Journal of Multi Disciplinary Engineering Technologies
Volume 18, Issue 01

Deep Learning for Sign Language: A CNN Approach to ASL Gesture Recognition

Syed Irfan[1] and Gurpreet Kaur[1*]

[1]Amity Institute of Information Technology, Amity University, Noida, India.

*Corresponding author(s). E-mail(s): gkaur10@amity.edu;
Contributing authors: humanic004@gmail.com;

Abstract

This paper presents a CNN model designed for ASL gesture recognition, aiming to improve accessibility solutions for deaf and hard-of-hearing individuals. The model developed in this work employs distinct deep learning methods to accurately identify hand gestures representing ASL alphabetical letters. The proposed CNN architecture is implemented using the Keras high-level API with TensorFlow as its backend. For image processing and visualization, OpenCV, NumPy, and Matplotlib are utilized. The architecture includes multiple convolutional layers, max-pooling layers, dropout layers, and densely connected layers, concluding with softmax layers for multi-class classification.

Training and testing are conducted on datasets of ASL gesture images, with data augmentation and validation strategies applied to enhance performance. The experimental evaluation shows promising accuracy, indicating that CNN-based models are well-suited for ASL gesture recognition. All model development and evaluation processes are carried out in Jupyter Notebook, while PyCharm is used for real-time testing with input images and a system-integrated camera. This work demonstrates that deep learning approaches can significantly contribute to ASL recognition and enhance ongoing efforts to improve accessibility solutions.

Keywords: Gesture recognition, American Sign Language (ASL), Convolutional Neural Networks (CNNs), Deep Learning, Accessibility Technologies.

1 Introduction

The concept of American Sign Language (ASL) is very important as a tool of interaction for individuals with hearing disabilities, enabling them to express themselves and communicate with others. However, there are challenges in the reception of ASL gestures. These difficulties often arise in environments where hearing-impaired individuals face obstacles, especially in situations where people may not understand, comprehend, or effectively utilize ASL.

To address this issue and ensure equality in learning environments, automated ASL gesture recognition systems have attracted significant attention as a viable solution. These systems employ computer vision and deep learning techniques to recognize hand signs and convert them into textual or auditory outputs.

Published on: November, 2024

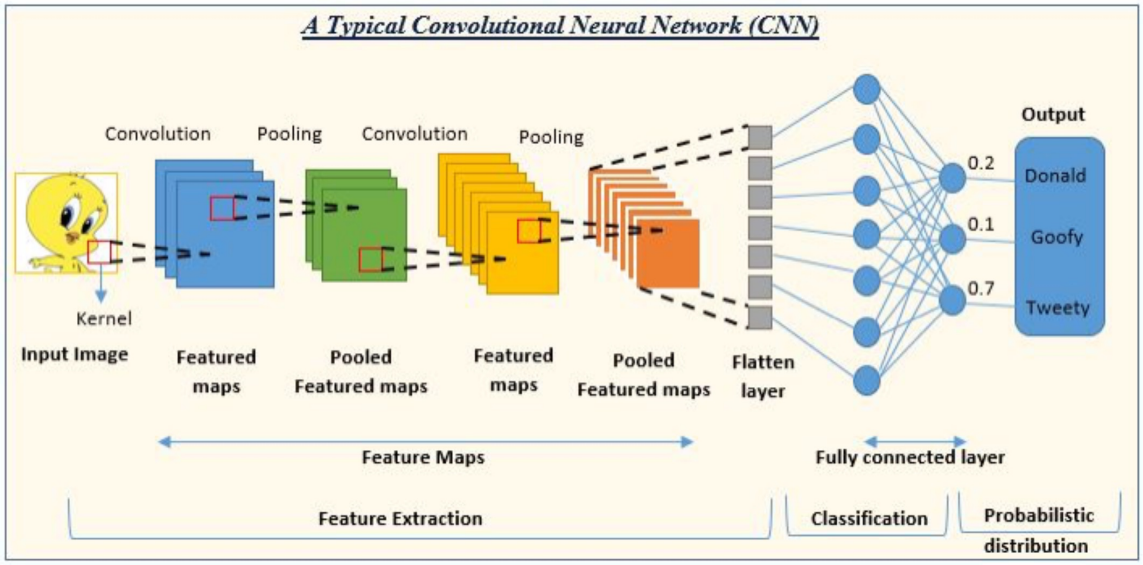

Fig. 1 CNN layers

In this work, we propose a technique for sign language gesture recognition using a deep learning model including CNN architecture which provides a manner of identifying the right hand signs that represent ASL alphabet characters. The model architecture is designed using Keras – a high-level application programming interface which makes creation of neural networks easy, and TensorFlow– a free deep-learning library that enables computation. Furthermore, OpenCV is used in image processing and capturing and NumPy in addition, subtraction, multiplication, division data handling and analysis. Matplotlib is used in visualizing the behavior of the model through training and evaluation of the metrics in model training and testing.

The proposed ASL gesture recognition model is trained and tested from the ASL gesture sets. Previous models are outdone due to the characteristics of the model, where the architecture applies stacked LSTM which has more LSTM layers. They shift their focus on the requisite of enhancing accuracy of the acquired information and the performance in generalization. Other methods are also used to increase the number of training examples, developing the data set quality to provide accurate validation of the model.

Overall, this research supports advancements in assistive technology by offering fast and accurate methods for ASL gesture recognition. The proposed approach, based on CNNs and deep learning, provides a practical solution to enhance accessibility for the hearing-impaired and other stakeholders, facilitating effective communication and knowledge sharing.

2 Literature Review

Sign language relies on visual cues such as hand shapes, gestures, body orientation, movement, facial expressions, and lip movements to convey meaning, much like spoken languages. Regional variations of sign language, such as Indian Sign Language (ISL), American Sign Language (ASL), and Portuguese Sign Language, exist, each with its own unique characteristics. Sign languages can be categorized into three types: fingerspelling, where individual letters are represented by handshapes; sign vocabulary, which uses hand and body movements, facial expressions, and lip movements to represent words; and continuous sign language, which involves sequences of gestures to form complete sentences. Research in sign language translation to text encompasses various approaches

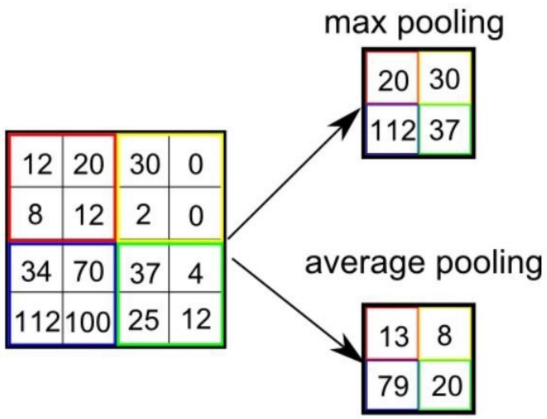

Fig. 2 Max Pooling and Average Pooling

and methodologies, aiming to bridge communication barriers for the deaf and hard-of-hearing community.

2.1 Related Works

The present work demonstrates how to detect and translate dynamic gestures from GDS by employing Microsoft Kinect for Windows v2. It employs two strategies for gesture recognition: template matching based on the GESTextractor and sequence matching using the Dynamic Time Warping (DTW) algorithm and the fusion of the Visual Gesture Builder with DTW. To assess these techniques, a group of eleven DGS gestures from an expert user from Germany is taken as the benchmark data set. To measure the performance of the two interfaces, the comparison considers the overall computation time as well as gesture identification accuracy.

However, most significantly, the computational time for DTW rises with the size of the dataset while it remains constant for the Visual Gesture Builder with DTW. However, when training the Visual Gesture Builder with DTW, the accuracy is 20.42% against the 65.45% of using only DTW. As a result, the study advises the application of the DTW on the smaller sets of data the Visual Gesture Builder with DTW on the large sets of data [1].

In Jing-hao Sun's work, the human hand was placed in a simple environment to be detected by the CamShift algorithm in real-time hand gestures. After that, a convolutional neural network (CNN) was used to detect and recognize the hand movement regions to obtain digits in real-time mode for ten digits which are used most frequently. The training images of the given system include in total 1600 images, where the hand gesture samples are 4000 where there are images of 400 hand gestures for each digit type. That is why, in this experiment, the accuracy indicator amounted to approximately 98.3%[2].

Hasan in his study utilized scaled normalization to identify the gestures through a process of brightness factor matching. Input images were segmented using technical thresholding properties on a black background. At the beginnings of the X and Y hub, the directions of any divided picture were moved to correspond to the position of the centroid of the hand unit and outline of the picture. Wysoski et al. [3, 4] constructed grids using boundary histogram and boundaries were normalized.

The purpose of this project is to develop an automated translator between ASL and textual English using readily available computing equipment like an ordinary computer and a standard webcam. Applying edge detection in conjunction with cross-correlation is possible and provides an accurate translation. The project entails the development of a live hand gesture recognition model whereby a number of image processing methods are integrated. Last but not the least; The gesture recognition stage is employed at the end of both approaches with the help of cross-correlation coefficient based scheme to detect the expressions [5, 6].

Table 1 CNN Layers

Layer (type)	Output Shape	Param #
conv2d layer (Conv2D)	(None, 126, 126, 128)	3584
max_pooling2d layer (MaxPooling2D)	(None, 63, 63, 128)	0
Dropout layer (Dropout)	(None, 63, 63, 128)	0
conv2d_1 layer (Conv2D)	(None, 61, 61, 256)	295168
max_pooling2d_1 layer (MaxPooling2D)	(None, 30, 30, 256)	0
dropout_1 layer (Dropout)	(None, 30, 30, 256)	0
conv2d_2 layer (Conv2D)	(None, 28, 28, 512)	1180160
max_pooling2d_2 layer (MaxPooling2D)	(None, 14, 14, 512)	0
dropout_2 layer (Dropout)	(None, 14, 14, 512)	0
conv2d_3 layer (Conv2D)	(None, 12, 12, 512)	2359808
max_pooling2d_3 layer (MaxPooling2D)	(None, 6, 6, 512)	0
dropout_3 layer (Dropout)	(None, 6, 6, 512)	0
dropout_4 layer (Dropout)	(None, 512)	0
dense_1 layer (Dense)	(None, 64)	32832
dropout_5 layer (Dropout)	(None, 64)	0
dense_2 layer (Dense)	(None, 256)	16640
dropout_6 layer (Dropout)	(None, 256)	0
dense_3 layer (Dense)	(None, 64)	16448
dropout_7 layer (Dropout)	(None, 64)	0
dense_4 layer (Dense)	(None, 256)	16640
dropout_8 layer (Dropout)	(None, 256)	0
dense_5 layer (Dense)	(None, 10)	2570

3 Methodology

3.1 Data collection

The data to this study was obtained from multiple sources as well as from GitHub data repositories and also hand extracted data. The main concentration was given out to collect image and video dataset about ASL gestures. The data set comprised of a set of stop frames of ASL hand signs in addition to the dynamic gestures were captured in the video format.

Several databases of ASL images were identified and used in this work. These repositories included multiple static images of hand sign which has targets captured in different lighting, hand pose orientations and backgrounds. Moreover, a custom dataset was also recorded by capturing images and videos from personal recording devices of the ASL gestures. This approach facilitated implementation of a rich source of new signs and variations not captured by other datasets. the hand captured data also made it easier to control some of these other variables such as the hand position, hand movement or even the background of the hand.

3.2 Algorithm Used

A Convolutional Neural Network (CNN), alternatively referred to as ConvNet, is a specialized form of Deep Learning model crafted specifically for the analysis and interpretation of visual data, particularly images. It processes input images by assigning significance to different features and objects within them through learnable weights and biases. Unlike traditional classification algorithms, CNNs require minimal preprocessing of input data. The architecture of a CNN is inspired by the organization of neurons in the human brain, particularly the Visual Cortex. In a CNN, neurons are activated by stimuli occurring in distinct sections of the visual field, termed Receptive Fields, which collectively span the entire visual space through overlapping. The CNN architecture consists of convolutional layers with filter sizes ranging from 128 to 512, utilizing kernel sizes of 3x3 pixels and ReLU activation functions. Max-pooling layers follow each convolutional layer to reduce dimensionality. Dropout layers with rates of 0.2 to 0.4 mitigate overfitting. The final fully connected layers consist of dense layers with 512 to 256 neurons and ReLU activation functions. The output layer employs softmax activation to predict class probabilities. This model comprises

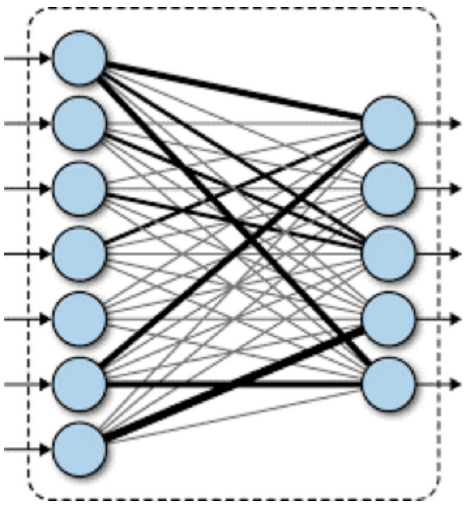

Fig. 3 Fully or Dense connected layer

approximately 13.4 million trainable parameters, facilitating the extraction of hierarchical features from input images and subsequent classification into 10 distinct classes.

3.2.1 Convolutional Layer

In the convolutional layers of the CNN model, filters with a small window size have been utilized, typically 3x3, that extend through the depth of the input image, which in my case is a 128x128 RGB image (depth of 3). These learnable filters are applied across the input image by sliding the window with a stride size of 1 pixel. At each position, the dot product of the filter entries and the corresponding input values is computed. Throughout training, the network learns to adjust these filters to detect various visual features such as edges of different orientations or regions with specific colours. Each filter specializes in recognizing certain patterns or features present in the input images. As the convolutional layers progress deeper into the network, they learn to extract increasingly abstract and complex features, allowing the model to discern intricate patterns and structures within the images. Ultimately, these learned features contribute to the model's ability to accurately classify images based on the presence of specific visual characteristics associated with American Sign Language gestures.

3.2.2 Fully Connected Layer

A fully connected layer, often referred to as a dense layer, is a neural network component where every neuron in the layer is linked to every neuron in the preceding layer. This type of layer ensures that all neurons from the preceding layer establish connections with each neuron in the current layer.

3.2.3 Final Output Layer

The final output layer in a neural network is responsible for producing the model's predictions or classifications based on the features learned from the preceding layers. The structure and activation function of the final output layer depend on the nature of the task being performed. The final output layer will predict or recognise the image of the input.

3.2.4 Training

We proceeded to train our Convolutional Neural Network (CNN) on the newly acquired dataset, utilizing the Keras ImageDataGenerator. This facilitated loading the training and test set data

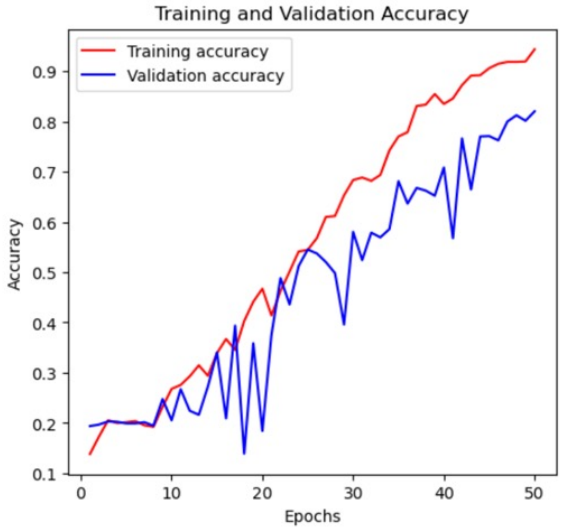

Fig. 4 Outputs from the first hidden layer

directly from directories, where each folder name represented a distinct class of images. Employing Reduce LR on Plateau and Early Stopping callbacks, the model's LR was dynamically adjusted based on validation dataset performance, preventing overshooting of loss minima. If validation loss failed to decrease, LR was reduced to enhance model performance. Our chosen optimization algorithm was Adam, which yielded superior accuracies. Ultimately, our model achieved 94% training accuracy and an 82% validation accuracy with 50 epochs.

Table 2 Training and Validation Metrics Across Epochs

Epoch nos.	Loss (%)	Accuracy (%)	Val_Loss (%)	Val_Acc (%)
1	2.33	13.83	2.2907	19.37
2	2.27	17.32	2.2844	19.69
3	2.25	20.55	2.2666	20.31
⋮	⋮	⋮	⋮	⋮
49	0.2144	91.95	0.6048	80.16
50	0.1829	94.38	0.6093	82.03

3.2.5 Testing and Validation

The pre-processed dataset was partitioned into training, validation, and testing sets, with the CNN model trained on the training data. Hyperparameters were fine-tuned through iterative experimentation to optimize model performance. The trained model's efficacy was evaluated using the validation set to assess metrics such as accuracy and precision. Additionally, the model's generalization capability and real-world performance were assessed by testing it on an independent testing set.

4 Results

During model training, fluctuations in validation accuracy and loss are common. Ideally, as epochs progress, loss decreases while accuracy improves. However, several scenarios may occur with validation metrics:

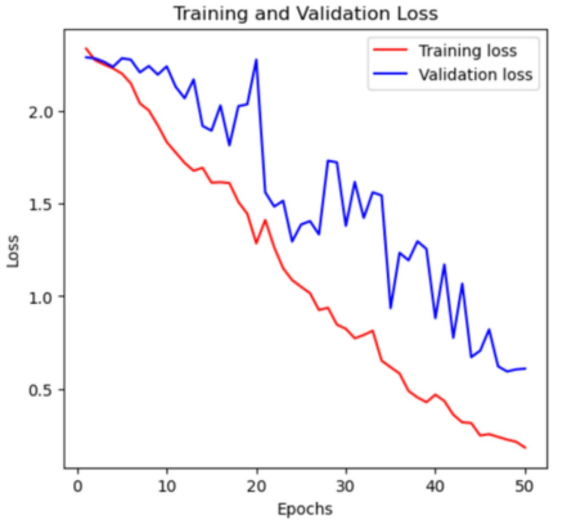

Fig. 5 Training and Validation Loss

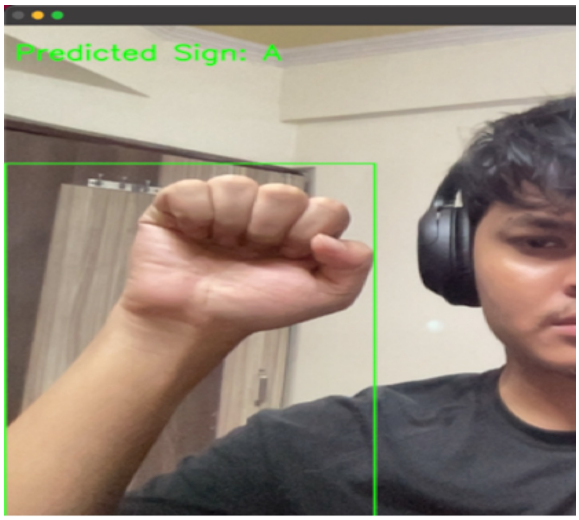

Fig. 6 Real-Time Testing output 1

1. Increasing validation loss accompanied by decreasing accuracy indicates the model is memorizing rather than learning.
2. Rising validation loss alongside increasing accuracy may signal overfitting or varying probability values, particularly when softmax is applied in the output layer.
3. Decreasing validation loss coupled with increasing accuracy indicates effective learning and model performance. Following model testing, we plotted accuracy and loss against epochs to visualize our results.

4.1 Limitations

Some limitations of the project are; that the classes of the dataset are only from A-J, so the model cannot recognise other alphabets. It cannot recognise words or phrases. The model's accuracy may be reduced in different backgrounds and different lighting or hand- poses.

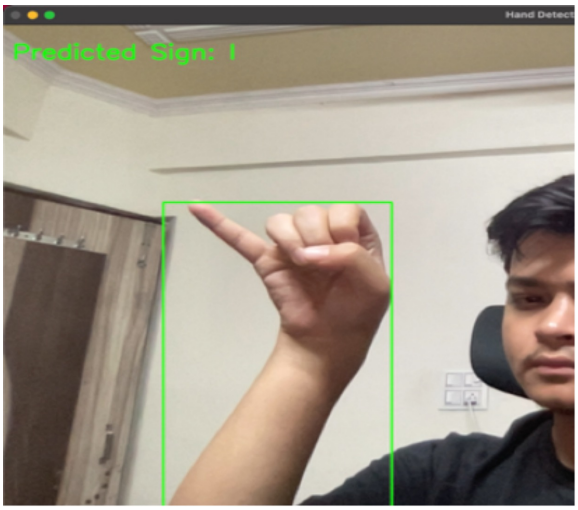

Fig. 7 Real-Time Testing output 2

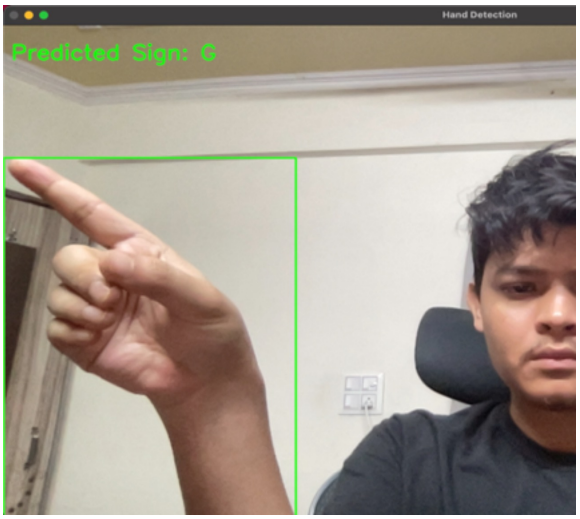

Fig. 8 Real-Time Testing output 3

4.2 Challenges

Some of the challenges faced during the project are:

1. Difficult in recognising different sizes and shape of hand sign.
2. A large computational power is required to process huge amounts of datas.
3. Time-consuming process of training and optimising the CNN model.

5 Conclusion and Future Directions

In this project, our research has demonstrated the efficacy of convolutional neural network (CNN) models in recognizing and interpreting American Sign Language (ASL) gestures. By leveraging deep learning techniques and extensive datasets comprising ASL images, we have developed a robust CNN model capable of accurately classifying hand gestures corresponding to various ASL letters. Through meticulous experimentation and evaluation, we have validated the model's effectiveness, achieving high levels of accuracy on both training and validation datasets. Our findings underscore

the potential of CNNs in facilitating communication for individuals with hearing impairments, paving the way for future advancements in sign language recognition technology.

Future work in this field could focus on several avenues to further enhance sign language recognition technology. Firstly, exploring more extensive and diverse datasets, including variations in hand poses, lighting conditions, and backgrounds, could improve the robustness of CNN models and their ability to generalize to real-world scenarios. Additionally, incorporating real-time video processing capabilities into sign language recognition systems would enable immediate feedback and interaction, facilitating more natural and seamless communication. Furthermore, integrating advanced techniques such as attention mechanisms and recurrent neural networks (RNNs) could enhance the temporal understanding of sign language gestures, leading to more accurate and context-aware interpretations. Lastly, collaboration with sign language experts and communities could provide valuable insights into the practical challenges and requirements of sign language users, guiding the development of more user-centred and inclusive solutions.

Declarations

- The authors received no specific funding for this study.
- The authors declare that they have no conflicts of interest to report regarding the present study.
- No Human subject or animals are involved in the research.
- All authors have mutually consented to participate.
- All the authors have consented the Journal to publish this paper.
- Authors declare that all the data being used in the design and production cum layout of the manuscript is declared in the manuscript.

References

[1] Amatya, K.S.G.M.P.: Translation of sign language into text using kinect for windows v2. In: The Eleventh International Conference on Advances in Computer-Human Interactions (2018)

[2] Sun, T.-T.J.S.-B.Z.J.-K.Y.G.-R.J.J.-H.: Research on the hand gesture recognition based on deep learning. In: 12th International Symposium on Antennas, Propagation and EM Theory (ISAPE) (2018)

[3] Hasan, P.K.M.M.M.: Brightness factor matching for gesture recognition system using scaled normalization. International Journal of Computer Science Information Technology (IJCSIT) **3**(2) (2011)

[4] A. I. Simei G. Wysoski, M.V.L.S.K.-.: A rotation invariant approach on static-gesture recognition using boundary histograms and neural networks. International Journal of Artificial Intelligence Applications (IJAIA) **3**(4) (2012)

[5] Joshi, H.S..E.A.A.: American sign language translation using edge detection and cross-correlation. In: IEEE Colombian Conference on Communications and Computing (COLCOM) (2017)

[6] Shiffman, D.: The Nature of Code: Simulating Natural Systems with JavaScript, (2024). No Starch Press