To Improve Accuracy of Indian Sign Language Recognition Using Custom Convolutional Neural Network model

Dr P Praveen[1], B Anu[2], P Shiva Sowmya[3], G Anjali Kiran[4], G Sai Srujan[5]

1. *Associate Professor in School of Computer Science and Artificial Intelligence, SR University, Warangal, India.*
2. *School of Computer Science and Artificial Intelligence, SR University, Warangal, India.*
3. *School of Computer Science and Artificial Intelligence, SR University, Warangal, India.*
4. *School of Computer Science and Artificial Intelligence, SR University, Warangal, India.*
5. *School of Computer Science and Artificial Intelligence, SR University, Warangal, India.*

ARTICLE INFO

Received:

Received in revised form:

Accepted:

Available online:

Keywords: Deep Learning, Convolutional Neural Network, Indian Sign Language Recognition, Open CV, Gaussian Blur, Grayscale, Normalization.

ABSTRACT

One of the vital challenges that deaf people face in the present society is communication. The use of sign language translators is not a viable solution. Recently Deep learning models showcased outstanding results in classification and recognition systems. Therefore, researchers are focusing more on developing deep learning models with at most accuracy. The study aims to represent a Convolutional neural network-based Indian Sign Language identification model for deaf persons and further convert it into the text, which is more accurate than prior studies. In this study, a CNN-based, ISL hand- gesture recognition model is developed, which outperforms many other existing models. A custom dataset was prepared, to train the model. The developed model can recognize 26 alphabets represented through hand gestures in ISL. We found that the model produced an accuracy of 95%.

Introduction

According to Census, India has over one million deaf people and over ten million people with hearing loss. To aid in the teaching and interpretation of sign language, a variety of programs have been developed. However, progress in recognizing sign languages with modern technology has been promising but very limited. Software that can recognize and interpret sign language is in high demand. It can also act as the bridge between sign language users and non-sign language users [1-4]. As a result, it is critical to establish the link between the deaf and the spoken by creating a deep learning model to translate sign language to text. Humans can communicate with others in different ways for example, speaking in some languages like telugu, hindi and many more. Sign language Recognition will reduce the communication gap between normal people and deaf or muted people.

Sign language is the combination of static and dynamic gestures. Static hand gestures don't require any hand movement [8,11]. The 26 alphabets, ten digits, and a few static word signs make up sign language. Each country employs its own as there is no universally acknowledged sing language [9]. Sign languages include American Sign Language, British Sign Language, Indian Sign Language, and many more. Even though the majority of deaf people in India uses Indian sign language, it is not widely employed in schools. Many organizations that work with deaf people have made efforts to promote Indian Sign Language, but most people are unaware of these signals due to their complexity, making communication between the deaf and

Corresponding Author: ……., Email:……

the spoken difficult. Throughout the previous few decades, studies have predominantly focused on the identification of American Sign Language [5,7]. American Sign Language represents the alphabet with a single hand, whereas Indian Sign Language (ISL) represents the alphabet with both hands [6,14].

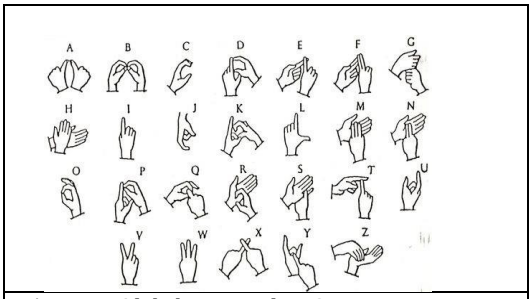

Figure 1. Alphabets in Indian Sign Language

Materials and Methods

Many researchers have conducted studies on Sign Language Recognition throughout the last few decades. Based on previous research, we found that there are two existing methodologies. They are glove-based systems and image processing and recognition. Many early Sign Language Recognition systems used data-gloves and accelerometers to acquire details of the hands [10,13]. The measurements were measured directly using Data Glove. In a glove-based system, a person has to wear a glove that consists of sensors like a flex sensor, which tracks the motion of the hand. It compares with the data received directly from sensors based on our finger movement [15]. It also contains an accelerometer and five flex sensors. The accelerometer detects hand movements, which generates the initial bit of binary number to be compared in a lookup table, which determines the language. For recognition, the data glove provided analog signal data to the microcontroller [16]. Finally, a pre-recorded voice matched with a recognized indication was used to demonstrate the outcome.

The authors, Borghetti, Sardini, and Serpelloni, created a numerical databased sign language recognition system. It consists of a 3-axis accelerometer and Hall sensors. The data glove made use of four Hall sensors that were attached to the fingers. An accelerometer is used to detect finger bending, while Hall sensors are used to monitor hand orientation. The analog sensor data sent into MATLAB code is used to identify indications given by the gloves. Only digits ranging from 0 to 9 were studied in this experimental setup. In digit recognition, the created system achieved 96 percent accuracy.

Figure 2. Example of Data Glove

The authors, Muthu and Gomathi, proposed hybrid CNN-RNN architecture to create a real-time Indian sign language (ISL) detection system. The ISL dataset was used to train the system. On the test data, the proposed model achieved 95.99%. Kishore, Prasad M, V, Prasad C.R, and Rahul suggested a 4-camera model for segmenting and classifying hand motions using features extracted from elliptical Fourier descriptors. Their system had a recognition rate of around 92.23 percent. Raghuveera, Deepthi, Mangalashri, Akshaya proposed a model to recognize ISL singlehanded signs, double-handed signs, and fingerspelling signs from 4,600 photos, with a 71.85% accuracy. Based on 3D skeletal point data from the Kinect sensor and SVM, few researchers recognized 37 Indian Sign Language signs. Furthermore, each of these sensors has advantages and disadvantages in terms of moving data. All of these methods, however, relied on sensors to detect the indications.

The image processing and recognition approach captures photos with a camera, compares them to existing images, and detects the sign a still hand picture frame is captured using a webcam. The processing of these frames improves them. Using feature extraction and classification techniques, the sign language is then translated into English text. This translation is converted to speech using the text-to-speech API. Before doing feature extraction, the images need to be processed such that only valuable information is analyzed while redundant, distracting noise and superficial data are ignored. For faster computation, the images are first reduced to 100 x 100 pixels [17]. The picture is then transformed to grayscale before being translated to binary [6]. The YCbCr model is used to detect skin color simultaneously. Finally, an edge detector, called a Canny edge detector, is used to detect edges [18].

Figure 3. Example of Kinect color and depth video stream

Recently Microsoft Kinect has offered a reasonable depth camera. Because of its capabilities, Kinect is now widely employed by scientists. Kinect is capable of delivering both color and depth video streams at the same time, and background segmentation is simple to accomplish with depth data. Kinect is used to recognize signs. Recently there are no data sets accessible as the outcomes are limited.

Data Collection

For Indian Sign Language recognition, to obtain good accuracy we have collected our own dataset through webcam using open CV. To collect the dataset, we have created two folders, train and test folders. The images present in train and test folder are used for training and testing the built model.

The train and test folders contains the sub folders, named as A, B, C……Z. These are the classes we have to predict. Any image, which is captured, belongs to one of these classes. Folder A consists of images in which, alphabet A is represented in Indian Sign Language. In addition, all other folders consists of their respective images.

Figure 4. Dataset collection

Fig.4 is the output display screen of data collection. The green square shown in the figure is region of interest; the sign that we represent with our hands should be placed in that region itself. Only the image covered in that region will be saved. While capturing the image, to save the image into its corresponding folder we have to press the alphabet to which it belongs, through keyboard.
In the left side of figure 3 we have labels like A: 90, B: 51 and so on, these are the count of images present in each folder. According to figure 3, we have 90 images in folder A, 51 images in folder B, etc. In the same way, data is collected for both testing and training.

Data Preprocessing

Before storing the image into its respective folder, the image is resized to 224 x 224 pixels and a RGB image (colored image) is transformed to grayscale image.

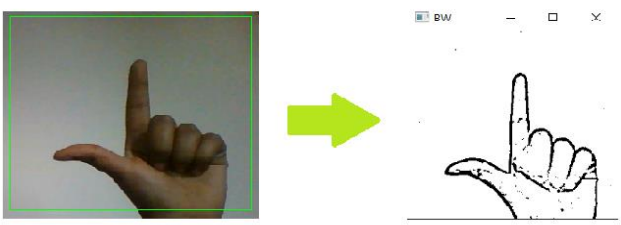

Figure 5. Image Preprocessing

As shown in the figure 5, after capturing the image through webcam, the RGB (colored image) is transformed into a grayscale image. In the grayscale image, the outline of the hand gesture is produced using the Gaussian blur approach and by applying threshold for the image.
Normalization, a preprocessing technique is applied to all the images before training or testing the model. Using this technique all pixel values in an image are ranged between 0 to 1. This is done through rescaling the image. To rescale the image, 225, in an image the maximum pixel value is 225, divide each pixel value.

Constructing the models

We developed a CNN model by adding three convolutional layers. The first layer consists of 32 convolutional filters with size 3x3 and with activation function ReLu. The pooling layer in which max-pooling operation is used follows this convolutional layer. In pooling layer, size of each filter is 2x2. The depth of the input image for the pooling layer determines the number of filters in the pooling layer. The output image of the convolutional layer functions as the pooling layer's input image. The number of filters employed in the convolutional layer will determine the depth of the output image of the convolutional layer. The number of filters in the pooling layer is indirectly equivalent to the number of filters in the convolutional layer. That is the cause we do not mention the number of filters in the pooling layer.

The architecture of the second and third convolutional layers is the same, with 64 convolutional filters in each. Each filter is 3x3 in size and has the relu activation function. Following each convolutional layer is a pooling layer identical to the first pooling layer.
The last pooling layer's output image is flattened and added to the fully connected network. Five hidden levels and one output layer make up a fully connected network. The first hidden layer contains 128 neurons, the second contains 122 neurons, 96,80, and the 64 neurons in the remaining hidden layers respectively. The ReLu activation function is employed in all of the hidden layers.

26 neurons are present in the output layer because we have 26 alphabets to predict. And we used the Softmax activation function in the output layer. The softmax activation function is widely used for categorical classification problems and we are dealing with a categorical classification problem, we used the softmax activation function.

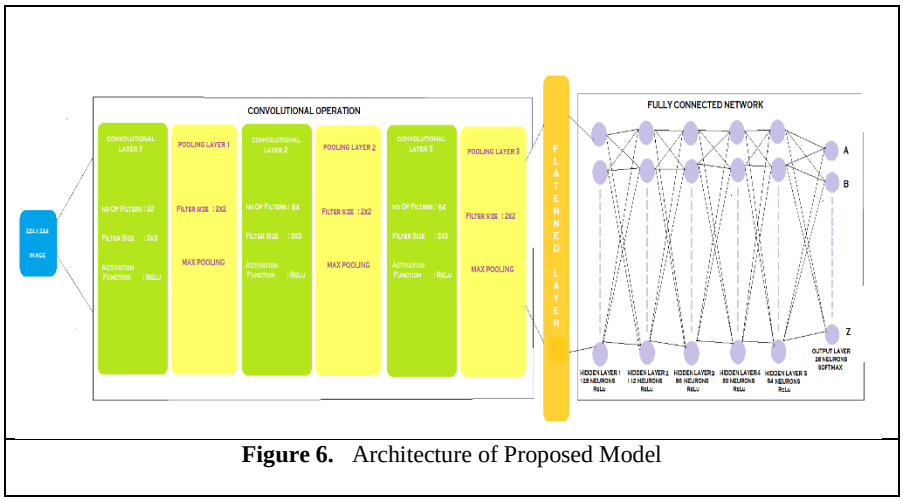

Figure 6. Architecture of Proposed Model

Implementation

In the proposed model, we 3 convolutional layers along with pooling layers. In 3 convolutional layers we used activation function as ReLu (Rectified Linear Unit). Convolutional layer produces a output image, and for each pixel of output image ReLu operation is applied.

(1) Output = max (0, input)

Max pooling operation is used in pooling layers. This max pooling operation considers the maximum value in the whole filter space as output. The output image size of pooling layer will be

(2) Output image size = $[\ \frac{n-f}{s}+1, \frac{n-f}{s}+1, d]$

In the above equation, n represents the size of input image to the pooling layer, f represents the filter size in pooling layer, s represents the stride value and d represents the depth of the input image to pooling layer.

While training the model we use an optimizer, for reducing the training time we use an optimizer. In our model, we used Adam as optimizer; Adam (Adaptive moment estimation) is formed by combining other two-optimization algorithms, RMSProp (Root Mean Square Propagation) and Momentum. In Adam optimizer the weights and bias parameters are updated as follows

$$(3)\ W = \mathrm{W} - \alpha \frac{\mathrm{V_{dw}}}{\sqrt{\mathrm{S_{dw}} + €}} \qquad (4)\ B = \mathrm{B} - \alpha \frac{\mathrm{V_{dB}}}{\sqrt{\mathrm{S_{dB}} + €}}$$

Where α represents learning rate, W represents weight , B represents bias parameter , €=10^{-8} and
V_{dw} = $\beta_1 V_{dw}$(prev) + (1- β_1) dW where β_1= 0.9
S_{dw} = $\beta_2 S_{dw}$(prev) + (1- β_2)$(dW)^2$ where β_2=0.999
V_{dB}= $\beta_1 V_{dB}$ (prev) + (1- β_1) dB where β_1= 0.9
S_{dB}= $\beta_2$$\mathrm{S_{dB}}$(prev) + (1- β_2)$(\mathrm{dB})^2$ where β_2=0.999

Initially, random weights and bias parameters will be considered, later on in each and every iteration the weights and bias parameters will get updated according the mentioned formulas, which helps the model in increasing its accuracy rate. And as the Adam is formed by combination of both the algorithms Momentum and RMSProp, it has the advantage of both these algorithms. Thus the Adam optimizer works better than many other optimization techniques.

Results and Discussion

Dataset

We collected approximately 6500 images with the help of laptop, webcam and opencv library. As there are many datasets available in internet. But they were not reliable. And our dataset is divided into 2 parts in the ratio 70:30. Training dataset contains 5200 images and testing dataset contains 1300 images.

Alphabet	Train Dataset	Test Dataset	Total
A	250	50	300
B	250	50	300
C	250	50	300
D	250	50	300
E	250	50	300
F	250	50	300
G	250	50	300
H	250	50	300
I	250	50	300
J	250	50	300
K	250	50	300
L	250	50	300
M	250	50	300
N	250	50	300
O	250	50	300
P	250	50	300
Q	250	50	300
R	250	50	300
S	250	50	300
T	250	50	300
U	250	50	300
V	250	50	300
W	250	50	300
X	250	50	300
Y	250	50	300
Z	250	50	300

Table 1. The above table depicts the number of images captured for each alphabet in training and testing datasets. This table gives clear information about the count of images in training and testing datasets which will be very useful in further steps.

Sample Datasets:

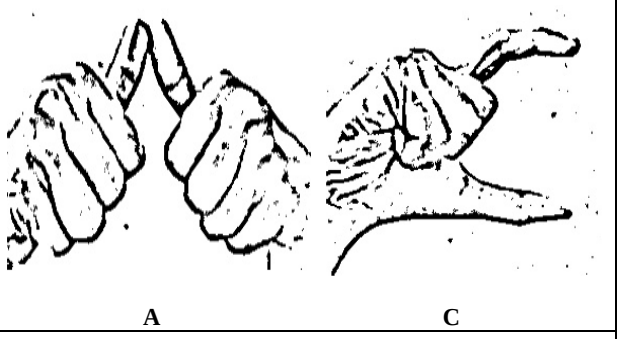

Figure 7. Alphabets in sign language

The above figure represents the Indian sign language for the alphabets A and C. These are the final outputs of the original images, stored in respective folders and given as input to custom CNN model. Actually, the original image is converted into gray scale image where the coloured images turns to black and white images which has 2 channels. Then after, we apply Gaussian blur that removes the high frequency components and thresholding is applied on the produced output.

Procedure

Training and Testing accuracy are the main factors for evaluating the performance of our proposed model. In order to get good accuracy, the model has been trained for different epochs. To train the data, we used custom CNN model. Finally, we observed that our model is performing well at batch size=10, epochs=,30. Our model is predicting the sign language with an accuracy of 98

BATCH SIZE 10

MODEL ACCURACY in %

Epochs	Train accuracy	validation accuracy
1	49.7	99.7
5	96.1	99.5
10	98.3	100
15	98.2	100
20	98.3	100
25	98.3	100
30	98.4	100

Table 2. Training and Validation accuracies at different epochs.

MODEL LOSS

Epochs	Train accuracy	Validation accuracy
1	1.64	0.02
5	0.12	0.02
10	0.06	0.02
15	0.03	0.02
20	0.03	0.02
25	0.01	0.02
30	0.01	0.02

Table 3: Train and Validation loss at at different epochs.

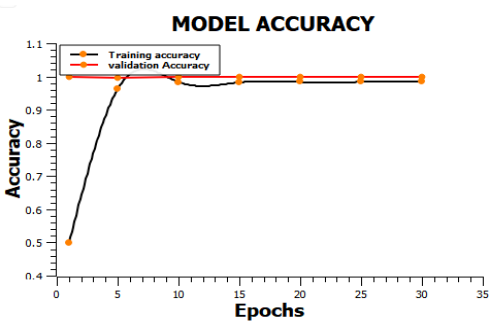

Figure 8. Graph between number of epochs and accuracy

Figure 8 gives information about the model accuracy and epochs. The accuracy value differs from one epoch to another. In the figure 8, as the epoch value increasing, accuracy is also increasing. This is one of the way to tune our model to get good results.

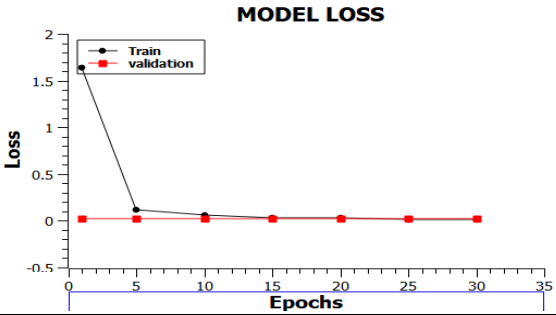

Figure 9. Graph between number of epochs and Loss

Figure 9 gives information about the model loss and epochs. The loss value differs from one epoch to another. In the figure 9, as the epoch value increasing, loss of data is also decreasing. This is one of the way to tune our model to get good results.

Figure 10. Predicting J

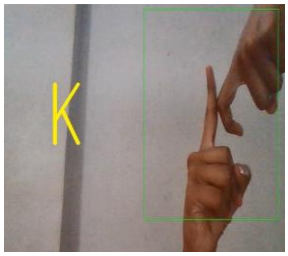

Figure 11. Predicting K

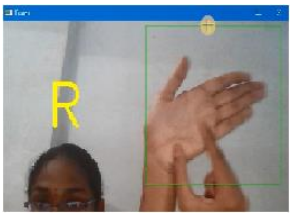

Figure 12. Predicting R

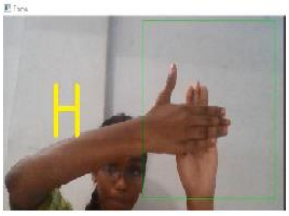

Figure 13. Predicting H

The above figures are the outputs of our proposed model. We can see the alphabet is displaying on the screen for the given input sign. The input sign should be captured in green color box which is of size 224x224.After the completion of image processing the modified image is given as input to CNN model.

Conclusion

The research presents a technique for classifying and recognizing Indian sign language signs using CNN. Our primary objective is to create a better real-time Sign language recognition system so that the system may be employed everywhere. It is accomplished by creating a custom dataset. We have collected our custom dataset and developed a custom CNN model to predict the images. Our model detects the alphabet, which is signed in the Indian signed language. We found the accuracy of our custom CNN model is about 98%. Further, extensions to our project would be converting these predicted alphabets to words and not only detecting static hand gestures, but we can also extend this model to detect dynamic hand gestures. In the future, we can also develop a two-way communication system by converting words to signs which will be very helpful for physically impaired people. A user-friendly web-based application can also be created in future that takes image as input and gives alphabet as output. We can also develop this model by adding speech channel too. After predicting the output, the text will be converted to speech.

References

1. K. Bantupalli and Y. Xie, "American Sign Language Recognition using DeepLearning and Computer Vision," 2018 IEEE International Conference on Big Data (BigData),Seattle,WA,USA,2018,pp. 4896-4899,doi:10.1109/BigData.2018.8622141.
2. W. L. Zheng and B. L. Lu, "Investigating Critical Frequency Bands and Channels for EEG-Based Emotion Recognition with Deep Neural Networks", *IEEE Transactions on Autonomous Mental Development*, vol. 7, no. 3, pp. 162-175, September 2015.
3. Pigou L., Dieleman S., Kindermans PJ., Schrauwen B. (2015) Sign Language Recognition Using Convolutional Neural Networks. In: Agapito L., Bronstein M., Rother C. (eds) Computer Vision - ECCV 2014 Workshops. ECCV 2014. Lecture Notes in Computer Science, vol 8925. Springer, Cham. https://doi.org/10.1007/978-3-319-16178-5_40
4. Jaoa Carriera, A. Z. (2018). Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset. Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on (pp. 4724-4733). IEEE. Honolulu.
5. V. N. T. Truong, C. Yang and Q. Tran, "A translator for American sign language to text and speech," 2016 IEEE 5th Global Conference on Consumer Electronics, 2016, pp. 1-2, doi: 10.1109/GCCE.2016.7800427.
6. H, Muthu & V, Dr. (2021). Indian Sign Language Recognition through Hybrid ConvNet-LSTM Networks. EMITTER International Journal of Engineering Technology. 9. 182-203. 10.24003/emitter.v9i1.613.
7. Kishore P.V., Prasad M.V., Prasad C.R., Rahul R. 4-Camera model for sign language recognition using elliptical fourier descriptors and ANN. 2015 International Conference on Signal Processing and Communication Engineering Systems, Guntur, India, 2015, pp. 34–38.
8. A depth-based Indian Sign Language recognition using Microsoft Kinect / T. Raghuveera, R. Deepthi, R. Mangalashri, R. Akshaya // Sādhanā. – 2020. – Vol. 45, N 1. – P. 34.

9. Y Ji, S Kim and KB Lee, "Sign Language Learning System with Image Sampling and Convolutional Neural Network", *Robotic Computing (IRC) IEEE*, 2017.
10. A. Phinyomark, R. N. Khushaba and E. Scheme, "Feature Extraction and Selection for Myoelectric Control Based on Wearable EMG Sensors", *Sensor*, pp. 1-17, June 2018.
11. S. Kwon and J. Kim, "Real-Time Upper Limb Motion Estimation From Surface Electromyography and Joint Angular Velocities Using an Artificial Neural Network for Human-Machine Cooperation", *IEEE Transactions on Information Technology and Biomedicine*, May 2011.
12. Ankita Wadhawan and Parteek Kumar, "Deep learning-based sign language recognition system for static signs", *Neural Computing and Applications*, vol. 32, no. 12, pp. 7957-7968, 2020.
13. Neel Kamal Bhagat, Y. Vishnusai and G. N. Rathna, "Indian Sign Language Gesture Recognition using Image Processing and Deep Learning", *In 2019 Digital Image Computing: Techniques and Applications (DICTA)*, pp. 1-8, 2019.
14. Imran Khan Mohd Jais, Amelia Ritahani Ismail and Syed QamrunNisa, "Adam optimization algorithm for wide and deep neural network", *Knowl. Eng. Data Sci*, vol. 2, no. 1, pp. 41-46, 2019.
15. H Cooper, B Holt and R. Bowden, "Sign Language Recognition", *Visual Analysis of Humans: Looking at People*, pp. 539-562, 2011.
16. VNT Truong, C Yang and Q. Tran, "A translator for American sign language to text and speech", *2016 IEEE 5th Global Conference on Consumer Electronics*, pp. 1-2, 2016.
17. SM Kamrul Hasan and M. Ahmad, "A new approach of sign language recognition system for bilingual users", *2015 International Conference on Electrical Electronic Engineering (ICEEE)*, pp. 33-36, 2015.
18. MM Islam, S Siddiqua and J. Afnan, "Real time Hand Gesture Recognition using different algorithms based on American Sign Language", *2017 IEEE International Conference on Imaging Vision Pattern Recognition (icIVPR)*, pp. 1-6, 2017.