. The threshold value may be handpicked or calculated using some algorithm. In our case, we used the Gaussian adaptive threshold method. The thresholding operation outputs a binary image where black pixels and white pixels represent background and foreground respectively.

In basic thresholding, we had to manually preset a threshold value and in Otsu’s thresholding, the optimal threshold value is automatically determined. But, both of these methods are global thresholding techniques, i.e. same threshold value is applied to segment all pixels in the image. Hence, it performs poorly in segmenting images that have variations in lighting and shading conditions in different local regions. Adaptive thresholding solves the problem by considering a small set of neighboring pixels at a time and computes the threshold.
based on that specific local region. So, the different threshold value is applied while thresholding different regions. In fact, the threshold value is calculated separately for each pixel of the image. Choosing the right size of pixel neighborhood is important as the thresholding largely depends on the assumption that, the neighboring pixels should have approximately uniform values. The neighborhood size should neither be too small to cover enough background and foreground pixels nor be too large to obey the assumption of local regions. The equation to calculate the threshold value for a pixel is defined in the Eq. (1).

\[ T(i,j) = GMean(i,j) - C \]  

(1)

Here, \( T(i,j) \) is the threshold value calculated for the pixel at position \((i,j)\) in the gray-scaled image. The mean value \( GMean(i,j) \) calculated here is the Gaussian mean which is a weighted arithmetic mean where pixel values farther away from the center pixel at \((i,j)\) are assigned lesser weights. \( C \) is a constant value that tunes the threshold. There may be cases where the mean value alone is insufficient to distinguish between the background and foreground. The result of the threshold can be improved by adding or subtracting some constant value \( C \). The value for \( C \) is entirely dependent on application and simple to tune by trial and error method. The expression for 2D Gaussian distribution is given in Eq. (2) where \( \sigma \) represents the standard deviation of the distribution and the distribution is assumed to have a mean of zero.

\[ G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2 + y^2}{2\sigma^2}} \]  

(2)

Eq. (3), is used to transform the grayscale image into a binary image where \( T(x,y) \) is a threshold that is computed for each pixel individually and \( I_{src}, I_{dst} \) denote the intensity value at \((x,y)\) co-ordinates for the source and destination image respectively.

\[ I_{dst}(x,y) = \begin{cases} 
0 & \text{if } I_{src}(x,y) > T(x,y) \\
255 & \text{otherwise}
\end{cases} \]  

(3)

After thresholding, we applied morphological opening and morphological closing transformations on the binary image to remove noise. A morphological opening of image \( A \) using the structuring element \( B \) is performed by executing two consecutive basic morphological transformations, an erosion operation followed by a dilation operation. The closing, on the other hand, is a dilation operation followed by an erosion operation. The equations of morphological erosion and dilation operations are stated in Eqs. (4) and (5) respectively.

\[ A \ominus B = \{ z | (B)_z \subseteq A \} \]  

(4)

\[ A \oplus B = \{ z | (B)_z \cap A \subseteq A \} \]  

(5)

A represents the set of foreground pixels and \( B \) is the structuring element. \( z \)'s are foreground values (1's represent foreground pixels and 0's represent background pixels). The equation of erosion inside the set of curly braces indicates that the erosion of \( A \) by \( B \) is the set of all points \( z \), such that \( B \), translated by \( z \), is contained in \( A \). This formulation results in the new foreground pixels of the image. Similarly, the dilation of \( A \) by \( B \) is the set of all displacements, \( z \), such that the foreground elements of \( B \) overlap at least one element of \( A \) (Gonzalez & Woods, 2018).

Opening mostly helps to remove noise in the background and closing mostly helps to remove noise in the foreground. As a result, we get a binary image map. After that, we apply the 'logical AND' operation on each of the three color channels of the original image and the binary image map i.e. if the pixel value to the image map is 0, then we set that pixel value of each three-color channel to 0. Otherwise, we keep the pixel value as it is. Finally, we get a well-segmented color image where the background is removed by black pixels, and the foreground is preserved as it is.

3.1.2. Data augmentation

The datasets we are working with are relatively small. The character dataset comprises 1005 images and the digit dataset comprises 1075 images. But for deep learning, we need a larger dataset to get good results. For this purpose, we used the data augmentation technique to produce a larger dataset. In real-world applications, the input data can be very different from the training data. Usually, training data are created in an ideal condition. But the training data does not always reflect the same condition as real-world natural data. As a result, it is usually ideal to augment the dataset to a margin so that it can replicate real-world natural data as accurately as possible while also reducing the model’s overfitting on the training set. The different types of augmentation techniques that are applied in our system include zoom, shear, horizontal flip, height and width shift, etc.

3.1.3. Data scaling

The stochastic gradient descent is used as the optimization technique in our model. Gradient descent is highly sensitive to the feature value. A high range of values of features causes irregular gradient descent updates. To ensure the update operation to be smooth, we normalize the feature value to a smaller range of 0 to 1 rather than a larger range of 0 to 255. For this, we multiplied all the pixels’ values by 1/255. It helps the algorithm to reach near the minima point in a lesser number of steps thus fastening the process.
6

### 3.2. Model description

A convolutional neural network-based approach is used for feature extraction in our system. CNN learns different features from images while training. Moreover, we applied the idea of transfer learning to assign the pre-trained weight in the model which helps enhance the overall performance.

#### 3.2.1. Convolutional Neural Network

Convolutional Neural Network is a bio-inspired process that took motivation from the organization of the human visual cortex. The process revolves around a mathematical operation called convolution operation which operates on two functions, a portion of digital image and a feature detector called filter, resulting in another output function. The overall performance.

#### 3.2.2. Random forest

The random forest algorithm can be seen as an improvement of another supervised machine learning algorithm called the decision tree. The decision tree uses the concept of the n-ary tree to solve classification and regression problems. Each of the leaf nodes in the tree corresponds to a class label. Features are represented on the internal nodes that are called decision nodes that help generate predictions from a series of feature-based splits. Random forest uses a collection of decision trees to build the forest that overcomes the limitation of the decision tree algorithm which has the tendency to overfit training data. Random forest builds multiple random decision trees and merges them together to make the model more accurate and less sensitive to training data. A random forest model is trained with the ‘Bagging method’ which is a combination of two steps, bootstrapping and aggregation.

Let, the training set $X = \{x_1, x_2, ..., x_n\}$ where $x_i$ is a single training sample and corresponding label set, $Y = \{y_1, y_2, ..., y_n\}$ where $y_i$ is the label of $x_i$'th training sample. Bootstrapping generates $m$ new random training sets each of size $n$ by random sampling with a replacement where repeated selections are allowed from the original training set $X$. This step ensures that we are using different datasets for every tree which makes the model less sensitive to training data. After that, $m$ decision trees are built using each of these random datasets separately with a random subset of features considered from the set of all features. This step of choosing a random subset of features is called ‘feature bootstrapping’ which helps reduce the correlation between trees. These random decision trees are collectively labeled as the random forest. The prediction for a new data point is calculated using the aggregation technique where the data point is passed to each of the random decision trees. In the case of regression, after training, for a new test sample $x$, the resulted prediction is calculated by averaging the output predictions.

**Table 1** Summary of different extracted layers used in the proposed model.

<table>
<thead>
<tr>
<th>Backbone networks</th>
<th>Extracted layers</th>
<th>Pre-trained</th>
<th>Character dataset</th>
<th>Parameters</th>
<th>Digit dataset</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>VGG16</td>
<td>block5_pool</td>
<td>ImageNet</td>
<td>64 x 64 x 3</td>
<td>17,501,888</td>
<td>128 x 128 x 3</td>
<td>23,793,344</td>
</tr>
<tr>
<td>VGG19</td>
<td>block5_pool</td>
<td>ImageNet</td>
<td>64 x 64 x 3</td>
<td>22,811,584</td>
<td>128 x 128 x 3</td>
<td>29,103,040</td>
</tr>
<tr>
<td>InceptionV3</td>
<td>mixed10</td>
<td>ImageNet</td>
<td>64 x 64 x 3</td>
<td>24,589,984</td>
<td>128 x 128 x 3</td>
<td>30,881,440</td>
</tr>
<tr>
<td>Xception</td>
<td>block14_seppool</td>
<td>ImageNet</td>
<td>64 x 64 x 3</td>
<td>40,425,896</td>
<td>128 x 128 x 3</td>
<td>55,105,960</td>
</tr>
<tr>
<td>ResNet50</td>
<td>conv5_block3_out</td>
<td>ImageNet</td>
<td>64 x 64 x 3</td>
<td>32,666,368</td>
<td>128 x 128 x 3</td>
<td>57,832,192</td>
</tr>
</tbody>
</table>

Fig. 3. Illustration of the proposed model architecture based on transfer learning.

Different convolutional layers extract different types of features, such as initial layers of convolution are known to capture low-level features like edges of images, and later convolutional layers are known to capture high-level composite features. The process helps reduce the number of trainable parameters and successfully analyzes position and orientation dependency in an image. We used five different powerful CNN models pre-trained on ‘ImageNet’ as the backbone of our sign language recognition system.
produced by individual decision trees. On the other hand, in the case of classification, the resultant prediction is calculated by majority voting where the prediction of the maximum number of random decision trees is taken as the resultant prediction.

3.2.3. Backbone networks

VGG16 (Simonyan & Zisserman, 2014) is a CNN-based networks taking motivation from LeNet-5 architecture. VGG16 is a 16-layer neural network with 13 convolutional layers and 3 fully connected layers. Five max-pooling layers have a pooling window of size 2 × 2 pixel. All the convolutional filters are of size 3 × 3 and stride of 1. VGG19 (Simonyan & Zisserman, 2014) is quite similar to architecture of VGG16. The main difference is the number of convolutional filters is 13 for VGG16 and 16 for VGG19.

The specialty of the inception network (Szegedy, Vanhoucke, Ioffe, Shlens, & Wojna, 2016) is the inception modules that ensure efficiency by keeping the number of parameters small without hurting the performance. Instead of picking the size of the convolutional filter by hand, it uses various sized filters along with pooling in the same layer. Factorizing large-size filters into multiple smaller filters to reduce computation is also a noticeable contribution by the Inception network.

Xception network (Chollet, 2017) is a modification of inception network. In the case of the Xception network, the inception module from the inception network is replaced by depth-wise separable convolutions. Depth-wise separable convolutions are depth-wise convolutions followed by point-wise convolutions. A depth-wise convolution is applied to a single channel at a time rather than all the channels. Then a point-wise 1 × 1 convolution is applied to these channels. This change enhances the performance of the network on the ImageNet dataset. As both architectures have the same number of parameters, it can be concluded that the enhancement of performance is due to the more efficient use of model parameters.

Training very deep neural networks sometimes cause exploding or vanishing gradient problem that results in degradation of accuracy. That is where ResNet network (He, Zhang, Ren, & Sun, 2016) comes in handy with the idea of residual blocks or shortcut connections that take the advantages of residual learning. The main idea is to take the activation from one layer and feed it directly to a deep layer of the network. That allows very deep layers of a neural network to learn significant complex patterns successfully.

3.2.4. Transfer learning

The main idea of transfer learning (Pan & Yang, 2009) is to use the knowledge gained from one domain to perform another related task of a similar domain. It is often looked at as an optimization where machines can use previously learned knowledge or experience to solve a problem of similar feature-space distribution. Though it does not guarantee enhanced performance unless the knowledge comes from solving a generalized task. The modern world is producing an abundance of data but there is a scarcity of the labeled ones. Manually generating a large amount of labeled data is expensive and sometimes transfer learning is the only way when we do not have enough data to train a model.

Let, the domain be a two-element tuple, \( D = (X, P(X)) \), where \( X \) is the feature space which is the set of all feature vectors of the training element. \( P(X) \) is a marginal probability distribution and \( X \) is a sample point where \( x = \{x_1, x_2, \ldots, x_n\} \) and \( x_i \) is the \( i \)th feature vector corresponding to a sample data point. Similarly, let the task be a two-element tuple, \( T = (Y, f(x)) \), where \( Y \) is the label space, set of all labels. \( f(x), f : X \rightarrow Y \) is the objective predictive function that is not observed but learned from training using data points. The training data comprises of the collection of pairs \( \{x_i, y_i\} \), where \( x_i \in X, y_i \in Y \).

The objective predictive function \( f(x) \) is also denoted as a conditional probability distribution, \( P(y|x) \) from the probabilistic viewpoint. Given, a source domain \( D_S \) and a target domain \( D_T \), a source learning task \( T_S \) and a target learning task \( T_T \), where \( D_S \neq DT, or \ T_S \neq TT \), the goal of transfer learning is to improve the learning of the target objective predictive function \( f_T(x) \) in \( DT \) using the knowledge from \( DS \) and \( TS \).

There are generally two conventional approaches for using transfer learning. The first one is to define a custom CNN model architecture and train it with a large dataset of the same domain that performs a related task. Then, re-train the model with the dataset of the target task to get the final model. The other more popular approach that we used for our model is to select a pre-trained model created by researchers that perform greatly in benchmark datasets. The model is then fine-tuned accordingly to the target task and a portion of the model or the whole model is re-trained with the dataset of the target task to get the final model.

CNN is a great addition to the field of computer vision and deep learning. There are no standardized rules globally accepted for defining a specific kind of model. Scientists did years of experiments and research and came up with some powerful CNN models, such as VGG16, VGG19, ResNet50, InceptionV3, Xception, etc. These models when pre-trained on benchmark datasets such as ImageNet to initialize weights of trainable parameters become a generalized powerful image classifier. Then, these models are trained again using a specific dataset after necessary modification in architecture to enhance performance even more. Finally, the model becomes a specialized classifier of the dataset.

3.2.5. Model architecture

Different CNN architectures such as VGG16, VGG19, Xception, InceptionV3, and ResNet50 with ‘ImageNet’ weight are applied as the backbone networks which take the advantages of transfer learning and extract useful features from images. We extracted layer till block5_pool, block5_pool, mixed10, block14_sepool2_act, and conv5_block3_out from VGG16, VGG19, InceptionV3, Xception, ResNet50 respectively for both character and digit recognition. Brief information about the backbone networks and extracted layers used for character and digit recognition is stated in Table 1. Some layers of the models were modified to fit our dataset. The input layer and the output layer (softmax) layer were changed, all the fully connected layers from the backbone network were removed and some custom fully connected layers were added according to our dataset. Lastly, a random forest classifier was added for the classification task. The architecture of the proposed model is presented in Fig. 3.

3.3. Model optimization and training

3.3.1. Stochastic Gradient Descent

Stochastic Gradient Descent (SGD) is one of the most common and popular optimization algorithms for optimizing neural networks. Although SGD means using an example at a time to calculate the gradient, it is also used interchangeably to refer to other kinds of gradient descents, mini-batch, and batch gradient descent. It is often used to minimize multi-variable objective function \( F(x) \) which is differentiable. To be more precise, it is the 1st order repetitive optimization technique to find a local minima of a differentiable function. It does so by taking iterative steps towards the direction of the negative gradient of function \( F(x) \). The main focus of the whole optimization process is to minimize the loss function. Eq. (7) states a single update of gradient descent. The gradient is denoted by the sign ‘\( \nabla \)’, which is a measure of the rate of change of a variable with respect to another variable. After a single update from a point \( a_n \), it gives a point \( a_{n+1} \) where \( F(a_{n+1}) \leq F(a_n) \).

\[
a_{n+1} = a_n - \gamma \nabla F(a_n) \tag{7}
\]

Here, \( \gamma \) is called the learning rate that defines the size of each step which requires careful choice of value.
3.3.2. Categorical cross-entropy

Categorical Cross-Entropy (CE) is an algorithm to calculate the loss of a multi-class classifier model, where the goal is to classify a single class from some number of classes. Here, the loss value represents the measure of the difference between two probability distributions. The higher the loss value, the larger the difference. Eq. (8) states the formula for calculating categorical cross-entropy loss. The two probability distributions are the distribution of the training dataset and the distribution of the model. The model is optimized accordingly using the loss value.

\[ CE = - \sum_{i=1}^{C} t_i \log(s_i) \]  
\[ J = \frac{1}{m} \sum_{i} CE_i \]  

Here, \( C \) is the number of total classes, \( t_i \) is the ground truth and \( s_i \) is the CNN predicted score. It is also called log loss. It is preferred over other loss functions such as squared error loss because it gives a convex optimization problem. The significance of the minus sign is that the loss gets smaller when the ground-truth value and the CNN predicted score is closer to each other. The loss is calculated to compute the overall cost \((J)\) of the model which is illustrated in Eq. (9). The loss function calculates the error for a single training example whereas the cost function evaluates performance error on the entire training set.

3.3.3. Model training and parameters

Consider, \( N \) training samples such as \( \{ (x_i, y_i) \}_{i=1}^{N} \) where each sample \( x_i \in \mathbb{R}^d \) is assign a label \( y_i \in \{ 1, 2, \ldots, l_N \} \). Every \( l_j \) represents a one-hot encoded vector such as \( l_j = [l_{j,1}, \ldots, l_{j,j}, \ldots, l_{j,l_N}] \in \mathbb{R}^{l_N} \). In this vector, only the \( j \)th element is set to one and other elements contain zeros. Here, \( N_c \) denotes the number of classes or categories in the training set. The model is trained with mini batch technique where the initial weight is assigned from the pre-trained model previously trained on the ImageNet. Through training, the re-learned features set \( Q = [q^{(1)}, q^{(2)}, \ldots, q^{(m)}] \) is fine-tuned for feeding the RF classifier.

In the training phase, the model was trained with SGD optimizer while keeping the initial learning rate \( 1 \times e^{-2} \) and categorical cross-entropy was used to calculate the loss. The model was trained for 300 epochs as after that the loss curves become approximately flat. The optimizer, loss function and other model hyper-parameters were chosen through trial and error and parameter tuning.

4. Evaluation metrices

Generally, ‘accuracy’ is considered to be the single most popular metric to evaluate a model performance. However, it can be tricky sometimes and accuracy alone cannot establish a model’s re-

\[ Accuracy = \frac{(TP + TN)}{(FP + FN + TP + TN)} \]

5. Experiments

NumPy, Pandas, Scikit-learn, TensorFlow (version: 2.0), and Python 3 (version: 3.6) have been utilized to develop the proposed system. The system is implemented on a core-i7 processor with NVIDIA GeForce GTX 1050 Ti GPU (CUDA Cores: 768) and 16 GB of physical memory. The developed models are also tested using the ‘Google Colab’ platform, which makes use of the NVIDIA Tesla K80 GPU.

5.1. Dataset collection and description

The models were tested on two different datasets. Accessing both datasets from Islam, Mousumi, Jessan and Hossain (2018), we obtained 1075 and 1005 image samples for Bangla sign digit and Bangla sign characters respectively. For digit dataset (Ishara-Bochon), each image has a resolution of 128 \( \times \) 128 pixels and is saved as a .jpg file. The character dataset (Ishara-Lipi) originally had over 1800 samples, however, it currently contains only 1005 samples. Bengali language contains 50 basic characters (11 vowels and 39 consonants) (Reza & Khan, 2012). However, some of the characters are not used as much as others. The character dataset contains 36 Bengali sign characters (6 vowels and 30 consonants) featuring most of the important phonemic characters where each of the 36 character classes includes on average 28 sign images. The images are taken from various angles, lighting conditions, and hand positions. These images are colored (RGB) images of size 64 \( \times \) 64 pixels each and saved in .jpg format. Examples of each class for both datasets are presented in Fig. 4 with appropriate labels.

5.2. Result analysis and discussion

Five different extended and modified pre-trained models were used to train both digit and character sign datasets. The training was performed separately on two different datasets.
To verify the effectiveness of the RF classifier on the features extracted by different pre-trained models, we trained different models without and with the RF classifier, as shown in Tables 2 and 3 respectively. In most cases, models with the RF classifier perform better than models without the RF classifier. The best performing model from Table 2 is VGG19 (batch size: 50) for character recognition and ResNet50 (batch size: 30) without RF and Xception (batch size: 50) without RF classifier for digit recognition. For character recognition, the hybrid model (VGG16 + RF) with batch size 40 achieved the best results with accuracy, precision, recall, and f1-scores of 97.33%, 97.71%, 97.33%, and 97.31% respectively. Besides, this model takes a reasonable training time of 820.46 s and 2858.73 s for character and digit recognition respectively.

Though both the character and the digit datasets are from the same authors, they are two separate datasets with different features and characteristics. Thus, their distribution is also different. As shown in Table 2, VGG19 performs best for character recognition. On the other hand, ResNet50 performs best for digit recognition. VGG16 and VGG19 were able to obtain good results in character recognition, with accuracies of over 80% for nearly all batch sizes and models based on other backbone networks failed to produce good results, with most accuracies falling below 80%. For digit recognition, all the models based on different backbone networks with various batch sizes perform satisfactorily. The worst performing models are ResNet50 (batch size: 60) without RF and Xception (batch size: 50) without RF classifier for character and digit recognition respectively as they achieve the least performance metrics i.e. accuracy, precision, recall, and f1-scores of (66.67%, 69.91%, 66.67%, 65.30%) and (87.33%, 89.12%, 87.33%, 87.33%) for character and digit recognition respectively.

Gradient-weighted Class Activation Mapping or Grad-CAM is a heatmap visualization process for a given class categories. Table 4 indicates the Grad-CAM visualization by highlighting the important regions of the images. It is the localization map that allows us to see where the model is looking in order to categorize the images. The heatmaps of the Grad-CAM are superimposed on the original images for better understanding.

Fig. 5 illustrates the ‘accuracy vs. epoch’ and ‘loss vs. epoch’ graphs for different backbone networks. From the graphs, it is evident that all the models based on different backbone networks are free of overfitting as the accuracy is increasing and loss is decreasing with the increase of each epoch. Besides, the ROC curve (Receiver Operating Characteristic curve) and precision-recall curve for the proposed model are shown in Fig. 6. For most of the categories, the area covered by both the ROC curve and precision-recall curve is 1.00 or very close to 1.00, which signifies the high ability of our model to distinguish between different classes. It proves that our model is robust enough to give a good evaluation for test samples. t-Distributed Stochastic Neighbor Embedding (t-SNE) is a non linear technique for visualizing high dimensional data. It represents each high-dimensional entity with a two or three-dimensional point in such a way that comparable things are described.

To see the detailed results for different combinations of backbone networks and different batch sizes, please refer to Table 2.
Table 3
Detailed results from the hybrid model for different combinations of backbone networks and batch sizes. The hybrid model is developed by feeding the features of different layers from various backbone networks to the Random Forest (RF) classifier.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Batch size</th>
<th>Character dataset</th>
<th>Digit dataset</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Accuracy (in %)</td>
<td>Precision (in %)</td>
</tr>
<tr>
<td>Hybrid (VGG16 + RF)</td>
<td>10</td>
<td>88.89</td>
<td>91.59</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>88.33</td>
<td>91.95</td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>89.00</td>
<td>91.87</td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>91.67</td>
<td>93.64</td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>91.67</td>
<td>93.64</td>
</tr>
<tr>
<td></td>
<td>60</td>
<td>88.89</td>
<td>91.05</td>
</tr>
<tr>
<td></td>
<td>70</td>
<td>85.00</td>
<td>87.80</td>
</tr>
<tr>
<td>Hybrid (VGG19 + RF)</td>
<td>10</td>
<td>88.89</td>
<td>90.31</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>83.33</td>
<td>87.59</td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>89.44</td>
<td>91.28</td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>87.78</td>
<td>89.70</td>
</tr>
<tr>
<td>Hybrid (InceptionV3 + RF)</td>
<td>10</td>
<td>88.89</td>
<td>90.31</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>83.33</td>
<td>87.59</td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>89.44</td>
<td>91.28</td>
</tr>
<tr>
<td>Hybrid (Xception + RF)</td>
<td>10</td>
<td>88.89</td>
<td>91.05</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>85.00</td>
<td>87.80</td>
</tr>
<tr>
<td>Hybrid (ResNet50 + RF)</td>
<td>10</td>
<td>88.89</td>
<td>91.05</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>85.00</td>
<td>87.80</td>
</tr>
</tbody>
</table>

Table 4
Grad-CAM visualization of different images showing important area of image for classification where $S_n$ represents different test samples.

<table>
<thead>
<tr>
<th>Image Label</th>
<th>Character Dataset</th>
<th>Digit Dataset</th>
</tr>
</thead>
<tbody>
<tr>
<td>Background</td>
<td>$S_1$</td>
<td>$S_2$</td>
</tr>
<tr>
<td>Eliminated</td>
<td>Images</td>
<td>$S_3$</td>
</tr>
<tr>
<td></td>
<td>$S_4$</td>
<td></td>
</tr>
<tr>
<td>Heatmap</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Grad-CAM</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 5
The detailed comparison between the existing systems and the proposed BSL recognition system is depicted in Table 5. Many of the systems have chosen neural network as the classifier. It is also evident from Table 5 that accuracy tends to fall when the number of classes increases. Systems working with only digit recognition were able to achieve higher accuracy than the systems working with character recognition. According to Table 5, VGG19 (batch size: 50) offers an accuracy of 97.33% for numerals and 91.11% for characters. On the other hand, the accuracy of the proposed hybrid model (VGG16 + RF) is 97.33% for numerals and 91.67% for characters. First of all, our goal is to develop a generalized model that can produce higher accuracy for both datasets. Secondly, we also have to consider the number of parameters that the model needs to learn during the training phase to produce the result. From Table 5, it is evident that the total trainable parameters for VGG16, VGG19 and ResNet50 are 17,501,888, 22,811,584, 32,666,368 for character dataset and 23,793,344, 29,103,040, 57,832,192 for digit dataset respectively. For both datasets, VGG16 has fewer trainable parameters than VGG19 and ResNet50. As shown in Tables 2 and 3, it is also apparent that the proposed model takes less training time. To be specific, the training time for the proposed model is $820.46$ s for...
numerals and 2858.73 s for character dataset, whereas, the training time for individual VGG19 (batch size: 50) and ResNet50 (batch size: 30) is 830.92 s for numerals and 3140.90 s for character dataset which are significantly longer than the training time of the proposed model. As the proposed model produces satisfactory accuracy for both datasets with less trainable parameters and less training time, the proposed hybrid model (VGG16 + RF) is better than the individual VGG19 or ResNet50 model.

The proposed system outperforms most of the existing systems in performance. Due to the unavailability of the complete Ishara-Lipi character dataset (Islam, Mousumi, Jessan, Rabby and Hossain, 2018; Islam, Mousumi, Rabby et al., 2018), we used currently existing 1005 samples. If we compare other systems that used the same dataset as ours, the proposed system surpasses Hasan and Ahsan (2019) and Islam, Mousumi, Rabby, Hossain et al. (2018) in digit recognition by achieving an accuracy of 97.33% whereas Hasan and Ahsan (2019) and Islam, Mousumi, Rabby, Hossain et al. (2018) achieved an accuracy of 95% and 94.74% respectively. In the case of character recognition, the proposed system achieved 91.67% accuracy and Islam, Mousumi, Rabby et al. (2018) achieved 92% accuracy. Although Islam, Mousumi, Rabby et al. (2018) obtained better accuracy than the proposed system, we must keep in mind that the proposed system utilized only (1005/1800) × 100 = 55.83% of the original Ishara-Lipi character dataset and reached (91.67/92)×100 = 99.64% of the system's accuracy proposed by Islam, Mousumi, Rabby et al. (2018), as shown in Table 6 and Fig. 8. Similarly, we analyzed other systems that exceed the proposed system and summarized the results in Table 6. From Table 6, we can undoubtedly say that the proposed system employed a limited amount of dataset and achieved most of the other system's accuracy.

The major goal of this research is to use an effective background elimination strategy and present a hybrid model that combines transfer learning and random forest to get better results. By automating the process of categorizing sign language, this model might serve as a bridge between the normal and hearing-impaired communities. If the hearing impaired and normal people can work together in the same place with no communication gap, a huge socio-economic change can happen.

6. Conclusion

This paper presented a hybrid transfer learning-based deep learning system with the RF classifier for Bangla sign language recognition. In the pre-processing phase, the proposed background elimination algorithm removes unnecessary background information from images, allowing the model to focus on the region of interest and thus improve the model’s inference speed. The proposed system provides a reliable architecture for smaller datasets as it uses pre-trained models, at the same time, the fine-tuning process with the lower learning rate enhances the model’s performance while preventing overfitting. Furthermore, the RF classifier utilizes random subsets of the fine-tuned features and averages several decision trees to reduce the variance of the final model caused by a single decision tree.

All of these aspects contributed to the development of the system that outperformed many of the state-of-the-art recognition systems in terms of accuracy. However, we feel that the results of character recognition could be improved. The data pre-processing such as background elimination produces better outcomes than data with no pre-processing. In the future, we plan to build a smartphone-based system for the hearing impaired community. For that, a reduced model architecture with a small set of parameters will be required to cope with the smartphone’s limited computational resources and power. We also have a plan to create a high-quality, complete, and open-source BSL dataset for future researchers.
Fig. 6. Multi-class ROC AUC and multi-class precision-recall AUC on Ishara-Lipi character and Ishara-Bochon digit dataset respectively where, the area under the curve (AUC) is a scalar statistic to evaluate the model. The ROC plot is generated by projecting the true-positive rate on the y-axis and the false-positive rate on the x-axis. On the contrary, the precision-recall plot is produced by projecting the precision on the y-axis and the recall on the x-axis.

Fig. 7. Color coded t-SNE visualization for both character and digit dataset.

Fig. 8. Visual comparison between the proposed system and other systems that used same datasets.
Table 5
Comparison among existing and the proposed BSL system for Bengali sign language recognition. The average accuracy is expressed as a percentage for ease of understanding.

<table>
<thead>
<tr>
<th>Developed BSL systems</th>
<th>Methodology</th>
<th>Dataset</th>
<th>Class</th>
<th>Average accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Islam, Mousseni, Rabby, Hossain et al. (2018)</td>
<td>CNN</td>
<td>Ishara-Bochon: 1075 samples</td>
<td>10 numerals</td>
<td>95%</td>
</tr>
<tr>
<td>Islam, Mousseni, Rabby et al. (2018)</td>
<td>CNN</td>
<td>Ishara-Lipi: 1800 samples</td>
<td>36 characters</td>
<td>92%</td>
</tr>
<tr>
<td>Rahman et al. (2012)</td>
<td>ANN and image processing</td>
<td>828 samples (train: 540, test: 288)</td>
<td>36 alphabets</td>
<td>80.90%</td>
</tr>
<tr>
<td>Uddin et al. (2017)</td>
<td>YCrC color model, Bag of features and SVM</td>
<td>800 samples</td>
<td>15 characters</td>
<td>86%</td>
</tr>
<tr>
<td>Rony et al. (2018)</td>
<td>CNN, Inception-v3</td>
<td>2050 samples (alphabets: 1900 samples, special characters: 150 samples)</td>
<td>38 alphabets, 3 special characters</td>
<td>92.85%</td>
</tr>
<tr>
<td>Jarman et al. (2015)</td>
<td>Fingertip finder algorithm with multilayered feed forward neural network</td>
<td>2300 samples</td>
<td>46 hand gestures (9 numerals, 9 vowels, 28 consonants)</td>
<td>86.69%</td>
</tr>
<tr>
<td>Hossen et al. (2018)</td>
<td>Deep CNN</td>
<td>1147 samples</td>
<td>37 characters</td>
<td>84.68%</td>
</tr>
<tr>
<td>Rafi et al. (2019)</td>
<td>VGG19</td>
<td>12,581 samples</td>
<td>38 alphabets</td>
<td>89.60%</td>
</tr>
<tr>
<td>Abedin et al. (2021)</td>
<td>Concatenated BdSL network (CNN based image network + pose estimation network)</td>
<td>12,581 samples (Train: 9959, validation: 1102, test: 1520)</td>
<td>38 alphabets</td>
<td>91.51%</td>
</tr>
<tr>
<td>Tasnere et al. (2020)</td>
<td>CNN</td>
<td>1674 samples (Train: 1066, validation: 271, test: 337)</td>
<td>10 numerals</td>
<td>97.63%</td>
</tr>
<tr>
<td>Hasan and Ahsan (2019)</td>
<td>HOG feature and multi-class Support Vector Machine</td>
<td>Ishara-Bochon: 1000 samples (train: 900, test: 100)</td>
<td>10 numerals</td>
<td>94.74%</td>
</tr>
<tr>
<td>Basnin et al. (2021)</td>
<td>Integrated CNN–LSTM</td>
<td>13,400 samples</td>
<td>36 alphabets</td>
<td>88.5%</td>
</tr>
<tr>
<td>Tabassum et al. (2020)</td>
<td>HOG features and KNN</td>
<td>1400 samples</td>
<td>35 alphabets</td>
<td>91.1%</td>
</tr>
<tr>
<td>Ahmad et al. (2019)</td>
<td>CNN</td>
<td>3200 samples</td>
<td>10 numerals</td>
<td>92%</td>
</tr>
<tr>
<td>Haque et al. (2019)</td>
<td>Principal Component Analysis (PCA) and KNN algorithm</td>
<td>130 samples</td>
<td>26 alphabets</td>
<td>77.88%</td>
</tr>
<tr>
<td>Shanta et al. (2018)</td>
<td>Scale Invariant Feature Transform (SIFT) and CNN</td>
<td>1700 samples</td>
<td>38 characters</td>
<td>90.63%</td>
</tr>
<tr>
<td>Khan et al. (2019)</td>
<td>CNN and customized ROI Segmentation</td>
<td>500 training samples, after augmentation: 4000 samples</td>
<td>5 sign gestures</td>
<td>9%</td>
</tr>
</tbody>
</table>

Ours: Best result from Table 2 | VGG19 (batch size: 50) | Ishara-Bochon: 1075 samples | 10 numerals | 97.33% |

Ours (proposed): Best result from Table 3 | Hybrid (VGG16 + RF) | Ishara-Bochon: 1075 samples | 10 numerals | 97.33% |

Table 6
Relative comparison between the proposed BSL system with other systems that ostensibly exceeds our system in performance with respect to percentage of dataset and percentage of accuracy.

<table>
<thead>
<tr>
<th>BSL systems</th>
<th>Recognition</th>
<th>% of dataset</th>
<th>% of accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Islam, Mousseni, Rabby et al. (2018)</td>
<td>Character</td>
<td>55.83</td>
<td>99.64</td>
</tr>
<tr>
<td>Tasnere et al. (2020)</td>
<td>Digit</td>
<td>64.22</td>
<td>99.69</td>
</tr>
<tr>
<td>Rony et al. (2018)</td>
<td>Character</td>
<td>49.02</td>
<td>98.73</td>
</tr>
</tbody>
</table>

CRediT authorship contribution statement

Sunanda Das: Conceptualization, Methodology, Investigation, Software, Validation, Writing – original draft, Writing – review & editing.
Md. Samir Imtiaz: Resources, Software, Data curation, Visualization.
Nieb Hasan Neom: Editing.
Sunanda Das: Conceptualization, Methodology, Investigation, Software, Validation, Writing – original draft, Writing – review & editing.
Md. Samir Imtiaz: Resources, Software, Data curation, Visualization.
Nieb Hasan Neom: Editing.
Hui Wang: Writing – review, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The dataset used in this research is an open-source dataset.

References

Berlitz (2021). The most spoken languages in the world.


