Limitations of ASL Letter Classification With and Without Segmentation Preprocessing

Mahyar Tajeri Wisdom Ng Fairuj Tabassum Henry Arboladora

Toronto Metropolitan University

***Abstract*—American sign language (ASL) is a visual gesture based language used by deaf and hard of hearing communities in the United States and Canada. classification of ASL letters using computer vision techniques provides an autonomous interface to non-signers and beyond. In this paper, we explore computer vision techniques in classifying static images of the ASL alphabet (A-Z, space, delete, nothing) first with a straightforward approach of transfer learning from the You Only Look Once (YOLO) v11 model. Then transfer learning from RestNet18 is coupled with a pre-trained hand-segmentation model in preprocessing to remove background noise as a factor in training. Based on our experimental results, both methods have significant drawbacks. A major limitation to the first approach is the irrelevant information in the background of both training and testing data. Conversely, the second approach limits the scope of the model to segmented images, is much slower in prediction, and is potentially bottlenecked by the segmenter model.**

***Index Terms*—American Sign Language, You Only Look Once, Hand Segmentation**

I. INTRODUCTION

Communication is a fundamental human need, though it presents a barrier for individuals that are hearing impaired. A 2005 study from Gallaudet University, done by Mitchell et al. estimated that the number of ASL users to be 250,000 to 500,000 adults in America [1]. Relying on signed communication such as ASL to communicate with non-signers is limited as most members of the public do not know sign language. This challenge highlights the impact emerging technologies could have in bridging the gap between ASL users and the hearing population. Given recent advances in computer vision and machine learning, automatic, real-time translation of ASL into English has become a feasible and impactful goal. ASL letters were chosen for this task as words and more complex gestures would require many frames per instance, and thus require more data and computer power than available to our team.

The work conducted here contrasts two main approaches to ASL letter classification, and analyses their strengths and weaknesses. The first approach takes the training data, makes basic augmentations, then employs transfer learning from the YOLOv11 model. This approach can be characterized as both simple and quick in making predictions, due to the nature of the model. YOLO, also known as “You Only Look Once”, is an object detection algorithm that aims to outperform more conventional models by only performing a single pass in the neural network for both object classification and object detection. The single pass in YOLO works by dividing the input image into a grid and predicts the bounding boxes and class probabilities for each cell simultaneously; this approach removes the need for iterative region proposals used in more conventional models like Fast R-CNN which significantly reduces computation time. Conversely, Fast R-CNN, utilizes a more conventional two-stage detection process. Firstly, it generate region proposals, which identifies regions of interest in the image (areas where an object might be), this is normally done by using an external method like Selective Search, Secondly, the image is then passed through a CNN to generate a feature map which in combination with the results of the first step results in feature extraction and classification. The combination of these two steps creates a lot of computational overhead for Fast R-CNN which slows down the model.. Additionally, YOLO utilizes transformers in combination with convolutional layers, this combination allows transformers to improve global feature extraction and convolutional layers to focus on spatial details [2]. As a result, YOLO Fast R-CNN with much faster inference times. Later YOLO algorithms also beat Fast R-CNN in

terms of accuracy. Lastly, YOLO models are also able to discern a lot more detail from less data [3].

High accuracy and simplicity are strengths of this method, but susceptibility to background noise is a significant flaw. As a potential fix for this problem, a pre-trained hand-segmenter model was used to preprocess all training data so that only the hand in the image was present, the rest of the image blacked out. On the other end, the segmenter would need to be used again on input data before a prediction. The objective in this case is to only feed the important information into the model, while simultaneously protecting the input of the model from noisy inputs. While on paper solving the noise problem, major challenges are raised: 1) The model is restricted to pre-segmented images as input. This severely limits the scope of the model unless the data is already pre-segmented, which is unlikely in a real-world scenario. If the model is not pre-segmented, the prediction speed is slow as multiple pre-processing steps are required. 2) The hand-segmenter itself struggles with some images, especially those with poor lighting and/or of a darker complexion. This leads to the introduction of junk data within the training. Some basic techniques can be used to limit this, but not entirely. The same can be said for input images for prediction since they must also be run through the segmenter model first.

II. RELATED WORK & LIMITATIONS

The area of Sign Language Recognition (SLR) has seen significant advancement in recent years. A comprehensive review of SLR done by Madhiaransan and Roy, at the Indian Institute of Technology Roorkee, highlights some of the limitations of SLR through a review of the SLR related work published within the last two decades [4]. The key limitations discussed are as follows:

1. Dataset Challenges: there exists a scarcity of datasets for the development of SLR systems. The review emphasizes the lack of large-scale and diverse datasets, specifically for continuous and non-manual SLR.

2. Illumination Variations and Dynamic Backgrounds: similar to most computer vision applications, environmental factors, such as illumination variations and dynamic backgrounds present significant challenges. The review highlights the difficulties RGB-trained SLR systems experience in lighting inconsistencies and non-uniform background. However, the study suggests converting RGB data to alternative colour spaces, such as HSV (Hue Saturation Value) or Ycbcr (Luminance Chrominance) and background subtraction methods to mitigate these challenges.

3. High Computation Time and Resource Requirements: computational overhead can present a challenge, especially when it comes to real-time SLR. High computational resources, such as GPUs, may be required to process real-time video, which creates resource-constraints and limits accessibility for all users.

4. Scaling and Image Orientation Problems: the system's output may vary depending on the distance and perspective of the input image. For example, gestures recorded with the hand tilted or rotated can cause misclassification by the model if it has not been trained on that specific orientation. Additionally, images captured at varying distances from the camera can lead to the model failing to accurately recognize the gesture if the input scale deviates from the training data.

III. METHOD

As the aim of the project is to increase accessibility, for Hardware tools, a basic external and built-in camera is used to test the model. In regards to Software tools, the project is implemented using Python as the primary programming language due to its extensive CV and ML libraries.

For both models, the ASL letters data is taken from popular online datasets.

A. Straightforward approach with YOLO 11

1) Data: Three datasets are used [5, 6, 7]. Together they comprise around 83 K images of closeups of hands forming ASL letters. Two of the data sets are not in YOLO format and are thus converted by creating the matching label file with the correct classification. YOLO also takes in a bounding box, but the Kaggle datasets did not provide these. As a workaround, the bounding box was assigned to encompass the entire image since most of the data was a cropped image of just the hand anyways.

2) Preprocessing: To give more generalizability to data, a series of out-of-the-box augmentations were applied.

Figure 1. Displayed in the top left is an instance of data before any preprocessing and the subsequent images are various different augmentations.

a) Horizontal Flip: Randomly flipped images horizontally, to be able to classify mirrored signs as well.

b) Rotation: Randomly rotated images up to 45 degrees, so the model is more robust to misaligned input.

c) RGB Shift: Shifted each colour channel by a random amount to account for variable lighting and the different capture of various cameras.

d) Gaussian Noise: Added Gaussian noise with a probability of 20%. This is to make the model more resilient to noisy inputs by forcing it to learn from more generalizable features rather than tiny details.

e) Random Brightness and Contrast: Adjusted the brightness and contrast randomly within a small limit, which simulates different lighting conditions.

f) Blur: Applied either motion, median, or classic blurring randomly to more closely represent real input data that might include camera-shake, or low-quality images.

This pipeline is designed to emulate a variety of realistic inputs, and helps the model focus on more invariant features. With these augmentations, the current split of data is 329630 training, 40975 validation, and 40620 testing instances, for a total of 411215 images.

3) Model Implementation: YOLOv11 is pulled pre-trained and fine tuned on the data described earlier for 34 epochs. The image size is set to 640 x 640 pixels. Other hyperparameters were not customized.

B. Hand-segmentation preprocessing approach

1) Data: The data is taken from [8] which is a Kaggle dataset consisting of about 220 K images (though later cut down by about 40% due to segmenting issues) of ASL letters. Although a distinct dataset, we believe that there is much overlap between this one and [5].

2) Hand-segmentation and other Preprocessing: A pre-trained hand segmentation model [9] is employed to keep only the desired part of each instance

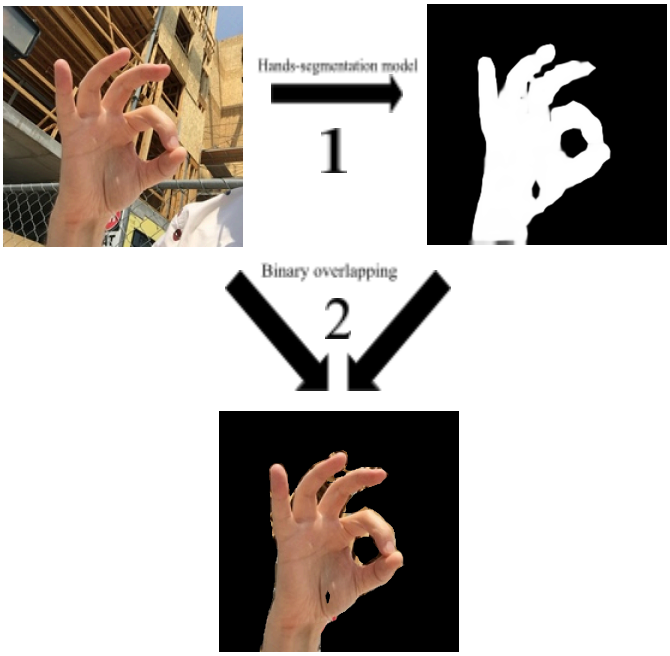

Figure 2. First the data instance is segmented by the pre-trained model and a mask is obtained. Then this mask is binary overlapped with the original to only keep the part of the data desired (the hand).

The image files are given as input, and the model outputs a binary mask where the white pixels indicate where the hand was detected. This mask is then taken and overlapped with the original image such that where the mask is white the pixels are kept, and where the mask is black the pixels are set to black.

To supplement, similar preprocessing was used as the first model, though less extensive. The training-validation split was made 80-20.

3) Model implementation: This model was implemented through transfer learning from ResNet18. ResNet18 is a strong candidate for transferring learning for our model due to several reasons. Firstly, it has a very efficient and simple architecture, which provides the necessary balance between complexity and performance, which with our limited resources for training, was important. Additionally, ResNet18 is also very robust and versatile due to its ability to extract useful features from a wide variety of different situations. It achieves this by utilizing residual connections, to bypass certain layers which can help its ability to learn more effectively. Furthermore, it is also a very modular architecture, which allows us to easily adapt ResNet18 to our use case. Lastly, ResNet-18 is pre-trained on large datasets like ImageNet, which allows us to leverage these learned features for cases with smaller datasets, like in our case [10]. YOLO was not used here because the likely slow prediction speed due to the segmentation step was thought to make the classification speed advantage of YOLO irrelevant.

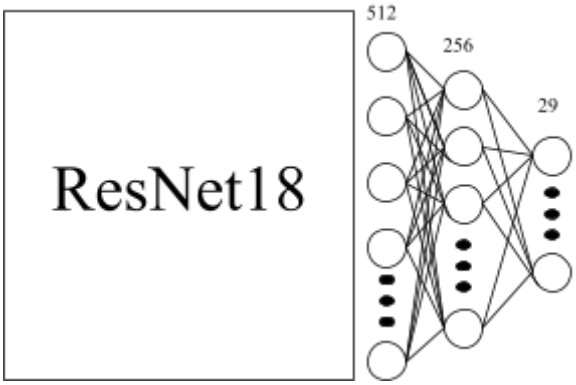

Figure 3. The architecture of the first iteration of transfer learning from ResNet. The 512 feature output of the pre-trained model is fully connected to a 256 and 29 layer sequentially.

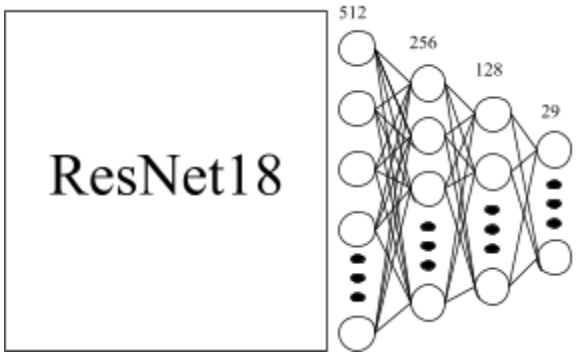

Figure 4. The architecture of the second iteration of transfer learning from ResNet. The 512 feature output of the pre-trained model is fully connected to a 256, 128, and 29 layer sequentially.

A pre-trained ResNet18 model was pulled from the torchvision package in Python. Then it was modified by adding fully connected layers of neurons after the 512 feature output inherent to the final layer of the pre-trained model. There were two iterations worth noting: One with a 256 neuron layer going to a 29 neuron output layer, and another with a 256 neuron going to a 128 neuron layer and then to the 29 neuron output layer.

In terms of hyperparameters, for both iterations A cross entropy loss was selected as the loss function, which combines a log soft max and negative log-likelihood loss. The learning rate was set to 0.001 and a momentum of 0.9 was used to help the optimizer converge faster and escape local minima. Furthermore, a learning rate scheduler was used to prevent overshooting with a $\gamma = 0.1$ every 7 epochs (gamma 0.1 means the learning rate is decreased by a factor of 0.1 every 7 epochs). Both iterations were trained for 22 epochs.

IV. EXPERIMENT

Both methodologies, straightforward transfer learning from YOLOv11 and transfer learning from ResNet18 with hand-segmentation preprocessing display impressive but also flawed results.

A. Straightforward approach with YOLO 11

During training, the YOLO model recorded box loss, class loss, and Distribution Focal Loss (DFL). Notably, the box loss displayed a value of "inf" during several epochs, likely due to an issue with the dataset.

Specifically, this may stem from annotating the bounding box as the entire image rather than accurately focusing on the hand.

For the class loss and DFL, the training curves remained relatively flat across all epochs, indicating consistent performance with minimal improvement over time. This behavior might be attributed to two factors: the large volume of data used for fine-tuning and the simplicity of the dataset, characterized by uniform and unchallenging backgrounds.

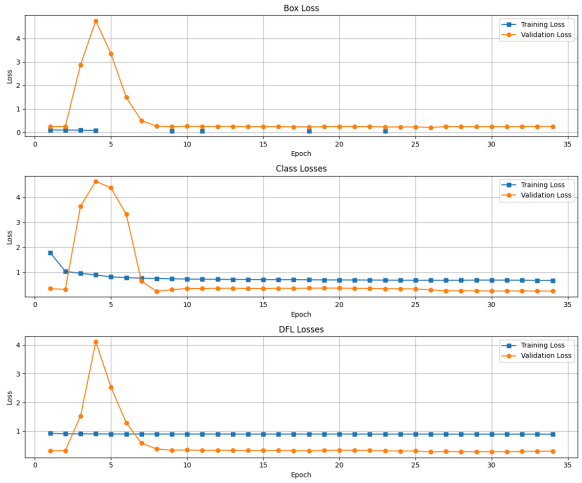

Figure 5. The three loss curves for the first model. In order from the top: box loss, class loss, Distribution Focal Loss (DFL). (All vs. epochs on the x-axis)

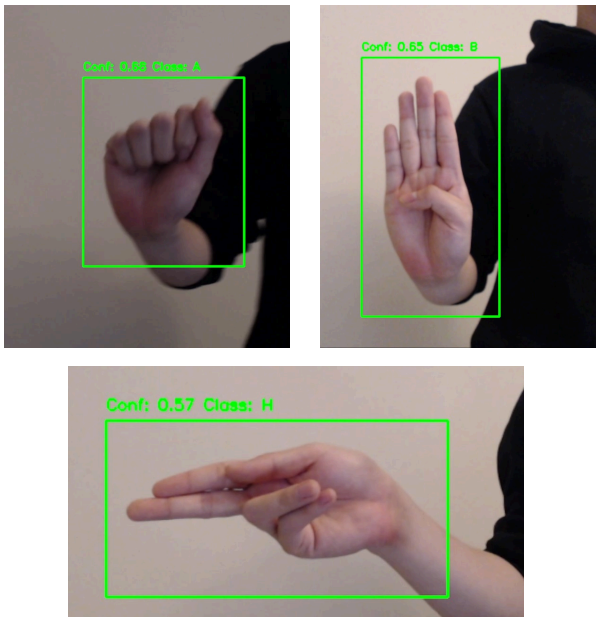

Figure 6. The model correctly identifies three signs - “A”, “B”, “H”.

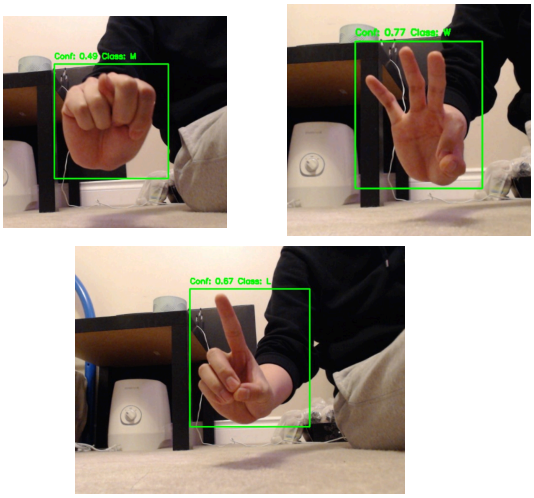

Figure 7. The model incorrectly identifies “D”, “F”, “N”.

The model was able to accurately identify hand signals against a plain background, as illustrated in Figure 6. In contrast, complex backgrounds pose a challenge and lead to incorrect predictions, as evident in Figure 7.

B. Hand-segmentation preprocessing approach

This more extensive approach shows potential in its ability to pick up on useful features within the data. Both iterations for this approach (512 to 256 to 29 and 512 to 256 to 128 to 29) demonstrate strong loss curves indicating good model fitting. Both curves flatten around 8 epochs.

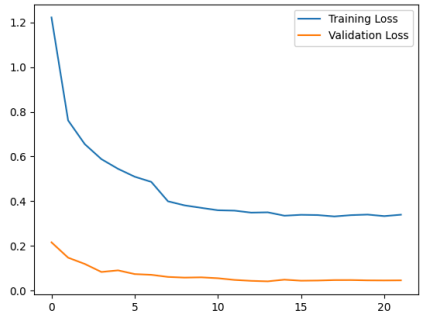

Figure 8. The loss vs. epoch curve of the first iteration of using hand-segmentation preprocessing (512 to 256 to 29). The Training loss is in blue and the validation loss is in orange.

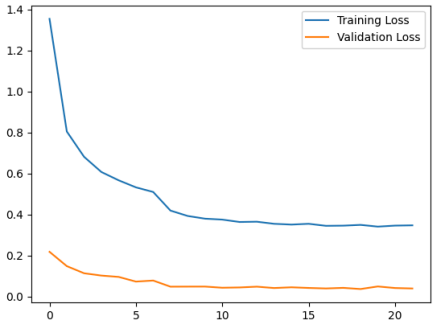

Figure 9. The loss vs. epoch curve of the second iteration of using hand-segmentation preprocessing (512 to 256 to 128 to 29). The Training loss is in blue and the validation loss is in orange.

Both iterations finished with a 99%+ validation accuracy as well a 90%+ training accuracy. It should be noted that lower validation loss is due to the use of dropout which artificially handicaps the model during training to prevent overfitting [11]

Although such accuracy sounds good on paper, when tested on unfamiliar data [7] (unfamiliar to this model, but was used in the YOLO model for training. Not part of the overlap mentioned earlier), the accuracy for both iterations was 42.4% and 45.5% respectively. This isn't abysmal considering there are 29 classes, but a far cry from the 99%+ boasted by the validation accuracy. We suspect the main cause of this dropoff to be from incorrectly segmented training and testing data. The segmenter used [9] often will be unable to determine where the hand in the image exactly matches, and will inadvertently crop out portions of the desired data. To combat this, a filter was issued upon the training, validation, and test data to remove any instance where the image was more than 90% black. Although undoubtedly deleting some good instances, This process would delete a lot of junk data that the segmenter produced. Despite this strategy, however, junk data still persists in the data set since sometimes the data would be ruined yet still under the threshold of 90% black pixels. We estimated that if we further lowered the threshold, too much good data would be deleted and it would be counter-productive.

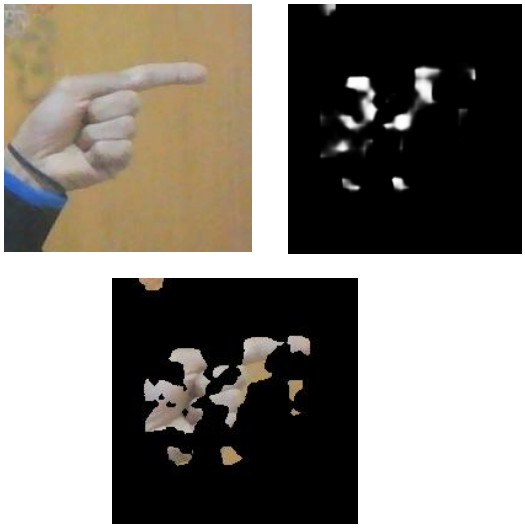

Figure 10. Results of hand segmentation. The top two images are the original and the mask respectively. Note that the mask is not accurate to the image, thus the final segmentation is useless. Furthermore, this falls under the 90% threshold and was not filtered out.

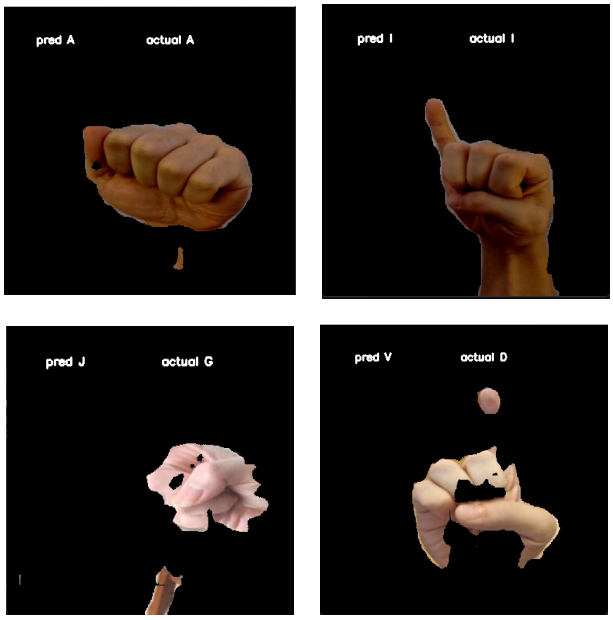

Figure 11. Results of our model on testing data. The top two images are examples of correct classification ("A" and "I"), whereas the bottom two images are incorrect ("G" was classified as "J" and "D" was classified as "V"). Note the poor segmentation in the incorrectly classified instances.

The model performs very well on correctly segmented data, and can accurately choose from amongst the 29 classes (based on our estimations, although we don't have an exact statistic on this because it's hard to input correctly segmented testing data in large quantities). Unfortunately, a large fraction of testing is segmented incorrectly and thus cannot be classified accurately by our model. This suggests that the bottleneck to the hand-segmented method is the hand-segmenter itself. This can be interpreted that the problem of background noise, poor lighting, etc. was not solved by using the segmented approach, but rather deferred to the strength of the segmentation model.

V. CONCLUSION

We constructed models via two different methods to classify images ASL letters. In the first method, we implemented a straightforward transfer learning approach from YOLOv11 to create a fast, real time classifier. Though quick and accurate under good conditions, The largest issue with this approach was that it was not very robust to occlusion and background noise, even when considerable augmentations were applied to training data. For our second approach, we implemented segmentation during preprocessing and predicting to only train and classify based on relevant information to overcome the issue of background inconsistency. Though promising results were obtained, including good accuracy on correctly segmented data, a good loss curve and validation accuracy, the accuracy was mediocre on unseen data. We surmise that until the off-the-shelf segmentation model is replaced with a better one, one cannot truly determine the effectiveness of this approach.

We think that segmenting before predicting is wise since the intricate poses of ASL letters (let alone gestures and words) can become difficult to classify with noisy data, and thus suggest further research within the segment-then-classify approach, when fast translation is not required. If responsiveness is crucial, we suggest further experimenting with the first method discussed and dealing with the noise problem in a manner that is more integrated with the model itself, such as broader training data including different skin tones and more occlusion. In terms of gestures, there is also room to extend the functionality of this model by allowing it to understand ASL phrases by incorporating dynamic gesture recognition. This would require the utilization of modeling techniques such as Recurrent Neural Networks (RNNs) or Long Short-Term Memory networks, in order to process gesture data effectively.

Repository links:

Simple approach: https://github.com/wisng/ASL-CV
Segmented approach: https://github.com/mahyartajeri/signitupmodel

REFERENCES

[1] R. E. Mitchell, T. A. Young, B. Bachleda, and M. A. Karchmer, "How Many People Use ASL in the United States? Why Estimates Need Updating," Sign Language Studies, vol. 6, no. 3, pp. 306–335, 2006. [Online]. Available: https://gallaudet.edu/wp-content/uploads/gcloud/gal-media/Documents/Research-Support-and-International-Affairs/ASL_Users.pdf. [Accessed: Nov. 30, 2024].

[2] "YOLOv11: An Overview of the Key Architectural Enhancements," ar5iv, Nov. 05, 2024. https://ar5iv.org/html/2410.17725 (accessed Nov. 30, 2024).

[3] A. Sharma, V. Kumar, and L. Longchamps, "Comparative performance of YOLOv8, YOLOv9, YOLOv10, YOLOv11 and Faster R-CNN models for detection of multiple weed species," Smart Agricultural Technology, pp. 100648–100648, Nov. 2024, doi: https://doi.org/10.1016/j.atech.2024.100648.

[4] M. Madhiarasan and P. P. Roy, "A Comprehensive Review of Sign Language Recognition: Different Types, Modalities, and Datasets," Apr. 07, 2022, *arXiv*: arXiv:2204.03328. doi: 10.48550/arXiv.2204.03328.

[5] Akash , "ASL Alphabet," *Kaggle*, 2017. [Online]. Available: https://www.kaggle.com/datasets/grassknoted/asl-alphabet

[6] A. Thakur, "ASL Dataset," *Kaggle*, 2018. [Online]. Available: https://www.kaggle.com/datasets/ayuraj/asl-dataset

[7] D. Lee, "American Sign Language Letters," *Roboflow*, 2020. [Online]. Available:

https://public.roboflow.com/object-detection/american-sign-language-letters

[8] D. Sau, "American Sign Language Alphabet Dataset," *Kaggle*, 2020. [Online]. Available: https://www.kaggle.com/datasets/debashishsau/aslamerican-sign-language-aplhabet-dataset

[9] G. Camporese, "hands-segmentation-pytorch," GitHub repository, [Online]. Available: https://github.com/guglielmocamporese/hands-segmentation-pytorch. [Accessed: 30-Nov-2024].

[10] M. Aruna, G. Kaneriya and P. Jain, "Leveraging Pre-trained ResNet-18 with Transfer Learning for Yoga Posture Classification," 2024 Second International Conference on Networks, Multimedia and Information Technology (NMITCON), Bengaluru, India, 2024, pp. 1-5, doi: 10.1109/NMITCON62075.2024.10699235.

[11] J. Shunk, “Neuron-Specific Dropout: A Deterministic Regularization Technique to Prevent Neural Networks from Overfitting & Reduce Dependence on Large Training Samples,” arXiv.org, 2022. https://arxiv.org/abs/2201.06938 (accessed Dec. 01, 2024).