A Media-pipe integrated deep learning model for ISL (Alphabet) recognition and converting Text to Sound with Video Input

TM Vishnu Mukundan[1], Aryan Gadhiya[1], Karthik Nadar[1], Rishita Gagrani[1], Niha Kamal Basha[1](✉), and Mufti Mahmud[2]

[1] Vellore Institute of Technology, Vellore, TN 632014, India
tm.vishnu.m@gmail.com, ary16.work@gmail.com, karthiknadar1204@gmail.com, rishitagagrani03@gmail.com
niha.k@vit.ac.in

[2] Department of Computer Science, Nottingham Trent University, Nottingham NG11 8NS, UK mufti.mahmud@ntu.ac.uk

Abstract. The present study showcases a novel deep learning-based vision application tasked with reducing the communication gap between sign language and non-sign language users. Speech and hearing impairments are a type of disability that restricts an individual's ability to communicate with others properly. Modern-day automation tools can be used to address this communication gap and allow people to communicate ubiquitously and in a variety of situations. The method defined in the paper involves loading a video file, extracting each frame, and detecting the hand landmarks in each frame using the Media-Pipe library. Then the frame is cropped, and the region of interest is pre-processed and stored in a new data directory for training purposes. The pre-processing involves the use of Gaussian blur, edge detection, morphological transformations, and signal processing functions. Data augmentation is then performed, and images are saved in a new directory. The images are then used to train a custom CNN model, which contains four convolutional layers along with two fully connected layers. The model is compiled using the categorical cross-entropy loss function, optimised using the RMSprop optimiser, and then evaluated using the evaluation metric, accuracy. The predicted sign language alphabet is displayed on the screen and is converted to speech using the Google Text-to-Speech library. The model achieves an overall accuracy of 93.96%. The findings indicate that the proposed approach can serve as a road-map to develop a real-time system capable of sign language recognition and Direct future investigations in this domain.

Keywords: Augmentation · Sign Language to Text conversion · Image Processing · Gesture recognition· Deep Learning · Text to Audio conversion

1 Introduction

Sign language is used as a primary mode of communication by Individuals with auditory impairments [1]. It enables, as well as facilitates, effective communication between hearing-impaired individuals and those who are not. It can also be used by individuals who are not hearing impaired but have communication difficulties due to speech impairments or developmental disorders. The Sign languages used around the world are of many regional types, such as American Sign Language, Auslan (Australian Sign Language), and British Sign Language, among others. In India, ISL or Indian sign language is used throughout the country as a medium of instruction in educational facilities [5]. In sign language, information transfer is facilitated through a combination of hand gestures, facial expressions, and body language. Words or concepts are represented by combining the different hand and finger movements, handshapes, and hand positions. The use of facial gestures and body language also plays a major part in conveying meaning, as they add nuance, emotions, and tone to the signs. Additionally, information can be conveyed through the speed, rhythm, and movement of signs as well.

Indian Sign Language or ISL can be separated from the other notable sign languages through factors like Vocabulary, Grammar etc. [5]. While there exist regional variations in Indian Sign Language, influenced by the spoken languages in India, the alphabet base used in teaching remains largely unchanged. It was estimated that in 2011, around 5 million people in India who identified as deaf or hard of hearing. ISL contains two types of signs, namely, static and dynamic signs. Concepts or words which can be expressed with a static position that does not involve any changes in handshape or movement are referred to as Static signs. On the other hand, Dynamic Signs involve movement or changes in handshape to represent the words or concepts. They may involve compound movements of hands, arms, or body along with the movement of fingers [2]. These gestures are auto-captured with the help of advanced computer vision techniques and systematic learning algorithms to automate sign language-to-text and text-to-sound conversion.

The main contributions of work include data augmentation, pre-processing, and a custom CNN model for sign language detection [14] on which hyperparameter tuning has been performed to efficiently classify the ISL Alphabets using a novel dataset. The dataset is extracted from pre-recorded videos, each segmented into frames and used for training purposes. The recognised symbols are converted to text and then to audio. This paper is organised into different sections where section two as Literature Survey, in which the existing research findings related to Indian sign language recognition have been discussed. In section three-Materials and Methods, the detailed description of data collection proposed work in terms of step-by-step processes such as Data Acquisition, Input Data Generator, CNN model generation and training, Testing the classification using model, and Text to speech [13]. In section four, relevant results on each step have been discussed and analysed. Finally, the conclusion and future work have been depicted in this paper with relevant references.

2 Existing Work

In this section, the major existing research finding on Indian Sign Language have been described in detail. Deep R. Kothadiya et al. [8] employs a vision transformer to classify static Indian sign language. The sign is divided into a sequence of positional embedding patches by the system, which is then fed into a transformer block that consists of a multilayer perceptron network and four self-attention layers. The dataset used is a culmination of a collection of publicly available ISL datasets. The method achieves a significant jump over other top-of-the-line convolutional architectures, and it does so in only the least amount of training epochs to reach around 99.29% accuracy. Different types of image augmentation, such as varied angular positions and brightness levels, can be detected by the proposed methodology. The multi-head attention-based encoding framework achieves good accuracy within a rather small number of training layers and epochs.

Chakraborty et al. examined the use of a MediaPipe Holistic to extract spatial features and different RNN models such as LSTM and GRU to train on temporal features [4]. The dataset used is an American sign language dataset that was recorded using existing videos on the internet, and it consists of 1600 examples. Both the GRU and LSTM models showed 100% accuracy on the training as well as the validation sets and 99% accuracy on the test set after 150 epochs. The proposed solution also provides an effective and accurate way to translate sign language to voice and text. Yasir et al. explored a methodology that employs machine learning to recognize Bangla Sign Language (BdSL) using a combination of Leap Motion Controller (LMC), Hidden Markov Model (HMM), and CNN [24]. The data are obtained from the unbroken hand movements, which are captured using the LMC. Feature extraction also takes place where non-linear features are calculated. Furthermore, the HMM is used to segment the continuous frames into specific states and identify new sign expressions. Finally, CNN is used to train the features and build a multilayer network for each sign expression recognition. The model produces an error rate of 3%, where without distortion, it reduces to 2%, for recognising basic sign expression recognition.

Nandi et al. [10] proposed a finger spelling recognition system of static ISL alphabets using CNN coupled with stochastic pooling, data augmentation methods, dropout, batch normalisation, and DiffGrad optimiser to optimise their model and reduce overfitting. The author creates an ISL dataset using a total of 26 static signs captured from multiple users, resulting in a dataset of 62,400 images to train and validate their model. The proposed model attained a peak training accuracy of 99.76% and a peak validation accuracy of 99.64%. Sharvani et al. [19] administers a model to translate the sign-language gestures in real-time into English using the TensorFlow object detection API. A webcam is used to capture the input, and the data acquisition is performed using Python and OpenCV. The data is then trained using a transfer learning approach where the method makes use of a TensorFlow detection model zoo, which is a pretrained model on the COCO 2017 dataset. The total loss incurred during the training was 0.25. They achieved 85.45% accuracy.

Amrita et al. [20] propose a model that is used to recognise ASL in real time and generate speech. Webcams are used to collect data. The data is pre-processed using the OpenCV and Pillow library of Python. The CNN model is used to train the model. The VGG-16 serves as the foundation for the model, having been pretrained on the ImageNet dataset to achieve a training accuracy of 99.65% and a loss of 0.0259. They achieved 99.62% test accuracy. Byeongkeun et al. Implements a study aimed to create a model capable of recognising American Sign Language fingerspelling in real-time using a CNN trained on depth maps. The dataset used for training the model consisted of 31,000 depth maps acquired with a Creative Senz3D camera that is equipped with a depth sensor [6]. The model is trained using a CafeNet Deep learning framework used in accordance with CNN's. Five convolutional layers, three max-pooling layers, and three fully connected layers form the basis of the model's architecture. A training and validation accuracy of 99.99% was attained by the authors, with samples corresponding to the test subject.

Sanket et al. [23] describe a model to recognise real-time sign language using deep learning. A conjunction of the Roboflow dataset is used along with their personalised dataset. CNN and YOLOv5 are used to train their model. YOLOv5 is employed to expedite the training process as the max pool layer of the CNN was decelerating the training. An accuracy of 53% was achieved from the CNN model, whereas for the YOLOv5 model, 88% accuracy was achieved. Priyadharsini et al. use CNN to transcribe gestures to spoken language, aiming to capture image features and complex nonlinear interactions for better accuracy [15]. A dataset of 280 images was used for sign-language recognition. The CNNs implemented for sign language recognition consists of convolution, pooling, non-linear, and fully connected layers, with pre-processing techniques including cropping, resizing, and Gray scaling. The training phase extracts and stores features of the processed image in a database for each sign, while the testing phase involves matching input images to the database and measuring differences in features for recognition. CNNs were trained with back-propagation, achieving a maximum accuracy of 100% for all alphabets.

Shirbhate et al. [17] developed a project to identify alphabets in ISL using computer vision and machine learning algorithms for those with hearing loss. The classification problem was tackled in three stages: skin segmentation, feature extraction, and supervised learning for classification, with a lack of standard datasets and variance in sign language with locality posing challenges. Skin segmentation was performed using SVM and Random Forest, with SIFT features for effective feature extraction. Multiclass SVMs using a linear kernel achieved the best accuracy, with hierarchical classification attempted with 53.23% four-fold CV accuracy. The 7Hu moments technique is used for gesture database creation, with both global and local visual features extracted to characterise manual alphabet letters. Depending on the application, either contour-based or region-based methods are used.

Pramada et al. [16] proposed an algorithm using image processing, machine learning, and artificial intelligence to recognise the number of fingers opened in a Binary Sign Language gesture. The detection system is recalibrated using colour calibration to accommodate varying lighting conditions by finding an area's maximum and minimum RGB values. Skin detection is performed using a predicate S(R.G.B) and allows for detection at 15fps. Pattern matching compares pixel values of each colour with templates stored in the database, and the area and coordinates of each colour pixel are calculated using an equation. Text-to-speech conversion translates the matched image into text and audio format, and finger values are assigned for alphabet recognition.

Ojha et al. [12] employed a CNN to capture and translate ASL gestures into text and speech in real-time. The CNN architecture consists of 11 layers, including 3 convolutional layers, a ReLU layer, a dense layer, max-pooling layers, and a dropout layer. The model is utilised to identify low-level features, such as lines, angles, and curves, and high-level features, such as gestures and shapes. Gaussian background subtraction is used to discover bounding boxes of various objects, and a CNN model is used to separate gesture signs from the background. An accuracy of 95% was achieved. Golekar et al. [**?**] have implemented a model which employs the Haar cascade classifier for hand segmentation and detection system that is used to construct a hand gesture recognition system using Neural networks, Support Vector Machine and Adaptive Boosting are algorithms used in hand gesture recognition. The recognition system utilises a convex hull for better fingertip detection. This paper claims to achieve better accuracy results compared to other existing hand gesture recognition systems. They also explain skin colour identification using the convex hull approach.

Triyano et al. [22] propose a method that first detects the background colour on 7 sections of the screen, and further, the system conducts pre-sampling steps to gather information about the background colour and the colour of the user's hand [14]. This colour data is then utilised to calculate the threshold for the input image. The hand segmentation is performed by logical operations on the binary image of the hand and the background, and then using morphological operations to obtain the optimal result. Results showed that the fingertip search method can be used to recognise a hand gesture. The hand-clenched search methods and Hu Moments value are used to figure gestures. Fingertip methods are more resilient in recognising gestures with 95% success.

The authors of a study by Khan et al. [7] proposed a new approach to improve object recognition performance by utilising multiple deep layers of pre-trained convolutional neural networks (CNNs). They utilised two popular pre-trained CNN models, VGG_16 and VGG_19, and extracted feature sets from three deep, fully connected layers named "fc6", "fc7", and "fc8". The authors used the Caltech-101 database from the Caltech object category datasets as a test bed for their proposed Deep Multi-Layer (DM-L) feature extraction technique in their study. The authors of the study found that among the three deep fully connected layers "fc6", "fc7", and "fc8" extracted from the pre-trained VGG_16 and VGG_19 CNN models, the "fc6" layer performed the best in terms of object recognition performance.

The paper by Bolme et al. [3] describes a new approach to object tracking using correlation filters. The proposed Minimum Output Sum of Squared Error (MOSSE) filter can track complex objects through rotations, occlusions, and other distractions at over 20 times the rate of current state-of-the-art techniques. MOSSE filters produce stable correlation filters when initialised using a single frame and are robust to variations in lighting, scale, pose, and non-rigid deformations. These results show that the advanced correlation filters track targets longer than the Naive method. This approach allows for the creation of effective filters using minimal training data. The paper by Song et.al [11] presents a lightweight Convolutional Neural Network (L-CNN) that is inspired by Depth-wise separable convolution and Single Shot Multi-Box Detector (SSD). The L-CNN approach narrows down the classifier's search space to focus on human objects, which results in faster and more accurate detection. The paper introduces a lightweight CNN for human object detection at the edge, leveraging the Depth-wise Separable Convolutional network. The model was trained on parts of ImageNet and VOC07 datasets. Experimental results showed that L-CNN achieved a Maximum of 2.03 and an Average of 1.79 FPS with a false positive rate of 6.6%.

3 Materials and Methodology used

This section presents the proposed work based on the methodology where the following process is carried out in a step-by-step manner to acquire the desired result. One of the extensively researched domains is recognising and classifying 26 alphabets in Indian sign language from real-time video [18,9,21]. A proposed methodology aims at sign language recognition and classification of subsequent text into different labels. Along with this, the labels are converted to audio using text-to-speech recognition by leveraging Google text-to-speech API. The implementation of the said scheme is divided into the following parts:

1. Data Acquisition
2. Input Data Generator
3. CNN model generation and training
4. Testing the classification model
5. Text to speech

The overall architecture is shown in Figure 1.

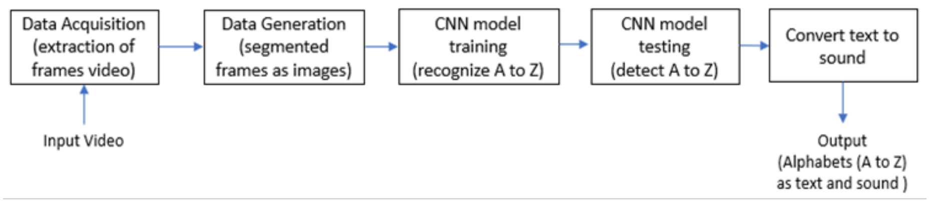

Data Acquisition (extraction of frames video)
Data Generation (segmented frames as images)
CNN model training (recognize A to Z)
CNN model testing (detect A to Z)
Convert text to sound
Input Video
Output (Alphabets (A to Z) as text and sound)

Fig. 1. Workflow of the proposed Sign Language Recognition system.

The input data has been generated by considering the recorded video as an input where an experienced signer has performed a sign language of alphabets (A to Z). From the video input, the frames are extracted and labelled automatically. Using these labelled inputs, training and testing data have been generated. Before generating these inputs, the extracted frames are pre-processed. In the pre-processing stage, images were re-sized into 224 by 224 pixels. Subsequently, every image was converted into grayscale. Around 2400 images were collected, with around 90-95 images distributed across every alphabet class. Using mediapipe hand detection, the landmarks of the hand were detected in each image by localising key points of the hands. Subsequently, the images were cropped to fit just the hand and a Gaussian blur filter with the kernel size 5 was applied to each image.

The filter was predominantly used to remove the noise from the image by replacing the noisy pixels in the images with the average value of the adjacent surrounding pixels based on Gaussian distribution. These denoised images were then subject to edge detection, where the change in the pixel values in the x-direction and y-direction were calculated. These values were utilised to calculate the gradient magnitude, which showcases the magnitude of the changes that occur in the image with respect to the pixel value. Then, the Image was sharpened using gradient magnitude and blur image. Sharpened images give a much clear idea of edges. To these sharpened images, a Top-hat morphological function was applied. Finally, data augmentation was performed on all the pre-processed images in the dataset. The augmentations consisted of the images being flipped horizontally and vertically. This was used to generate a new derived dataset consisting of the original pre-processed images along with their augmented counterparts. The size of the dataset was 9600 images. The mathematical equations for the preprocessing techniques used are referenced in Equations 1, 2, 3 and 4.

$$M(x,y) = \sqrt{I_x^2(x,y) + I_y^2(x,y)} \tag{1}$$

where $M(x,y)$ is the gradient magnitude at pixel (x,y), I_x and I_y are the partial derivatives of the image function in x and y directions respectively.

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

where $G(x, y)$ is the value of the Gaussian function at pixel (x,y) with standard deviation σ.

$$T(x, y) = I(x, y) - (I(x, y) \circ B)^o \tag{3}$$

where $T(x, y)$ is the result of applying the top-hat morphological function to the image, $I(x, y)$ is the original image, $\circ$ represents the morphological dilation operation, B is the structuring element used for dilation, and $(\cdot)^o$ represents the morphological opening operation.

$$S(x, y) = I(x, y) + kM(x, y) \tag{4}$$

where $S(x, y)$ is the sharpened image, $I(x, y)$ is the original image, $M(x, y)$ is the gradient magnitude calculated using equation (1), and k is a constant that controls the amount of sharpening applied.

The abstract diagrammatic representation for the preprocessing is shown in Figure 2.

The dataset was derived using the ImageDataGerenrator under the Keras pre-processing library, which has been traditionally used to generate the dataset from the folders. The images were subject to augmentation, where techniques such as rotation (to rotate the images by a certain angle), shear (to rectify the perception angles), zoom (to magnify into the frame), and width shift and height shift (to control the amount of horizontal and vertical shifts) were implemented. The actions were used to create both the Train and the Test datasets. After segregating the images into labelled train and test folders, the data is fit into a custom Convolution Neural Network model. They are a category of deep, feed-forward artificial neural networks, that are predominantly utilised for visual imagery analysis. They incorporate a variation of multi-layer perceptron that aims to minimise the requirement for pre-processing.

The custom CNN architecture includes multiple Max-pool, Dense, and Convolution layers. The model consists of two convolution layers that take the input image of shape 224 by 224. A Max-pooling layer is then applied to scale down the image shape by 50%. The second Convolution layer is added on top of this, which takes the input shape 108 by 108. This is further succeeded by a Max-pool layer, which is used to decrease the image size by 50%. The model is then subject to flattening, and a flattening layer is added to modify the image array. Finally, two dense layers are added, with the first dense layer used to divide the image into 64 different categories, and the last dense layer is utilised to further categorise the image into 26 different alphabet categories. Here, the saved train and validation data folders are used to train the model in Google Colaboratory. At first, Google Drive has been used to host the dataset folders, and then we generated input data using ImageDataGenerator of the Keras pre-processing package. Then, the model that was described in Figure 3 was used to train on the Indian sign language dataset using the Keras TensorFlow library in Python. Later for testing the model new real-time dataset was used. The algorithmic steps are shown below.

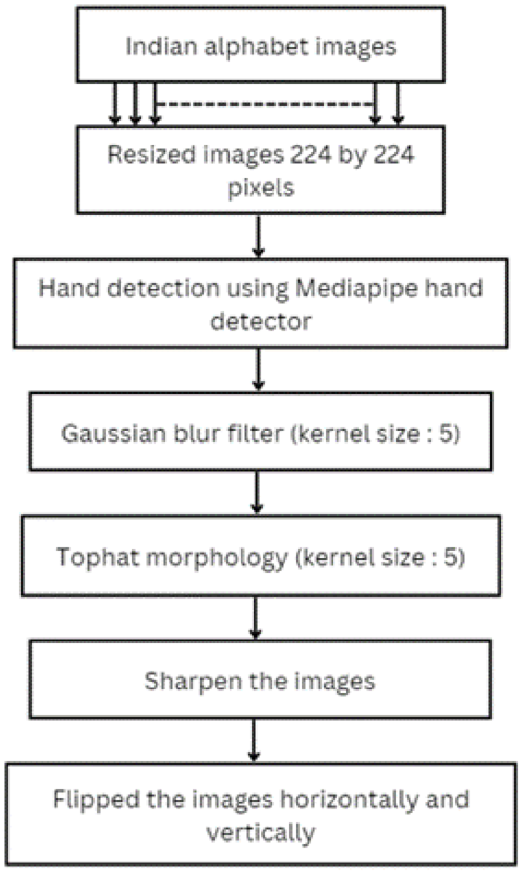

Indian alphabet images
Resized images 224 by 224 pixels
Hand detection using Mediapipe hand detector
Gaussian blur filter (kernel size : 5)
Tophat morphology (kernel size : 5)
Sharpen the images
Flipped the images horizontally and vertically

Fig. 2. Pre-processing and Augmentation of the proposed work.

In this, the signer would start to record the video. That video would be converted into frames. For each frame, we detected the hand using a mediapipe hand detector and pre-processed that specific hand segment. The trained model was loaded, and we passed the frame to the model for prediction. Using the classification methods provided by the Keras library, the model predicted a value. Then, A category index value like 0, 1, 2, 3, ... (up to 25) was obtained. The index value would serve as a key in a dictionary that corresponds to each category. For instance, 0 would be linked to A, 1 would be linked to B, and so forth. As the last stage, this feature converts the predicted text to speech. For this, google text to speech library has been used. A predicted alphabet was passed to GTTS, and it returned an mp3 file of that alphabet. But to run it in real-time, pygame module was used. Pygame would load an mp3 file, so the signer can listen to the predicted alphabet in real time. The obtained results are analysed in the next section.

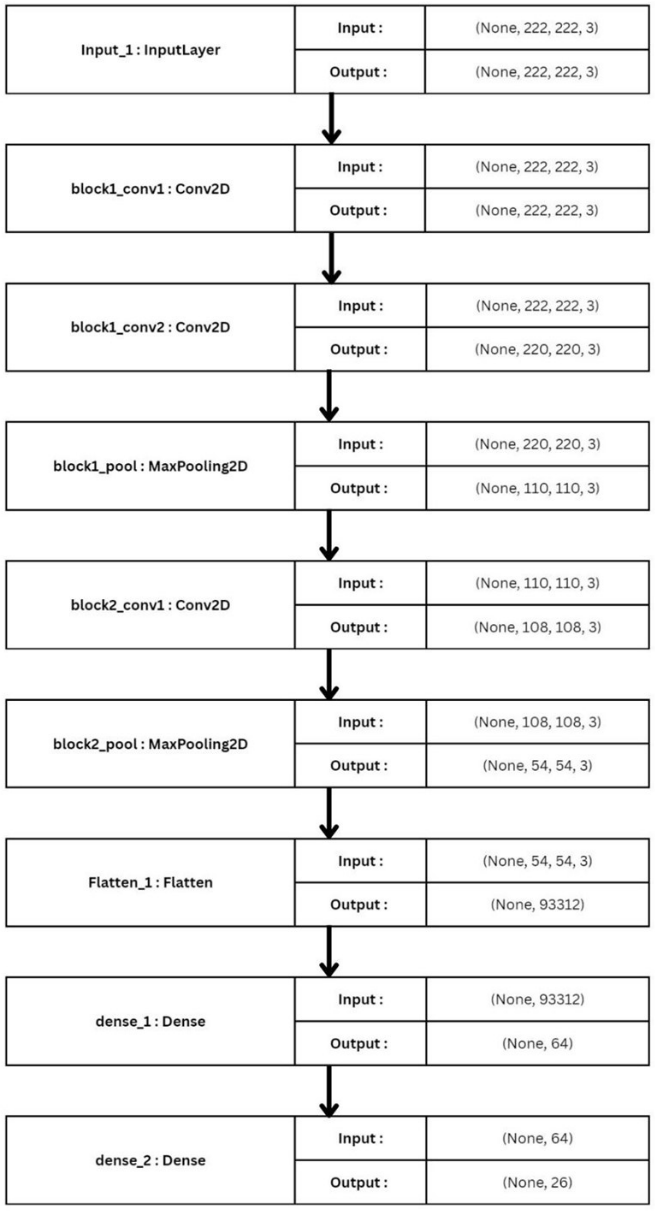


Fig. 3. Convolutional Neural Network model for training and testing.

Algorithm 1 Algorithm for Estimating Hand Gesture

```
1: procedure ESTIMATEGESTURE($Video$)
2:     Convert video into N frames
3:     Use $mp\_model$ to detect hand in video frames
4:     Crop frame around the hand region
5:     Detect coordinates of landmarks on frames and calculate the min and max values
6:     Convert frames to grayscale images
7:     Apply Gaussian blur to frames to improve edge detection and sharpen images, producing preprocessed frames
8:     Pass preprocessed video frame to trained model to predict correct alphabet
9:     Create an array of predicted values of 10 consecutive frames: $D = \{F_1, F_2, F_3, ..., F_{10}\}$
10:    if Value $P_i$ of all elements of array $D$ is same then
11:        Convert predicted value into speech using $GTTS$ only once and wait for next different letter
12:    end if
13:    Output: Predict sign language and convert it into speech
14: end procedure
```

4 Result and Analysis

The Google Colaboratory GPU has been used to train the model on Indian Sign Language alphabet data. The categorical Cross Entropy function was utilised to calculate the loss. Adam optimiser with learning rate 1e-4 was used. The model was trained for 50 epochs on the training dataset with a batch size of 32 and used the test dataset as the validation dataset. The data set consists of images of alphabets from A to Z. 93.96% accuracy was achieved, and loss was 0.2132 after 50 epochs. For the validation data set, accuracy and loss were 77.25% and 0.9902, respectively. The plotted training, test accuracy, and loss curves are shown in Figure 4. The results of each step of the proposed work have been shown in Figure 5, which shows the extracted video frame from an input video, the result after applying the Gaussian filter, detected coordinates of hard gesture, and a recognised text output of the sign language.

5 Conclusion

The focus of this paper is on speech and hearing impairments. These are a type of disability that restricts an individual's ability to communicate with others properly. Many automation tools are used to address this communication gap. After evaluating existing works in terms of Indian sign language recognition, the method defined in the paper has been proposed. The steps involved are loading a video file, extracting each frame, and detecting the hand landmarks in each frame using the Media-Pipe library. Then the frames are cropped, and the region of interest is pre-processed and stored in a new data directory for training

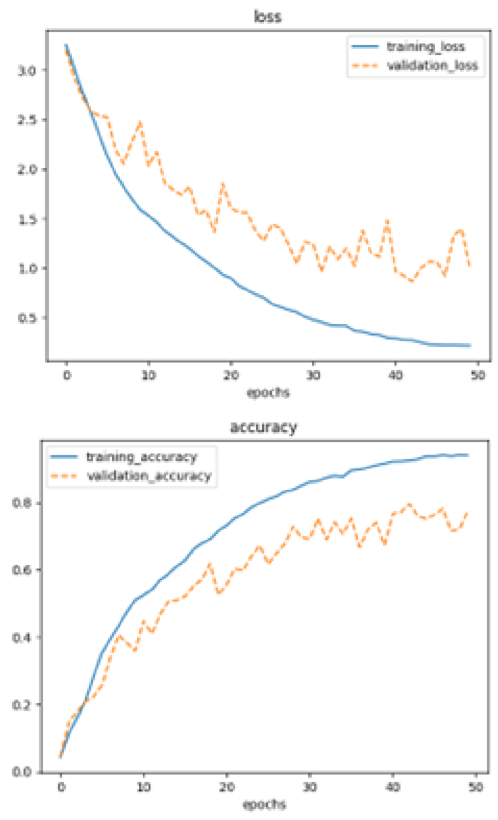
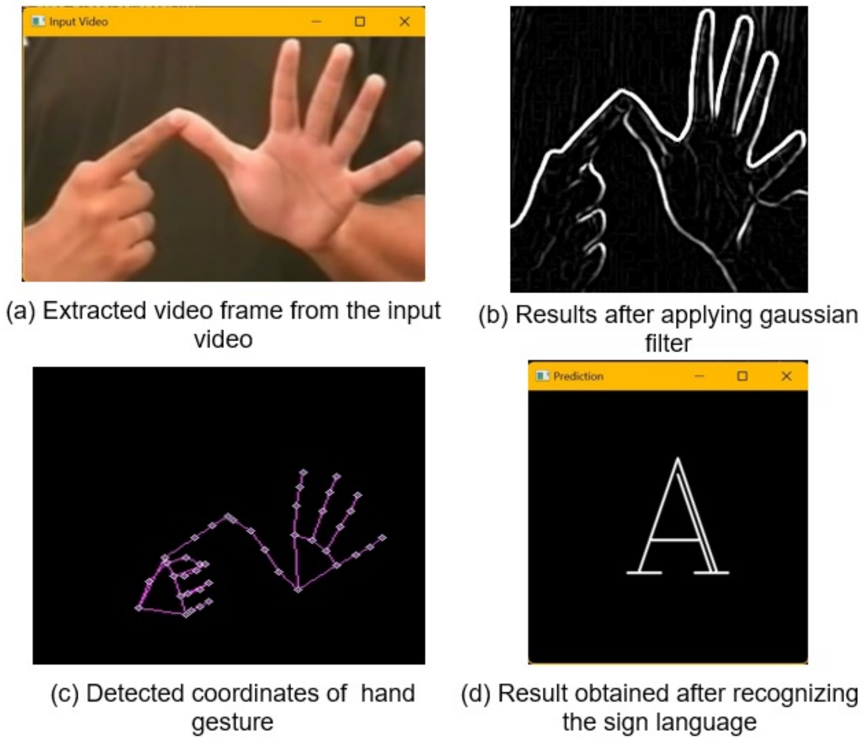


Fig. 4. Proposed models training and validation accuracy and loss.

purposes. The pre-processing involves the use of Gaussian blur, edge detection, morphological transformations, and signal processing functions. Data augmentation is then performed, and images are saved in a new directory. The images are then used to train a custom CNN model, which contains four convolutional layers along with two fully connected layers. The model is compiled using the categorical cross-entropy loss function, optimised using the RMSprop optimiser, and the evaluation metric considered is accuracy. The predicted sign language alphabet is displayed on the screen and is converted to speech using the Google Text-to-Speech library. The model achieves an overall training accuracy of 93.96% and a validation accuracy of 77.25%. The findings indicate that the proposed approach can serve as a roadmap to develop a real-time sign language recognition system. Even though the validation accuracy was less comparatively, it can be improved in the near future by increasing the quantity of data and fine-tuning the model's hyperparameters.



(a) Extracted video frame from the input video

(b) Results after applying gaussian filter

(c) Detected coordinates of hand gesture

(d) Result obtained after recognizing the sign language

Fig. 5. Pictorial representation of the hand-sign detection process

Acknowledgement

The authors sincerely thank VIT and NTU management for their encouragement during this work.

References

1. Adeyanju, I., Bello, O., Adegboye, M.: Machine learning methods for sign language recognition: A critical review and analysis. Intelligent Systems with Applications **12**, 200056 (2021)
2. Aronoff, M., Meir, I., Sandler, W.: The paradox of sign language morphology. Language **81**(2), 301 (2005)
3. Bolme, D.S., Beveridge, J.R., Draper, B.A., Lui, Y.M.: Visual object tracking using adaptive correlation filters. In: 2010 IEEE computer society conference on computer vision and pattern recognition. pp. 2544–2550. IEEE (2010)

4. Chakraborty, S., Bandyopadhyay, N., Chakraverty, P., Banerjee, S., Sarkar, Z., Ghosh, S.: Indian sign language classification (isl) using machine learning. American Journal of Electronics & Communication **1**(3), 17–21 (2021)
5. Goswami, S., GGR, A.R., Sharma, K.: Introduction of indian sign language in inclusive education. Disability, CBR & Inclusive Development **30**(4), 96–110 (2019)
6. Kang, B., Tripathi, S., Nguyen, T.Q.: Real-time sign language fingerspelling recognition using convolutional neural networks from depth map. In: 2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR). pp. 136–140 (2015)
7. Khan, H.A.: Dm-l based feature extraction and classifier ensemble for object recognition. Journal of Signal and Information Processing **9**(2), 92–110 (2018)
8. Kothadiya, D.R., Bhatt, C.M., Saba, T., Rehman, A., Bahaj, S.A.: Signformer: Deepvision transformer for sign language recognition. IEEE Access **11**, 4730–4739 (2023)
9. Mariappan, H.M., Gomathi, V.: Real-time recognition of indian sign language. In: 2019 international conference on computational intelligence in data science (ICCIDS). pp. 1–6 (2019)
10. Nandi, U., Ghorai, A., Singh, M.M., Changdar, C., Bhakta, S., Kumar Pal, R.: Indian sign language alphabet recognition system using cnn with diffgrad optimizer and stochastic pooling. Multimedia Tools and Applications **82**(7), 9627–9648 (2023)
11. Nikouei, S.Y., Chen, Y., Song, S., Xu, R., Choi, B.Y., Faughnan, T.R.: Real-time human detection as an edge service enabled by a lightweight cnn. In: 2018 IEEE International Conference on Edge Computing (EDGE). pp. 125–129 (2018)
12. Ojha, A., Pandey, A., Maurya, S., Thakur, A., Dayananda, P.: Sign language to text and speech translation in real time using convolutional neural network. Int. J. Eng. Res. Technol.(IJERT) **8**(15), 191–196 (2020)
13. Papastratis, I., Chatzikonstantinou, C., Konstantinidis, D., Dimitropoulos, K., Daras, P.: Artificial intelligence technologies for sign language. Sensors **21**(17), 5843 (2021)
14. Park, K., Chae, M., Cho, J.H.: Image pre-processing method of machine learning for edge detection with image signal processor enhancement. Micromachines **12**(1), 73 (2021)
15. Pigou, L., Dieleman, S., Kindermans, P.J., Schrauwen, B.: Sign language recognition using convolutional neural networks. In: Computer Vision-ECCV 2014 Workshops: Zurich, Switzerland, September 6-7 and 12, 2014, Proceedings, Part I 13. pp. 572–578 (2015)
16. Pramada, S., Saylee, D., Pranita, N., Samiksha, N., Vaidya, M.: Intelligent sign language recognition using image processing. IOSR Journal of Engineering (IOSRJEN) **3**(2), 45–51 (2013)
17. Shirbhate, R.S., Shinde, V.D., Metkari, S.A., Borkar, P.U., Khandge, M.A.: Sign language recognition using machine learning algorithm. International Research Journal of Engineering and Technology (IRJET) **7**(03), 2122–2125 (2020)
18. Singha, J., Das, K.: Recognition of indian sign language in live video. arXiv preprint arXiv:1306.1301 (2013)
19. Srivastava, S., Gangwar, A., Mishra, R., Singh, S.: Sign language recognition system using tensorflow object detection api. In: Advanced Network Technologies and Intelligent Computing: First International Conference, ANTIC 2021, Varanasi, India, December 17–18, 2021, Proceedings. pp. 634–646 (2022)
20. Thakur, A., Budhathoki, P., Upreti, S., Shrestha, S., Shakya, S.: Real time sign language recognition and speech generation. Journal of Innovative Image Processing **2**(2), 65–76 (2020)

21. Tripathi, K.M., Kamat, P., Patil, S., Jayaswal, R., Ahirrao, S., Kotecha, K.: Gesture-to-text translation using surf for indian sign language. Applied System Innovation **6**(2), 35 (2023)
22. Triyono, L., Pratisto, E., Bawono, S., Purnomo, F., Yudhanto, Y., Raharjo, B.: Sign language translator application using opencv. In: IOP Conference Series: Materials Science and Engineering. vol. 333, p. 012109 (2018)
23. Wahane, A., Gadade, R., Hundekari, A., Khochare, A., Sukte, C.: Real-time sign language recognition using deep learning techniques. In: 2022 IEEE 7th International conference for Convergence in Technology (I2CT). pp. 1–5 (2022)
24. Yasir, F., Prasad, P., Alsadoon, A., Elchouemi, A., Sreedharan, S.: Bangla sign language recognition using convolutional neural network. In: 2017 International Conference on Intelligent Computing, Instrumentation and Control Technologies (ICICICT). pp. 49–53. IEEE (2017)