Weekly Report

Week 4

Nguyen Huu Huy Thinh

2025-02-24

Abstract

This report summarizes the tasks completed, challenges faced, and progress made during the past week. Additionally, it outlines the next steps and any issues that require further discussion.

1 Introduction

Last week, I focused on training a YOLO 11 model for segmentation to distinguish between walls and obstacles. After training the model, I integrated it with the OAK 1 camera to capture real-time segmented images. These images were then used as input for a deep predictive model applied to plastering work.

2 Data Collection For Segmentation

The dataset consists of 640×640 images captured directly using the OAK 1 camera. A total of 250 images were collected, which were then labeled using Roboflow.

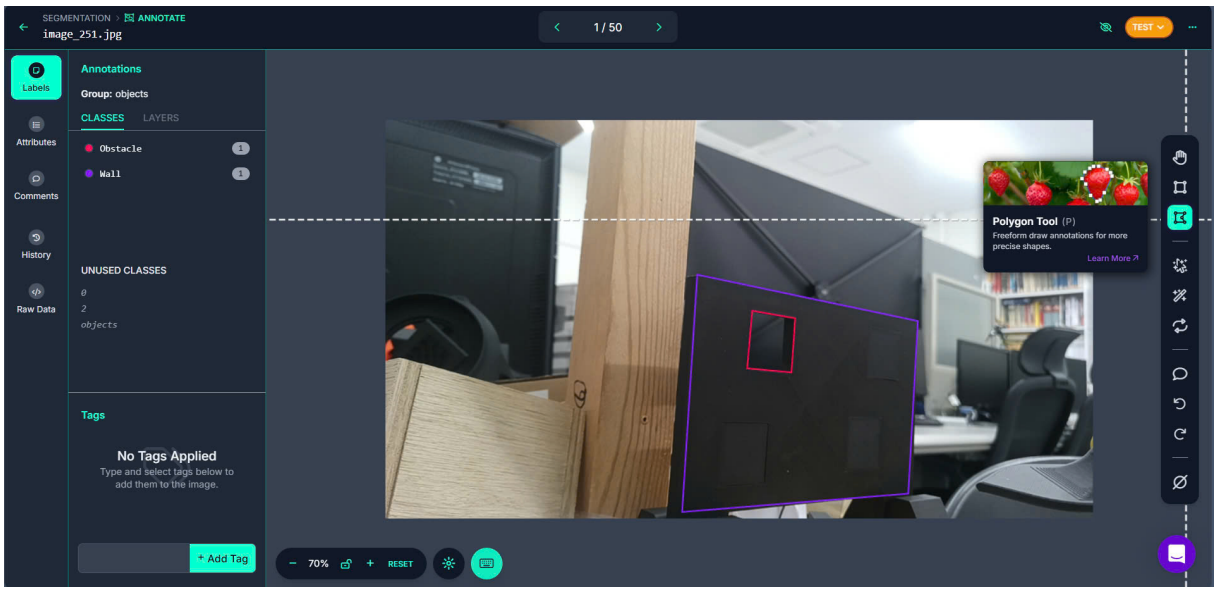


Figure 1: Labeling using Roboflow

To expand the dataset, various augmentation techniques such as rotation, flipping, and brightness adjustment were applied, increasing the total number of images to 516. The dataset was then split into 75% for training, 15% for validation, and 10% for testing.

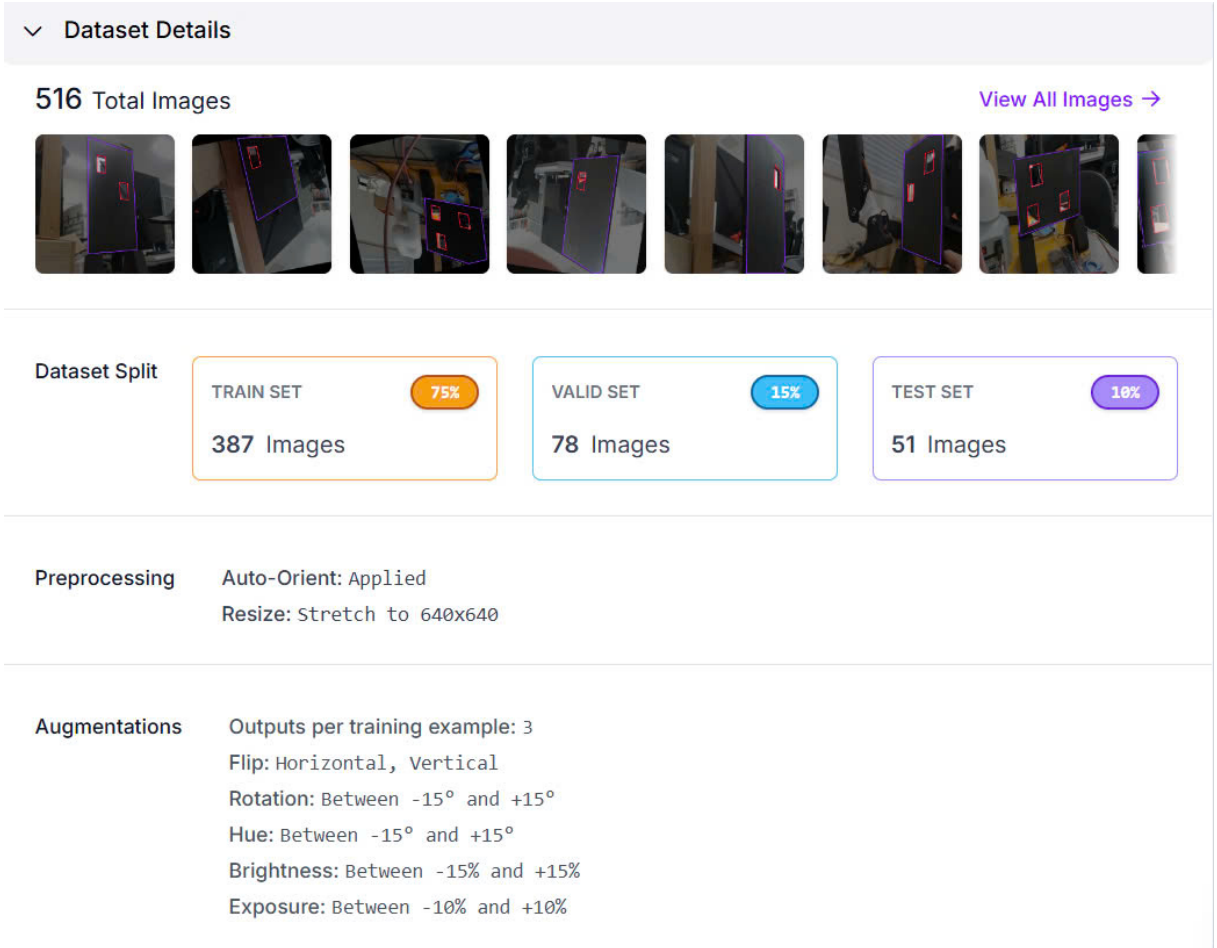
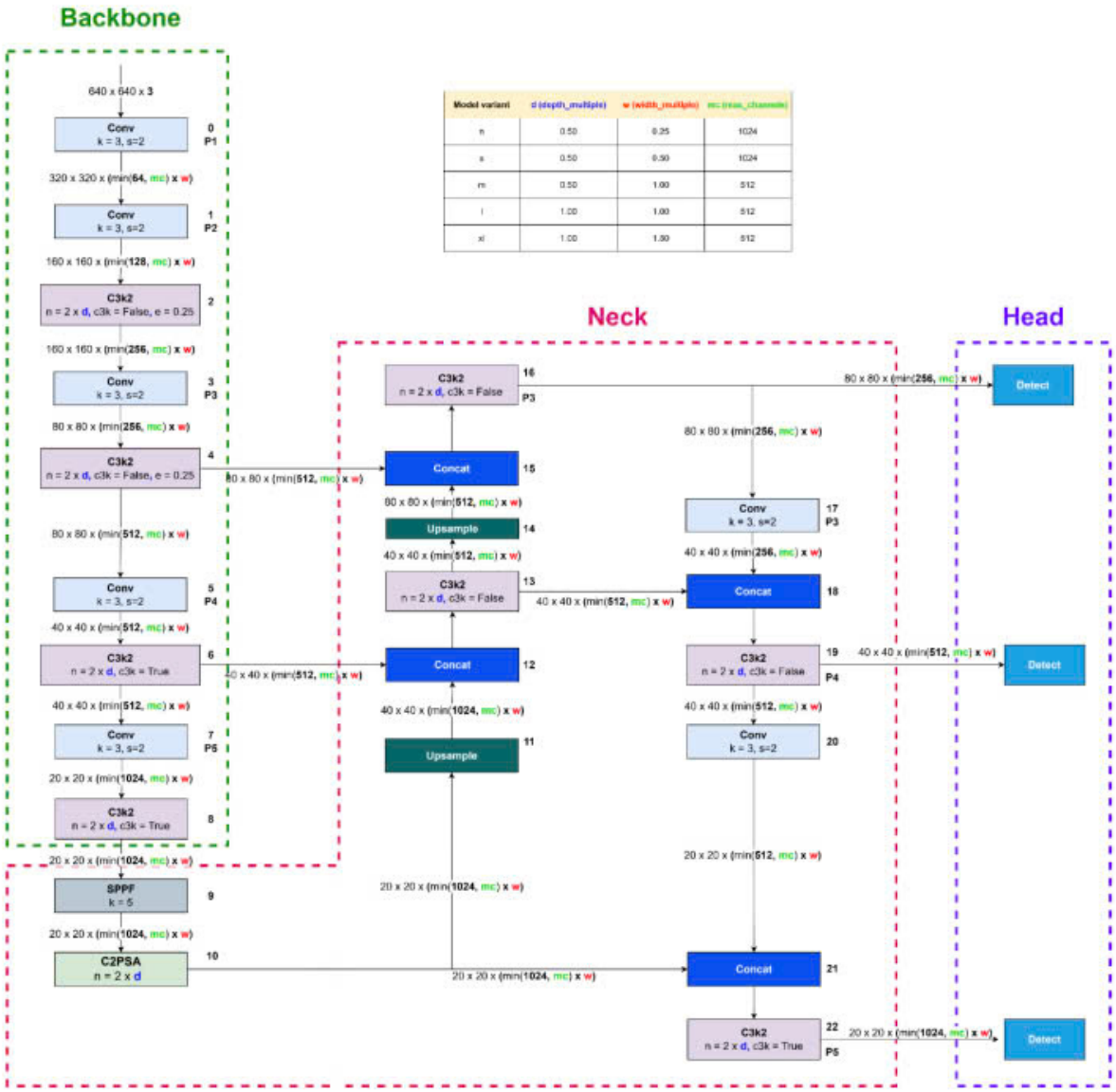


Figure 2: Dataset Details

3 YOLO 11 Model

YOLO (You Only Look Once) is a deep learning-based object detection model known for its speed and accuracy. YOLO 11, the latest iteration, builds upon previous versions by improving segmentation performance and real-time processing capabilities.



Model variant	d (depth_multiple)	w (width_multiple)	mc (max_channels)
n	0.50	0.25	1024
s	0.50	0.50	1024
m	0.50	1.00	512
l	1.00	1.00	512
xl	1.00	1.50	512

Figure 3: YOLO11 Architecture [1]

YOLO 11 follows a modular architecture consisting of three main components: Backbone, Neck, and Head, each playing a crucial role in object segmentation and detection.

- Backbone: The backbone is a crucial component of the YOLO architecture, responsible for extracting features from the input image at multiple scales. This process involves stacking convolutional layers and specialized blocks to generate feature maps at various resolutions.

- Neck: The neck combines features at different scales and transmits them to the head for prediction. This process typically involves upsampling and concatenation of feature maps from different levels, enabling the model to capture multi-scale information effectively.

- Head: The head of YOLOv11 is responsible for generating the final predictions in terms of object detection and classification. It processes the feature maps passed from the neck, ultimately outputting bounding boxes and class labels for objects within the image.

Together, these components enable YOLO 11 to perform fast and accurate segmentation of walls and obstacles, making it suitable for real-time applications.

4 Training and Validation

For training, the model was set up with the following parameters:

- Epochs: 100
- Batch size: 16
- Learning Rate: 0.01
- Optimizer: AdamW

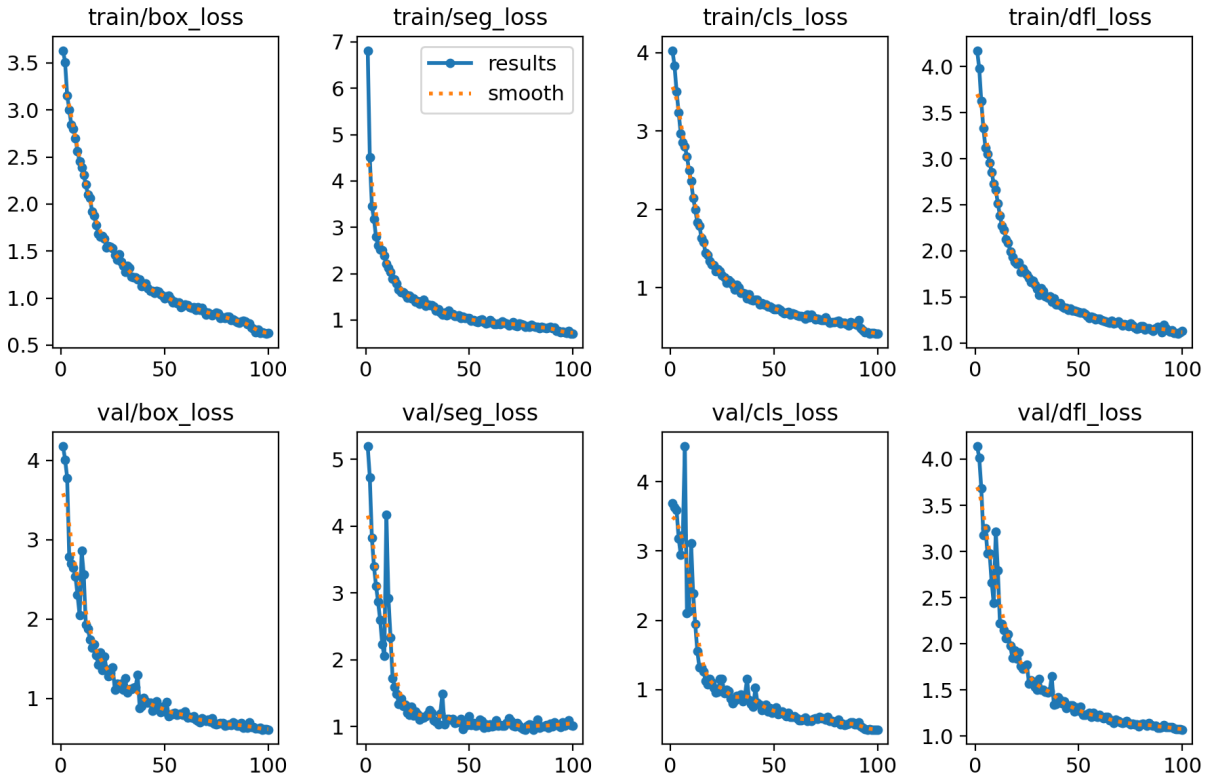


Figure 4: Training Graphs

The training process shows a gradual decrease in all key loss components, including segmentation loss (seg loss), bounding box loss (box loss), classification loss (cls loss), and distribution focal loss (dfl loss). Throughout the training, these losses steadily reduce, indicating that the model is progressively improving its ability to segment objects, predict bounding boxes, and classify object types. By the end of epoch 100, the losses appear to converge, indicating that the model has stabilized in terms of performance and has reached a point where further training may result in minimal improvements.

In the validation process, the mAP (mean Average Precision) graph shows a rapid increase in performance, reflecting that the model quickly improves its ability to detect and classify objects. By around epoch 50, the graph converges, suggesting that the model has reached its optimal performance. At this point, the model achieves high precision and recall, meaning it is correctly identifying and localizing objects with minimal false positives and false negatives.

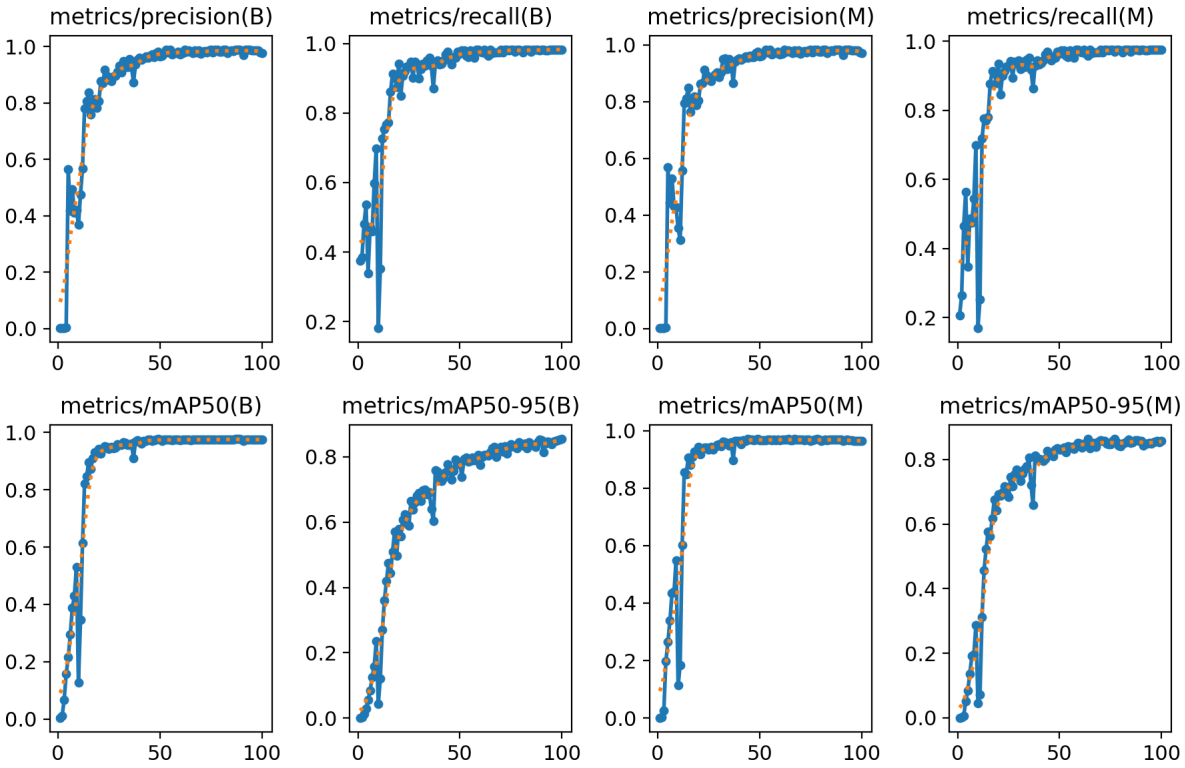


Figure 5: Validation Graphs

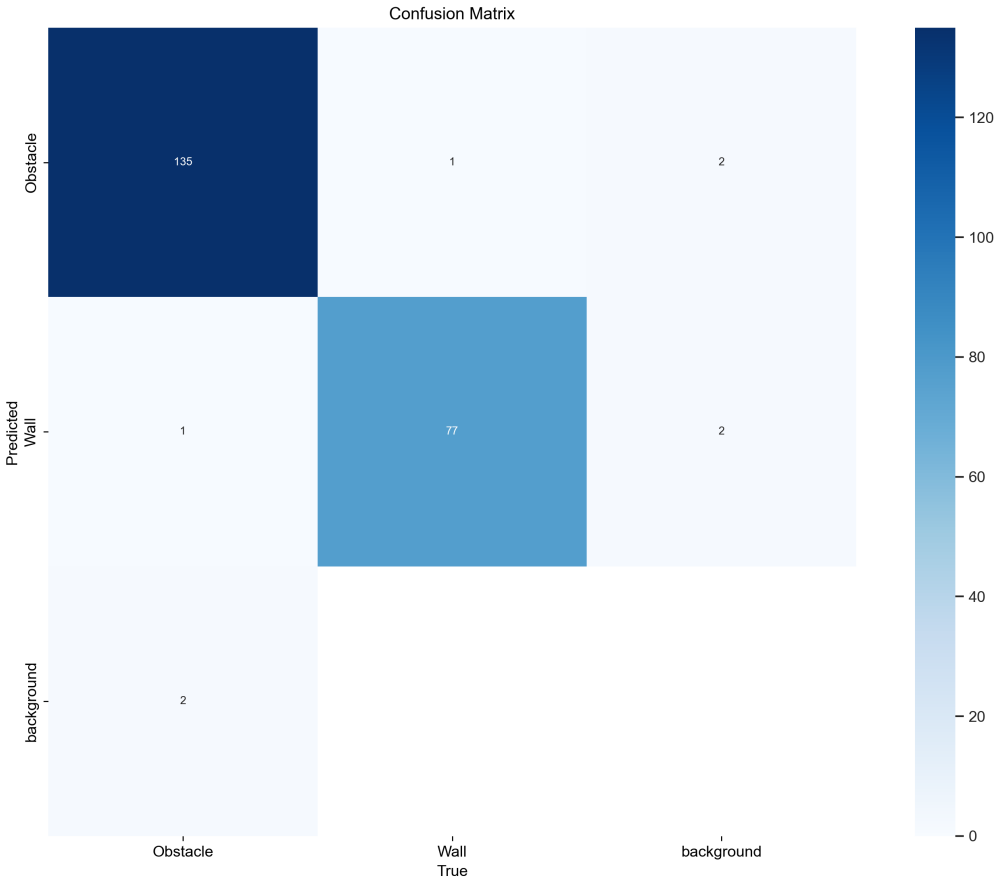


Figure 6: Confusion Matrix

5 AI Integration With OAK 1 Camera

To integrate AI with the OAK 1 camera, the DepthAI framework was used to preview the scene and capture images. Since the camera was fixed to capture the entire scene, there was no need to run the AI model continuously. Instead, the segmentation process was applied after image capture.

After training the YOLO 11 model, it was deployed on the Roboflow framework. Using the Roboflow API, segmentation was applied to the captured images. The workflow was straightforward:

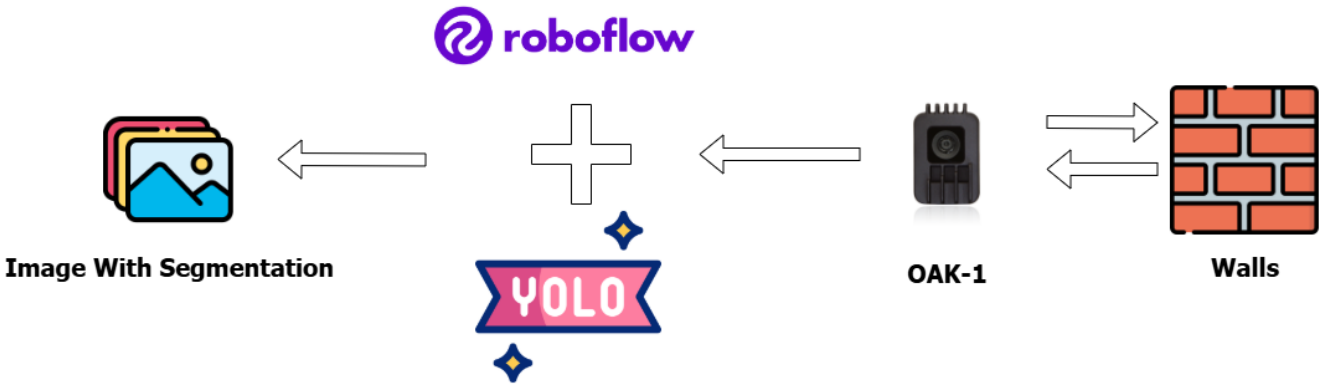


Figure 7: AI Integration With OAK 1 Camera

1. The OAK 1 camera captures the scene.
2. The captured image is sent to the Roboflow API, which returns segmentation results.
3. A simple masking algorithm is applied to overlay the segmentation information onto the original image.

6 Data Collection For Deep Predictive Model

Using the segmented images, a dataset was created to train a deep predictive model for robot arm movement in plastering tasks. Each data point in the dataset consists of the following components:

- Image: The segmented image captured from the OAK 1 camera.
- Distance (Integer): A numerical value representing the distance between the Robot Arm and the wall.

- Label (2D Array): A structured set of movements for the robot arm, where each row in the array corresponds to a specific movement step.

To store this data efficiently, the distance and action labels are saved in a JSON file, ensuring structured access for training and inference in the predictive model.

7 Next Steps

- Continue collecting and processing images with segmentation to complete the dataset.
- Design and implement a custom deep predictive model that will use the collected data for training.

8 Conclusion

This week, significant progress was made in training a YOLO 11 model for segmentation to detect walls and obstacles, which was integrated with the OAK-1 camera for real-time image capture. The segmented images were then used to create a dataset for training a deep predictive model that will guide robot arm movements in plastering tasks. Next steps include completing the data collection, developing a custom predictive model, and proceeding with training and validation to refine the system's performance.

References

[1] P. Hidayatullah, N. Syakrani, M. R. Sholahuddin, T. Gelar, and R. Tubagus, "YOLOv8 to YOLO11: A Comprehensive Architecture In-depth Comparative Review".