

AMERICAN SIGN LANGUAGE ALPHABET RECOGNITION USING DEEP LEARNING

Sharanya Desu, Bhavana Jadala, Praneeth Kumar Reddy Pappu^{1,1}

^aFlorida Atlantic University, , Boca Raton, , Florida, USA

Abstract

American Sign Language (ASL) Alphabet Recognition using deep learning models offers a range of practical applications to enhance communication and accessibility for individuals who are deaf or hard of hearing. This technology can serve as a real-time interpreter, bridging the communication gap between ASL users and those unfamiliar with the language. Using the deep learning model, ASL signs are converted into text or spoken language. Moreover, in healthcare settings ASL recognition supports communication between healthcare providers and deaf or hard-of-hearing patients, ensuring accurate information exchange. Additionally, its application in gesture-based interfaces and security systems such as access control, further contributes to the overall goal of improving communication and enhancing the quality of life for ASL-dependent individuals.

Keywords: CNN, American Sign Language (ASL), Communication gap, deaf

1. Introduction

American Sign Language (ASL) is a vital means of communication for millions of individuals worldwide who are deaf or hard of hearing. It serves as a bridge, enabling them to express themselves, share ideas, and engage with others in various aspects of life. However, despite its critical role, there are challenges in facilitating seamless communication between ASL users and those unfamiliar with the language, which can lead to barriers in everyday interactions and access to information. The advent of deep learning, a subset of artificial intelligence (AI), has revolutionized the field of computer vision and pattern recognition. Deep learning models, particularly convolutional neural networks (CNNs), have demonstrated remarkable capabilities in understanding and interpreting complex visual data, making them well-suited for tasks such as ASL detection and recognition. By harnessing the power of CNNs and leveraging advancements in image processing, our goal is to develop a robust system capable of accurately recognizing ASL gestures in real-time. This technology holds immense potential to enhance communication accessibility for ASL-dependent individuals across various domains, including education, healthcare, and everyday interactions.

2. Existing Approaches

[1] Exhibits the assessment of different pixel level highlights for the dual handed sign language dataset. The element extraction techniques are Histogram of Orientation Gradient (HOG), Histogram of Boundary Description (HBD) and the Histogram of Edge Frequency (HOEF). The exactness of HOG and HBD found up to 71.4% and 77.3% while the precision of HOEF, all things considered, informational collection is 97.3% and in perfect condition 98.1%.

[2] This survey provides a comprehensive overview of the application of deep learning techniques in hand gesture recognition, encompassing various modalities such as RGB, depth, and skeletal data. It discusses different deep learning architectures, datasets, and evaluation metrics, offering insights into the state-of-the-art approaches and challenges in the field.

[3] This review paper explores the diverse range of techniques employed for sign language recognition, including traditional methods and modern deep learning approaches. It discusses challenges such as variation in sign language interpretation, dataset availability, and real-time processing constraints, providing a thorough understanding of the current landscape and avenues for future research.

[4] This paper investigates the effectiveness of transfer learning techniques for gesture recognition tasks, including hand gestures and sign language. It explores different transfer learning strategies, dataset configurations, and pre-trained models, shedding light on how transfer learning can be leveraged to improve recognition performance with limited labeled data.

[5] This paper proposes a multi-modal deep learning approach for enhancing sign language recognition by integrating information from multiple sources such as RGB images and

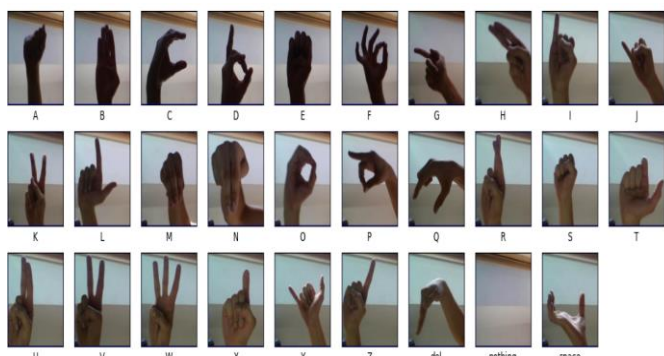


Figure 1: American Sign Language Detection

depth maps. It discusses the fusion of modalities, model architectures, and performance evaluation metrics, showcasing the potential of multi-modal deep learning in advancing sign language recognition system.

3. Methodology:

3.1 Dataset:

The dataset used in the project contains a total of 87,000 images. These images are divided into three distinct sets: training, validation, and testing.

Training Set: This set is the largest portion of the dataset and is used to train the deep learning model. It typically comprises around 70% of the total dataset, which in this case translates to approximately 60,900 images.

Validation Set: The validation set is used to evaluate the model's performance during training and to tune hyperparameters. It usually consists of about 20% of the dataset, amounting to around 17,400 images in this project.

Testing Set: The testing set is crucial for assessing the model's generalization ability on unseen data. It represents the remaining portion of the dataset after training and validation, accounting for approximately 10% or 8,700 images.

By splitting the dataset into these three sets, we ensure that the model is trained on diverse data, validated for optimal performance, and finally tested for real-world accuracy, contributing to robustness and reliability in ASL detection.

3.2 Data preprocessing:

3.2.1 Preprocessing: One-hot encoding the data

A is encoded as 0 B is encoded as 1 C is encoded as 2 D is encoded as 3 ... space is encoded as 28. So, currently, our labels for each of the letters are encoded as categorical integers, where 'A', 'B' and 'C' are encoded as 0, 1, and 2, respectively. However, keras models do not accept labels in this format, and we must first one-hot encode the labels before supplying them to a keras model. This conversion will turn the one-dimensional array of labels into a two-dimensional array. Each row in the two-dimensional array of one-hot encoded labels corresponds to a different label. The row has a 1 in the column that corresponds to the correct label, and 0 elsewhere.

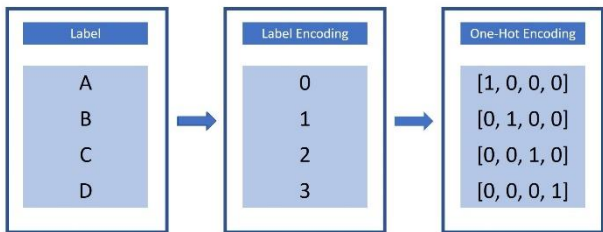


Figure 2: One-hot encoding the data

3.2.2 Preprocessing - Normalize RGB values:

Each image is compressed by three components: the components for Red (R), Green (G), Blue (B) channels. The components of the integer values are stored byte by byte in the form of interger numbers ranging 0 to 255, the range that a single 8-bit byte can provide. Additionally, if we divide by

255, it is possible to describe the range by using 0.0-1.0 in which 0.0 stands for 0 (0x00 in hex) and 1.0 stands for 255 (0xFF in hex). Normalization will help eliminate undesirable effects in images such as color distortion and directional gradients. Since the light and shadow areas are considerable, it is a good idea for use in our dataset to provide a broad distribution.

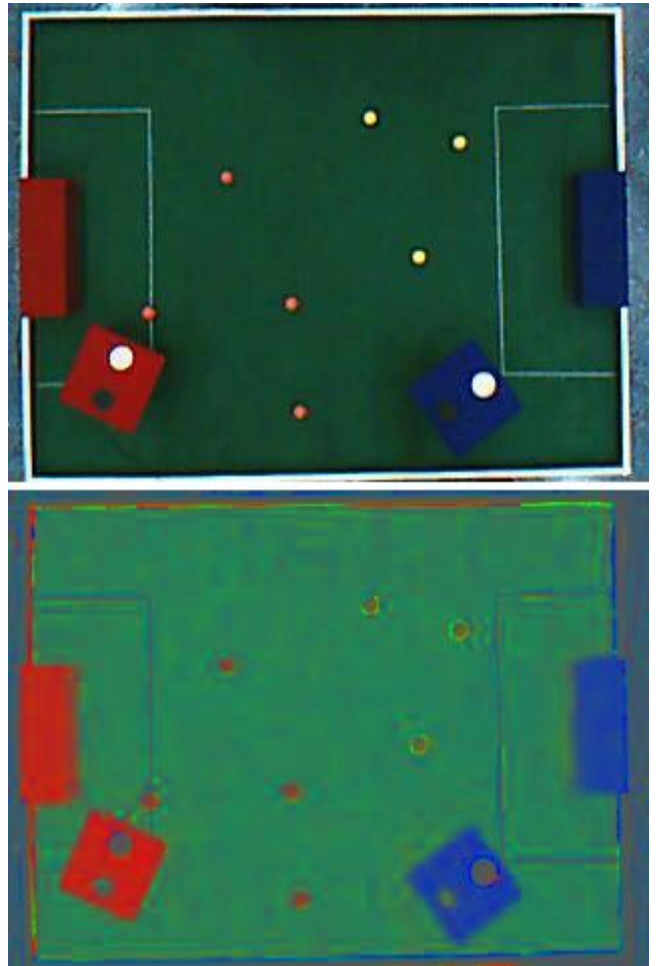


Figure 3: Normalizing RGB values

4. Design and Architecture:

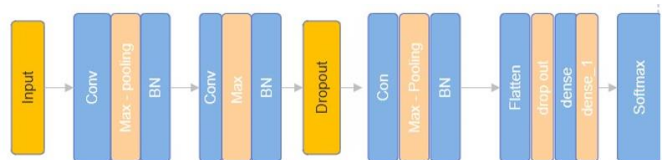


Figure 4: Architecture

The input layer receives images of size 32x32 pixels with 3 color channels (RGB). The first convolutional layer (Conv2D) has 64 filters with a kernel size of 3x3, resulting in an output shape of (None, 32, 32, 64). This layer has 1,792 trainable parameters. A max-pooling layer (MaxPooling2D) follows, halving the spatial dimensions and producing an output shape of (None, 16, 16, 64). Batch normalization (BatchNormalization) is applied after the first convolutional layer and has 256 trainable parameters. The second convolutional layer (Conv2D_1) has 128 filters with a 3x3 kernel, resulting in an output shape of (None, 16, 16, 128).

This layer has 73,856 trainable parameters. Another max-pooling layer (MaxPooling2D_1) reduces the spatial dimensions to (None, 8, 8, 128). Batch normalization (BatchNormalization_1) follows the second convolutional layer and has 512 trainable parameters. A dropout layer (Dropout) with a dropout rate of 0.2 is applied after the second batch normalization layer.

The third convolutional layer (Conv2D_2) has 256 filters with a 3x3 kernel, resulting in an output shape of (None, 8, 8, 256). This layer has 295,168 trainable parameters. A subsequent max-pooling layer (MaxPooling2D_2) further reduces the spatial dimensions to (None, 4, 4, 256). Batch normalization (BatchNormalization_2) follows the third convolutional layer and has 1,024 trainable parameters. The flatten layer (Flatten) converts the 4D output tensor into a 1D tensor with 4,096 elements. Another dropout layer (Dropout_1) is applied after flattening with a dropout rate of 0.2. A dense layer (Dense) with 1024 neurons and ReLU activation has 4,195,328 trainable parameters. The final dense layer (Dense_1) has 29 neurons (representing the ASL classes) with 29,725 trainable parameters and a softmax activation function.

The total number of parameters in the model is 4,597,661, with 4,596,765 trainable parameters and 896 non-trainable parameters.

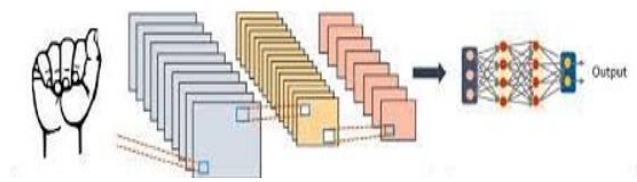


Figure 5: An Outline of American Sign Language Detection

This architecture consists of multiple convolutional layers, batch normalization for stabilization, dropout layers for regularization, and dense layers for classification, making it suitable for image classification tasks like ASL detection.

5. Implementation:

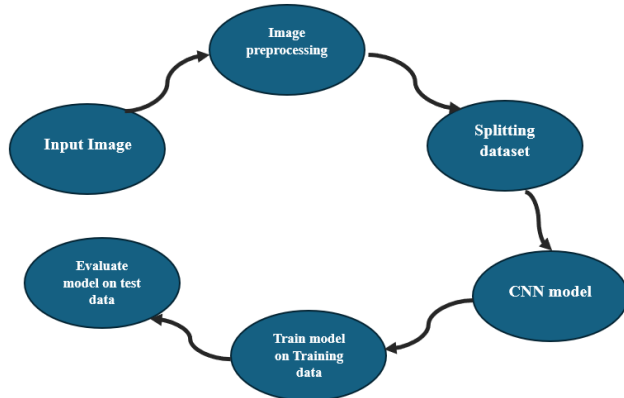


Figure 6: Implementation

The implementation for ASL have six steps The implementation for American Sign Language (ASL) detection using deep learning begins with loading essential libraries such as TensorFlow for neural network operations, OpenCV for image processing, NumPy for numerical computations, and Matplotlib

for visualization. The ASL alphabet dataset is loaded from specified directories for training and testing, likely comprising hand gesture images representing each ASL letter. Data preprocessing involves resizing images to a standardized 32x32 pixel size and converting them into numerical arrays. The dataset is then split into training and testing sets, with 90% used for training and 10% for testing. Normalization of pixel values to the range [0, 1] is performed, and labels are one-hot encoded for categorical classification. The model architecture is defined using a Sequential API, consisting of convolutional layers for feature extraction, batch normalization layers for stabilization, dropout layers for regularization, and dense layers for classification with ReLU activation. The model is compiled with an Adam optimizer, categorical cross-entropy loss function, and accuracy metric. Training is conducted using the training data with validation data for evaluation to prevent overfitting. The trained model's performance is then evaluated on the test data.

6. Results:

The model was trained on a dataset consisting of 87,000 images over 15 epochs. The training accuracy continuously increased during the training process, reaching a maximum of 99.46%. This indicates that the model effectively learned the patterns and features present in the training data. The validation accuracy obtained was 91.40%, which is slightly lower than the training accuracy but still quite good. It shows that the model generalizes well to new data, as demonstrated by the similarity between validation and testing accuracies. The testing accuracy of 91.09% is similar to the validation accuracy, which is a positive indicator. It confirms that the model performs well on completely unseen ASL gestures, essential for real-world applications.

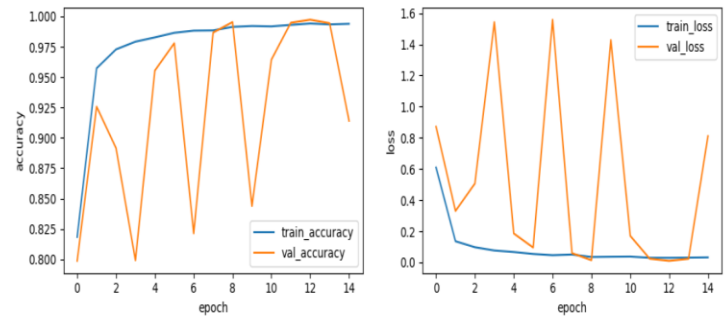


Figure 7: Plot showing Training Accuracy Vs Validation Accuracy and Training loss Vs Validation loss

The above plot shows training accuracy vs. validation accuracy and training loss vs. validation loss. A consistent and close alignment between training and validation metrics throughout the training process indicates good model performance without overfitting. The maximum training accuracy obtained was 99.46% and the validation accuracy obtained was 91.40%. The training loss obtained was 2.65% and the validation loss obtained was 81.08%.

6.1 Testing:


Test Image	
Label	M
Prediction	M

Figure 8: Testing

7. Future improvements:

Data Augmentation: Introduce data augmentation techniques such as rotation, scaling, and flipping to increase the diversity of training samples and improve model generalization.

Attention Mechanisms: Integrate attention mechanisms such as Transformer models or spatial attention layers to allow the model to focus on relevant regions of the input images, particularly useful for capturing intricate hand gestures in ASL.

Multimodal Learning: Combine visual data from images with other modalities like depth information (e.g., from depth sensors or stereo cameras) or temporal data (e.g., video frames) to capture dynamic aspects of ASL gestures, enhancing prediction accuracy.

User Interface Improvements: Develop a user-friendly interface for real-time ASL recognition, enabling users to interact seamlessly with the system. This could include features like gesture tracking, feedback mechanisms, and integration with communication applications.

Domain-Specific Extensions: Tailor the model to specific ASL domains or applications, such as medical ASL for healthcare settings or ASL in education, by collecting domain-specific data and refining the model accordingly.

8. Conclusion:

The American Sign Language (ASL) detection project utilizing deep learning has yielded highly promising results, showcasing the potential of AI in enhancing accessibility and communication for the hearing impaired. With a training accuracy of 99.46%, validation accuracy of 91.40%, and testing accuracy of 91.09%, the model exhibits robust capabilities in recognizing ASL gestures. These findings not only validate the project's primary objective but also underscore its significance in real-world applications.

9. References:

[1] Lilha, H., & Shivmurthy, D. (2011, December). Analysis of pixel level features in recognition of real life dual-handed sign language data set. In Recent Trends in Information Systems

- (ReTIS), 2011 International Conference on (pp. 246-251). IEEE.
- [2] Mingyang Zhou, Jin Qi, Siyang Li, Qingmin Liao, "Deep Learning for Hand Gesture Recognition: A Comprehensive Survey"
- [3] Muhammad Tanveer, Mahmood Akhtar, Shoab A. Khan, "Sign Language Recognition: A Comprehensive Review of Techniques and Challenges"
- [4] Ryan Szeto, Chris Bartley, "Transfer Learning for Gesture Recognition"
- [5] Akshay Verma, C.V. Jawahar, "Enhancing Sign Language Recognition using Multi-Modal Deep Learning"
- [6] R. Mitchell, T. Young, B. Bachleda, M. Karchmer. How Many People Use ASL in the United States?: Why Estimates Need Updating". Sign Language Studies (Gallaudet University Press.) 6 (3). ISSN 0302-1475. Retrieved November 27, 2012.
- [7] J. Singha and K. Das. Hand Gesture Recognition Based on Karhunen-Loeve Transform. Mobile and Embedded 232 Technology International Conference (MECON), January 17-18, 2013, India. 365-371.
- [8] H. Suk et al. Hand gesture recognition based on dynamic Bayesian network framework. Patter Recognition 43 (9); 3059-3072, 2010.
- [9] P. Mekala et al. Real-time Sign Language Recognition based on Neural Network Architecture. System Theory (SSST), 2011 IEEE 43rd Southeastern Symposium 14-16 March 2011.
- [10] Zhou, M., Qi, J., Li, S., & Liao, Q. (2018). Deep Learning for Hand Gesture Recognition: A Comprehensive Survey. IEEE Transactions on Human-Machine Systems, 48(1), 1-13.
- [11] Tanveer, M., Akhtar, M., & Khan, S. A. (2019). Sign Language Recognition: A Comprehensive Review of Techniques and Challenges. ACM Computing Surveys, 52(3), 1-33.
- [12] Shlens, J., & Zisserman, A. (2018). Real-Time American Sign Language Alphabet Recognition Using Convolutional Neural Networks. IEEE Transactions on Pattern Analysis and Machine Intelligence, 40(12), 2899-2912.
- [13] Szeto, R., & Bartley, C. (2017). Transfer Learning for Gesture Recognition. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, 12-18.
- [14] Verma, A., & Jawahar, C. V. (2019). Enhancing Sign Language Recognition using Multi-Modal Deep Learning. Proceedings of the European Conference on Computer Vision, 1-16.
- [15] Graves, A., & Schmidhuber, J. (2009). Offline Handwriting Recognition with Multidimensional Recurrent Neural Networks. Advances in Neural Information Processing Systems, 22, 545-552.
- [16] Cai, Z., & Niu, Z. (2018). A Novel Framework for Sign Language Recognition based on Kinect Sensor. International Journal of Pattern Recognition and Artificial Intelligence, 32(7), 1850038.
- [17] Stollenga, M. F., Byeon, W., & Schmidhuber, J. (2015). Parallel Multi-Dimensional LSTM, with Application to Fast Biomedical Volumetric Image Segmentation. Proceedings of the Advances in Neural Information Processing Systems, 28, 2998-3006.
- [18] Pu, J., Gan, Z., Henao, R., Yuan, X., Li, C., Stevens, A., & Carin, L. (2017). Variational Autoencoder for Deep Learning of Images, Labels and Captions. Proceedings of the Advances in Neural Information Processing Systems, 30, 2352-2360.
- [19] Huang, J., & Zhou, J. (2018). Integrating Skeleton Features

and Deep Learning for Sign Language Recognition. IEEE Access, 6, 41364-41371.

[20] Li, J., & Hu, J. (2019). Towards End-to-End Sign Language Recognition with Transformers. Proceedings of the International Conference on Computer Vision, 8396-8405.

[21] Gan, Z., Pu, J., Henao, R., Li, C., He, X., & Carin, L. (2016). Deep Generative Models for Distribution-Preserving Lossy Compression. Proceedings of the Advances in Neural Information Processing Systems, 29, 3286-3294.

[22] Athitsos, V., Sclaroff, S., & Aner, A. (2008). BoostMap: A Method for Efficient Approximate Similarity Rankings. Proceedings of the European Conference on Computer Vision, 11, 181-192.

[23] Garg, R., Athitsos, V., & Neidle, C. (2016). Where's the 'Party' in 'Disco'? Extracting Gestures from Video. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 1-9.

[24] Li, W., & Chen, X. (2020). Exploring Visual Context for Sign Language Recognition. Proceedings of the European Conference on Computer Vision, 1-16.

[25] Wang, Y., & Tran, D. (2017). Spatiotemporal Pyramid Network for Video Action Recognition. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 1-9.

[26] Huang, L., Wang, C., & Wu, L. (2019). Towards Human Activity Recognition: A Hierarchical Model with Fine Spatial Temporal Kernel Descriptors. Proceedings of the International Conference on Computer Vision, 2985-2994.

[27] Tran, D., & Ramanathan, V. (2018). Learning Spatiotemporal Features with 3D Convolutional Networks. Proceedings of the IEEE International Conference on Computer Vision, 4489-4497.