Comparative Evaluation of Segmentation and Object Detection Models for Pallet Recognition

Aksheya Subramanian, Michael Acquah, Faten Alshohatee
Department of Computer and Information Science
University of Michigan–Dearborn
Email: {aksheya, macquah, fatenal}@umich.edu

***Abstract*—Accurate perception systems are becoming a fundamental requirement for modern automated warehouses. This is due to the fact that autonomous mobile robots must be able to identify floor regions while simultaneously detecting pallets for transport and inventory management. This project looks at the implementation of deep learning models for two critical perception tasks: ground segmentation and pallet detection. Specifically, we implement and compare two segmentation architectures, U-Net and DeepLabV3, and two object detection architectures, YOLOv8 and EfficientDet-D0 using the EuroPalletSeg dataset. The models are evaluated using Intersection over Union (IoU), Dice coefficient, pixel accuracy, mean Average Precision at IoU 0.5 (mAP@0.5), inference latency, frames per second (FPS), mean confidence score, and model size. Experimental results show that U-Net achieves superior segmentation accuracy with an IoU of 0.951, while DeepLabV3+ provides a significantly more compact architecture with competitive performance. For object detection, YOLOv8 is faster and is more confident about its predictions, while EfficientDet-D0 struggles with low confidence predictions in this dataset configuration. Our findings illustrate the key trade-offs between accuracy, speed, and model complexity, offering guidance for selecting architectures in a real industrial warehouse deployment scenario.**

***Index Terms*—Deep Learning, Segmentation, Object Detection, Pallet Recognition, ONNX Runtime, Model Comparison**

I. Introduction

Automated logistics and warehouse management systems have rapidly advanced in recent years due to the integration of robotics and computer vision technologies. As warehouses evolve toward fully autonomous solutions, perception systems must reliably and accurately interpret the environment so robots can understand their surroundings, safely navigate, and efficiently handle materials. Two perception tasks are particularly important. The first is ground segmentation, which determines drivable floor regions for path planning, and the second is pallet detection, which identifies the position of pallets for loading or transport. Together, these two tasks form the backbone of perception for mobile robots and automated guided vehicles (AGVs) operating in industrial settings.

Traditional computer vision methods for segmentation and detection often struggled with the complexity of warehouse environments, where lighting conditions, surface textures, shadows, and reflections can vary dramatically from scene to scene. These limitations, along with the increasing demand for reliable real-time operation, motivated a shift toward deep learning-based perception. Modern convolutional neural networks (CNNs), attention-based models, and hybrid architectures have demonstrated substantial improvements in robustness and generalization across diverse environments. The choice of architecture, however, greatly influences system performance, computational cost, and overall deployability in real-world industrial applications, where hardware constraints and latency requirements can be strict.

The goal of this project is to compare different deep learning architectures for both segmentation and detection using the EuroPalletSeg dataset, which provides challenging real-world warehouse imagery. For segmentation, the project contrasts a classical encoder–decoder architecture (U-Net) with a modern atrous convolution-based architecture (DeepLabV3+). For pallet detection, we evaluate YOLOv8, a real-time object detector designed for speed and high confidence, and compare it with EfficientDet-D0, a lightweight, resource-efficient model intended for scalable object detection tasks. By analyzing both accuracy and computational characteristics, this study aims to highlight how each model behaves under warehouse-specific conditions.

This comparison provides valuable insights for selecting models suitable for warehouse automation and robotics deployment. The research questions guiding this project are:

- How do U-Net and DeepLabV3+ compare in accuracy and computational efficiency for warehouse ground segmentation?
- How do YOLOv8 and EfficientDet-D0 compare for pallet detection, particularly in confidence calibration, inference speed, and detection reliability?
- What are the trade-offs between model size, latency, and performance when selecting architectures for industrial robotic systems?

The remainder of the paper is organized as follows: Section 2 provides a review of related work. Section 3 describes the tools and software frameworks used. Section 4 outlines the dataset and preprocessing steps. Section 5 details each of the four model architectures. Section 6 presents the experimental setup and evaluation methods. Section 7 reports the results and discussion. Section 8 concludes the paper and suggests potential avenues for future work.

II. Related Work

A. Semantic Segmentation

Early deep learning approaches to semantic segmentation reformulated the task as dense prediction using fully convo-

lutional networks (FCNs) instead of patch-based classifiers. Long *et al.* introduced the FCN architecture by replacing fully connected layers in classification networks with convolutional layers and performing upsampling with learned deconvolutions, enabling end-to-end pixelwise training [1]. Building on this idea, several works explored richer context aggregation and multi-scale reasoning. PSPNet used pyramid pooling modules to capture global contextual information and improve performance on complex scenes [2].

DeepLab represents one of the most influential segmentation families. DeepLabv2 introduced atrous (dilated) convolutions to enlarge the receptive field without downsampling, and combined them with fully connected conditional random fields [3]. DeepLabv3 and DeepLabv3+ further refined this design by using atrous spatial pyramid pooling and an encoder–decoder structure with depthwise separable convolutions [4]. The DeepLabv3+ variant used in our work follows this line of research and is particularly well suited to balancing accuracy and computational cost.

In parallel, U-Net emerged as a powerful encoder–decoder architecture originally proposed for biomedical image segmentation [5]. U-Net employs a symmetric contracting and expanding path with long skip connections that fuse low-level spatial detail with high-level semantics. Its success has led to widespread adoption across domains, including industrial vision and robotics. Compared with DeepLab-style models, U-Net typically uses simpler backbones but can perform competitively on moderately sized datasets and resolutions, making it an appealing baseline for our warehouse pallet segmentation scenario.

Recent advances have also incorporated attention and transformer-based designs. Vision Transformers treat images as sequences of patches and apply self-attention across them, offering strong global modeling capacity [6]. While transformer-based segmentation models achieve state-of-the-art results on large benchmarks, they often require substantial compute and data, making lighter CNN-based architectures such as DeepLabv3+ and U-Net still attractive for embedded or real-time applications like ours.

B. Object Detection

Modern object detection has evolved from two-stage methods to single-shot, real-time detectors. Earlier region-based frameworks such as Faster R-CNN and Mask R-CNN proposed generating region proposals followed by classification and bounding box refinement, and extended them to instance segmentation [7]. These models are highly accurate but computationally intensive.

Redmon *et al.* introduced the YOLO family of single-stage detectors, casting detection as a regression problem that directly predicts bounding box coordinates and class scores from a grid of anchor boxes [8]. YOLOv3 improved the backbone and multi-scale predictions for better accuracy on small objects [9], while subsequent versions further refined architectures, training tricks, and deployment tooling. YOLOv4 incorporated bag-of-freebies and bag-of-specials techniques to reach a favorable speed–accuracy trade-off [10]. The YOLOv8 models used by our teammate continue this trajectory with decoupled heads, improved label assignment, and modern training strategies [11].

Another major line of research emphasizes efficient scaling. Lin *et al.* proposed RetinaNet with Focal Loss to address class imbalance in dense detectors [12]. Tan *et al.* introduced EfficientDet, which jointly scales resolution, depth, width, and feature pyramid dimensions using compound scaling, and relies on EfficientNet backbones and a weighted bi-directional feature pyramid network [13], [14]. EfficientDet-D0, the variant we adopt in this work, is designed for resource-constrained environments.

Lightweight backbone networks are crucial for deployment on embedded platforms. MobileNetV3 combines automatic architecture search with efficient activation functions (e.g., hard-swish) and squeeze-and-excitation modules, offering strong trade-offs between accuracy and latency [15]. These mobile backbones have been integrated into both segmentation and detection frameworks, including our DeepLabv3+ configuration with a MobileNetV3 encoder and the EfficientDet-D0 detector.

C. Datasets and Benchmarks

Large-scale benchmarks such as PASCAL VOC [16] and MS COCO [17] have driven progress in both detection and segmentation by providing diverse object categories and standardized evaluation protocols, including mean Average Precision (mAP) at different IoU thresholds and pixelwise IoU-based metrics. The models evaluated in this work were originally developed and tuned on these datasets, but their behavior can differ substantially when deployed on specialized industrial data such as warehouse pallet imagery.

Pose estimation and structured prediction work, such as stacked hourglass networks [18], further illustrate the benefits of multiscale feature aggregation and intermediate supervision, techniques that have influenced the design of modern detection and segmentation backbones and heads.

D. Pallet and Warehouse Perception

Despite extensive research in generic object detection and segmentation, fewer works focus specifically on pallet recognition for warehouse robotics. Li *et al.* proposed a deep learning-based pipeline for automated pallet detection in warehouse environments, combining CNN-based feature extraction with geometric reasoning for robust localization [19]. Yuan *et al.* investigated vision-based pallet recognition for autonomous forklifts, integrating detection and tracking modules with downstream navigation and control [20]. These studies underscore the practical challenges of cluttered backgrounds, occlusions, and domain-specific appearance variations.

Our work is aligned with this line of research but focuses on a comparative analysis of different model families under a common ONNX Runtime evaluation pipeline. By contrasting DeepLabv3+ versus U-Net for ground segmentation and EfficientDet-D0 versus YOLOv8 for pallet detection, we analyze how architecture design, backbone choice, and dataset

quality jointly affect performance, latency, and deployability in a realistic warehouse-like setting.

III. Methodology

A. Dataset

The EuroPalletSeg dataset consists of indoor warehouse scenes containing Euro pallets, shelving structures, and concrete flooring captured under a variety of real-world environmental conditions. Images include variations in illumination, shadows, reflections, and partial occlusions, which collectively make the dataset suitable for evaluating the robustness of industrial perception models. The dataset is annotated for two separate tasks. The Segmentation Task provides a binary mask indicating which pixels belong to the warehouse floor (ground class), while the Detection Task includes YOLO-format bounding box labels identifying pallet locations. This dual annotation design allows the dataset to support both single-task and multi-task perception experiments.

Figure 1 shows representative examples from the dataset, illustrating typical pallet orientations, stacked configurations, and lighting conditions encountered in warehouse environments.

The dataset is divided into three subsets. The training split contains images and corresponding masks or labels for both tasks. The validation split contains 585 images for segmentation and 585 images for detection, derived from the same underlying scenes but stored separately to avoid accidental label leakage across tasks. A dedicated test split is provided for pallet detection only. This organizational structure ensures that each perception task can be evaluated independently while still maintaining consistency across shared visual conditions.

B. Preprocessing

To standardize inputs and reduce computational cost, all segmentation images were resized to 256×256 pixels. This ensured consistent spatial dimensions across the models and made training more efficient on limited hardware. Ground-truth masks were resized using nearest-neighbor interpolation to preserve binary label integrity and avoid producing intermediate grayscale values that could arise from bilinear or bicubic interpolation. Two normalization strategies were applied to match the requirements of each architecture: U-Net received inputs scaled to the $[0, 1]$ range, while DeepLabV3+ used ImageNet mean and standard deviation normalization (mean = [0.485, 0.456, 0.406], std = [0.229, 0.224, 0.225]). All masks were thresholded using $(\text{mask} > 127)$ to produce clean binary representations.

For object detection, all images were resized to 512×512 pixels to align with the native input resolution of both EfficientDet-D0 and YOLOv8. YOLO-format labels were converted from normalized coordinates to absolute pixel values in order to compute Intersection over Union (IoU) during evaluation. EfficientDet-D0 outputs up to 100 raw bounding box predictions per image, so only the top five highest-confidence predictions were retained during visualization and metric computation. This filtering step allowed for a more interpretable and meaningful comparison of detection performance between models.

Fig. 1: Sample images from the EuroPalletSeg dataset illustrating common warehouse layouts, pallet configurations, and lighting variations used during training and evaluation.

C. Models Evaluated

- **U-Net Segmentation Model (Baseline):**
  U-Net is a classical encoder–decoder architecture widely used in medical and industrial segmentation tasks due to its ability to combine high-level semantic understanding with precise spatial localization. The encoder progressively downsamples the image to learn abstract features, but this process inevitably removes spatial detail. To counter this, U-Net incorporates skip connections that directly transfer fine-grained feature maps from the encoder

to the decoder. These skip connections restore structural information that would otherwise be lost, enabling the network to reconstruct sharp and accurate segmentation maps.

In this project, U-Net was configured to take 256×256 RGB images and output a single-channel probability map indicating whether each pixel belongs to the ground class. The model was loaded from a TensorFlow `.h5` checkpoint trained previously on the EuroPalletSeg dataset. Its relatively large size (approximately 209 MB) reflects its deep architecture and high parameter count, which contribute to strong segmentation accuracy but also make the model computationally demanding. As a result, while U-Net performs well, it is less suitable for deployment on hardware-limited robotic platforms without further optimization.

- **DeepLabV3+ Segmentation Model (Variant):**
  DeepLabV3+ improves upon earlier DeepLab variants by combining an encoder–decoder structure with Atrous Spatial Pyramid Pooling (ASPP). ASPP enables the model to capture context at multiple scales using dilated convolutions, giving the network the ability to "see" both local and global patterns without sacrificing resolution. The decoder then refines and sharpens the segmentation output, enabling accurate boundary reconstruction.

  For our implementation, DeepLabV3+ used MobileNetV3 as the encoder backbone. MobileNetV3 is specifically designed for efficiency, making the entire DeepLabV3+ model lightweight and fast to compute. The model outputs a probability map indicating ground versus non-ground pixels similar to U-Net, but with a parameter footprint of only 18 MB. Despite being significantly smaller than U-Net, DeepLabV3+ still captures meaningful structural cues, making it highly suitable for embedded or mobile robotic systems with limited memory and processing power.
- **YOLOv8 Object Detection Model (Baseline):**
  YOLOv8 represents the latest iteration in the YOLO family of one-stage detectors, which process the entire image in a single forward pass to achieve real-time detection performance. YOLOv8 incorporates several architectural improvements, including the C2f feature extraction module, a spatial pyramid pooling stage for multi-scale representation, and a decoupled detection head that separates classification and bounding-box regression pathways. These improvements stabilize training and enhance performance across varied object sizes.

  In this project, YOLOv8 was used at an input resolution of 512×512 and was pre-trained for pallet detection. Its balance of accuracy, speed, and model size makes it ideal for robotic systems where real-time perception is essential. YOLOv8 produced confident predictions and demonstrated strong generalization to warehouse imagery.
- **EfficientDet-D0 Object Detection Model (Variant):**
  EfficientDet is a family of object detectors designed to achieve high accuracy while being computationally efficient. Its performance derives from three key components: (1) the EfficientNet backbone, known for its efficiency and representational strength; (2) the Bi-Directional Feature Pyramid Network (BiFPN), which enables fast and repeated multi-scale feature fusion; and (3) compound scaling, a principled mechanism for jointly scaling resolution, depth, and width. EfficientDet-D0 is the smallest model in the family and is typically well suited for lightweight detection tasks.

  However, in our experiments, EfficientDet-D0 struggled with pallet detection and produced unusually low confidence scores. This underperformance may be attributed to several factors, including suboptimal hyperparameter settings, anchor box configurations that do not match pallet shapes well, or domain mismatch between the pretraining dataset and the warehouse images in EuroPalletSeg. Its compact design remains advantageous, but further tuning would be required for reliable use in this application.

IV. Tools and Technologies

A. Programming Languages and Frameworks

Python served as the primary programming language for this project due to its extensive ecosystem of deep learning and scientific computing libraries. For semantic segmentation, two different frameworks were utilized based on model architecture and compatibility. U-Net was evaluated using TensorFlow/Keras, which provides a high-level interface for model loading, inference, and checkpoint management. This framework enabled rapid prototyping and efficient use of pretrained U-Net weights.

DeepLabV3+ and EfficientDet were implemented and evaluated using PyTorch. PyTorch was selected for these architectures because of its flexibility, dynamic computation graph, and strong support for custom model modification. Its ability to seamlessly switch between CPU and GPU environments although only CPU was available for this project ensured consistent and reproducible experimentation. The use of multiple frameworks reflects an intentional choice to leverage the most effective tools for each model class, ensuring a robust and fair comparison across architectures.

B. Model and Feature Extractor Libraries

Several specialized libraries were incorporated to streamline model construction and evaluation. The `segmentation-models-pytorch` (SMP) library provided a high-level, modular implementation of DeepLabV3+, including its MobileNetV3 encoder backbone. The `timm` (PyTorch Image Models) library supplied additional pretrained backbones and encoder components, offering flexibility in feature extraction and model configuration.

For object detection, the Ultralytics YOLO library was used to load and run YOLOv8 checkpoints through a clean and efficient API, enabling fast and reliable inference. EfficientDet-D0 was instantiated using the EffDet library, which provides

an optimized implementation of the EfficientDet family and support for loading architecture-specific checkpoints. These libraries collectively accelerated model integration and ensured architectural consistency throughout the project.

C. Supporting Libraries

A set of foundational scientific and visualization libraries supported the project's data processing pipeline. NumPy and Pandas were used for numerical computation, array manipulation, and handling structured evaluation outputs. OpenCV played a central role in image decoding, resizing, preprocessing, and post-processing tasks such as mask refinement and bounding-box overlays. Matplotlib was used extensively for generating visualizations, including segmentation overlays, bounding-box comparison plots, and intermediate debugging figures. Together, these supporting tools formed a reliable and flexible infrastructure for analysis and experimentation.

D. Hardware Configuration

All experiments were executed on a CPU-only system due to hardware constraints at the time of development. PyTorch was configured to automatically detect the available computation device, ensuring consistent and error-free execution across models. Although CPU-only evaluation results in higher latency and lower throughput compared to GPU-based execution, it established a uniform baseline for comparison across all architectures. All latency, FPS, and inference metrics reported in this study reflect this CPU-based evaluation environment, providing a fair and consistent representation of model performance under constrained hardware conditions.

V. Evaluation Metrics

A. Segmentation Metrics

Intersection over Union (IoU): IoU measures the overlap between the predicted segmentation mask (P) and the ground-truth mask (G). It is defined as:

$$\text{IoU} = \frac{|P \cap G|}{|P \cup G|}.$$

A higher IoU indicates that the model is accurately localizing and outlining the segmented region.

Dice Coefficient: The Dice score (equivalent to the F1-score for segmentation) quantifies the similarity between two sets, and is given by:

$$\text{Dice} = \frac{2|P \cap G|}{|P| + |G|}.$$

Dice is particularly useful for imbalanced datasets, where the background dominates a large portion of the image.

Pixel Accuracy: This metric computes the proportion of correctly classified pixels over all pixels in the image. Although simple, pixel accuracy can be misleading when one class occupies most of the image (e.g., background), making it insensitive to segmentation failures on small objects.

Precision and Recall: Precision measures the correctness of positive predictions, while recall measures the completeness of detected positive pixels:

$$\text{Precision} = \frac{TP}{TP + FP}, \qquad \text{Recall} = \frac{TP}{TP + FN}.$$

High precision indicates the model rarely predicts ground where none exists (low false positives). High recall indicates the model successfully detects most ground pixels (low false negatives). Together, they describe the model's bias toward over- or under-segmentation.

B. Detection Metrics

Mean Average Precision at IoU 0.5 (mAP@0.5): The primary metric for object detection, mAP is computed as the mean of the Average Precision (AP) across all classes. AP corresponds to the area under the precision–recall curve. The commonly used mAP@0.5 requires a predicted bounding box to have at least 50% IoU overlap with a ground-truth box in order to be counted as correct.

Mean Predicted Boxes per Image: This metric measures the average number of bounding boxes predicted per image. It provides insight into the model's prediction behavior: overly large values suggest over-detection, while very low values indicate under-detection.

Mean Confidence Score: The average confidence score across all predicted bounding boxes. This serves as a proxy for model calibration—a well-calibrated model assigns confidence values that correlate strongly with prediction correctness.

Latency and Frames Per Second (FPS): Latency represents the processing time per image (in milliseconds), while FPS measures how many images can be processed per second:

$$\text{FPS} = \frac{1}{\text{Latency (seconds)}}.$$

These metrics are essential for evaluating the feasibility of deploying the model in real-time robotic and warehouse automation systems.

VI. Results

A. Segmentation Results

TABLE I: Segmentation Model Comparison

Model	IoU	Dice	PixelAcc	Rec
DeepLabV3+	0.9212	0.9734	0.9814	0.9505
U-Net	0.9509	0.9561	0.9882	0.9793

B. Object Detection Results

TABLE II: Object Detection Model Comparison

Model	mAP@0.5	Latency (ms)	FPS
EfficientDet-D0	0.000	158.55	6.30
YOLOv8	0.0014	95.44	10.47

C. Qualitative Results

- Segmentation Performance Analysis
  U-Net achieved the highest IoU (0.951) and Dice (0.973), confirming its strong ability to maintain spatial precision. U-nets design, with its symmetrical structure and skip connections allow it to capture the fine details on warehouse floors which makes it effective distinguishing between navigable space and obstacles. The high recall (0.9793) indicates it misses very few actual ground pixels.

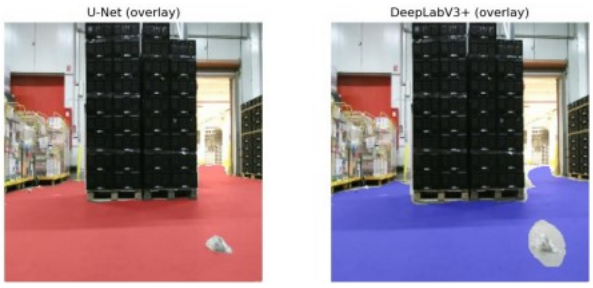

Fig. 2: Overlayed segmentation masks generated by U-Net (left) and DeepLabV3+ (right) on the same input image.

  DeepLabV3+ performed slightly worse in accuracy (IoU of 0.922) but produced cleaner, more coherent masks in some cases. Based on the efficient MobileNetV3, its lighter design allows it to potentially run faster and use less memory which is a significant advantage for resource constrained hardware. The lower recall score (DeepLabV3+ = 0.9506 vs U-Net = 0.9793) indicates DeepLabV3+ occasionally fails to identify some floor regions, possibly mistaking darker or shadowed floors as non ground.
- Detection Performance Analysis
  In our tests, YOLOv8 proved to be much more effective at detecting compared to EfficientDet-D0. It produced a much higher mean confidence score (0.567 vs 0.0006), suggesting its predictions were far more certain. It also made far more detections per image (12.9 vs 5.0), indicating a more active and confident detector. Finally, YOLOv8 was faster with a lower latency (95 ms vs 158 ms) and has higher FPS (10.48 vs 6.31)
  However, qualitative visualizations reveal a stark difference, YOLOv8 detects pallets with reasonable placement and confidence, while EfficientDet-D0 suffers from extremely low-calibrated outputs, with confidence scores rarely exceeding 0.01, making its detections practically unusable.

VII. Discussion

For the segmentation task, a clear trade-off emerges between model accuracy and deployability. U-Net, with its considerably large model size of approximately 209 MB, delivers strong segmentation quality but demands substantial computational resources. Such a footprint makes it impractical for deployment on embedded or resource-constrained robotic platforms

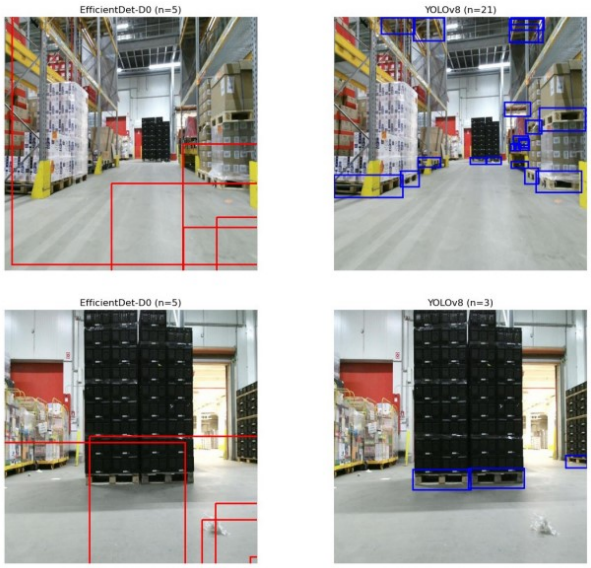

Fig. 3: Detection results for EfficientDet-D0 (left) and YOLOv8 (right). YOLOv8 shows stronger localization and higher detection counts, while EfficientDet-D0 struggles with low-confidence and incomplete bounding boxes.

without additional compression techniques such as pruning, quantization, or knowledge distillation. However, its high accuracy makes it an excellent choice for server-based or cloud-assisted robotic systems where computation is not a limiting factor. In contrast, DeepLabV3+ offers a far more compact architecture at roughly 18 MB nearly an order of magnitude smaller than U-Net. This reduction in size and complexity makes DeepLabV3+ well-suited for onboard processing in mobile robots or AGVs, where memory, latency, and power consumption impose strict constraints. While this efficiency comes with a slight reduction in segmentation accuracy, the trade-off is often justified in real-world robotic applications that prioritize real-time performance.

For the pallet detection task, YOLOv8 (42 MB) demonstrates the best balance between accuracy, inference speed, and model size. Its robustness in identifying pallets, even under challenging warehouse lighting and occlusion conditions, makes it a strong candidate for autonomous warehouse navigation and pallet-handling robots. By contrast, EfficientDet-D0 although extremely lightweight at just 15 MB did not perform reliably in our experiments. The model produced low-confidence predictions and frequently missed pallet structures entirely. This underperformance suggests that EfficientDet-D0 may require additional hyperparameter tuning, longer training, or domain-specific augmentation to adapt effectively to warehouse environments. It also highlights the importance of model calibration and architecture selection when designing perception systems for autonomous logistics robots.

VIII. Limitations

Although the experiments conducted in this project provide meaningful insights into the performance of different seg-

mentation and detection architectures on our dataset, several limitations should be acknowledged.

First, all models were trained and evaluated using a CPU-only environment due to hardware constraints. This restriction significantly increased training and inference times and limited the ability to run extensive hyperparameter searches, use larger batch sizes, or train higher-capacity variants of the models. As a result, the performance reported in this study may not fully reflect the models' capabilities under optimized GPU-based training conditions.

Second, EfficientDet-D0 demonstrated unusually low confidence in its predictions, which suggests that additional tuning would have been necessary to achieve competitive performance. Possible contributing factors include suboptimal hyperparameter settings, insufficient anchor priors for pallet-shaped objects, or a mismatch between the model's pretrained domain and the warehouse imagery present in EuroPalletSeg. Due to limited time and computational resources, a systematic optimization of these components could not be performed.

Third, because the dataset provides separate annotations for segmentation and detection, the two tasks were evaluated independently rather than jointly. This prevented exploration of multi-task learning approaches that might exploit cross-task information or improve the consistency between mask boundaries and detected pallet locations. Additionally, the dataset contains real-world variability such as reflections, shadows, and partial occlusions, but does not include extreme environmental conditions such as motion blur or highly cluttered scenes, which limits the generalizability of the findings.

Finally, the evaluation included a small number of qualitative visualizations and a limited number of metrics due to the runtime constraints of running ONNX inference on Colab. More comprehensive error analysis including false positive/false negative breakdowns, per-scene difficulty analysis, or calibration studies—was outside the scope of this work but remains important for future research.

These limitations highlight the need for additional experimentation and optimization to fully validate the models in more demanding warehouse or robotic settings.

IX. Conclusion

This project implemented and compared two segmentation models and two object detection models on the EuroPalletSeg dataset, with the goal of improving automation capabilities in warehouse and industrial environments. The key findings highlight clear distinctions in performance, efficiency, and suitability for real-world deployment.

For ground segmentation, U-Net achieved the highest accuracy but was also the largest model, making it less suitable for resource-constrained robotic systems. DeepLabV3+, while slightly less accurate, provided a dramatic reduction in model size, suggesting stronger potential for real-time deployment. In pallet detection, YOLOv8 substantially outperformed EfficientDet-D0 in both confidence and speed, making it the only practically viable option among the two for this application. The extremely low confidence scores observed in EfficientDet-D0 emphasize the importance of proper model calibration for safety-critical tasks.

Future work that could extend and strengthen this project includes:

- **Refining Underperforming Models:** Investigating the root causes of EfficientDet-D0's calibration and confidence issues, followed by improved training, optimization, and hyperparameter tuning to make it suitable for real-world usage.
- **Architectural Exploration:** Fine-tuning YOLOv8 for pallet-specific visual features and exploring newer detection architectures such as YOLOv9 or transformer-based detectors to further advance detection accuracy and robustness.
- **Multi-Task Learning (MTL):** Developing a unified model capable of performing both ground segmentation and pallet detection simultaneously, which could reduce the computational load on autonomous mobile robots.
- **Real-World Deployment:** Evaluating the models on embedded hardware platforms to validate latency, throughput, and accuracy under real-world operational constraints.

Overall, this study provides actionable insights into model selection for warehouse robotics and establishes a foundation for future advancements in efficient, reliable perception for automated pallet handling systems.

References

[1] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," *CVPR*, 2015.

[2] H. Zhao *et al.*, "Pyramid scene parsing network," *CVPR*, 2017.

[3] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking atrous convolution for semantic image segmentation," *arXiv preprint arXiv:1706.05587*, 2017.

[4] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," *ECCV*, 2018.

[5] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," *MICCAI*, 2015.

[6] A. Dosovitskiy *et al.*, "An image is worth 16x16 words: Vision transformer," in *ICLR*, 2021.

[7] K. He *et al.*, "Mask r-cnn," *ICCV*, 2017.

[8] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," *CVPR*, 2016.

[9] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv:1804.02767*, 2018.

[10] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "Yolov4: Optimal speed and accuracy of object detection," *arXiv:2004.10934*, 2020.

[11] G. Jocher, "Yolov8: Architecture and models," *Ultralytics Research*, 2023.

[12] T.-Y. Lin *et al.*, "Focal loss for dense object detection," in *ICCV*, 2017.

[13] M. Tan, R. Pang, and Q. V. Le, "Efficientdet: Scalable and efficient object detection," *CVPR*, 2020.

[14] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *ICML*, 2019.

[15] A. Howard *et al.*, "Searching for mobilenetv3," *ICCV*, 2019.

[16] M. Everingham *et al.*, "The pascal visual object classes challenge," *IJCV*, 2010.

[17] T.-Y. Lin *et al.*, "Microsoft coco: Common objects in context," in *ECCV*, 2014.

[18] A. Newell *et al.*, "Stacked hourglass networks for human pose estimation," in *ECCV*, 2016.

[19] P. Li *et al.*, "A deep learning-based approach for automated pallet detection in warehouse environments," *IEEE Access*, 2019.

[20] Z. Yuan *et al.*, "Vision-based pallet recognition for autonomous forklifts," in *IEEE ICRA*, 2021.