Sign Language Recognition: A Comparative Analysis of Deep Learning Models

Aswathi Premkumar, R. Hridya Krishna, Nikita Chanalya, C. Meghadev, Utkrist Arvind Varma, T. Anjali, and S. Siji Rani

Abstract Sign language is the primary means of communication used by deaf and dumb people. Learning this language could be perplexing for humans; therefore, it is critical to develop a system that can accurately detect sign language. The fields of deep learning and computer vision with recent advances are used to make an impact in sign language recognition with a fully automated deep learning architecture. This paper presents two models built using two deep learning algorithms; VGG-16 and convolutional neural network (CNN) for recognition and classification of hand gestures. The project aims at analysing the models' performance quantitatively by optimising accuracy obtained using limited dataset. It aims at designing a system that recognises the hand gestures of American sign language and detects the alphabets. Both the models gave excellent results, VGG-16 being the better. VGG-16 model delivered an accuracy of 99.56% followed by CNN with an accuracy of 99.38%.

Keywords American sign language · Deep learning · Sign language · VGG-16 · Convolutional neural network · Neural network · Feature extraction

1 Introduction

Sign language is a method by which the deaf and/or dumb individuals communicate through visual gestures. It is expressed using hand gestures, movements, orientation of palm and face expressions executed by humans. It is the most expressible way for communication for the individuals with hearing or speech impairment. Sign languages have their sign grammar and lexicon. It is not common universally, and thus, each country has its own sign language system. There are about 150 sign languages as per the 2021 edition of Ethnologue. American sign language (ASL) is one of the most leading sign languages of the deaf and dumb individuals of USA as well as of Anglophone Canada. ASL uses ASL manual alphabet/ASL fingerspelled alphabet. Fingerspelling is the method in which a particular word is spelled out

A. Premkumar · R. Hridya Krishna (✉) · N. Chanalya · C. Meghadev · U. A. Varma · T. Anjali · S. Siji Rani
Department of Computer Science and Engineering, Amrita School of Engineering, Amrita Vishwa Vidyapeetham, Amritapuri, India

S. Smys et al. (eds.), *Inventive Computation and Information Technologies*, Lecture Notes in Networks and Systems 336, https://doi.org/10.1007/978-981-16-6723-7_1

using hand gestures. Each sign shown corresponds to a letter of the word. One of the challenges faced is that normal people find it difficult to understand the gestures of sign language, thus making communication burdensome. Deep learning models have given efficient results in sign language recognition from hand gesture images [1]. Much research has been done in deep learning to find an efficient method of sign language detection. There are various neural networks such as the convolutional neural network which comprises of various layers such as convolutional layers, pooling layers and fully connected layers, fully convolutional network (FCN) in which all learnable layers are convolutional, thus having lesser number of parameters and maintaining the spatial information of the input hand gestures images. Considering the importance of sign language and its efficient recognition to help deaf and dumb individuals to communicate with society, comparative research was conducted in two different deep learning modes, namely VGG-16 and a CNN model, by training and testing each model with hand gesture images.

2 Related Works

American sign language recognition and detection is not a novel concept. Over the past two decades, researchers have made use of various classifiers that belong to different categories such as linear classifiers, Bayesian, neural networks (Table 1).

3 Dataset

The initial step of the proposed system is to collect the data. The dataset consists of 17,113 American sign language hand gesture images from 27 classes (26 alphabets + 1 space class denoted as '0') out of which 12,845 images were used for training and 4268 images for validation (Fig. 1).

4 Data Augmentation

Image data augmentation technique is done using the ImageDataGenerator class imported from K. This is used in expanding the dataset in order to boost the performance and strengthen the model to generalise. Thus, more data result in better accuracy and efficiency of the model.

Table 1 Related works

References	Classification model	Focus
[2]	SqueezeNet architecture	The system uses RGB images which are then used to train the SqueezeNet architecture, so that it could be run on mobiles
[3]	SVM and ANN	Develops a system for Indian sign language recognition using feature extraction techniques, scale invariant feature transform (SIFT) and histogram of oriented gradients (HOG), and then classifies
[4]	Support vector machine (SVM)	Researches explain skin segmentation can be completed using YCbCr systems. This was then classified using SVM
[5]	Histogram matching and ORB algorithm	Developed an android application that captures the hand gesture and detects the sign into digits and alphabets. Proposed methodology involves preprocessing the real-time image, recognising gestures using histogram matching and ORB algorithm
[6]	PNN and KNN	Hand gestures are recognised and translated into text and speech (Hindi as well as English) by using MATLAB. The classification is done using two models and results compared
[7]	SVM and ANN	Involves a review of various steps involved in hand gesture recognition. The methods used for data acquisition, image processing, segmentation and feature extraction were compared. And, the models were then classified
[8]	HMM	Involves comparison of various vision based sign recognition systems, which mostly uses HMM as base. Detailed common challenges of these models
[9]	CNN (SignNet)	A CNN model, SignNet, is proposed by combining high-and low-resolution network CNNs. It works at various spatial resolutions and detects hand gesture images using a synthetic dataset

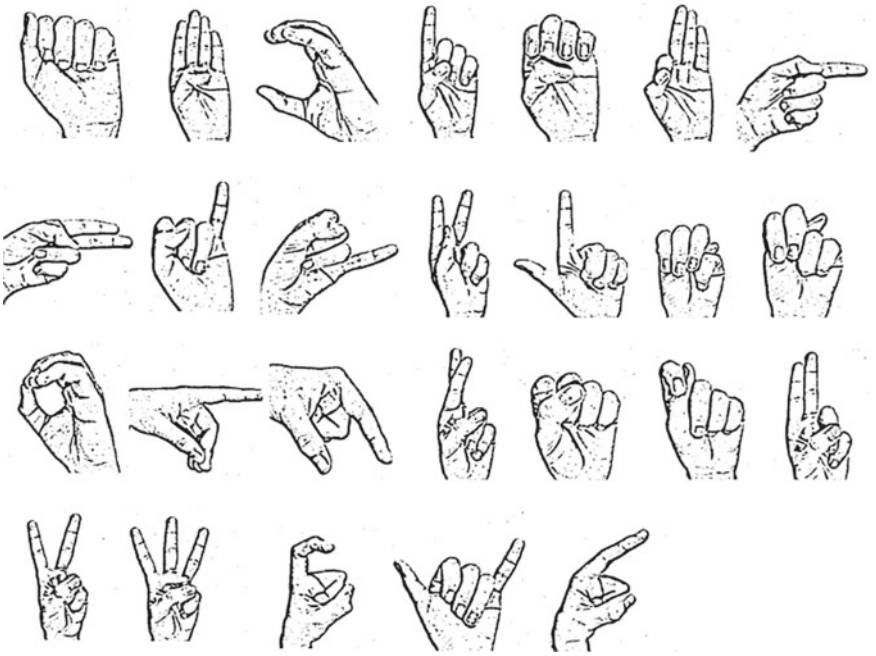

Fig. 1 Dataset

5 Methodology

Two deep learning models were developed for the problem statement, and a comparative analysis was performed on these models based on their training statistics and results.

5.1 Model 1: CNN Model

Introduction

Convolutional neural network (CNN) is a class of deep neural networks that is used in object detection, image recognition, image classification, etc. [10, 11] CNN architecture has a similar architecture to the nerve cells that communicate with the interconnected neurons in the body. The important core layers in the CNN architecture include Input, Padding, Convolution + Activation/ReLU, Pooling, Flatten/Dense, Fully Connected + Softmax. In CNN, the input image is assigned importance to certain features in it to differentiate one from another. Each input image was passed through a series of convolutional layers followed by chosen parameters, and filters

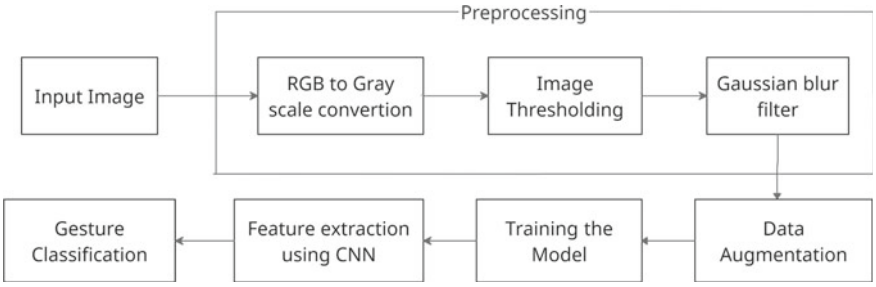

Fig. 2 CNN flow process

with strides were applied and padded whenever required. It gives the highest accuracy in comparison with other image processing algorithms.

Flow Process

A 2D convolutional neural network (CNN) with tensor flow library was developed for training the required models in order to do the detection.

Steps for classification (Fig. 2): Step I: Importing Keras libraries and packages. Step II: Data loading and preprocessing. Step III: Building CNN model with two convolutional layers with rectified linear unit (ReLU), which is the activation function that extracts different features of the input, two MaxPooling layers to gradually decrease the spatial size of the image representation in order to decrease the number of parameters, thus decreasing computational complexity in the model network, the flatten layer, and then, finally, a fully connected layer in which the last dense layer has Softmax as activation function which will execute the classification based on extracted features. Then, the CNN model was compiled with loss 'categorical_crossentropy' and 'adam' as optimiser. Step IV: Using ImageDataGenerator to apply transformations and augmentation on images for training.

Preprocessing

Preprocessing of coloured images is necessary to extract features from the images. The coloured images were first converted to grayscale as the grey scale images are easier to process than coloured images which will take plenty of resources and time for training the data. Image thresholding was applied to detect the boundaries, thus separating the required object from the background pixels. This was followed by the application of a Gaussian blur filter, which helps to reduce the random noise and for cutting the extreme outliers [12].

Architecture

The images pass through the following layers (Fig. 3): (i) one convolutional layer of size $126 \times 126 \times 32$, succeeded by a pooling layer of size 2×2 that decreases the height and width of the image into $63 \times 63 \times 32$; (ii) one convolutional layer of size $61 \times 61 \times 32$ succeeded by a pooling layer of size 2×2 that further decreases the height and width of the image into $30 \times 30 \times 32$; (iii) flatten layer; (iv) one dense

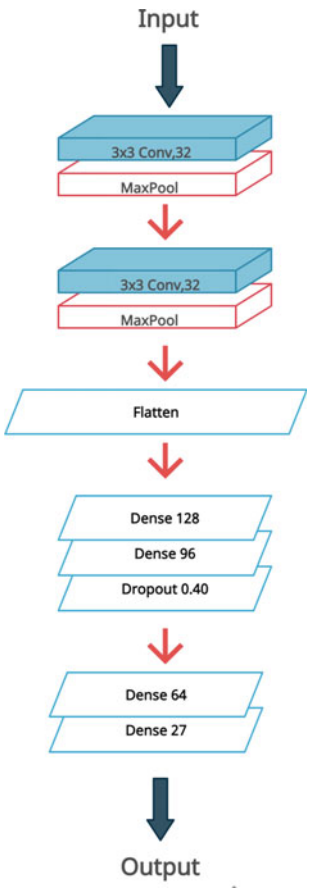

Fig. 3 CNN network architecture

layer with 96 units and along with a dropout after that of 96 units; (v) two dense layers, first one with 64 units and the second one with 26 units, 1 for each class. The flatten layer and dense layer reduce the data into one dimension and identify the class into which it belongs.

5.2 Model 2: VGG-16

Introduction

VGG-16 (OxfordNet), which stands for Visual Geometry Group, is an object recognition model in deep learning [13]. It is a convolutional neural network architecture that is 16 layers deep. The default input size of this model is 224 × 224 pixels with three channels for the image in RGB format [14]. The receptive field used by the

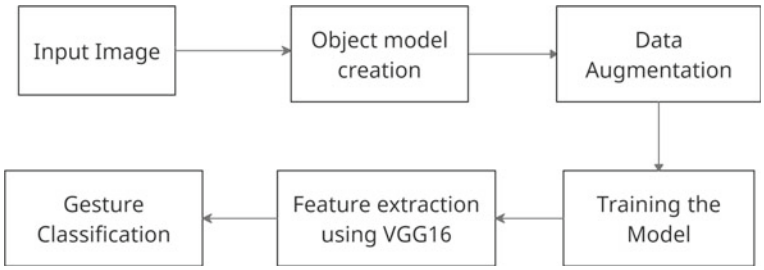

Fig. 4 VGG-16 flow process

convolutional layer in the VGG-16 model is very small, i.e., 3 × 3. To retain the spatial resolution after convolution, the convolution stride is set to 1 pixel. It has pooling layers of size 2 × 2. VGG-16 has three fully connected layers. The ReLU activation unit is used by all hidden layers.

Flow Process

The steps for classification are (Fig. 4): Step I: Imported Keras libraries and packages. Step II: Loaded the dataset. Step III: The VGG-16 model was built. Convolutional layers with small size convolution filters were added so as to have a large number of weighted filters. After each block of convolutional filters, one max pooling layer was added which helps to reduce the amount of data sent to the next layer by a factor of 4. After the 5 blocks of convolution and pooling layers, the flatten layer was added in order to change the data dimension into a one-dimensional input into the dense layers. Lastly the dense layers were added in which the neurons of different layers are connected into a network. A sigmoid activation unit is added to the last dense layer. Step IV: Use ImageDataGenerator to apply transformations and augmentation on images for training.

Architecture

VGG-16 architecture (Fig. 5) comprises 13 convolutional layers and 3 fully connected layers, hence not a fully convolutional network.

The images pass through the following layers:

(i) two convolutional layers each of size 224 × 224 × 64 succeeded a pooling layer of size 2 × 2, which thus decreases the image size into 112 × 112 × 64; (ii) two convolutional layers each of size 112 × 112 × 128, succeeded by a pooling layer of size 2 × 2 that further decreases the image size into 56 × 56 × 128; (iii) three convolutional layers each of size 56 × 56 × 256 succeeded by a pooling layer of size 2 × 2 that further decreases the image size into 28 × 28 × 256; (iv) three convolutional layers each of size 28 × 28 × 512 succeeded by a pooling layer of size 2 × 2 which further decreases the image size into 14 × 14 × 512; (v) three convolutional layers each of size 14 × 14 × 512 succeeded by a pooling layer of size 2 × 2 which further decreases the size of the image into 7 × 7 × 512; vi) flatten layer; and (vii) three dense layers, first two with 4096 units and the last with 26 units,

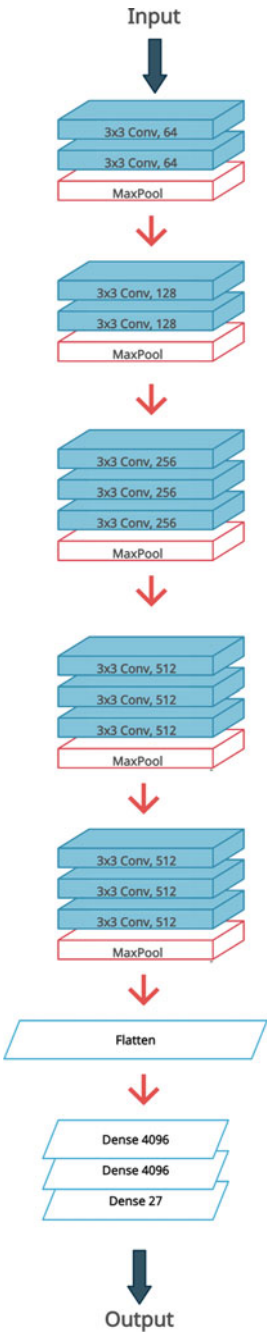

Fig. 5 VGG-16 network architecture

1 for each class. The flatten layer and dense layer reduce the data into one dimension and identify the class into which it belongs.

6 Result

After training the model, a set of validation images were passed to test the prediction. Then, the performances of both the models were optimised by selecting the appropriate number of epochs and steps per epochs. More epochs give better accuracies, but it could possibly increase the complexity of the model too. For the CNN model, 30 epochs and 200 steps per epoch were used, and for VGG-16 model, 10 epochs and 100 steps per epoch were applied which gave the best results that balanced between accuracy and complexity.

6.1 CNN Model

The test set consists of 4268 images. The proposed model was trained using 30 epochs. From the tests done, an accuracy of 99.38 was obtained for the validation set. The accuracies in various epochs varied from 57.10 to 99.95%. An evaluation on these gave an average of 99.38%.

The train and test accuracy as well as losses were plotted across the number of epochs (Figs. 6 and 7, respectively). Both the accuracies escalated as the number

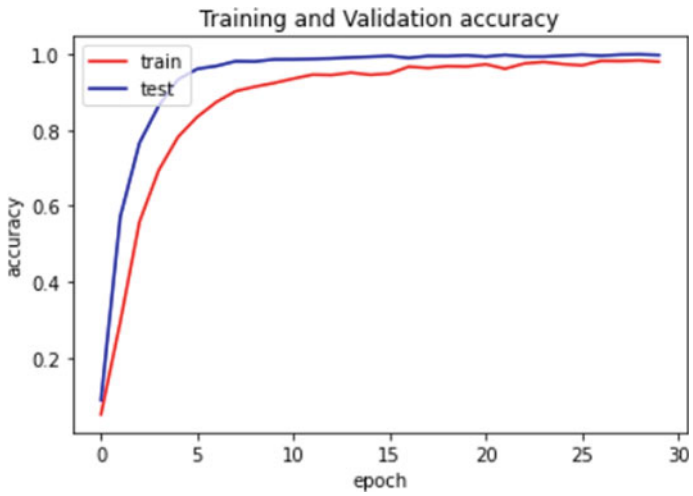

Fig. 6 Plot of estimated training and validation accuracy (CNN model)

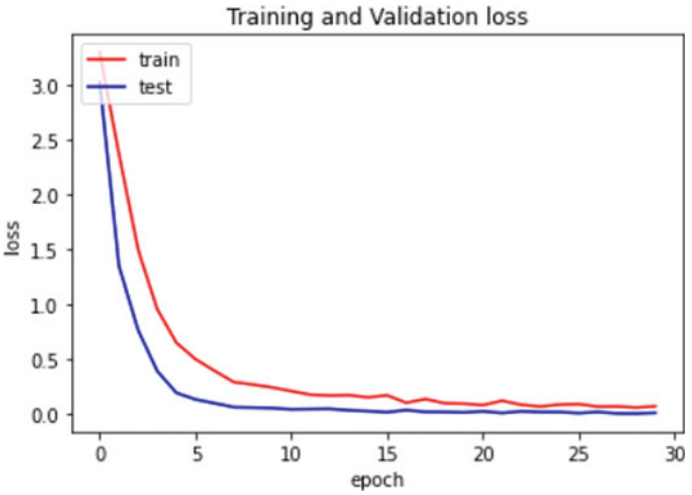

Fig. 7 Plot of estimated training and validation loss (CNN model)

of epochs progressed. The number of losses of both training and testing reduced in subsequent epochs.

The plot of training and testing accuracy (Fig. 6) showed that the accuracy of prediction got higher for higher epochs. The validation accuracy at the first epoch was found to be 57.10%, and it increased sharply till the 5th epoch, with an accuracy of 93.32% after which it increased smoothly till a maximum of 99.95%.

The validation loss decreased sharply (Fig. 7) from the first epoch till the 6th epoch after which it decreased smoothly.

6.2 VGG-16 Model

The validation set (400 images) was passed through the model, and accuracies were optimised by using 10 epochs. As a result, an accuracy of 99.56% was obtained. The accuracies in various epochs varied from 95.88 to 100%. An evaluation on these gave an average of 99.56%. The accuracies of the training and validation were plotted and evaluated (Figs. 8 and 9, respectively). The graphs accuracy of both training and testing increased with increase in the number of epochs. The losses occurred during training and testing were also plotted and evaluated. The losses of both training and testing decreased as the number of epochs got higher.

The plot (Fig. 8) shows that the validation accuracy at the first epoch was found to be 98.62%, and it increased till the 3rd epoch. It then gave an accuracy of 95.88% in the 4th epoch after which it increased sharply reaching a maximum of 100%. The validation loss decreased sharply after the first epoch (Fig. 9).

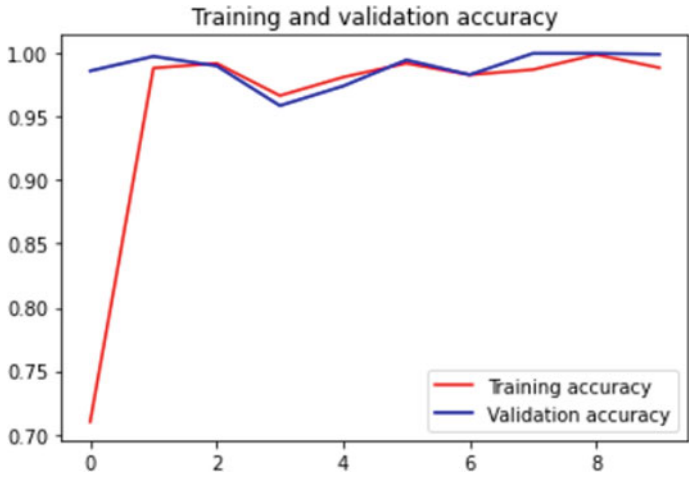

Fig. 8 Plot of estimated training and validation accuracy (VGG-16 model)

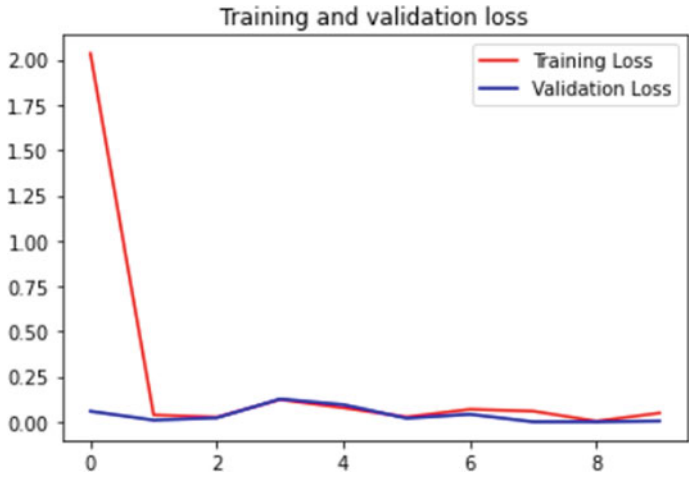

Fig. 9 Plot of estimated training and validation loss (VGG-16 model)

7 Conclusion and Future Works

Considering the complexities of various combinations of hand gestures and its understandability to normal people, there are many challenges in this domain. With this project, two efficient deep learning models were analysed for American sign language detection and the best one was thus recognised. The accuracy of both the models improved as the number of epochs got higher. This is because for each subsequent epoch, the neural network updates the weight estimated in the first epoch with the

values that reduce overall loss. Choosing an appropriate number of epochs, balancing complexity and accuracy produced excellent results. The two models gave excellent results in recognising the gestures correctly. The VGG-16 model with fixed residual blocks gave a better result compared to the CNN model built from scratch. The VGG-16 model fits the data more accurately as there are more weighted layers in VGG which thus had much more parameters which extracted features better than the CNN model, thus classifying better.

The project focuses on recognising the hand gestures that correspond to the alphabet from A to Z using the dataset containing hand gesture images. This project can be developed further to detect images in real time. This will involve the use of LeapMotion API that would help in real-time data generation from hand gestures of people. This could also be developed to recognise hand gestures for words and numbers along with the finger spelling. The project has a scope in further development by using various other models, such as ResNet, which has been proved to be faster and efficient due to its deeper layers.

Acknowledgements The authors feel obliged in taking the opportunity to sincerely thank Anjali T. (Assistant Professor, Computer Science Engineering, Amrita School of Engineering, Amritapuri) for assisting time to time throughout the project duration as well as Amrita Vishwa Vidyapeetham University to provide the golden opportunity to work on this outstanding project on the topic sign language recognition. The authors are overwhelmed with gratitude and humility and acknowledge gratitude to all those who have assisted in bringing these thoughts and ideas far beyond the simplicity and into something substantial.

References

1. R.C. Gonzalez, R.E. Woods, *Digital Image Processing*, 2nd edn. (Prentice Hall, New Jersey, 2008), p. 693
2. N. Kasukurthi, B. Rokad, S. Bidani, A. Dennisan, *American Sign Language Alphabet Recognition using Deep Learning*. arXiv:1905.05487 [cs.CV] (2019)
3. J. Ekbote, M. Joshi, Indian sign language recognition using ANN and SVM classifiers, in *2017 International Conference on Innovations in Embedded and Communication System (ICIIECS)* (2017)
4. S. Lahoti, S. Kayal, S. Kumbhare, I. Suradkar, V. Pawar, Android based American sign language recognition system with skin segmentation, in *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)* (2017)
5. M. Mahesh, A. Jayaprakash, M. Geetha, Sign language translator for mobile platforms, in *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)* (IEEE, 2017), pp. 1176–1181
6. U. Patel, A.G. Ambedkar, Moment based sign language recognition for Indian language, in *2017 International Conference on Computing, Communication, Control and Automation (ICCUBEA)* (2017)
7. M.J.Z. Omar, M.H. Jaward, A review of hand gesture and sign language recognition techniques. Int. J. Mach. Learn. Cyber. **10**, 131–153 (2019)
8. N. Aloysius, M. Geetha, Understanding vision-based continuous sign language recognition. Multimed. Tools Appl. **79**, 22177–22209 (2020)

9. N. Aloysius, M. Geetha, An ensembled scale-space model of deep convolutional neural networks for sign language recognition, in *Advances in Artificial Intelligence and Data Engineering Advances in Intelligent Systems and Computing*, vol 1133, eds. by N. Chiplunkar, T. Fukao (Springer, Singapore, 2021)
10. J. Herazo, *Sign Language Recognition Using Deep Learning* (2020)
11. D.A. Sharath Kumar, Sign language recognition with convolutional neural network. Int. Res. J. Eng. Technol. (IRJET) (2020)
12. S.A. Khan, A.D. Joy, S.M. Asaduzzaman, M. Hossain, *An efficient sign language translator device using convolutional neural network and customized ROI segmentation*, in *2019 2nd International Conference on Communication Engineering and Technology (ICCET)*, pp. 152–156 (2019)
13. H. Qassim, A. Verma, D. Feinzimer, Compressed residual-VGG16 CNN model for big data places image recognition, in *2018 IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC)*, pp. 169–175 (2018)
14. S. Masood, H.C. Thuwal, A. Srivastava, American sign language character recognition using convolutional neural network, in *Smart Computing and Informatics. Smart Innovation, Systems and Technologies*, vol. 78, eds. by S. Satapathy, V. Bhateja, S. Das (Springer, Singapore, 2018)