International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 07 Issue: 05 | May - 2023 Impact Factor: 8.176 ISSN: 2582-3930

American Sign Language Recognition and Conversion using CNN

Aditya Thakar[1], Ameya Katgeri[2], Devdatta Patil[3],

Warepam Richard Singh[4], Prof. P. S. Jadhavar[5]

[1,2,3,4] Student, Department Of Computer Engineering, P.V.P.I.T, Pune, Maharashtra, India
[5] Professor, Department Of Computer Engineering, P.V.P.I.T, Pune, Maharashtra, India.

Abstract - This paper discusses the development of an application that can recognize and translate American Sign Language (ASL) gestures into text to facilitate communication between ASL users and non-native speakers. The application utilizes computer vision techniques and natural language processing, including convolutional neural networks and data augmentation techniques, to interpret and translate ASL gestures into text. The paper explores the challenges in ASL recognition due to variability in signing styles, dialects, and lighting conditions. The data for training the model was collected manually using OpenCV and the HandTrackingModule. The paper highlights the advantages and limitations of existing approaches and provides a comparative study of various works in recognizing ASL using vision-based techniques.

***Key Words*:** ASL translation, Deep Learning, CNN, Mediapipe

1.INTRODUCTION

American Sign Language or commonly known as ASL is a visual, complete natural language. This means that it has linguistic properties as spoken languages, like grammatical rules, syntax, and vocabulary. It uses hand gestures. facial expressions, and body language to communicate. It is used by the deaf and hard-of-hearing individuals to communicate with others. However, it becomes difficult to communicate with the non-native speakers creating a language barrier. This will lead to feelings of isolation and exclusion for these individuals.
ASL is complex and complete language of communication for hearing impaired people and it has massive varieties of sign gestures of hand, finger, palm, and fist with and without the support of facial expressions and body movements. In ASL, there are no specific sign gestures for several English words and sentences. Thus, those English words and sentences can be signed by spelling out the set of American Manual Alphabets (AMA) of 26 gestures from A to Z.

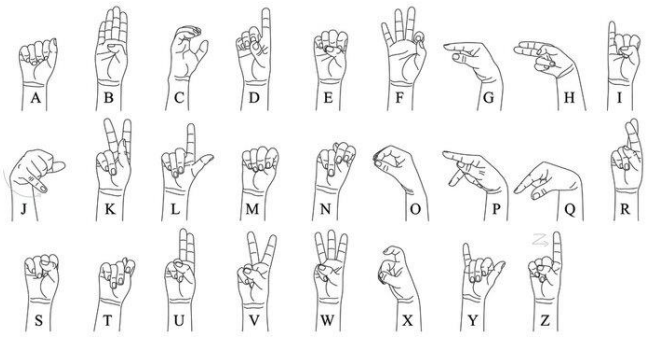

Fig. - 1 Static signs for 26 alphabets

In this project we have created an application that recognizes the different signs performed by a native speaker of the language and translates it to text. This text can be then read by the non-native speaker. Basically, the application translates ASL to English. Thus, facilitating effective communication between the users. This application uses CNN, computer vision techniques and natural language processing to interpret and translate ASL gestures into text. These approaches include data augmentation techniques, deep learning algorithms, and the use of multiple modalities, such as incorporating facial expressions and body movements into the recognition process.

Mainly due to the advances of deep learning, more concretely convolutional neural networks [1], the quality of image recognition and object detection has been progressing at a dramatic pace. There have been significant advancements in these technologies, however despite the this there are still significant challenges to be addressed. One of the most significant challenges is the high degree of variability in ASL gestures, which can be affected by factors such as speed with which the user displays signs, lighting conditions, and variations in signing styles. Additionally, there are variations in ASL dialects, which can complicate the recognition and conversion process.

2. Literature Survey

The field of American Sign Language (ASL) recognition and conversion has received significant attention in recent years, with many researchers exploring the use of machine learning, computer vision, and deep learning techniques to build effective systems.

Paper [2] presents a deep learning-based approach for recognizing American Sign Language (ASL) signs from

images, with the objective of deploying the system on mobile applications or embedded single-board computers. By using image processing to convert RGB data to grayscale, the storage requirements and training time of the convolutional neural network are reduced. The authors compare different network architectures, including LeNet-5, AlexNet, Vgg16, and MobileNet v2, and achieve a testing accuracy of 87.5% using a final architecture with only 10 layers, including a dropout layer.

Paper [3] presents a method for translating and recognizing American Sign Language (ASL) video hand gestures into human or machine-readable English text using deep neural networks. The recognition process begins with fetching the input gestures from a video and using Gaussian Mixture Model (GMM) for background elimination and foreground detection. Various feature extraction techniques, such as SURF, Zernike Moment (ZM), Discrete Cosine Transform (DCT), Radon Features (RF), and RGB levels, are used to extract the hand features from the video frames. The extracted features are then used for classification and recognition using stacked autoencoders in the deep neural networks.

Paper [4] proposes a vision-based application for sign language recognition. Using deep learning and computer vision techniques, the proposed model takes video sequences of sign language gestures and extracts both temporal and spatial features. The spatial features are recognized using a Convolutional Neural Network (Inception) while the temporal features are trained using a Recurrent Neural Network (RNN). The American Sign Language Dataset is used for the proposed model.

In paper [5] authors present a software that focuses on the translation of 24 static hand gestures of American Sign Language into human or machine-readable English text. The proposed approach involves pre-processing of the signed input gesture, followed by the calculation of various properties of the pre-processed gesture image. The transliteration of the signed gesture into text is then carried out based on the calculated properties.

Authors of paper [6] presents a real time system that was trained using a dataset collected by Massey University in 2011, and achieved 100% accuracy in the testing phase. The real-time system determines the skin colour of a frame to identify the hand, and uses the convex hull algorithm to determine the hand gesture, which is then translated into text using a pre-recorded convolutional neural network model and weights. The accuracy of the real-time system is 98.05%.

Paper [7] presents a new approach for the translation of 24 static gestures of the American Sign Language alphabet into a human or machine-readable English manuscript. The gestures are location, background, background colour, illumination, angle, distance, time, and camera resolution invariant. The recognition process involves pre-processing and clear segmentation stages, resulting in an average recognition rate of 98.21%, which is an outstanding accuracy compared to state-of-the-art techniques.

Paper [8] is a comparative study of various works done by researchers in recognizing American Sign Language (ASL) using vision-based techniques. The goal of the American Sign Language Recognition (ASLR) system is to automatically understand the gestures of ASL and convert them into equivalent human-readable or machine-readable text. The paper highlights the various works done in this field and compares the results.

3. Data Collection

The data to train the model was collected manually. A script was written which uses OpenCV [9] library and the “HandTrackingModule” [10], [11] module to detect and track a hand in real-time through the webcam. The code contains an infinite loop that reads frames from the webcam and detects a hand in the view of the webcam. If a hand is detected the program crops the image that includes only the hand and some margin around it which is manually specified. The image is then resized to 244 x 244 pixels because the model expects this size [12]. The aspect ratio of the image is constant even after resizing of the image. The program displays the webcam stream, cropped hand image and the resized hand image in separate windows. When the key “s” is pressed the resized hand image is saved to the disk. These saved images were then used for training the model. 500 images for each of the 26 alphabets were captured for training the model. To help the model generalize the data was collected with plain and complex backgrounds, signer, invariant location, illumination (natural and artificial) and distance from the webcam [7].

4. System Architecture

The application takes input frames from a webcam connected to the computer using the OpenCV library. Then the application uses hand detector module that is used to identify and locate the position of the hand within the frame. This module tracks the hand after it has been identified and senses any changes in the signs made by the signer. These frames are resized and given to the pre-trained CNN model for identification and classification of the signs shown by the signer. This model is trained on a large dataset of images of sign language gestures. This model is used to classify gesture in real time. The predicted letter of model is displayed on the webcam stream. The sentence that has been registered is also displayed on the webcam stream

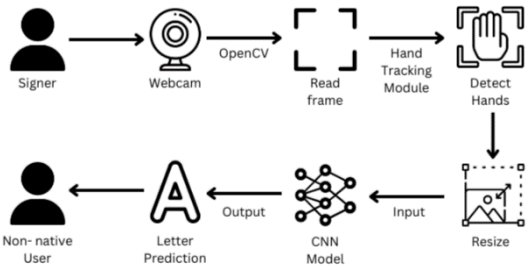

Fig. - 2 System Architecture Diagram

5. Technologies Used

HandTrackingModule

The HandTrackingModule is a computer vision library that provides us with the functionality to detect and track human hands in real-time using deep learning algorithms. The module uses MideaPipe framework for hand detection and tracking [10]. MideaPipe achieves this by using a deep learning model that is trained to detect and track 21 landmarks (keypoints) on a hand. The landmarks include points on fingers, palm, and wrist [12] [11].

The HandTrackingModule takes an input image i.e., a from a webcam, processes it using MediaPipe to detect hands, and then tracks the detected hands in subsequent frames. The module is capable of detecting and tracking a single hand or multiple hands simultaneously. We have specifically used the HandDetector class from the HandTrackingModule which uses the BlazeHand model of MediaPipe for hand detection [13].

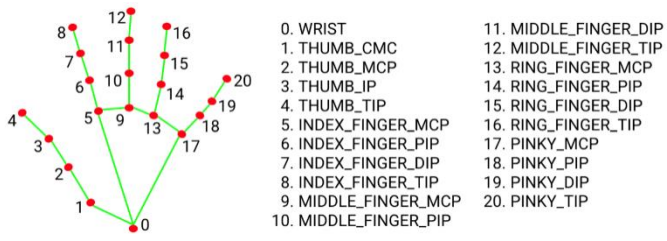

Fig. - 3 Hand landmarks using MediaPipe [11]

CNN

MobileNet is a convolutional neural network architecture specifically designed to run on devices that with limited computational resources. It was proposed by Google in their 2017 paper titled "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications" [14]. This efficiency is achieved by using a depthwise separable convolutions. Depthwise separable convolutions were first introduced in a 2014 paper by Chollet [15], and were later adapted for use in the MobileNet architecture by Howard et al. in their 2017 paper [14].

A standard convolutional layer applies a set of filters to the entire input volume, resulting in an output volume that is typically larger than the input volume. In contrast, a depthwise separable convolution decomposes the convolution operation into two separate steps: a depthwise convolution and a pointwise convolution.

In the depthwise convolution step, a separate filter is applied to each input channel independently. This results in a set of output feature maps that have the same spatial dimensions as the input, but with a reduced number of channels. This step reduces the number of computations required compared to a standard convolution, as there are fewer filters to apply.

In a standard convolutional layer, a square input feature map with spatial width and height of D_F and M input channels respectively is transformed into a square output feature map of spatial width and height D_G with N output channels. The layer is defined by a convolution kernel K of size $D_K \times D_K \times M \times N$, where D_K is the spatial dimension of the kernel and M and N are the number of input and output channels respectively. Standard convolutions have a computational cost that depends on the kernel size, the number of input and output channels, and the feature map size [14].

MobileNet models reduce the computational cost by using depthwise separable convolutions, which decompose the standard convolution operation into two separate operations: a depthwise convolution, which applies a single filter to each input channel separately, and a pointwise convolution, which applies a 1x1 convolution to combine the outputs of the depthwise convolution. This reduces the computational cost of the convolution operation by breaking the interaction between the number of output channels and the size of the kernel [14] [16].

In the pointwise convolution step, a 1x1 convolution is applied to combine the output feature maps from the depthwise convolution. This step allows for non-linear combinations of the features from the depthwise convolution and allows for a greater degree of expressiveness in the network [14]. The use of depthwise separable convolutions allows for a significant reduction in the number of computations required compared to a standard convolution, while still maintaining high accuracy.

Architecture

We first defined the base model which is a pretrained MobileNet model that has been trained on a large dataset of images called ImageNet. The original MobileNet architecture consists of 28 layers, including a single input layer and several convolutional layers with batch normalization and ReLU activation. After each depthwise separable convolutional layer there is a pointwise convolutional layer with a 1x1 kernel size and ReLU activation function. This is followed by batch normalization and a ReLU activation function, before

proceeding to the next depthwise separable convolutional layer [17]. The model expects images of width and height 244 x 244 pixels. A global average pooling layer is also added. This layer reduces the spatial dimensions of feature maps produced by the base model and averages them over each channel. This helps to extract condensed representation of the input image. After this a fully connected layer of 128 neurons is added. The activation function used for the fully connected layer is ReLU. Finally a output layer is added with a number of neurons equal to the number of classes in the training dataset. The final layer uses softmax activation function [17].

ReLU Activation Function

The Non-Linearity operation is used after the convolution operation mentioned above. ReLU stands for Rectified Linear Unit and is a non-linear operation. It is applied to each element individually and it replaces all negative pixel values in the feature map to zero [17]. The purpose of the ReLU is to introduce non-linearity since real world training is non-linear and the CNN should model to that.

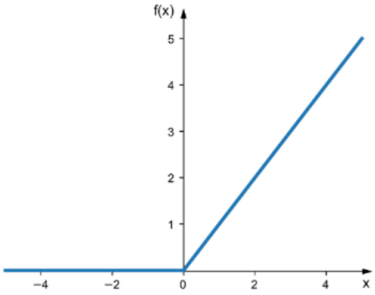

Fig. - 4 Graph of ReLU Activation Function [18]

ReLU function gives a max between an input number and 0.

$$A(x) = \max(0, x)$$

ReLU is a computationally cheap activation function unlike other activation functions such as sigmoid and tanh because it requires simpler mathematical operations which is a good point to consider when designing neural networks [19]. Deep convolutional neural networks with ReLUs train several times faster than their equivalents with tanh units [17].

Softmax Activation Function

The softmax activation function is a commonly used activation function in neural networks, particularly in multi-class classification tasks. It is a generalization of the logistic sigmoid activation function and is used to convert a vector of arbitrary real-valued numbers into a vector of probabilities that sum to 1 [19] [20].

The softmax function takes an input vector z of length k and outputs a vector of length k with values ranging between 0 and 1, where the sum of all values in the output vector is equal to 1 [20].

Mathematically, the softmax function is defined as:

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

where,

σ = softmax

$\vec{z}$ = input vector

e^{z_i} = standard exponential function for input vector

K = number of classes in the multi-class classifier

e^{z_j} = standard exponential function for output vector

e^{z_j} = standard exponential function for output vector [21].

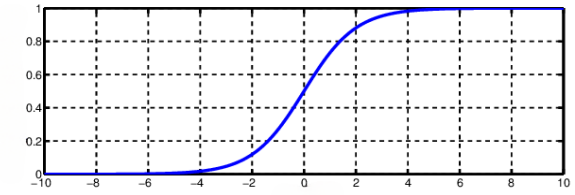

Fig. - 5 Graph of Softmax Function [22]

6. Evaluation of Model

The proposed model achieved a test accuracy of 98% and a test loss of 0.04% on the manually collected dataset of gestures. These results indicate the fact that the model is able to accurately recognize and convert ASL gestures with very high accuracy and low error. If the results are compared to other models, the proposed model outperforms many other models in terms of accuracy and computational complexity. MobileNet CNN architecture has fewer parameters than other popular CNN architectures, allows for faster training and inference while still maintaining high accuracy.

7. Conclusion

In this paper, an application has been developed that utilizes CNN, computer vision, and natural language processing techniques to recognize and translate sign language into English text. The approach involves data augmentation, deep learning algorithms, and incorporating multiple variations to improve recognition accuracy. Despite challenges posed by variability in ASL gestures and dialects, our application achieves high accuracy in recognizing ASL gestures and translating them into English text. It thus can provide a means for effective communication between the native and non-native ASL speakers.

REFERENCES

1. M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part I 13*, 2014, pp. 818–833.
2. S. Chavan, X. Yu, and J. Saniie, "Convolutional neural network hand gesture recognition for American sign language," in *2021 IEEE International Conference on Electro Information Technology (EIT)*, 2021, pp. 188–192.
3. S. Shivashankara and S. Srinath, "American Sign Language Video Hand Gestures Recognition using Deep Neural Networks," *Int. J. Eng. Adv. Technol.*, vol. 5, no. 8, pp. 2742–2750, 2019.
4. K. Bantupalli and Y. Xie, "American sign language recognition using machine learning and computer vision," 2019.
5. S. Shivashankara and S. Srinath, "American sign language recognition system: an optimal approach," *Int. J. Image, Graph. Signal Process.*, vol. 11, no. 8, p. 18, 2018.
6. M. Taskiran, M. Killioglu, and N. Kahraman, "A real-time system for recognition of American sign language by using deep learning," in *2018 41st international conference on telecommunications and signal processing (TSP)*, 2018, pp. 1–5.
7. S. Shivashankara and S. Srinath, "An american sign language recognition system using bounding box and palm FEATURES extraction techniques," *Int. J. Recent Technol. Eng.*, vol. 7, no. 4, 2018.
8. S. Shivashankara and S. Srinath, "A review on vision based American sign language recognition, its techniques, and outcomes," in *2017 7th International Conference on Communication Systems and Network Technologies (CSNT)*, 2017, pp. 293–299.
9. G. Bradski and A. Kaehler, *OpenCV: Computer vision with the OpenCV library*. " O'Reilly Media, Inc.," 2008.
10. D. Victor and S. Panchal, "HandTrackingModule - GitHub." https://github.com/victordibia/handtracking/blob/master/README.md
11. "Hand landmarks detection guide," *MediaPipe*. https://developers.google.com/mediapipe/solutions/vision/hand_landmarker
12. C. Lugaresi *et al.*, "Mediapipe: A framework for building perception pipelines," *arXiv Prepr. arXiv1906.08172*, 2019.
13. V. Bazarevsky, I. Grishchenko, K. Raveendran, T. Zhu, F. Zhang, and M. Grundmann, "Blazepose: On-device real-time body pose tracking," *arXiv Prepr. arXiv2006.10204*, 2020.
14. A. G. Howard *et al.*, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv Prepr. arXiv1704.04861*, 2017.
15. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1251–1258.
16. Szegedy *et al.*, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.
17. A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, no. 6, pp. 84–90, 2017.
18. M. RADU, I. COSTEA, and V. Stan, *Automatic Traffic Sign Recognition Artificial Inteligence - Deep Learning Algorithm*. 2020. doi: 10.1109/ECAI50035.2020.9223186.
19. S. Mhatre R., D. A. Save M., and N. V. Sunita, *Deep Learning*. TechKnowledge Publications, 2023.
20. C. M. Bishop and N. M. Nasrabadi, *Pattern recognition and machine learning*, vol. 4, no. 4. Springer, 2006.
21. T. Wood, "Softmax Function," *Deep AI*. https://deepai.org/machine-learning-glossary-and-terms/softmax-layer
22. B. Chen, W. Deng, and J. Du, "Noisy softmax: Improving the generalization ability of dcnn via postponing the early softmax saturation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 5372–5381.