Near-Perfect Alpha-Numeric ASL Recognition Using Salient Object Detection and 2D CNNs

Forrest Moulin
B.Sc. in Information Sciences and Technology
Independent Researcher
contact@mainstreamstudios.ai

Abstract

This research presents SignWave, an innovative alpha-numeric American Sign Language (ASL) recognition system developed by the author, which uniquely combines computer vision, machine learning, and salient object detection techniques. Leveraging background noise removal preprocessing and a deep convolutional neural network (CNN), SignWave recorded an outstanding 99.97% validation accuracy on a compact dataset of 2.25 GB, representing 36 ASL gesture classes A-Z and 1-10. Trained on a cloud virtual machine without a graphics processing unit (GPU), The model's efficiency is highlighted by its use of only 20 epochs, and the results demonstrate that high-accuracy sign language recognition (SLR) can be achieved with accessible computing resources. Designed for real-time sign language interpretation software, SignWave aims to enhance communication between sign language users and non-signers. By showcasing noticeable accuracy, efficiency, and a broader range of classes than some contemporary methods, this research has the potential to advance communication accessibility.

Introduction

American Sign Language is a visual-spatial language with unique linguistic rules, used commonly in deaf communities of North America [2, 3], making it an ideal language for computer vision interpretation use cases. In the United States, approximately 11 million individuals were reported to be deaf or have serious difficulty hearing in 2021 [15]. The language has seen significant growth in popularity, with ASL courses experiencing a 6,583% enrollment increase from 1990 to 2016, making it the third most-studied language on U.S. college campuses [3].

Sign language characters are communicated via fingerspelling, which involves manual representation of letters or numbers using specific handshapes. Fingerspelling serves as a critical component of ASL communication, allowing signers to spell out proper nouns, acronyms, and words that lack a corresponding sign [9]. In the context of SLR systems like SignWave, fingerspelling classification is fundamental, as it provides the initial framework for identifying simpler signs, paving the way for more advanced word gesture recognition and interpretation capabilities.

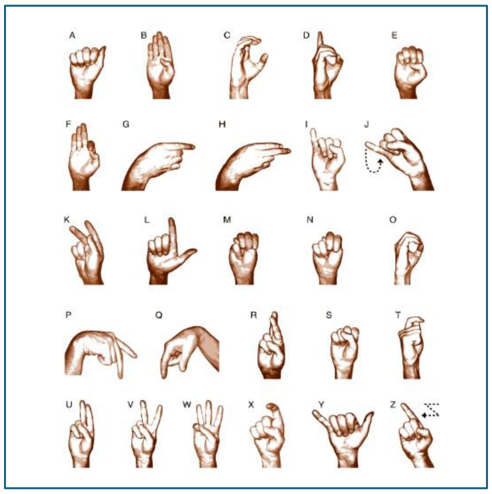


Figure 1: ASL Alphabet Fingerspelling [18]

SLR systems have potential to play a crucial role in bridging communication gaps between deaf individuals and the hearing community worldwide. According to the National Geographic Society [16], over 300 distinct sign languages are used worldwide, serving a community of more than 72 million deaf or hard-of-hearing individuals. These systems can enhance accessibility, improve educational outcomes, and foster social inclusion.

One challenge with existing SLR systems is that datasets may have diverse backgrounds that introduce noise and prevent effective training results. Furthermore, traditional methods often require larger datasets as well as extensive preprocessing and data augmentation, making them resource intensive. For example, preprocessing of the CNN model published by Kumar et al. [8] required that 49,000 images be removed from the initial dataset.

SignWave aims to address this gap by utilizing a single preprocessing step—background noise removal—resulting in a more lightweight model that operates efficiently with smaller, non-augmented datasets. Unlike traditional SLR methods that rely on extensive datasets, which can exceed 100,000 images and require 100 epochs, the proposed approach achieved significant results with only 36,000 images and 20 epochs [4].

Potential SLR applications include sign language proficiency assessments as well as vision-based translation and interpretation through Sign-to-Text (STT) or Sign-to-Speech (STS) systems for transcription or vocalization. The vision-based approach is notably more accessible compared to sensor-based glove systems, which can be impractical due to the inconvenience of wearing and removing gloves lack of portability, and the expenses associated with custom hardware for various hand sizes.

Alternatively, widespread availability of smartphones and computers with front-facing cameras and sufficient random-access memory (RAM) specifications enhances the feasibility of vision-based sign language solutions. According to Pew Research Center [17], approximately 9 out of 10 Americans own a smartphone. Due to the high level of accessibility of smartphone and computer cameras, vision-based systems are a promising avenue for improving communication and inclusivity for deaf or hard-of-hearing individuals.

Related Work

Recent advances in machine learning and computer vision have significantly enhanced the accuracy and sophistication of SLR systems, particularly for alpha-numeric recognition. Contemporary SLR systems often leverage state-of-the-art open-source tools such as MediaPipe for hand landmark detection or gesture recognition. MediaPipe, an open-source framework developed by Google, facilitates the construction and deployment of machine learning pipelines and is widely used in SLR research [12].

In the early stages of SLR research, models struggled with lower accuracy rates due to limited computational power and less sophisticated algorithms. For example, early systems in the 1990s and early 2000s achieved recognition rates around 80-90% using techniques like Hidden Markov Models (HMMs) and basic feature extraction methods [20, 8]. These systems were often limited by the small datasets and the simplistic nature of the models employed.

Sundar and Bagyammal [21] achieved 99% accuracy on 26 ASL classes by combining MediaPipe with Long Short-Term Memory (LSTM) networks. However, their dataset was relatively shallow, comprising an equivalent total of 3,380 images. Kumar et al. [8] reported an impressive 99.95% accuracy on the same number of ASL classes using MediaPipe with a CNN. Similarly, Barbhuiya et al. [1] recorded a 99.82% accuracy on 36 ASL classes using a combination of the pre-trained AlexNet model and Support Vector Machines (SVM).

Method	**Classes**	**Parameters**	**Accuracy (%)**
MediaPipe & CNN[8]	26	Not specified	99.95
Simple CNN [4]	29	2,029,150	99.89
AlexNet & SVM[1]	36	Not specified	99.82
MediaPipe & LSTM[21]	26	188,090	99.00

Table 1: Related SLR Model Comparison

Zhou et al. [23] explored a sensor-based approach, achieving 98.63% accuracy on 660 sign gestures. Despite its high accuracy, the method's limitations in terms of portability, hardware costs, and the inconvenience of using specialized gloves pose significant challenges. Additionally, Li et al. [11] proposed a word-level deep sign language recognition system from video, demonstrating the potential of large-scale datasets and comparing different methods for enhanced performance in SLR tasks.

Methodology

The dataset used for training and validation was sourced from Synthetic ASL Alphabet and Synthetic ASL Numbers [5, 6]. Initially, the sourced dataset required 9.97 GB of storage and consisted of 37,000 images with a resolution of 512 x 512 pixels, including 'Blank' classes with random backgrounds. After the 'Blank' classes were removed, the dataset was organized into 36 classes representing ASL letters A-Z and numbers 1-10. Each class contained 900 images for training and 100 images for validation.

Preprocessing involved background removal using salient object detection via the rembg Python library to crop the hand from the image foreground and remove the background. This approach significantly reduced dataset size and complexity, resulting in a streamlined dataset of 2.25 GB — over a 75% reduction from the original size. This preprocessing step aimed to enhance model performance by eliminating background noise and focusing the CNN on relevant gesture features.

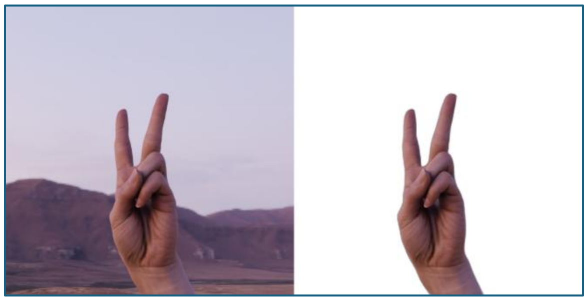


Figure 2: ASL Letter V Background Removal with Salient Object Detection

SignWave was developed using the Python 3.11.2 programming language and implemented with TensorFlow 2.16.1, a versatile deep learning framework providing robust support for designing and training neural networks. This study used a Google Cloud E2 virtual machine configured with 16 GB RAM, 4 virtual CPUs, and 20 GB mounted storage, which contrasts with more resource intensive environments commonly used in similar studies

The development process also incorporated several open-source libraries to enhance functionality and streamline tasks. Matplotlib 3.9.0 was utilized for line graph visualizations, providing clear and informative visual representation of data. NumPy 1.26.2 facilitated numerical operations, ensuring efficient computation and manipulation of large datasets. Pandas 2.1.4 was employed for organizing training history into DataFrames, making data handling more manageable. Rembg 2.0.57 was used for background removal, a crucial preprocessing step in this research's image data preparation. Scikit-learn 1.5.1 enabled the computation of confusion matrices, essential for evaluating model performance, and Seaborn 0.13.2 was leveraged for visualizing these confusion matrices, enhancing the interpretability of classification results.

SignWave includes three distinct variants and architectures:

Model	**Classes**	**CNN Layers**	**Trainable Parameters**
ABC	26 (A-Z)	10	1,014,394
123	10 (1-10)	10	1,012,330
ABC-123	36 (A-Z & 1-10)	15	2,086,116

Table 2: SignWave Model Comparison

The ABC and 123 models were designed using a simpler 10-layer CNN architecture most comparable to the LeNet-5 structure. These models incorporate two convolutional layers followed by pooling layers, and two fully connected layers. ABC-123 utilized a 15-layer CNN architecture similar to a simplified version of VGG16, with multiple convolutional layer pairs followed by pooling layers for handling the larger number of classes [19, 22]. All three models were trained for 20 epochs with a batch size of 25 images, which were rescaled to 50 x 50 pixels to reduce the total number of parameters and accelerate the training process.

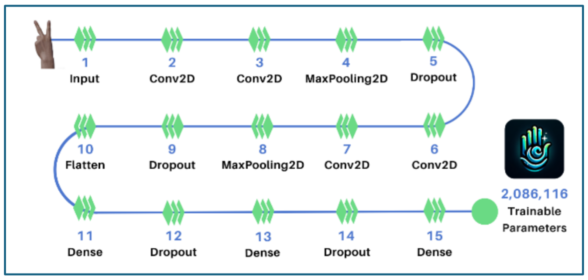


Figure 3: SignWave ABC-123 Deep CNN Architecture

A custom Keras Callback class was implemented to monitor model performance. This callback generated and updated accuracy and loss curve plots for both training and validation datasets using Matplotlib. The plots, along with relevant tables, were compiled into a single PDF report updated at the end of each epoch. This approach provided real-time insights into the model's progress and facilitated the analysis of training dynamics. After each epoch, the custom callback saved the model with a unique file name that included the model type, version, epoch number, and timestamp This naming convention ensured efficient management of model iterations, making it easier to retrain or utilize the Keras files in an organized manner.

To provide a comprehensive view of model performance, several visualizations were generated, including accuracy curves, cross-entropy loss curves, confusion matrices, and data tables. Model performance was analyzed by comparing training and validation metrics to ensure consistent accuracy. This evaluation process involved cross-referencing validation accuracy with training results to validate model reliability and guide necessary adjustments. This methodology ensured continuous assessment of performance metrics, enabling iterative improvements and robust model development.

Performance evaluation included plotting a confusion matrix to analyze prediction performance by class. The results were visualized using Matplotlib and Seaborn, providing a comprehensive view of the model's effectiveness in recognizing and classifying ASL gestures.

Results

The SignWave models achieved exceptional performance metrics, indicating their effectiveness and efficiency in ASL recognition tasks. Notably, the models recorded near-perfect validation accuracy as well impressive cross-entropy loss values. The validation results were as follows:

Model	**Validation Accuracy (%)**	**Validation Loss**
ABC	99.96	0.0023
123	100.0	0.0002
ABC-123	99.97	0.0009

Table 3: SignWave Model Validation Metrics

The training and validation accuracy curves for SignWave ABC-123 depicted in Figure 4 illustrate the model's effective learning process. The plateauing difference between the training and validation curves suggests slight underfitting, which can lead to improved generalization of unseen data [7].

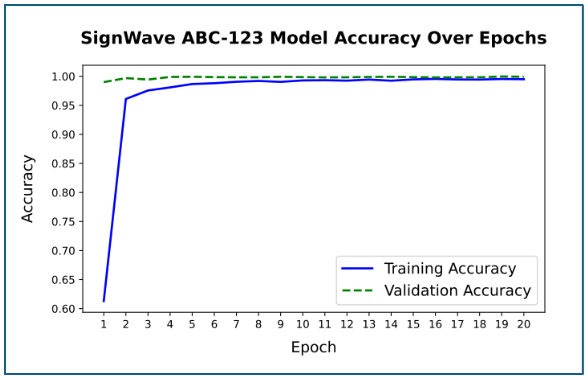


Figure 4: SignWave ABC-123
Training and Validation Accuracy Curves

The confusion matrix for SignWave ABC-123 presented in Figure 5 details the model's classification performance across the 36 ASL gesture classes. The true prediction values along the diagonal indicate the model perfectly classified 35/36 classes of the ABC-123 model. The lone misclassification occurred with the letter R, which was mistaken for the letter C only once.

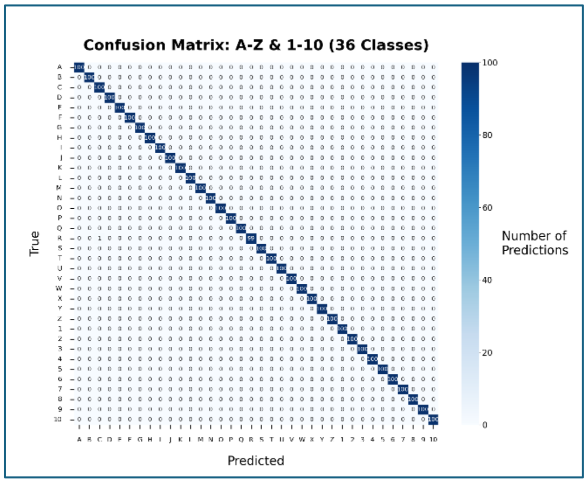


Figure 5: SignWave ABC-123 Confusion Matrix

Class	Precision	Recall	F1-Score
R	0.99	1.0	0.99
C	1.0	0.99	0.99
All Others	1.0	1.0	1.0

Table 4: ABC-123 Model
Class Performance Metrics

Precision, recall, and F1-scores were calculated for each class, shown in Table 4. Precision is the ratio of correctly predicted positive observations to the total predicted positives, while recall is the ratio of correctly predicted positive observations to all observations in the actual class. F1-score is the weighted average of precision and recall.

Discussion

The SignWave models were benchmarked against known ASL recognition methods to assess their relative performance. Traditional methods typically rely on extensive datasets, prolonged training periods, and advanced computational resources, such as GPUs.

For instance, SignWave utilized a streamlined dataset of 2.25 GB for the ABC-123 model, which contained 60% less images than the dataset used by a comparable model that only classified 29 ASL classes [4]. This reduction in dataset size without compromising accuracy underscores the effectiveness of the salient object detection technique to remove background noise.

Unlike traditional models that often require more costly CPUs and GPUs for training, SignWave models were trained on a cloud virtual machine without a GPU. For example, research by Elsayed et al. [4] utilized an Intel Core i9 CPU and NVIDIA GeForce RTX 2080 Ti GPU, suggesting a significantly more complex and costly computing environment. By utilizing a more accessible configuration, our approach demonstrates that high-performance results can be achieved without the necessity for such expensive hardware. This approach highlights the potential for data scientists and machine learning specialists to conduct advanced research and development with more affordable computing resources.

Traditional ASL recognition models often involve 50 or more epochs for training [4, 8, 21]. Conversely, SignWave's achieved superior validation accuracy in less than 20 epochs, demonstrating a substantial improvement in training efficiency when background noise is removed. In fact, the ABC model reached 99.96% accuracy in just 5 epochs, and the 123 model reached 100% accuracy in just 13 epochs.

However, a limitation of using salient object detection as a preprocessing step, is that this same preprocessing is required before the image can be passed to the model for prediction in the context of a live ASL interpretation or video-based ASL translation. While this technique is near perfect with static images, performance with video feed frames may vary. Thus, at least 16 GB RAM is recommended for deployment in a live recognition system. Additionally, background removal does reduce the dataset storage size, but the process can be time consuming, potentially taking longer than the training process itself. Lastly, while the data appears to be representational of real-world images of hands, it was synthetically generated.

With a sufficient RAM configuration, the ABC-123 model can be deployed in conjunction with a real-time Python app that offers Sign-to-Text to transcribe the processed gesture, or Sign-to-Speech to read it aloud. In real-time SLR, there are limitations to the speed at which signs can be predicted and displayed due to the sequential nature of the process. Initially, the system must detect that a hand movement or sign change has occurred, which involves processing the video feed to identify and track hand gestures.

Once a sign is detected, the system then classifies the gesture, analyzing the movement and matching it to known sign patterns using machine learning models. After recognizing the gesture, the system generates and displays the prediction, converting the recognized sign into text or another form of output and updating the display [13]. Each of these stages introduces some delay, contributing to the overall latency of the system. This latency impacts how quickly the system can respond to and display the recognized signs, affecting the flow of communication in real-time SLR contexts.

The ABC-123 model can also predict 36 classes from appropriately cropped static images when backgrounds have already been removed. Because images were rescaled at training, dataset images of larger sizes can be rescaled down to 50 x 50 pixels before being passed to the model for class prediction.

Conclusion

This research has confirmed the efficacy of the SignWave system for recognizing alpha-numeric American Sign Language (ASL) gestures using salient object detection and 2D convolutional neural networks (CNNs). The successful application of these techniques in a static image processing setting underscores the potential of resource-efficient solutions for sign language recognition (SLR).

SignWave's high accuracy, achieved without the need for a GPU, underscores the potential of leveraging public cloud infrastructure for developing advanced SLR systems. This research demonstrates that high-performance models can be effectively developed and deployed in the cloud, highlighting the logistical and cost benefits associated with cloud computing for SLR. The successful application of resource-efficient techniques, such as innovative preprocessing methods and streamlined model architectures, showcases how public cloud environments can provide a practical and economical solution for creating high-accuracy SLR applications. These findings suggest that cloud-based approaches can significantly reduce the need to purchase expensive hardware while still achieving competitive performance in machine learning tasks.

Currently, most SLR systems are capable of handling basic Sign-to-Text for individual letters or numbers, which is analogous to vocalizing each letter or number sequentially. However, a more advanced, word-based solution would provide a more conversational experience, similar to Speech-to-Text technologies. This would enable a more natural and fluid communication style, similar to actual conversation rather than discrete sign recognition.

Future research can build upon these findings to explore more complex scenarios, such as live-feed gesture recognition and the development of word-based systems. The next steps include adapting the system for real-time sign language interpretation and Sign-to-Speech functionality, as well as expanding its capabilities to support a broader range of gestures and sign languages. This research paves the way for continued advancements in SLR, aiming to enhance communication accessibility and inclusivity for sign language users worldwide.

References

[1] A. A. Barbhuiya, R. K. Karsh, and R. Jain, "CNN-based feature extraction and classification for sign language," Multimedia Tools and Applications, vol. 80, no. 2, pp. 3051–3069, 2021. Available: https://doi.org/10.1007/s11042-020-09095-1.

[2] Canadian Association of the Deaf, "Language," 2022. [Online]. Available: https://cad-asc.ca/issues-positions/language/. [Accessed: Jul. 22, 2024].

[3] Clemson University, "American Sign Language," 2024. [Online]. Available: https://www.clemson.edu/cah/academics/languages/languages/asl.html. [Accessed: Jul. 22, 2024].

[4] N. Elsayed, A. Ibrahim, and M. Saleh, "Vision-based American Sign Language classification approach via deep learning," arXiv, 2022. [Online]. Available: https://arxiv.org/pdf/2204.04235. [Accessed: Jul. 22, 2024].

[5] O. Fahey, "Synthetic ASL Alphabet [Data set]," Kaggle, Jun. 17, 2022. [Online]. Available: https://www.kaggle.com/datasets/lexset/synthetic-asl-alphabet. [Accessed: Jul. 22, 2024].

[6] O. Fahey, "Synthetic ASL Numbers [Data set]," Kaggle, Jun. 17, 2022. [Online]. Available: https://www.kaggle.com/datasets/lexset/synthetic-asl-numbers. [Accessed: Jul. 22, 2024].

[7] I. Goodfellow, Y. Bengio, and A. Courville, "Deep learning," MIT Press, 2016. Available: https://www.deeplearningbook.org/.

[8] R. Kumar, A. Pandey, and A. Gupta, "Mediapipe and CNNs for real-time ASL gesture recognition," arXiv, 2023. [Online]. Available: https://arxiv.org/pdf/2305.05296. [Accessed: Jul. 22, 2024].

[9] B. Lee and K. Secora, "Fingerspelling and its role in translanguaging," Languages, vol. 7, no. 4, p. 278, 2022. Available: https://doi.org/10.3390/languages7040278.

[10] R. Liang and M. Ouhyoung, "A real-time continuous gesture recognition system for sign language," in Proceedings of the Third International Conference on Automatic Face and Gesture Recognition, 1998, pp. 558-565. Available: https://doi.org/10.1109/AFGR.1998.670869.

[11] D. Li, C. Rodriguez, X. Yu, and H. Li, "Word-level deep sign language recognition from video: A new large-scale dataset and methods comparison," in Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, 2020, pp. 1459-1469. Available: https://doi.org/10.1109/WACV45572.2020.9072872.

[12] C. Lugaresi, J. Tang, H. Nash, C. McClanahan, E. Uboweja, M. Hays, F. Zhang, C.-L. Chang, M. Yong, J. Lee, W.-T. Chang, W. Hua, M. Georg, and M. Grundmann, "MediaPipe: A framework for perceiving and processing reality," in Proceedings of the Third Workshop on Computer Vision for AR/VR at IEEE Computer Vision and Pattern Recognition (CVPR), 2019. [Online]. Available: https://research.google/pubs/mediapipe-a-framework-for-perceiving-and-processing-reality/. [Accessed: Jul. 22, 2024].

[13] F. Moulin, "python-asl-detection," GitHub, Jun. 2024. [Online]. Available: https://github.com/ffm5113/python-asl-detection. [Accessed: Jul. 22, 2024].

[14] National Association of the Deaf, "Learning American Sign Language," 2024. [Online]. Available: https://www.nad.org/resources/american-sign-language/learning-american-sign-language/. [Accessed: Jul. 22, 2024].

[15] National Deaf Center on Postsecondary Outcomes, "How many deaf people live in the United States?" 2024. [Online]. Available: https://nationaldeafcenter.org/faq/how-many-deaf-people-live-in-the-united-states/. [Accessed: Jul. 22, 2024].

[16] National Geographic Society, "Sign language," 2024. [Online]. Available: https://education.nationalgeographic.org/resource/sign-language/. [Accessed: Jul. 22, 2024].

[17] Pew Research Center, "Mobile fact sheet," 2024. [Online]. Available: https://www.pewresearch.org/internet/fact-sheet/mobile/. [Accessed: Jul. 22, 2024].

[18] Pocket Sign, "Sign language alphabet – ASL fingerspelling," n.d. [Online]. Available: https://www.pocketsign.org/alphabet. [Accessed: Jul. 22, 2024].

[19] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," International Journal of Computer Vision, vol. 120, no. 3, pp. 142-158, 2015. Available: https://doi.org/10.1007/s11263-015-0816-y.

[20] T. Starner, J. Weaver, and A. Pentland, "Real-time American Sign Language recognition using desk and wearable computer-based video," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 20, no. 12, pp. 1371-1375, 1997. Available: https://doi.org/10.1109/34.650113.

[21] B. Sundar and T. Bagyammal, "American Sign Language recognition for alphabets using MediaPipe and LSTM," Procedia Computer Science, vol. 215, pp. 642–651, 2022. Available: https://doi.org/10.1016/j.procs.2022.12.066.

[22] Z. Tao, Z. Yang, B. Chen, W. Bao, and H. Cheng, "Protein sequence classification with LetNet-5 and VGG16," in Intelligent Computing Theories and Applications, D. S. Huang, K. H. Jo, J. Jing, P. Premaratne, V. Bevilacqua, and A. Hussain, Eds. Springer, 2022, vol. 13394, pp. 621-633. Available: https://doi.org/10.1007/978-3-031-13829-4_60.

[23] Z. Zhou, K. Chen, X. Li, S. Zhang, Y. Wu, Y. Zhou, and J. Chen, "Sign-to-speech translation using machine-learning-assisted stretchable sensor arrays," Nature Electronics, vol. 3, pp. 571–578, 2020.

References

[1] A. A. Barbhuiya, R. K. Karsh, and R. Jain, "CNN-based feature extraction and classification for sign language," Multimedia Tools and Applications, vol. 80, no. 2, pp. 3051–3069, 2021. Available: https://doi.org/10.1007/s11042-020-09095-1.

[2] Canadian Association of the Deaf, "Language," 2022. [Online]. Available: https://cad-asc.ca/issues-positions/language/. [Accessed: Jul. 22, 2024].

[3] Clemson University, "American Sign Language," 2024. [Online]. Available: https://www.clemson.edu/cah/academics/languages/languages/asl.html. [Accessed: Jul. 22, 2024].

[4] N. Elsayed, A. Ibrahim, and M. Saleh, "Vision-based American Sign Language classification approach via deep learning," arXiv, 2022. [Online]. Available: https://arxiv.org/pdf/2204.04235. [Accessed: Jul. 22, 2024].

[5] O. Fahey, "Synthetic ASL Alphabet [Data set]," Kaggle, Jun. 17, 2022. [Online]. Available: https://www.kaggle.com/datasets/lexset/synthetic-asl-alphabet. [Accessed: Jul. 22, 2024].

[6] O. Fahey, "Synthetic ASL Numbers [Data set]," Kaggle, Jun. 17, 2022. [Online]. Available: https://www.kaggle.com/datasets/lexset/synthetic-asl-numbers. [Accessed: Jul. 22, 2024].

[7] I. Goodfellow, Y. Bengio, and A. Courville, "Deep learning," MIT Press, 2016. Available: https://www.deeplearningbook.org/.

[8] R. Kumar, A. Pandey, and A. Gupta, "Mediapipe and CNNs for real-time ASL gesture recognition," arXiv, 2023. [Online]. Available: https://arxiv.org/pdf/2305.05296. [Accessed: Jul. 22, 2024].

[9] B. Lee and K. Secora, "Fingerspelling and its role in translanguaging," Languages, vol. 7, no. 4, p. 278, 2022. Available: https://doi.org/10.3390/languages7040278.

[10] R. Liang and M. Ouhyoung, "A real-time continuous gesture recognition system for sign language," in Proceedings of the Third International Conference on Automatic Face and Gesture Recognition, 1998, pp. 558-565. Available: https://doi.org/10.1109/AFGR.1998.670869.

[11] D. Li, C. Rodriguez, X. Yu, and H. Li, "Word-level deep sign language recognition from video: A new large-scale dataset and methods comparison," in Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, 2020, pp. 1459-1469. Available: https://doi.org/10.1109/WACV45572.2020.9072872.

[12] C. Lugaresi, J. Tang, H. Nash, C. McClanahan, E. Uboweja, M. Hays, F. Zhang, C.-L. Chang, M. Yong, J. Lee, W.-T. Chang, W. Hua, M. Georg, and M. Grundmann, "MediaPipe: A framework for perceiving and processing reality," in Proceedings of the Third Workshop on Computer Vision for AR/VR at IEEE Computer Vision and Pattern Recognition (CVPR), 2019. [Online]. Available: https://research.google/pubs/mediapipe-a-framework-for-perceiving-and-processing-reality/. [Accessed: Jul. 22, 2024].

[13] F. Moulin, "python-asl-detection," GitHub, Jun. 2024. [Online]. Available: https://github.com/ffm5113/python-asl-detection. [Accessed: Jul. 22, 2024].

[14] National Association of the Deaf, "Learning American Sign Language," 2024. [Online]. Available: https://www.nad.org/resources/american-sign-language/learning-american-sign-language/. [Accessed: Jul. 22, 2024].

[15] National Deaf Center on Postsecondary Outcomes, "How many deaf people live in the United States?" 2024. [Online]. Available: https://nationaldeafcenter.org/faq/how-many-deaf-people-live-in-the-united-states/. [Accessed: Jul. 22, 2024].

[16] National Geographic Society, "Sign language," 2024. [Online]. Available: https://education.nationalgeographic.org/resource/sign-language/. [Accessed: Jul. 22, 2024].

[17] Pew Research Center, "Mobile fact sheet," 2024. [Online]. Available: https://www.pewresearch.org/internet/fact-sheet/mobile/. [Accessed: Jul. 22, 2024].

[18] Pocket Sign, "Sign language alphabet – ASL fingerspelling," n.d. [Online]. Available: https://www.pocketsign.org/alphabet. [Accessed: Jul. 22, 2024].

[19] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," International Journal of Computer Vision, vol. 120, no. 3, pp. 142-158, 2015. Available: https://doi.org/10.1007/s11263-015-0816-y.

[20] T. Starner, J. Weaver, and A. Pentland, "Real-time American Sign Language recognition using desk and wearable computer-based video," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 20, no. 12, pp. 1371-1375, 1997. Available: https://doi.org/10.1109/34.650113.

[21] B. Sundar and T. Bagyammal, "American Sign Language recognition for alphabets using MediaPipe and LSTM," Procedia Computer Science, vol. 215, pp. 642–651, 2022. Available: https://doi.org/10.1016/j.procs.2022.12.066.

[22] Z. Tao, Z. Yang, B. Chen, W. Bao, and H. Cheng, "Protein sequence classification with LetNet-5 and VGG16," in Intelligent Computing Theories and Applications, D. S. Huang, K. H. Jo, J. Jing, P. Premaratne, V. Bevilacqua, and A. Hussain, Eds. Springer, 2022, vol. 13394, pp. 621-633. Available: https://doi.org/10.1007/978-3-031-13829-4_60.

[23] Yale University, "American Sign Language FAQ," 2024. [Online]. Available: https://ling.yale.edu/academics/american-sign-language-asl/faq. [Accessed: Jul. 22, 2024].

[24] Z. Zhou, K. Chen, X. Li, S. Zhang, Y. Wu, Y. Zhou, and J. Chen, "Sign-to-speech translation using machine-learning-assisted stretchable sensor arrays," Nature Electronics, vol. 3, pp. 571–578, 2020. Available: https://doi.org/10.1038/s41928-020-0428-6.

\

Pseudo APA with numbers

[1] Barbhuiya, A. A., Karsh, R. K., & Jain, R. (2021). CNN-based feature extraction and classification for sign language. Multimedia Tools and Applications, 80(2), 3051–3069. https://doi.org/10.1007/s11042-020-09095-1.

[2] Canadian Association of the Deaf. (2022). Language. Retrieved from https://cad-asc.ca/issues-positions/language/.

[3] Clemson University. (2024). American Sign Language. Retrieved from https://www.clemson.edu/cah/academics/languages/languages/asl.html.

[4] Elsayed, N., Ibrahim, A., & Saleh, M. (2022). Vision-based American Sign Language classification approach via deep learning. arXiv. Retrieved from https://arxiv.org/pdf/2204.04235.

[5] Fahey, O. (2022, June 17). Synthetic ASL Alphabet [Data set]. Kaggle. Retrieved from https://www.kaggle.com/datasets/lexset/synthetic-asl-alphabet.

[6] Fahey, O. (2022, June 17). Synthetic ASL Numbers [Data set]. Kaggle. Retrieved from https://www.kaggle.com/datasets/lexset/synthetic-asl-numbers.

[7] Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep learning. MIT Press. https://www.deeplearningbook.org/.

[8] Kumar, R., Pandey, A., & Gupta, A. (2023). Mediapipe and CNNs for real-time ASL gesture recognition. arXiv. Retrieved from https://arxiv.org/pdf/2305.05296.

[9] Lee, B., & Secora, K. (2022). Fingerspelling and its role in translanguaging. Languages, 7(4), 278. https://doi.org/10.3390/languages7040278.

[10] Liang, R., & Ouhyoung, M. (1998). A real-time continuous gesture recognition system for sign language. In Proceedings of the Third International Conference on Automatic Face and Gesture Recognition (pp. 558-565). https://doi.org/10.1109/AFGR.1998.670869.

[11] Li, D., Rodriguez, C., Yu, X., & Li, H. (2020). Word-level deep sign language recognition from video: A new large-scale dataset and methods comparison. In Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (pp. 1459-1469). https://doi.org/10.1109/WACV45572.2020.9072872.

[12] Lugaresi, C., Tang, J., Nash, H., McClanahan, C., Uboweja, E., Hays, M., Zhang, F., Chang, C.-L., Yong, M., Lee, J., Chang, W.-T., Hua, W., Georg, M., & Grundmann, M. (2019). MediaPipe: A framework for perceiving and processing reality. In Proceedings of the Third Workshop on Computer Vision for AR/VR at IEEE Computer Vision and Pattern Recognition (CVPR). Retrieved from https://research.google/pubs/mediapipe-a-framework-for-perceiving-and-processing-reality/.

[13] Moulin, F. (2024, June). python-asl-detection. GitHub. Retrieved from https://github.com/ffm5113/python-asl-detection.

[14] National Association of the Deaf. (2024). Learning American Sign Language. Retrieved from https://www.nad.org/resources/american-sign-language/learning-american-sign-language/.

[15] National Deaf Center on Postsecondary Outcomes. (2024). How many deaf people live in the United States? Retrieved from https://nationaldeafcenter.org/faq/how-many-deaf-people-live-in-the-united-states/.

[16] National Geographic Society. (2024). Sign language. Retrieved from https://education.nationalgeographic.org/resource/sign-language/.

[17] Pew Research Center. (2024). Mobile fact sheet. Retrieved from https://www.pewresearch.org/internet/fact-sheet/mobile/.

[18] Pocket Sign. (n.d.). Sign language alphabet – ASL fingerspelling. Retrieved from https://www.pocketsign.org/alphabet.

[19] Simonyan, K., & Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. International Journal of Computer Vision, 120(3), 142-158. https://doi.org/10.1007/s11263-015-0816-y.

[20] Starner, T., Weaver, J., & Pentland, A. (1997). Real-time American Sign Language recognition using desk and wearable computer-based video. IEEE Transactions on Pattern Analysis and Machine Intelligence, 20(12), 1371-1375. https://doi.org/10.1109/34.650113.

[21] Sundar, B., & Bagyammal, T. (2022). American Sign Language recognition for alphabets using MediaPipe and LSTM. Procedia Computer Science, 215, 642–651. https://doi.org/10.1016/j.procs.2022.12.066.

[22] Tao, Z., Yang, Z., Chen, B., Bao, W., & Cheng, H. (2022). Protein sequence classification with LetNet-5 and VGG16. In D. S. Huang, K. H. Jo, J. Jing, P. Premaratne, V. Bevilacqua, & A. Hussain (Eds.), Intelligent Computing Theories and Applications (Vol. 13394, pp. 621-633). Springer. https://doi.org/10.1007/978-3-031-13829-4_60.

[23] Yale University. (2024). American Sign Language FAQ. Retrieved from https://ling.yale.edu/academics/american-sign-language-asl/faq.

[24] Zhou, Z., Chen, K., Li, X., Zhang, S., Wu, Y., Zhou, Y., & Chen, J. (2020). Sign-to-speech translation using machine-learning-assisted stretchable sensor arrays. Nature Electronics, 3, 571–578. https://doi.org/10.1038/s41928-020-0428-6.

References

[1] Barbhuiya, A. A., Karsh, R. K., & Jain, R. (2021). CNN-based feature extraction and classification for sign language. *Multimedia Tools and Applications, 80*(2), 3051–3069. https://doi.org/10.1007/s11042-020-09095-1

[2] Canadian Association of the Deaf. (2022). Language. Retrieved from https://cad-asc.ca/issues-positions/language/

[3] Clemson University. (2024). American Sign Language. Retrieved from https://www.clemson.edu/cah/academics/languages/languages/asl.html

[4] Elsayed, N., Ibrahim, A., & Saleh, M. (2022). Vision-based American Sign Language classification approach via deep learning. *arXiv*. Retrieved from https://arxiv.org/pdf/2204.04235

[5] Fahey, O. (2022, June 17). Synthetic ASL Alphabet [Data set]. Kaggle. Retrieved from https://www.kaggle.com/datasets/lexset/synthetic-asl-alphabet

[6] Fahey, O. (2022, June 17). Synthetic ASL Numbers [Data set]. Kaggle. Retrieved from https://www.kaggle.com/datasets/lexset/synthetic-asl-numbers

[7] Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT Press. https://www.deeplearningbook.org/

[8] Kumar, R., Pandey, A., & Gupta, A. (2023). Mediapipe and CNNs for real-time ASL gesture recognition. *arXiv*. Retrieved from https://arxiv.org/pdf/2305.05296

[9] Lee, B., & Secora, K. (2022). Fingerspelling and its role in translanguaging. *Languages, 7*(4), 278. https://doi.org/10.3390/languages7040278

[10] Liang, R., & Ouhyoung, M. (1998). A real-time continuous gesture recognition system for sign language. In *Proceedings of the Third International Conference on Automatic Face and Gesture Recognition* (pp. 558-565). https://doi.org/10.1109/AFGR.1998.670869

[11] Li, D., Rodriguez, C., Yu, X., & Li, H. (2020). Word-level deep sign language recognition from video: A new large-scale dataset and methods comparison. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision* (pp. 1459-1469). https://doi.org/10.1109/WACV45572.2020.9072872

[12] Lugaresi, C., Tang, J., Nash, H., McClanahan, C., Uboweja, E., Hays, M., Zhang, F., Chang, C.-L., Yong, M., Lee, J., Chang, W.-T., Hua, W., Georg, M., & Grundmann, M. (2019). MediaPipe: A framework for perceiving and processing reality. In *Proceedings of the Third Workshop on Computer Vision for AR/VR at IEEE Computer Vision and Pattern Recognition (CVPR)*. Retrieved from https://research.google/pubs/mediapipe-a-framework-for-perceiving-and-processing-reality/

[13] Moulin, F. (2024, June). python-asl-detection. GitHub. Retrieved from https://github.com/ffm5113/python-asl-detection

[14] National Association of the Deaf. (2024). Learning American Sign Language. Retrieved from https://www.nad.org/resources/american-sign-language/learning-american-sign-language/

[15] National Deaf Center on Postsecondary Outcomes. (2024). How many deaf people live in the United States? Retrieved from https://nationaldeafcenter.org/faq/how-many-deaf-people-live-in-the-united-states/

[16] National Geographic Society. (2024). Sign language. Retrieved from https://education.nationalgeographic.org/resource/sign-language/

[17] Pew Research Center. (2024). Mobile fact sheet. Retrieved from https://www.pewresearch.org/internet/fact-sheet/mobile/

[18] Pocket Sign. (n.d.). Sign language alphabet – ASL fingerspelling. Retrieved from https://www.pocketsign.org/alphabet

[19] Simonyan, K., & Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. *International Journal of Computer Vision, 120*(3), 142-158. https://doi.org/10.1007/s11263-015-0816-y

[20] Starner, T., Weaver, J., & Pentland, A. (1997). Real-time American Sign Language recognition using desk and wearable computer-based video. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 20*(12), 1371-1375. https://doi.org/10.1109/34.650113

[21] Sundar, B., & Bagyammal, T. (2022). American Sign Language recognition for alphabets using MediaPipe and LSTM. *Procedia Computer Science, 215*, 642–651. https://doi.org/10.1016/j.procs.2022.12.066

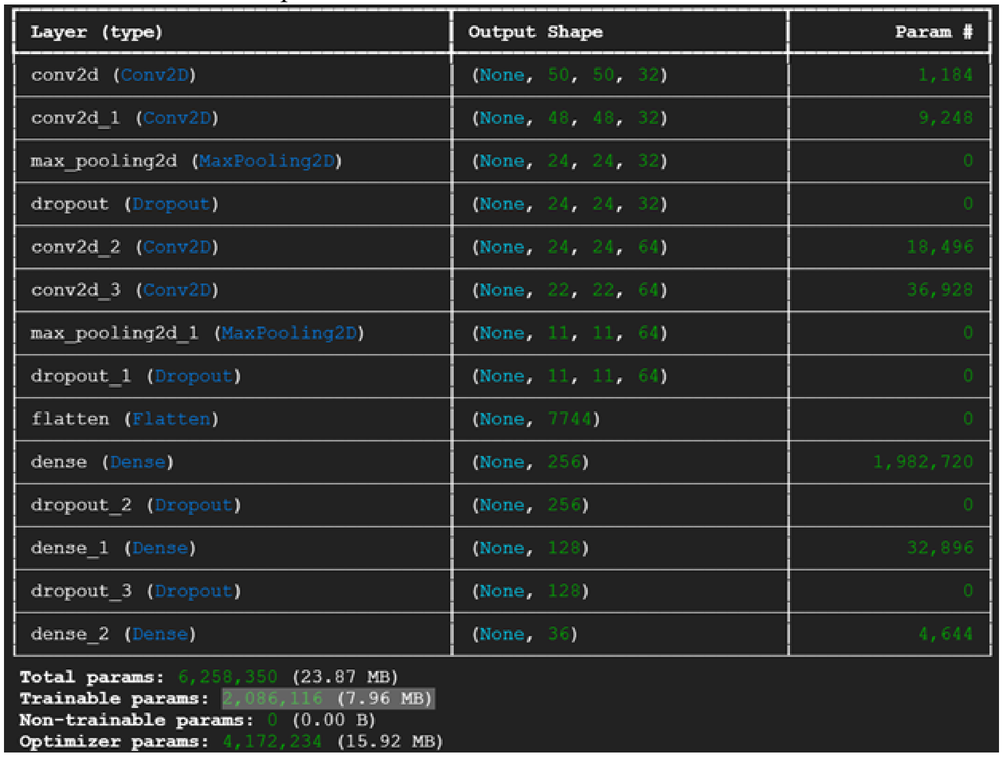
[22] Tao, Z., Yang, Z., Chen, B., Bao, W., & Cheng, H. (2022). Protein sequence classification with LetNet-5 and VGG16. In D. S. Huang, K. H. Jo, J. Jing, P. Premaratne, V. Bevilacqua, & A. Hussain (Eds.), *Intelligent Computing Theories and Applications* (Vol. 13394, pp. 621-633). Springer. https://doi.org/10.1007/978-3-031-13829-4_60

[23] Yale University. (2024). American Sign Language FAQ. Retrieved from https://ling.yale.edu/academics/american-sign-language-asl/faq

[24] Zhou, Z., Chen, K., Li, X., Zhang, S., Wu, Y., Zhou, Y., & Chen, J. (2020). Sign-to-speech translation using machine-learning-assisted stretchable sensor arrays. *Nature Electronics, 3*, 571–578. https://doi.org/10.1038/s41928-020-0428-6

MISC

let-num-v9.2-tf2.16.1-ep01-val-acc-0.9997-07-15-2024-23-57.keras



```
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━┓
┃ Layer (type)                         ┃ Output Shape                ┃         Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━┩
│ conv2d (Conv2D)                      │ (None, 50, 50, 32)          │           1,184 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ conv2d_1 (Conv2D)                    │ (None, 48, 48, 32)          │           9,248 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ max_pooling2d (MaxPooling2D)         │ (None, 24, 24, 32)          │               0 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ dropout (Dropout)                    │ (None, 24, 24, 32)          │               0 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ conv2d_2 (Conv2D)                    │ (None, 24, 24, 64)          │          18,496 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ conv2d_3 (Conv2D)                    │ (None, 22, 22, 64)          │          36,928 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ max_pooling2d_1 (MaxPooling2D)       │ (None, 11, 11, 64)          │               0 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ dropout_1 (Dropout)                  │ (None, 11, 11, 64)          │               0 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ flatten (Flatten)                    │ (None, 7744)                │               0 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ dense (Dense)                        │ (None, 256)                 │       1,982,720 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ dropout_2 (Dropout)                  │ (None, 256)                 │               0 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ dense_1 (Dense)                      │ (None, 128)                 │          32,896 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ dropout_3 (Dropout)                  │ (None, 128)                 │               0 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ dense_2 (Dense)                      │ (None, 36)                  │           4,644 │
└──────────────────────────────────────┴─────────────────────────────┴─────────────────┘
 Total params: 6,258,350 (23.87 MB)
 Trainable params: 2,086,116 (7.96 MB)
 Non-trainable params: 0 (0.00 B)
 Optimizer params: 4,172,234 (15.92 MB)
```

Epoch	Training Time (s)
1	118
2	115
3	115
4	115
5	114
6	114
7	114
8	114
9	115
10	115
11	114
12	114
13	114
14	115
15	115
16	114

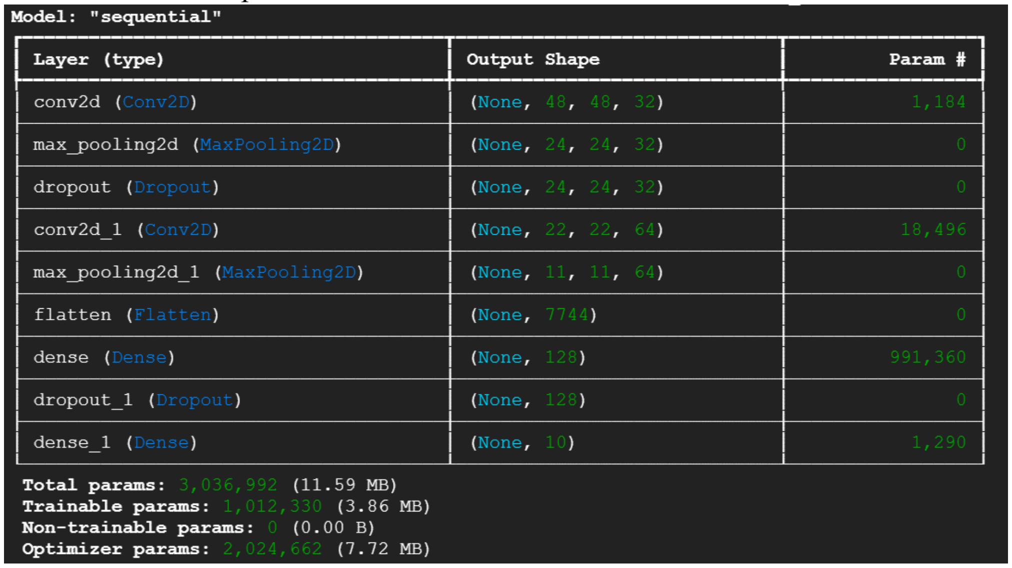
17 114

18 114

19 114

20 114

num-v2.1-tf2.16.1-ep09-val-acc-1.0000-07-17-2024-01-13.kerasc



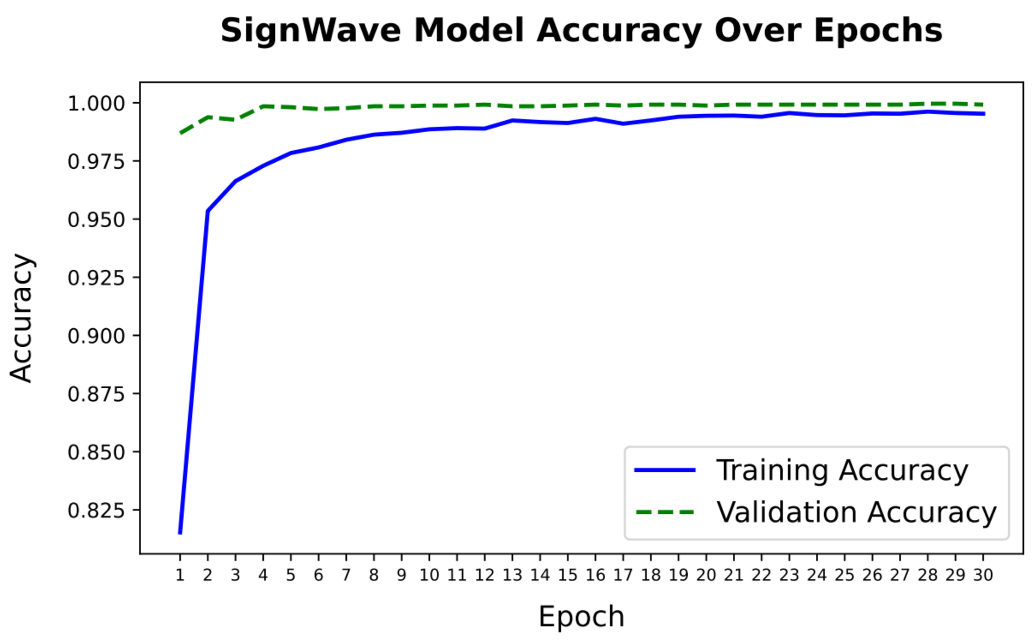
```
Model: "sequential"
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━┓
┃ Layer (type)                         ┃ Output Shape                ┃         Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━┩
│ conv2d (Conv2D)                      │ (None, 48, 48, 32)          │           1,184 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ max_pooling2d (MaxPooling2D)         │ (None, 24, 24, 32)          │               0 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ dropout (Dropout)                    │ (None, 24, 24, 32)          │               0 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ conv2d_1 (Conv2D)                    │ (None, 22, 22, 64)          │          18,496 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ max_pooling2d_1 (MaxPooling2D)       │ (None, 11, 11, 64)          │               0 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ flatten (Flatten)                    │ (None, 7744)                │               0 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ dense (Dense)                        │ (None, 128)                 │         991,360 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ dropout_1 (Dropout)                  │ (None, 128)                 │               0 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ dense_1 (Dense)                      │ (None, 10)                  │           1,290 │
└──────────────────────────────────────┴─────────────────────────────┴─────────────────┘
 Total params: 3,036,992 (11.59 MB)
 Trainable params: 1,012,330 (3.86 MB)
 Non-trainable params: 0 (0.00 B)
 Optimizer params: 2,024,662 (7.72 MB)
```

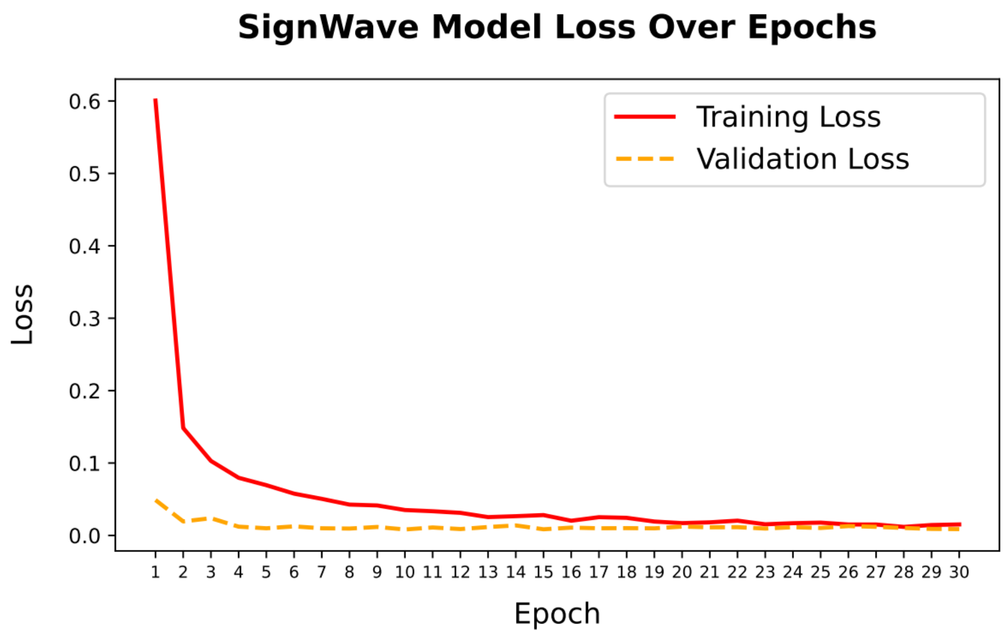
SignWave ASL Letter Classification Computer Vision Model

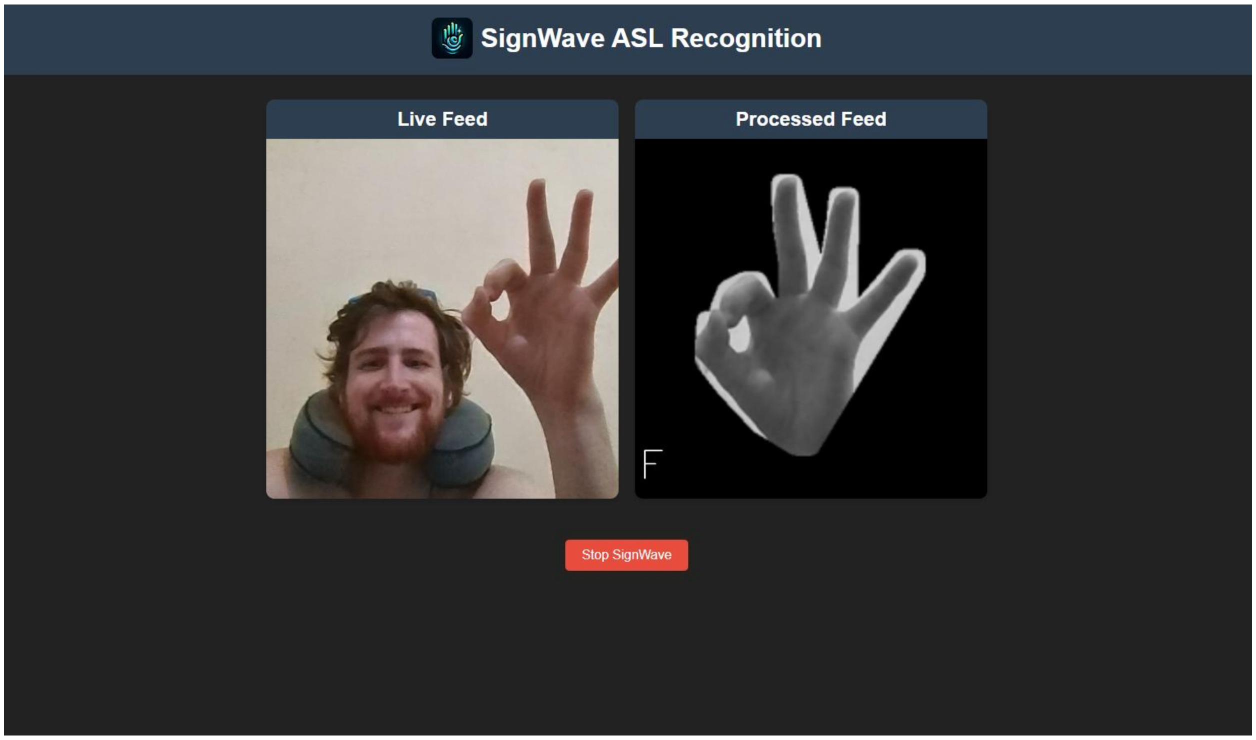
Layer Type	**Output Shape**	**# Parameters Calculation**
Input	(175, 175, 4)	0 (No parameters)
Conv2D	(173, 173, 32)	(3 * 3 * 4 + 1) * 32 = 1184
MaxPooling2D	(86, 86, 32)	0 (No parameters)
Dropout	(86, 86, 32)	0 (No parameters)
Conv2D	(84, 84, 64)	(3 * 3 * 32 + 1) * 64 = 18496
MaxPooling2D	(42, 42, 64)	0 (No parameters)
Flatten	(113568,)	0 (No parameters)
Dense	(128,)	(113568 + 1) * 128 = 14505984
Dropout	(128,)	0 (No parameters)

Dense	(26,)	(128 + 1) * 26 = 3354
Total		**14,527,218 parameters**

Layer Type	**Output Shape**	**Parameters Calculation**	**Description**
Input	(None, 50, 50, 4)	No parameters	Input layer with shape (50, 50, 4).
Conv2D	(None, 48, 48, 32)	(3 * 3 * 4 + 1) * 32 = 1,184	Conv2D layer with 32 filters, kernel size 3x3, and 4 input channels.
MaxPooling2D	(None, 24, 24, 32)	No parameters	MaxPooling2D layer splits each spatial dimension in half.
Dropout	(None, 24, 24, 32)	No parameters	Dropout layer for regularization to prevent overfitting.
Conv2D	(None, 22, 22, 64)	(3 * 3 * 32 + 1) * 64 = 18,496	Conv2D layer with 64 filters, kernel size 3x3, and input channels 32.
MaxPooling2D	(None, 11, 11, 64)	No parameters	MaxPooling2D layer splits each spatial dimension in half.
Flatten	(None, 7744)	No parameters	Flatten layer converts MaxPooling2D output to a single vector of size 7744.
Dense (128 Units)	(None, 128)	(7744 + 1) * 128 = 991,360	Dense layer with 128 units, taking an input size of 7744.
Dropout	(None, 128)	No parameters	Dropout layer for regularization to prevent overfitting.
Dense (26 Units)	(None, 26)	(128 + 1) * 26 = 3,354	Dense layer with 26 units (classes A-Z), taking input size of 128.
Total Parameters	–	1,014,394 parameters	Total number of parameters used to train the model.


SignWave Model Accuracy Over Epochs
Accuracy
Epoch
Training Accuracy
Validation Accuracy
1.000
0.975
0.950
0.925
0.900
0.875
0.850
0.825
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30


SignWave Model Loss Over Epochs
Loss
Epoch
Training Loss
Validation Loss
0.6
0.5
0.4
0.3
0.2
0.1
0.0
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30


SignWave ASL Recognition
Live Feed
Processed Feed
F
Stop SignWave

Vision-Based American Sign Language Classification Approach via Deep Learning (arxiv.org)

Abstract

Hearing-impaired is the disability of partial or total hearing loss that causes a significant problem for communication with other people in society. American Sign Language (ASL) is one of the sign languages that most commonly used language used by Hearing impaired communities to communicate with each other. In this paper, we proposed a simple deep learning model that aims to classify the American Sign Language letters as a step in a path for removing communication barriers that are related to disabilities.

Hearing-impaired is the description of a hearing disorder that is considered as any degree of hearing loss (Demorest and Erdman 1987). Sign language is one of the most common communication ways that is used among hearing-impaired people (Pfau, Steinbach, and Woll 2012).

Sign language includes gestures, body movements, and facial expressions to represent words, tone, and emotions instead of using sounds (Sandler and Lillo Martin 2006). According to the World Federation of the Deaf (WFD), over 72 million people around the earth are deaf. In addition, there are more than 300 different sign languages that exist and are used by different deaf and hard-of-hearing people around the world (United Nations 2021).

Sign languages do not maintain the same grammatical properties as spoken languages. Nevertheless, the sign languages maintain similar linguistic properties as the spoken languages (Battison 1974). The American Sign Language (ASL) is the most commonly used sign language in the United States and several parts of Canada (Hill, Lillo-Martin, and Wood 2018). The ASL is considered to be originated in 1817 at the American School of Deaf (ASD) (Bahan 1996) where the signs have been adopted from the French sign language (Valli and Lucas 2000).

Figure 1 shows the ASL alphabet signs (APSEA 2021).

Fingers spelling is a standard system used in different sign languages to spell names, locations, words, and phrases that do not have a specific sign and also to clarify words when a specific sign was not well provided. Interpreting sign language to a speech is essential to remove communication barriers and provide a higher quality of life for deaf and hard of hearing people worldwide. The ASL letters classification is a complex problem due to the large variety of different representations for the same letter due to the different physical abilities to move fingers to represent the letter and the length of the fingers. There were few attempts to solve that problem, such as (Abdulhussein and Raheem 2020) that targeted the ASL letters classification by applying a deep learning model and edge detection for the hand and fingers. However, this work did not summarize prediction results on a classification problem regarding each alphabet letter. In addition, the dataset contained only 240 images where ten different samples represented each letter. (Ameen and Vadera 2017) proposed a convolution model that attempted to classify the ASL letters. This work focused on using different types of features that were used in (Rioux-Maldague and Giguere 2014) combined by the convolution neural network. The model had higher accuracy than the (Rioux-Maldague and Giguere 2014) approaches. However, the lack of a dataset and the model implementation complexity was the major issue of this model.

This paper proposed an ASL classification approach via a deep convolution neural network. The proposed model can classify ASL hand postures images to their corresponding letters. In this paper, we addressed the major issues of the ALS sign classification problem. The contribution of this paper is as follows: we employed data augmentation (Antoniou, Storkey, and Edwards 2017) via multiple augmentation approaches to solving the data limitations in the ASL letters classification problem. In addition, we empirically designed a simple convolution neural network-based model that achieved a classification accuracy and can be trained rapidly compared to the other existing models. Moreover, this model does not require any data preprocessing other than image size adjustment. Furthermore, the model performs the classification without additional segmentation algorithms or transfer learning techniques. Finally, this model is employed within a system that interprets the American Sign Language letters to caption words that help readers understand the ASL speakers and remove the communication barriers.

Proposed Model

The proposed model is mainly based on the convolution neural network architecture (CNN) as a robust algorithm for images classification task (Lawrence et al. 1997; Howard 2013; Yadav and Jadhav 2019). The proposed model design has been selected based on empirical evaluations of different convolution layers models. The choice has been set to the lightest model design while maintaining comparable accuracy. The proposed model design consists of 13-layered architecture is shown in Figure 2. For all the convolution layers, kernel sizes were set to 3×3. The number of kernels (filters) in the convolution layers was set to 32, 64, 128, and 256. We used the glorot uniform (Hanin and Rolnick 2018; Glorot and Bengio 2010) to initialize the kernel weights and the biases were initialized by zeros. The dropout was set to 20%. We used the Adam optimization function (Kingma and Ba 2014) with initial learning rate $\alpha = 0.01$, and $\beta1 = 0.9$, and $\beta2 = 0.999$. The maximum pooling pool was set to 2×2, the padding was set as valid, and the strides were set to 1×1. The Flatten layer is used to adjust the input data size before the fully connected dense layer. The dense layer units were set to 64. The weights of the dense layer were initialized using glorot uniform function, and the ReLU function was used as the activation function as it has minimal cost compared to the other non-linear activation functions (Teh and Hinton 2000; Elsayed, Maida, and Bayoumi 2018). Finally, the softmax layer was used to classify the 26 letters of the ASL and the three gesture signs: space, delete, and nothing. The proposed model summary is shown in Table 1.

Data Preparation

Data augmentations are techniques that aim to increase the existing data by performing some different modifications on the copies of the original data. Data augmentation has been used in several data analysis and machine learning tasks where there was a lack of training data availability to train the model. In addition, data augmentation acts as a model regularizer that helps reduce and prevent the overfitting problem during the model training. Our model applied four types of data augmentation: gaussian noise (Lopes et al. 2019), image rotation by 90 degrees, image rotation by 30 degrees, and image rotation by -60 degrees (Shorten and Khoshgoftaar 2019). Each augmentation type was applied to a randomly selected quarter of the dataset, maintaining the uniqueness of each image selection for the augmentation. Applying augmentation increased the dataset size. We used the ASL Alphabet dataset, which is available on Kaggle (Sai 2021). The augmentation process is shown in Figure 3. The original dataset size was 87,000 images, and the dataset size was increased to 108,627 images after employing the data augmentation. Then, we performed the data normalization and image cropping to 50×50 as a data preprocessing stage before the data splitting. After augmentation and data preprocessing, we split the dataset into 60% for training, 20% for validation, and 20% for testing. The images in this dataset have different pixels intensity.

Experiments and Results Our experiments were performed on a Window 10 OS, Intel(R) Core(TM) i-9 CPU @ 3.00 GHz processor, and NVIDIA GeForce RTX 2080 Ti. We used Tensorflow 2.4.0, Python 3.8, and NumPy 1.19.5. The model was trained for 100 epochs. The batch size was set to 128. The RMSProp has been used as the model optimization function (Hinton, Srivastava, and Swersky 2012). The loss function was set to the categorical crossentropy function (Ketkar 2017). The training versus validation accuracy is shown in Figure 4. The loss of training versus validation is shown in Figure 5. The empirical results of our proposed model are shown in Table 2. Figure 6 shows the confusion matrix of the proposed model, where the numbers 0 to 29 indicate the alphabet letter starting from the letter A to letter Z, in addition to the nothing, delete, and space gesture signs. Table 3 shows the comparison results between our proposed model with other research works that address the sign language gesture classification problem using different gesture-based datasets.

Conclusion and Future Work

The communication gap between the hearing-impaired and hearing people has been one of the significant issues in all societies for decades. The proposed model aims to reduce the misunderstanding gap between hearing-impaired and hearing people by understanding the American Sign Language (ASL) letter via classification. The proposed model achieved significantly high accuracy for the correct classification of the ASL letters. As future work, this model is a part of a project to translate the classified letters into transcript words that can also be converted to voice. That could help eliminate the communication gap between hearing-impaired and hearing people and provide a better quality of life for deaf and hard of hearing people, which can significantly improve social communication and understanding.

Table 2: The empirical result of our proposed model. Comparison Value Train Accuracy 99.949% Test Accuracy 99.889% Train Time 55.77(min) Test Time 0.2363(sec) Precision 0.9849 Recall 0.9924 F1-score 0.9925 Total Parameters 2,029,470 Trainable Parameters 2,029,150 Non-trainable Parameters 320

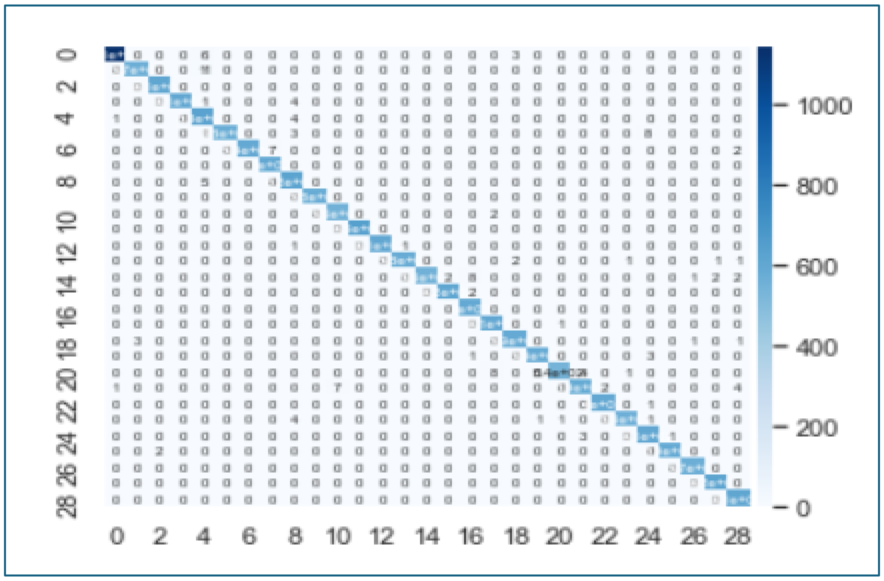


Figure 6: The confusion matrix of the proposed model testing results.

Table 3: A comparison between different gestures classification models from the method, dataset, number of signs, and accuracy

Author	**Method**	**Dataset**	**#Signs**	**Accuracy(%)**
(Elpeltagy et al. 2018)	(HOG–PCA) + (COV3DJ) + (CCA)	ChaLearn	20	83.12%
(Ansari and Harit 2016)	ANNs, 100 − neurons	Indian-reduced	20	37.27%
(Ansari and Harit 2016)	ANNs, 400 − neurons	Indian-reduced	2	90.979%
(Elpeltagy et al. 2018)	(HOG–PCA) + (CCA)	Indian	140	60.40%
(Ansari and Harit 2016)	SIFT	Indian	140	49.07%
(Quinn and Olszewska 2019)	HOG − SVM − RBF	British	26	99.00%
(Quinn and Olszewska 2019)	HOG − SVM − Linear	British	26	98.89%
(Nagarajan and Subashini 2013)	EOH + SVM	British	26	93.75%
(Barkoky and Charkari 2011)	ANN − Backpropagation	Digits	10	96.62%
(Barbhuiya, Karsh, and Jain 2021)	VGG16 + SVM	ASL	36	99.76%
(Barbhuiya, Karsh, and Jain 2021)	AlexNet + SVM	ASL	36	99.82%
Our	Simple CNN	ASL	29	**99.94**%

PAPER

Figure NUMBERHERE: Synthetic ASL Alphabet Dataset Collage (Fahey, 2022)