REAL-TIME SIGN LANGUAGE TRANSLATION USING YOLO

Crystal Wen
University of Minnesota
wen00015@umn.edu

Lulin Liu
University of Minnesota
liu02721@umn.edu

Matt He
University of Minnesota
he000295@umn.edu

Kevin Lin
University of Minnesota
lin00747@umn.edu

1 PROBLEM STATEMENT

Our everyday lives are filled with tasks that require talking, from ordering food to setting up doctor's appointments. However, what if you cannot speak and rely on your hands to communicate? Members of the deaf and mute community routinely encounter such communication barriers, often leading to frustrating, isolating, and degrading experiences. Due to limited availability of qualified interpretation services, many deaf individuals face challenges such as underemployment, social isolation, and significant public health disadvantages.

To address this critical issue, our project aims to develop a real-time sign language translation system using the You Only Look Once (YOLO) object detection framework. Specifically, we focus on accurately recognizing static American Sign Language (ASL) alphabet hand signs from images or live video streams. By reliably detecting and classifying all 26 ASL letters under various lighting conditions, diverse backgrounds, and different hand shapes, our system establishes a crucial foundation for comprehensive ASL translation. Ultimately, this innovation aims to enhance communication accessibility, empowering the deaf community in everyday interactions.

2 WHY THIS PROBLEM IS INTERESTING

This problem is particularly interesting because real-time sign language translation can dramatically improve accessibility and inclusion for the deaf and hard-of-hearing community. As online communication increasingly becomes central to education, professional environments, and social interactions, an effective automated translation tool could bridge language gaps in digital spaces such as video conferencing and remote learning platforms. Additionally, this technology can significantly benefit learners of ASL by providing immediate, real-time feedback on their signing accuracy, facilitating a more effective learning experience.

3 RELATED WORK

American Sign Language (ASL) is the primary sign language used by the deaf community in the United States, characterized by its own unique grammar, syntax, and vocabulary according to Liddell (2003). It is distinct from other sign systems, which vary in their linguistic structure and cultural in-

fluences. Garcia et al. (2016) proposed a real-time ASL fingerspelling translator leveraging transfer learning with a pre-trained GoogleNet on multiple ASL datasets, and achieved consistent classification for letters a–e and robust performance for letters a–k. Rahman et al. (2019) Rahman and his colleagues proposed a novel CNN model for ASL recognition using preprocessed images from four publicly available datasets, and achieved a significant 9% improvement in accuracy compared to existing methods. Pu et al. (2025) proposed a kinematic hand pose rectification method combined with a feature-isolated mechanism and input-adaptive inference to enhance skeleton-based sign language recognition. They achieved new state-of-the-art performance on WLASL100 and LSA64, with top-1 accuracies of 86.50% and 99.84%, respectively.

4 PROJECT GOALS

The primary goal of this project is to develop a reliable, real-time translation system that recognizes static American Sign Language (ASL) alphabet hand signs using the YOLO (You Only Look Once) object detection framework. Our focus is on accurately detecting and classifying individual ASL letters from images or video frames across a range of lighting conditions, backgrounds, and hand shapes.

The project specifically aims to:

- **Accurate Alphabet Detection:** Train a YOLO-based object detection model to recognize and classify the 26 ASL alphabet hand signs with high precision and recall.
- **Robustness Across Environments:** Ensure consistent model performance across varied lighting conditions, signing styles, and backgrounds by using a diverse and well-annotated dataset.
- **Live Letter Sequencing:** Extend the model to process sequences of ASL letters from live video or image streams, enabling conversion of letter sequences (e.g., C-A-T) into conherent words (e.g., "CAT").
- **Foundation for Future Work:** Establish a solid baseline for full ASL translation by focusing on highly accurate alphabet detection, which could later support dynamic gesture recognition and sentence-level interpretation.

This comprehensive approach aims to address the technical challenges of gesture recognition and translation by emphasizing practical deliverables and measurable outcomes.

5 YOLO-BASED ASL CLASSIFICATION EXPERIMENTS

5.1 DATASET PREPARATION

The experiments utilized the American Sign Language Alphabet image dataset. we primarily utilized a publicly available YOLO-formatted ASL dataset from Kaggle[1], which contains labeled hand sign images for 26 classes (A–Z). The data were split into training, validation, and test sets (e.g., roughly 80%/10%/10%), ensuring each class was well-represented in each subset. No explicit preprocessing beyond resizing and labeling was required, as the images were already segmented to a single hand on typically simple backgrounds. All images were resized (with padding) to the YOLOv8 input resolution (e.g. 640×640) for training.

5.2 MODEL TRAINING AND AUGMENTATION SETUP

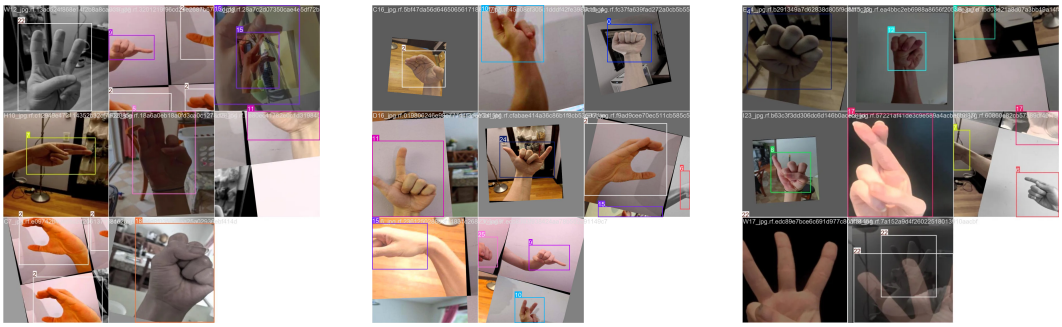
For the augmented model, we applied multiple on-the-fly augmentation strategies during training to enrich data diversity:

- **Color jitter:** Hue, saturation, and value were randomly adjusted to simulate diverse lighting and skin tones, improving generalizability and recall.
- **Geometric transformations:** Transformations such as rotation, translation, scaling, and horizontal flips were applied for varied perspectives to help reduce false negatives.

[1] https://www.kaggle.com/datasets/daskoushik/sign-language-dataset-for-yolov7

- **Mosaic augmentation:** We used mosaic augmentation to merge four images and their annotations into one, introducing multiple hands and varied backgrounds. This enhanced context and scale diversity, boosting recall with a slight precision trade-off.
- **Mixup augmentation:** We applied mixup, blending image pairs via linear interpolation to boost data variability and model robustness.

In Figure 1, we show some result after we perform data augmentation.



(a) Augmentation A (b) Augmentation B (c) Augmentation C

Figure 1: Visualization of different augmented images

In contrast, the non-augmented model was trained on the same images but with all these augmentations turned off. This baseline thus learns only from the original images without additional variability. All other hyperparameters (learning rate schedule, optimizer, etc.) were kept constant between the two runs to isolate the impact of data augmentation.

(a) Augmented Model

Parameter	**Value**
Epochs	100
Batch size	8
Device	GPU
Initial learning rate (lr_0)	0.01
Final learning rate factor (lrf)	0.1
Warmup epochs	3
Hue variation (hsv_h)	0.015
Saturation variation (hsv_s)	0.7
Value variation (hsv_v)	0.4
Rotation (degrees)	15
Translation	0.1
Scale	0.5
Horizontal flip probability ($fliplr$)	0.5
Mosaic probability	0.5
Mixup probability	0.1
Close mosaic (epoch)	10

(b) Non-Augmented Model

Parameter	**Value**
Epochs	100
Batch size	8
Device	GPU
Initial learning rate (lr_0)	0.01
Final learning rate factor (lrf)	0.1
Warmup epochs	3
Data Augmentation	None

Table 1: Comparison of Hyperparameters for Augmented vs. Non-Augmented Models

5.3 Description of Hyperparameter Tables

Tables 1a and 1b list the key training hyperparameters for our two YOLOv8 experiments—baseline (no augmentation) versus augmented.

In the **Augmented Model** (Table 1a), all core settings—epochs, batch size, device, and learning-rate schedule—are identical to the baseline. We then add on-the-fly augmentations to enrich data diversity:

- **Color Jitter:** hue ± 0.015, saturation ± 0.7, value ± 0.4.
- **Geometric Transforms:** random rotation $\pm 15°$, translation $\pm 10\%$, scaling up to $\pm 50\%$, horizontal flips $p = 0.5$.
- **Mosaic Augmentation:** $p = 0.5$ until epoch 10 ("close mosaic").
- **Mixup:** $p = 0.1$.

In the **Non-Augmented Model** (Table 1b), we train for 100 epochs with a batch size of 8 on GPU hardware. The learning rate starts at $lr_0 = 0.01$ and decays by a factor $lrf = 0.1$, with a 3-epoch warmup. No data augmentations are applied, establishing our baseline performance.

By keeping every other hyperparameter fixed and only toggling augmentations, we isolate their impact: as shown in Section 5, this yields higher recall and mAP without sacrificing precision.

5.4 YOLOV8 PERFORMANCE EVALUATION

Table 2 contrasts the *best-validation* epochs for the two YOLOv8 models. Data augmentation boosts every headline number: precision (+0.038), recall (+0.029), and therefore F1 (+0.033). The mean Average Precision rises from 0.943 to 0.971 at an IoU threshold of 0.5, and the more stringent mAP@0.5:0.95 also edges upward. In short, augmentation yields a model that detects more true letters while slightly reducing false alarms.

Table 2: Test-set metrics (best epoch) for YOLOv8 on the ASL alphabet.

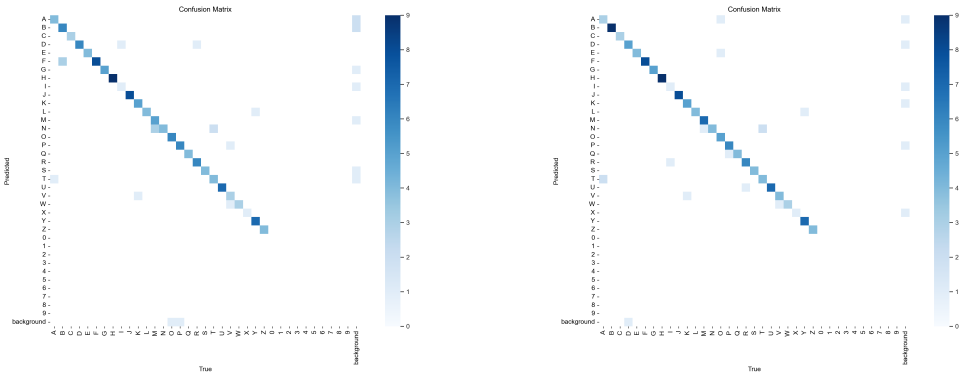
Model	**Precision**	**Recall**	**F1-score**	**mAP@0.5**	**mAP@0.5:0.95**
Non-augmented	0.879	0.879	0.879	0.943	0.783
Data-augmented	**0.917**	**0.908**	**0.912**	**0.971**	**0.786**

Training-time cost. The baseline finished in $\sim$21 ks (5.9 h), whereas the augmented run required $\sim$45 ks (12.6 h). The extra cost stems from (i) the heavier on-the-fly image synthesis pipeline and (ii) training for 86 rather than 31 epochs to reach its optimal checkpoint. In our use-case this increase is acceptable because the augmented model will be frozen for real-time inference on edge hardware where latency, not training throughput, is the bottleneck.

Error-pattern shift. As detailed in Sec. 5 and Fig. 2, augmentation eliminates most confusion within shape-similar pairs—*M/N*, *A/S*, *U/V*, *F/O*—while retaining near-perfect accuracy on distinctive letters such as *L* and *Y*. The remaining hard case is *Q* vs. *P*, whose distinction relies on thumb placement and fingertip curvature; excessive geometric jitter may occasionally blur those cues, explaining the small precision dip for *Q*. Future work could incorporate hand–pose key-point supervision to disambiguate these subtleties without sacrificing robustness.

5.5 CONFUSION MATRIX ANALYSIS

To further analyze class-wise performance, we examined the confusion matrices for both models (Figure 2). The confusion matrix for the baseline (non-augmented) model (Figure 2a) reveals that while many letters are correctly identified (strong diagonal entries), there were several notable confusions between letters with similar hand shapes. For example, the letter M was sometimes misclassified as N, and vice versa. These letters involve very similar fist shapes, which the model struggled with when trained on limited variations. Similarly, the model often confused A vs. S, A vs. T, B vs. F, etc. Without augmentation, the model had trouble distinguishing these, leading to false positives between those classes. Letters with unique shapes like L or Y were almost always classified correctly even by the baseline model – evidenced by near-100% correct predictions for those classes. This indicates that clear, distinct gestures are learned well even without augmentation, whereas ambiguous gestures benefit from additional training variety. The augmented model's confusion matrix (Figure 2b) shows a much cleaner classification result. Some diagonal cells are noticeably stronger and the off-diagonal misclassification counts are greatly reduced. After training with augmented data, the model distinguishes similar hand shapes much better. For instance, the confusion between M and N was greatly reduced with each letter being correctly recognized with high recall. The



(a) Confusion matrix for nonaugmented model

(b) Confusion matrix for augmented model

Figure 2: Visualization of confusion matrices for non-augmented (left) and augmented (right) models.

model learned finer nuances like the thumb placement, as augmentation exposed it to varied hand orientations and lighting that highlight these features. The A vs S and A vs T confusion was also mitigated; in the augmented model's matrix, virtually no S or T gestures are predicted as A or vice versa. The F vs B mix-up was likewise corrected. However, P and Q had an increase in misclassification compared to the baseline model. This may be due to the augmented parameters. For example, when rotated in a certain way, P would look like Q leading to a decrease in accuracy and recall for these classes. Overall, almost every letter shows improved precision and recall in the confusion matrix with augmentation. Some classes that were previously among the hardest, such as M and F, (noted in other studies as prone to false negatives) saw a marked improvement. Meanwhile, letters like L and Y that were already easy to classify retained their perfect (or near-perfect) performance – augmentation did not degrade any of the well-learned classes.

5.6 YOLOV10 PERFORMANCE EVALUATION

To expand on our YOLOv8 experiments, we evaluated several variants of the YOLOv10 architecture (Nano, Small, and Medium models)Wang et al. (2024), both with and without augmentation. These models were trained and tested using the same ASL dataset on MSI GPUs. Table 3 Table 4 and summarizes the key performance metrics across all variants.

Table 3: YOLOv10 Validation Performance (Accuracy Metrics)

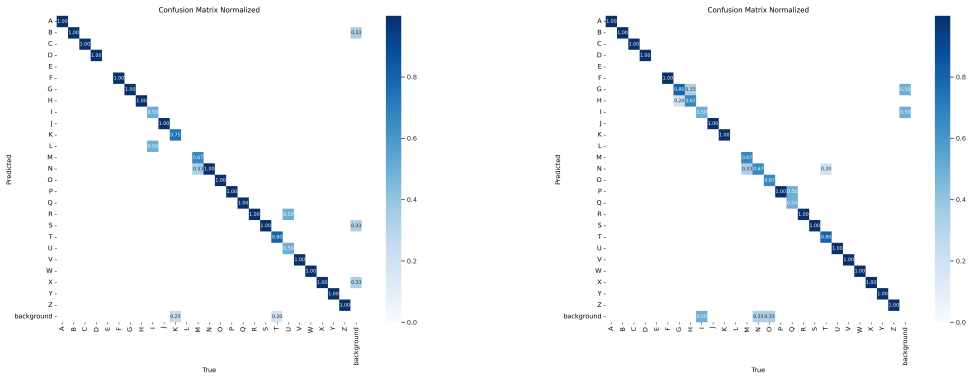
Model Variant	**Augmented**	**Precision**	**Recall**	**mAP@0.5**	**mAP@0.5:0.95**
YOLOv10n (Nano)	No	0.850	0.817	0.916	0.769
YOLOv10s (Small)	No	0.885	0.928	**0.981**	**0.794**
YOLOv10s (Small)	Yes	**0.920**	0.865	0.924	0.792
YOLOv10m (Medium)	No	0.862	0.807	0.934	0.777
YOLOv10m (Medium)	Yes	0.901	0.817	0.927	0.792

Table 4: YOLOv10 Training Setup Summary

Model Variant	**Epochs (Completed)**	**Training Time**
YOLOv10n (Nano)	193	∼5 min
YOLOv10s (Small)	145	∼20 min
YOLOv10s (Small, Aug)	300	∼1 hr
YOLOv10m (Medium)	140	∼30 min
YOLOv10m (Medium, Aug)	300	∼1.3 hr

5.6.1 PERFORMANCE INSIGHTS

As shown in Table 3, the **YOLOv10s (Small)** model with augmentation achieved the highest precision (0.920), indicating its strength in minimizing false positives. However, this model had a slightly lower mAP compared to its non-augmented counterpart, suggesting that the augmentation may have introduced noise or over-regularization of the already clean dataset. Table 4 highlights that augmentation substantially increases training time (from $\sim$20 minutes to $\sim$1 hour for YOLOv10s), although it did offer modest gains in robustness. For example, the **YOLOv10m** model with augmentation achieved higher precision and slightly higher recall compared to the non-augmented version, supporting its utility in larger-capacity models. Overall, the YOLOv10s model—both with and without augmentation—performed similarly to our previously trained YOLOv8n model (Table 2). YOLOv10s achieved slightly higher precision (up to 0.920) and mAP@0.5 (0.981) compared to YOLOv8n (0.917 precision, 0.971 mAP@0.5). These results suggest that the architectural changes in YOLOv10 offer incremental improvements but not a substantial leap in performance. Additionally, the impact of augmentation appears mixed: while it slightly improved precision in some YOLOv10 configurations, it reduced mAP in others. This indicates that augmentation strategies may need further tuning depending on model scale and dataset characteristics.



(a) Confusion matrix for non-augmented Yolov10s model

(b) Confusion matrix for augmented Yolov10s model

Figure 3: Visualization of confusion matrices for non-augmented (left) and augmented (right) models.

Figure 3 compares the class-wise performance of the YOLOv10s model with and without data augmentation. The non-augmented model (Figure 3a) performs well overall, though it shows confusion between visually similar gestures—most notably between **M and N**. The augmented model (Figure 3b) shows a mixed outcome. While some classes, such as **K** and **U**, show improved consistency, others suffer from reduced accuracy. In particular, confusion between **G and H** increases, and predictions for **M**, **N**, and **P** become less reliable. These results suggest that while augmentation can enhance robustness in certain cases, it may also introduce variability that harms performance if not carefully tuned. When the base model already generalizes well, aggressive augmentation may lead to diminished returns or new sources of error.

6 LIVE DETECTION RESULTS

We evaluated our real-time system on both single-letter trials and multi-letter spelling tasks. Each letter is "collected" over a 3 second countdown window (90 frames), and the most confident and frequently predicted letter in that window is chosen as the final output.

6.0.1 SINGLE-LETTER RECOGNITION

Success cases include hand gestures with clear, distinguishable shapes. The augmented model recognized signs like C, T, and J with high confidence. However, it still struggled with signs such as A and R, which were misclassified as M and P.

6.0.2 WORD SPELLING TRIALS

Table 5: Live Spelling Results

Word	Target	Recognized	Outcome
CAT	C–A–T	C–A–T	Success
HELLO	H–E–L–L–O	H–E–L–L–O	Success
WORLD	W–O–R–L–D	W–O–P–L–K	FAIL

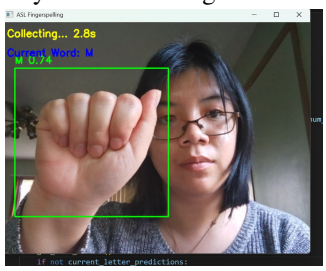
While the system was able to spell certain words correctly, R and D were misrecognized as P and K respectively when spelling "WORLD". This is likely due to the scale and mosaic augmentation during training. The scale parameter might have cropped off or distorted finger edges and the mosaic parameter blended scenes together, making classification difficult for some small hand gestures. Another possibility is that the transition from O to R and L to D may have caused some confusion during the collection period.

6.1 SIMPLE ANALYSIS OF LIVE DETECTION

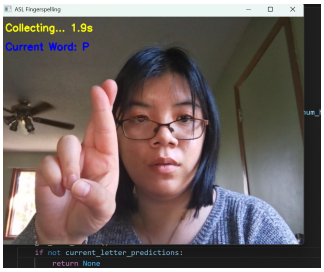
Overall, our real-time system demonstrates strong performance on clearly distinct ASL letters (e.g. C, T, J), but struggles on pairs with very similar hand shapes (M vs. N, P vs. Q). There are three main sources of error:

- **Gesture Ambiguity:** Some letters differ only by slight thumb or finger positions (M/N, P/Q). In the absence of extremely fine spatial detail, the model can confuse these signs (see Figure 4a and 4b).
- **Lighting and Contrast:** Indoor lighting variations sometimes wash out key finger boundaries, causing misclassification (e.g. the "P" in Figure 4b looks more like a "Q" when the thumb contour is under-lit).
- **Fixed Collection Window:** A rigid 3second voting window may cause the system to be unable to capture rapid transitions. This may have lead to the misses in "WORLD" Figure 5a).

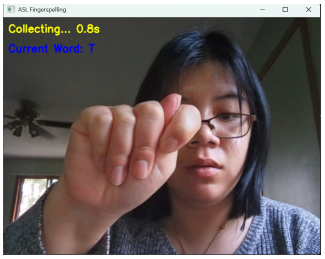
These errors could be fixed with targeted data augmentation for confusing pairs, adaptive lighting normalization, and dynamic buffering. These solutions will be explored in future iterations.



(a) Failure: A (misclassified as M)

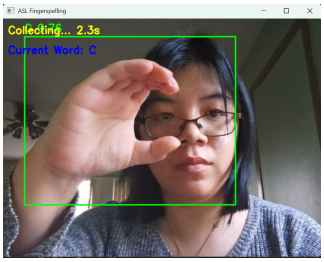


(b) Failure: R (misclassified as P)



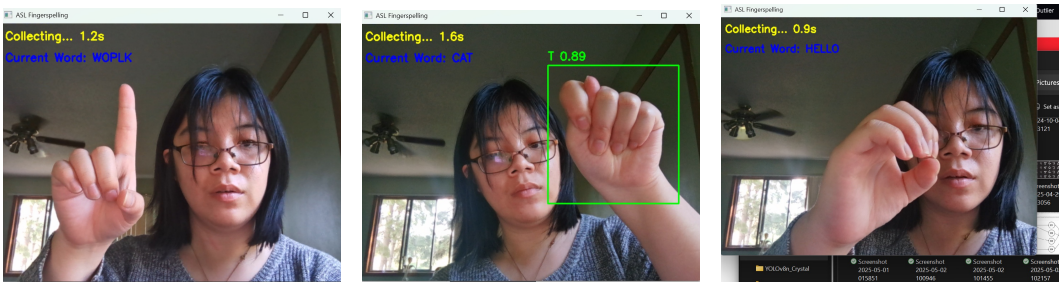
(c) Success: T

(d) Success: J



(e) Success: C

Figure 4: Examples from our live-detection demo. *Top row:* failure cases (3 s collection window → incorrect output). *Bottom row:* successful letter detections (chosen by majority vote).



(a) Failure: "WORLD" → "WOPLK"

(b) Success: "CAT" spelled correctly

(c) Success: "HELLO" spelled correctly

Figure 5: Live-detection word-level examples. *Left:* failure in multi-letter sequencing. *Center/Right:* successful word spellings.

7 Next Steps

As a next step toward improving generalization and robustness, we experimented with image segmentation using the Ayuraj dataset[2]. This dataset features hand signs with black backgrounds, simplifying the segmentation problem and reducing background noise.

We trained YOLOv10s and YOLOv10m models on this dataset without any augmentation. The models achieved near-perfect performance on the validation set:

- **YOLOv10s:** Precision = 0.995, Recall = 0.996, mAP@0.5 = 0.995, mAP@0.5:0.95 = 0.995
- **YOLOv10m:** Precision = 0.999, Recall = 1.000, mAP@0.5 = 0.995, mAP@0.5:0.95 = 0.995

Despite these strong metrics, the models struggled during real-world predictions. This was likely due to the dataset including hand signs for both the alphabet and digits **0–9**. Several number and letter signs share similar visual forms (e.g., "V" vs. "2", "W" vs. "3"), leading to frequent misclassifications. This highlights the need for label refinement or class restriction in training datasets when targeting a specific alphabet subset.

In future iterations, we would aim to explore different hybrid strategies to better improve real-world predictions.

The complete codebase and results are avaiable at our public GitHub repository: **`https://github.com/Wen-KaiXuan/YoloASL`**

References

Brandon Garcia, Sigberto Alarcon Viesca, et al. Real-time american sign language recognition with convolutional neural networks. *Convolutional Neural Networks for Visual Recognition*, 2 (225-232):8, 2016.

Scott K. Liddell. *Grammar, Gesture, and Meaning in American Sign Language*. Cambridge University Press, Cambridge, UK, 2003. ISBN 9780521816205.

Muxin Pu, Mei Kuan Lim, and Chun Yong Chong. Siformer: Feature-isolated transformer for efficient skeleton-based sign language recognition, 2025. URL `https://arxiv.org/abs/2503.20436`.

Md. Moklesur Rahman, Md. Shafiqul Islam, Md. Hafizur Rahman, Roberto Sassi, Massimo W. Rivolta, and Md Aktaruzzaman. A new benchmark on american sign language recognition using convolutional neural network. In *2019 International Conference on Sustainable Technologies for Industry 4.0 (STI)*, pp. 1–6, 2019. doi: 10.1109/STI47673.2019.9067974.

Ao Wang, Hui Chen, Lihao Liu, Kai Chen, Zijia Lin, Jungong Han, and Guiguang Ding. Yolov10: Real-time end-to-end object detection. *arXiv preprint arXiv:2405.14458*, 2024.

[2] `https://www.kaggle.com/datasets/ayuraj/asl-dataset`