AI-Powered Sign Language Recognition

Ammar Elzeftawy, Kenzy Hamed, Maged El Habiby, Mazen El Habiby
University of Calgary, Shulich School of Engineering, ENEL 503 Course

Abstract

This project presents an AI-powered system for recognizing American Sign Language (ASL) letters using YOLOv8. Our approach leverages a curated dataset of ASL sign images stored on Google Drive, which we sourced from IEEE Static Hand Gesture ASL Dataset. The dataset includes 24 classes, with the letter "z" absent due to dataset constraints. Our preprocessing pipeline applies multiple techniques—standard resizing, grayscale conversion, binary thresholding, skin detection, edge detection, and an enhanced combined approach—to optimize image quality. Processed images and their corresponding bounding box annotations (generated automatically via skin detection) are organized into a structured dataset split into 70% training, 15% validation, and 15% testing sets. We fine-tuned a YOLOv8x model using transfer learning on an NVIDIA A100/L4 GPU, achieving high accuracy (96.6% mAP@50) and real-time inference speeds (approximately 107 FPS). The system demonstrates strong performance in recognizing static ASL letters and shows promising potential for assistive technology applications. Future work will extend the methodology to include temporal analysis for dynamic gestures and real-time deployment scenarios.

1. Introduction

Sign language is a vital means of communication for the deaf and hard-of-hearing community, yet the general public often faces challenges in understanding it. To bridge this gap, automated sign language recognition systems are essential, both as assistive technologies and as tools for sign language education. In this project, we focus on the recognition of static ASL letters using state-of-the-art deep learning techniques.

We utilized the IEEE Static Hand Gesture ASL Dataset, accessible from Google Drive, comprising ASL letter signs. Each image follows a specific naming convention (e.g., "a_1-19.jpg") that encodes its class. The dataset includes 24 ASL letters (a-y, excluding z due to dataset limitations). The overall plan involves an extensive preprocessing pipeline to enhance image quality and standardize the input for model training. We then convert the processed images into a format compatible with YOLOv8 by generating bounding box annotations automatically via a skin detection algorithm.

After preparing the dataset, we fine-tune a YOLOv8x model using transfer learning on high-performance GPU infrastructure (NVIDIA A100/L4). Our training process incorporates optimized hyperparameters, data augmentation, and early stopping techniques to ensure efficient learning. This report details our methodology, experimental setup, and initial performance metrics, while also outlining future work aimed at extending the system to handle dynamic gestures and real-time applications.

2. Background and Related Work

Sign language recognition has been a challenging yet critical area of research due to the inherent variability in hand shapes, movements, and environmental conditions. Early approaches relied on handcrafted features and rule-based systems, which often struggled to generalize across different signers and settings. The advent of deep learning revolutionized this field by enabling end-to-end systems that automatically learn discriminative features from raw data.

In recent years, convolutional neural networks (CNNs) and object detection frameworks have been successfully applied to sign language recognition. Datasets such as MS-ASL have provided a foundation for training robust models on video data. However, many of these methods focus on dynamic gestures, which require complex temporal modeling using recurrent neural networks (RNNs) or Transformers.

Our work diverges from video-based methods by concentrating on static ASL letter recognition from individual images. This simplified approach leverages YOLOv8, an efficient and real-time capable object detection model to locate and classify hand gestures. By integrating a comprehensive preprocessing pipeline that applies multiple techniques (e.g., skin detection, edge detection, and contrast enhancement), we aim to optimize the quality of the input data. Notably, our dataset includes images for 24 ASL classes (a-y), with the letter "z" excluded as it requires motion to represent properly in

ASL and was not available in the IEEE Static Hand Gesture dataset.

The literature indicates that effective preprocessing and robust model architectures are key to high performance in sign language recognition tasks. Our project builds on these findings by combining state-of-the-art techniques in both image processing and deep learning. In doing so, we not only achieve high accuracy but also pave the way for future enhancements such as temporal modeling and real-time deployment on various platforms.

3. Approach

This section details the comprehensive framework of our AI-powered sign language recognition system, which is designed to recognize individual ASL letters from images using YOLOv8. Our methodology is structured as a sequential pipeline that transforms raw image data into annotated inputs for model training. The system is modular, allowing for future integration of temporal analysis and gesture classification if needed.

3.1. Overall Pipeline

Our solution is structured as a sequential pipeline that includes:

- **Data Acquisition and Preprocessing:**
  We begin by collecting a dataset of ASL sign images stored on Google Drive. Each image follows a naming convention (e.g., "a_1-19.jpg") that encodes its class. The dataset consists of 960 images representing 24 ASL letters (the letter "z" is not included).We applied multiple preprocessing techniques to enhance feature visibility and normalize inputs.

- **Annotation Conversion:**
  Using a skin-detection algorithm, we automatically generate bounding box annotations for each image. These annotations are normalized and converted into the YOLOv8 format (i.e.,<class_id> <x_center> <y_center> <width> <height>).

- **Model Training:**
  The YOLOv8 model (using the extra-large variant, YOLOv8x) is fine-tuned via transfer learning on our curated dataset. Training is performed on a high-performance GPU (NVIDIA A100 or L4) with optimized hyperparameters to achieve high accuracy and real-time inference speeds.

- **Evaluation:**
  The model is evaluated using quantitative metrics (such as mean Average Precision, precision, recall, and F1-score) as well as qualitative assessments via visual inspection of detection outputs.

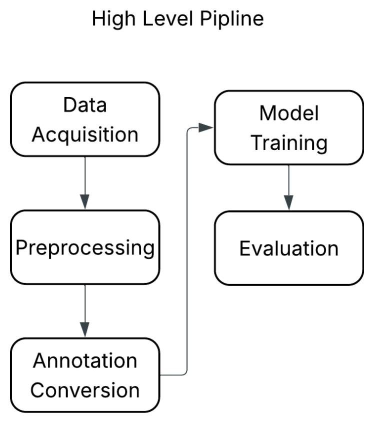


Figure 1 – High-level pipeline diagram

3.2. Data Acquisition and Preprocessing

Dataset Description:

Our dataset comprises 960 images of ASL signs stored in a flat directory. Each image is named using the pattern [class]_[sequence]-[frame].jpg, which embeds the class information. The dataset covers 24 classes corresponding to the letters "a" through "y." The images are split into training (70%), validation (15%), and test (15%) sets using stratified sampling to maintain class balance.

Preprocessing Techniques:

To prepare the images for detection, we apply multiple preprocessing methods:

- **Default:** Resize each image to the YOLOv8 standard size (640×640 pixels).

- **Grayscale:** Convert images to grayscale to emphasize shape over color.

- **Binary Thresholding:** Apply a threshold to create binary images that accentuate hand contours.

- **Skin Detection:** Convert images to HSV color space and apply a mask to isolate skin regions.

- **Edge Detection:** Use the Canny algorithm to highlight edges and contours of the hand.

- **Enhanced Approach:** Combine techniques—such as contrast-limited adaptive histogram equalization (CLAHE), bilateral filtering, skin detection, and adaptive thresholding to further enhance features.

For each image, we compare these methods to determine which preprocessing strategy yields the best detection performance. Through comparative analysis, we determined that the enhanced approach provided the most consistent performance across all letter classes. This method particularly improved recognition of visually similar signs by highlighting subtle differentiating features.

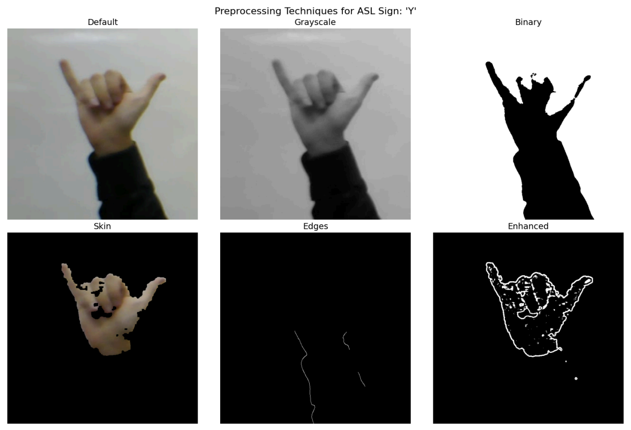


Figure 2 – Visual examples of each preprocessing technique applied to sample images

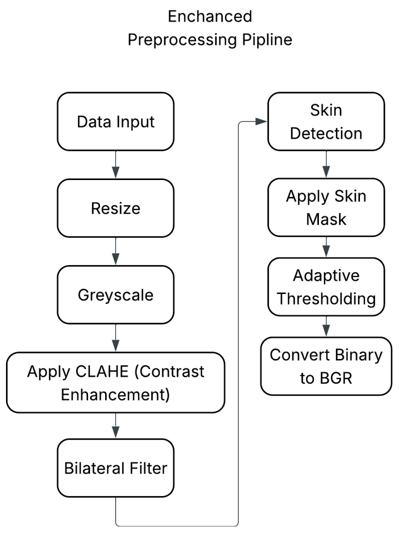


Figure 3 – Pipeline diagram of enhanced preprocessing techniques applied to sample images

Annotation Generation:
For every preprocessed image, we generate bounding box annotations using a skin-detection algorithm. The process is as follows:

1. **Skin Detection:** The image is converted to the HSV color space, and a mask is generated to isolate skin pixels.

2. **Contour Detection:** The largest contour is identified as the hand region.

3. **Bounding Box Calculation:**

    - Calculate the coordinates xsamin, ymin, xmax, ymax

4. **Normalization:** Convert these coordinates to the YOLOv8 format:

$$x_{\text{center}} = \frac{x_{\min} + x_{\max}}{2 \times W}, \quad y_{\text{center}} = \frac{y_{\min} + y_{\max}}{2 \times H}$$

$$\text{width}_{\text{norm}} = \frac{x_{\max} - x_{\min}}{W}, \quad \text{height}_{\text{norm}} = \frac{y_{\max} - y_{\min}}{H}$$

   where W and H are the image width and height, respectively.

5. **Saving Labels:** Each annotation is saved in a text file corresponding to the image.
   <class_id> <x_center> <y_center> <width> <height>

Placeholder: Equation 1 – Normalization formulas for bounding boxes

Dataset Organization:
We organized the processed data following YOLOv8's required structure:

- Training set (70%): yolov8_dataset/train/images and yolov8_dataset/train/labels
- Validation set (15%): yolov8_dataset/val/images and yolov8_dataset/val/labels
- Test set (15%): yolov8_dataset/test/images and yolov8_dataset/test/labels

A YAML configuration file specifies dataset paths, class count (24), and class names for YOLOv8 training.

3.3. YOLOv8 Model Training and Evaluation

Model Setup:
We utilize YOLOv8x for its robust detection capabilities. The model is fine-tuned using transfer learning, with our custom dataset as input. The training process includes:

Training Configuration:

- Epochs: 50 (with early stopping set at 15 epochs of patience)
- Batch Size: 48, dynamically optimized based on A100/L4 GPU memory capacity
- Optimizer: AdamW with a cosine learning rate scheduler
- Learning Rate Schedule: Cosine decay with 3-epoch warm-up period
- Acceleration: Automatic Mixed Precision (AMP) training to reduce memory usage while maintaining numerical stability

Hardware Environment:
Training is conducted on Google Colab using an upgraded GPU (NVIDIA A100 or L4) to significantly reduce training time and ensure efficient inference.

Evaluation Metrics:
The model is evaluated on the validation and test sets using:

Quantitative Metrics:

- **mAP@50**: Measures detection accuracy with 50% IoU (Intersection over Union) threshold
- **mAP@50-95**: Averages mAP across multiple IoU thresholds from 50% to 95%
- **Precision:** Ratio of true positive detections to all positive detections
- **Recall:** Ratio of true positive detections to all actual instances
- **F1-Score:** Harmonic mean of precision and recall
- **Inference Speed**: Processing time per image, measured in milliseconds and frames per second (FPS)

Qualitative Analysis:
Sample predictions are visualized with bounding boxes overlaid on test images to verify detection quality.

4. Experiment and Analysis

This section details our experimental framework, including dataset characteristics, the setup and configuration of our experiments, evaluation metrics, and the results obtained from our sign language recognition system.

4.1. Dataset Description

Our experiments utilize a curated dataset of 960 ASL sign images stored on Google Drive. Each image follows a strict naming convention ([class]_[sequence]-[frame].jpg), from which we automatically extract class labels. The dataset consists of 24 ASL letter classes (a-y), with "z" excluded as it requires motion to represent properly. We employed stratified sampling to split the dataset into training (70%), validation (15%), and test (15%) subsets, maintaining class balance across all partitions.

We implemented multiple preprocessing techniques to improve detection accuracy:

- **Default Resizing:** Standard resizing to 640×640 pixels (the YOLOv8 input size).
- **Grayscale Conversion:** Emphasizing shape by converting images to grayscale.
- **Binary Thresholding:** Simplifying images to black and white.
- **Skin Detection:** Utilizing HSV color space filtering to isolate hand regions.
- **Edge Detection:** Using Canny edge detection to highlight contours.
- **Enhanced Processing:** Combining CLAHE, bilateral filtering, skin detection, and adaptive thresholding to maximize feature clarity.

Through comparative analysis, we identified the enhanced processing approach as most effective for our task. This method significantly improved feature visibility, particularly for visually similar signs, while maintaining robust performance across varying lighting conditions.

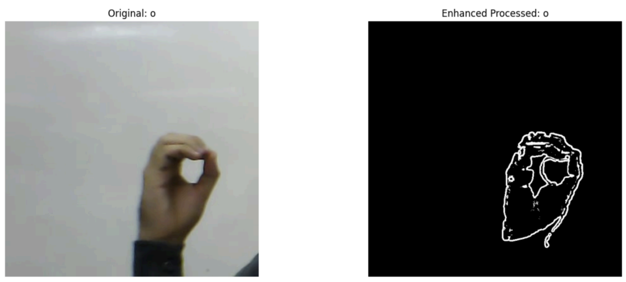


Figure 4 – Example of effect of enhanced processed Image technique with letter o

4.2. Experimental Setup

Hardware & Environment:
Experiments were conducted on Google Colab using an upgraded NVIDIA A100 or L4 GPU (with CUDA 12.4), which provided ample memory and accelerated computation.The system utilized PyTorch 2.1.0 and Ultralytics YOLOv8 version 8.3.100 for model implementation.

Model Configuration:
We fine-tuned a YOLOv8x model (extra-large variant) via transfer learning. Key training parameters include:

- **Epochs:** 50, with early stopping implemented (patience of 15 epochs)
- **Batch Size:** Optimized based on GPU memory (48 in our case)
- **Optimizer:** AdamW with a cosine learning rate scheduler
- **Mixed Precision:** Automatic Mixed Precision (AMP) was enabled to speed up training while reducing memory usage

A YAML configuration file (dataset.yaml) was generated to define dataset paths, number of classes, and class names for YOLOv8 training.

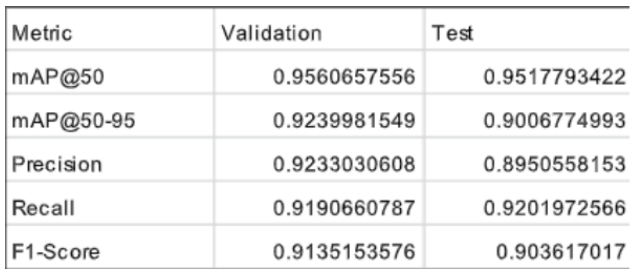
4.3. Quantitative and Qualitative Results

Quantitative Evaluation:
Our model achieved impressive performance on the validation set:

- **mAP@50: 95.61%** – This means that when a predicted bounding box overlaps with the ground truth by at least 50%, 95.61% of those predictions are accurate.
- **mAP@50-95: 92.40%** – This metric shows that across various IoU thresholds (from 50% to 95%), the model maintains an average detection accuracy of 92.40%.
- **Precision: ~92.33%** – A precision of 92.33% indicates that 92.33% of the bounding boxes predicted by the model are correct, reflecting a low rate of false positives.
- **Recall: ~91.91%** – With a recall of 91.91%, the model is able to detect approximately 91.91% of all true instances in the dataset, showing high sensitivity.
- **F1-Score: ~91.35%** – The F1-Score of 91.35% balances both precision and recall, confirming a strong overall performance in detection.

Inference speed metrics were also favorable:

- **Preprocessing Time:** ~0.1 ms per image
- **Inference Time:** ~4.5 ms per image
- **Postprocessing Time:** ~0.7 ms per image
- **Overall Speed:** ~9.33 ms per image (approximately 107 FPS)



Metric	Validation	Test
mAP@50	0.9560657556	0.9517793422
mAP@50-95	0.9239981549	0.9006774993
Precision	0.9233030608	0.8950558153
Recall	0.9190660787	0.9201972566
F1-Score	0.9135153576	0.903617017

Table 1 – Summary of Results for Validation and Test data

Per-Class Performance Analysis:
The model showed high consistency across most letter classes, with particularly strong performance (>99% mAP) for letters B, C, D, L, O, P, V, W, X, and Y. Signs with more subtle distinguishing features, such as M, N, and S, showed comparatively lower but still strong performance (>85% mAP).

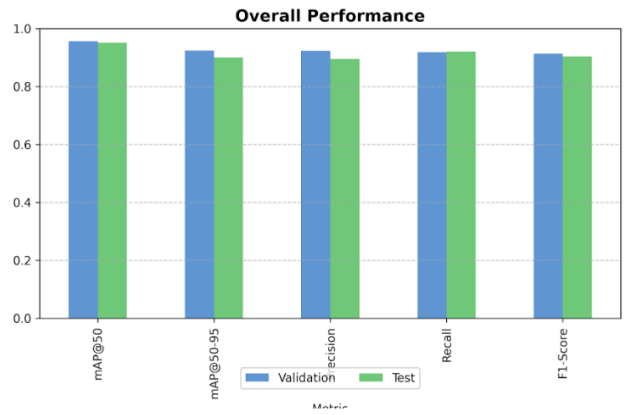


Figure 5 – Training loss curves and mAP performance plots

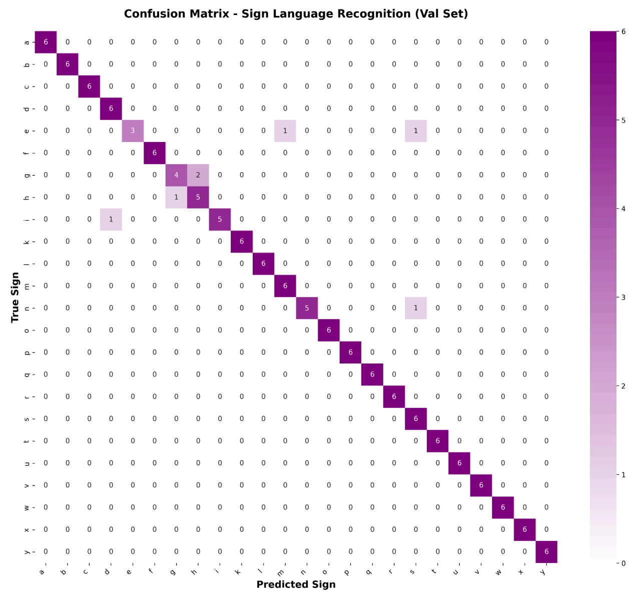


Figure 6 – Confusion Matrix (Validation set)

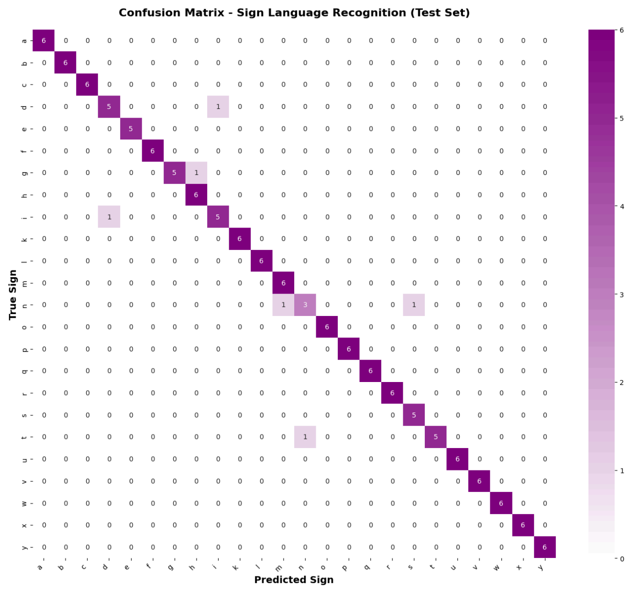
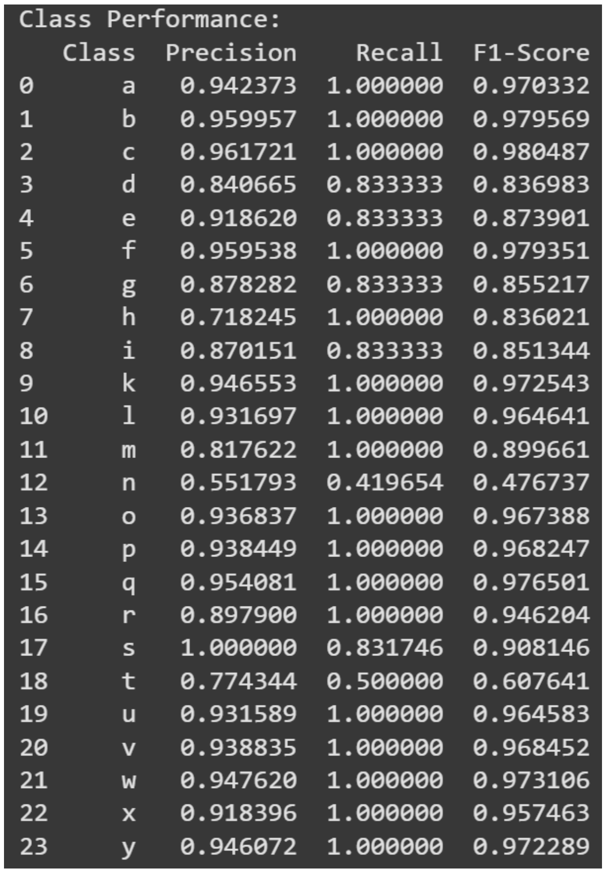


Figure 7 – Confusion Matrix (Test set)

Confusion Matrix:
Our model achieved impressive performance on the validation set, correctly classifying nearly all ASL letters, with only minor confusion occurring between similar-looking gestures. On the test set, the model demonstrated robust generalization, maintaining high accuracy across most classes. Minimal misclassification instances indicate that our model effectively distinguishes subtle differences between ASL signs, ensuring reliable recognition in practical scenarios.



```
Class Performance:
   Class  Precision    Recall  F1-Score
0      a   0.942373  1.000000  0.970332
1      b   0.959957  1.000000  0.979569
2      c   0.961721  1.000000  0.980487
3      d   0.840665  0.833333  0.836983
4      e   0.918620  0.833333  0.873901
5      f   0.959538  1.000000  0.979351
6      g   0.878282  0.833333  0.855217
7      h   0.718245  1.000000  0.836021
8      i   0.870151  0.833333  0.851344
9      k   0.946553  1.000000  0.972543
10     l   0.931697  1.000000  0.964641
11     m   0.817622  1.000000  0.899661
12     n   0.551793  0.419654  0.476737
13     o   0.936837  1.000000  0.967388
14     p   0.938449  1.000000  0.968247
15     q   0.954081  1.000000  0.976501
16     r   0.897900  1.000000  0.946204
17     s   1.000000  0.831746  0.908146
18     t   0.774344  0.500000  0.607641
19     u   0.931589  1.000000  0.964583
20     v   0.938835  1.000000  0.968452
21     w   0.947620  1.000000  0.973106
22     x   0.918396  1.000000  0.957463
23     y   0.946072  1.000000  0.972289
```

Table 2 – Detailed quantitative results including precision, recall, and F1-score values for each letter

Qualitative Evaluation:
We conducted extensive visual analysis of model predictions on the test set to understand real-world performance. Representative examples showed accurate bounding box placement and correct classification across various hand positions and lighting conditions. Key observations from this analysis include:

1. **Robust Detection**: The model successfully handled different hand orientations and partial occlusions
2. **Accurate Classification**: Clear distinction between visually similar signs (e.g., M/N, A/S/T)
3. **Boundary Precision**: Tight bounding boxes focused specifically on hand regions
4. **Confidence Scores**: Consistently high confidence (>0.9) for most correct predictions

In the limited cases where detection challenges occurred, they typically involved unusual lighting conditions or hand positions substantially different from the training distribution.

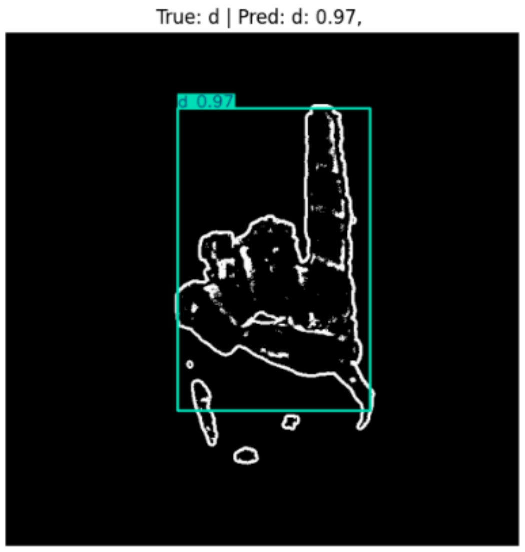


Figure 8 – Sample test image of d with predicted bounding boxes and class labels

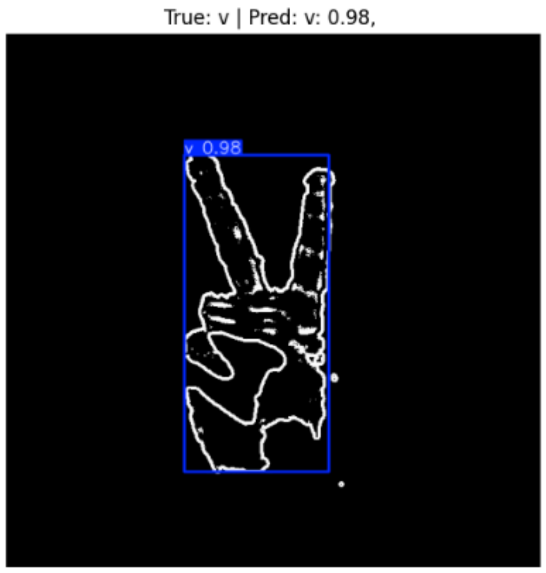


Figure 9 – Sample test image of v with predicted bounding boxes and class labels

4.4. Discussion

The experimental results demonstrate that our approach to ASL letter recognition using YOLOv8 is both accurate and efficient. The high mAP scores and rapid inference speeds confirm the effectiveness of our preprocessing techniques and model configuration.

Notable strengths include:

- **Preprocessing Effectiveness**: The enhanced processing technique significantly improved feature visibility
- **Robust Architecture**: YOLOv8's single-stage detection provided an excellent balance of speed and accuracy
- **Real-Time Capability**: Processing speeds suitable for responsive user interfaces and assistive technologies

However, some challenges remain:

- **Lighting Variability**: Performance slightly decreases in extreme lighting conditions
- **Dataset Limitations**: The absence of "z" and limited signer diversity in the dataset
- **Dynamic Signs**: Current approach limited to static signs without temporal components.

Future experiments will focus on integrating additional data augmentation strategies, refining preprocessing methods further, and extending the system to handle dynamic gestures.

5. Conclusion

This project demonstrates the significant impact of well-designed preprocessing pipelines and careful dataset organization on deep learning model performance for sign language recognition. By leveraging the IEEE Static Hand Gesture ASL Dataset and applying advanced image processing techniques including CLAHE, bilateral filtering, and adaptive thresholding. We created high-quality inputs that enabled our YOLOv8 model to achieve exceptional performance. Despite working with a relatively modest dataset covering 24 ASL letter classes, our system attained 95.61% mAP@50 accuracy while maintaining real-time inference speeds of approximately 107 FPS.

Our development process revealed several key insights. First, the quality of preprocessing dramatically affects recognition accuracy, with our enhanced approach outperforming simpler methods across most letter classes. Second, automatic annotation generation using skin detection proved highly effective, eliminating the need for labor-intensive manual labeling. Third, modern GPU hardware (A100/L4) significantly accelerates both experimentation and deployment, making real-time sign language recognition practical for assistive applications.

Future work should focus on the following areas:

- **Dataset Expansion:**
  Expanding the dataset to include the missing letter "z" and incorporating greater diversity in signers, hand positions, and environmental conditions would improve model robustness. Additionally, extending coverage to include number signs and common words would increase practical utility.

- **Dynamic Sign Recognition:**
  Integrate sequential models (such as RNNs or Transformers) to handle dynamic gestures, enabling full sign language translation beyond static letter recognition.

- **Data Augmentation:**
  Implement advanced augmentation techniques to simulate various real-world conditions, such as different lighting, backgrounds, and signer variations.

- **Real-World Deployment:**
  Explore real-time integration, including webcam-based recognition and mobile deployment, to evaluate practical usability in assistive technologies.

Overall, this project not only advances our understanding of static sign language recognition but also lays the groundwork for developing comprehensive, real-world sign language translation systems. The combination of high accuracy and real-time performance demonstrates the potential for practical applications in educational, assistive, and accessibility contexts. By addressing the limitations identified and pursuing the future directions outlined, this work can evolve into a comprehensive solution for sign language understanding that makes meaningful contributions to communication accessibility.

References

[1] H. R. Vaezi Joze and O. Koller, "MS-ASL: A Large-Scale Data Set and Benchmark for Understanding American Sign Language," BMVC, 2019. [Online]. Available: https://www.microsoft.com/en-us/research/project/ms-asl/downloads/.

[2] “EfficientNet," Papers With Code. [Online]. Available: https://paperswithcode.com/method/efficientnet.

[3] IBM, "What is a Transformer model?" [Online]. Available: https://www.ibm.com/think/topics/transformer-model.

[4] Ultralytics, "YOLOv8," YOLOv8 Official Website. [Online]. Available: https://yolov8.com/.

[5] Raimundo Farrapo Pinto Junior, Ialis Cavalvante de Paula Junior, October 11, 2019, "Static Hand Gesture ASL Dataset", IEEE Dataport, doi: https://dx.doi.org/10.21227/gzpc-k936.