Comparative Evaluation of Segmentation and Object Detection Models for Pallet Recognition

Aksheya Kannan Subramanian
Michael Acquah
Faten Alshohatee
CIS 583 – Deep Learning

Each slide in this presentation has an accompanying audio recording. Feel free to play them manually or view the presentation in SlideShow mode for the full experience.

Motivation

AUTONOMOUS WAREHOUSES REQUIRE RELIABLE PERCEPTION FOR SAFE ROBOT NAVIGATION

TWO KEY TASKS: **GROUND SEGMENTATION** AND **PALLET DETECTION**

TRADITIONAL CV STRUGGLES WITH LIGHTING, SHADOWS, REFLECTIONS

DEEP LEARNING PROVIDES STRONGER ROBUSTNESS FOR REAL INDUSTRIAL ENVIRONMENTS

GOAL: COMPARE SEGMENTATION AND DETECTION MODELS FOR WAREHOUSE DEPLOYMENT

Research Questions

How do **U-Net** and **DeepLabV3+** compare for warehouse ground segmentation?

How do **YOLOv8** and **EfficientDet-D0** differ in pallet detection performance?

What are the key trade-offs in **accuracy**, **latency/FPS**, **confidence calibration**, and **model size** for real-time deployment?

Dataset: EuroPalletSeg

- Real indoor warehouse scenes with pallets, shelving, and concrete flooring
- Challenging visual conditions: **shadows**, **reflections**, **partial occlusions**
- **Ground segmentation masks**
- **YOLO-format pallet bounding boxes**
- Separate train/validation splits with **585 validation images** for each task

Dataset Creation

How the Ground Masks Were Created

→ Generated using Meta AI's **Segment Anything Model (SAM)**

→ Manually drew bounding boxes to guide SAM

→ Inspected and verified segmentation previews for accuracy

→ Refined or created additional masks for ~2000 images using **LabelMe**

Final Output

→ A **golden-quality, high-fidelity ground segmentation dataset**

→ Fully aligned with the original images and optimized for warehouse perception tasks

Preprocessing

Segmentation Pipeline

- Resize images to **256×256**
- Apply model-specific normalization
  - U-Net → scale to **[0,1]**
  - DeepLabV3+ → **ImageNet mean/std**
- Resize masks with **nearest-neighbor** interpolation
- Apply binary **thresholding (>127)** to clean the mask

Detection Pipeline

- Resize images to **512×512**
- Convert YOLO labels from **normalized → pixel coordinates**
- Use absolute coordinates for IoU + evaluation consistency

EfficientDet Postprocessing

- EfficientDet-D0 outputs many raw predictions
- Only the **top 5 highest-confidence** boxes per image were kept for comparison + visualization

Models Evaluated

Segmentation Models

U-Net — 209 MB

Encoder–decoder with skip connections; high spatial precision

DeepLabV3+ (MobileNetV3) — 18 MB

Atrous Spatial Pyramid Pooling; lightweight + efficient

Detection Models

YOLOv8 — 42 MB

One-stage detector with fast, confident predictions

EfficientDet-D0 — 15 MB

Compact detector using EfficientNet + BiFPN

U-Net Overview

Architecture

- Classical **encoder–decoder** structure
- Uses **skip connections** to recover spatial detail lost during downsampling

Strengths

- Produces **sharp, accurate segmentation masks**
- Well-suited for tasks requiring precise boundary reconstruction
- **Large model size (209 MB)** → less efficient for embedded or real-time deployment

DeepLabV3+ Overview

Architecture

- **Atrous Spatial Pyramid Pooling (ASPP)** for multi-scale context
- **MobileNetV3 encoder** for efficient feature extraction
- Decoder refines segmentation boundaries

Strengths

- **Lightweight (18 MB)** and fast
- Captures both local detail and global context
- Well-suited for **resource-constrained** or real-time robotics

YOLOv8 Overview

Architecture

- **One-stage detector** with a modern C2f backbone
- Decoupled detection head improves box + class predictions

Strengths

- **Fast inference** and low latency on CPU
- **High confidence scores** and stable predictions
- **Robust pallet detection**, even under reflections and occlusions

EfficientDet-D0 Overview

Architecture

- **EfficientNet-B0 backbone** for compact, high-quality feature extraction
- **BiFPN** (Bidirectional Feature Pyramid Network) for efficient multi-scale fusion
- Designed with compound scaling for balanced depth, width, and resolution

Strengths

- **Very small model size** (~15 MB)
- Highly **efficient and lightweight**, ideal for low-power or embedded devices
- Optimized for speed while maintaining reasonable accuracy on general object detection tasks

Evaluation Metrics

Segmentation Metrics

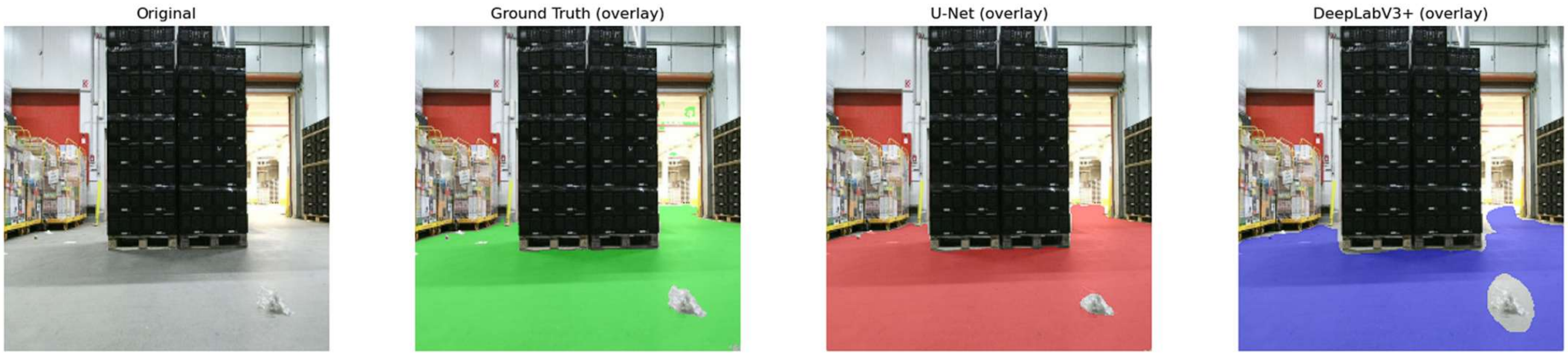
- **IoU (Intersection over Union)** – overlap quality
- **Dice Coefficient** – foreground accuracy
- **Pixel Accuracy** – overall correctness
- **Precision / Recall** – false positives vs. false negatives

Detection Metrics

- **mAP@0.5** – bounding box quality
- **FPS** – inference speed
- **Latency** – per-image processing time
- **Confidence Scores** – model certainty in detections

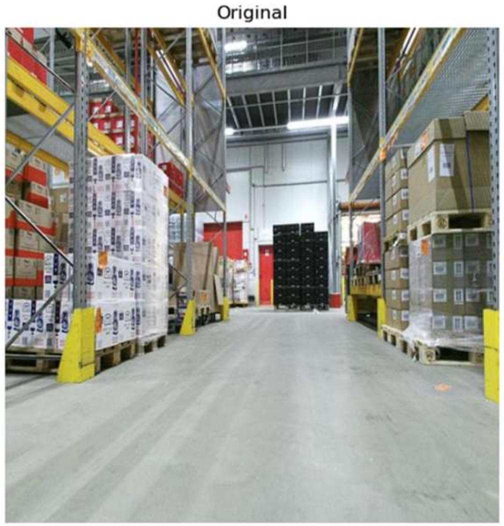
Segmentation Results

Model	IoU	Dice
U-Net	0.951	0.973
DeepLabV3+	0.921	0.973



Detection Results

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



Qualitative Results

U-Net: produces sharper, more detailed ground masks

DeepLabV3+: smooth and coherent masks, slightly less precise at boundaries

YOLOv8: consistent and reliable pallet bounding boxes

EfficientDet-D0: weak detections with very low confidence, often missing pallets

Limitations

📌 **Experimental Constraints**

CPU-only evaluation, limiting achievable speed

Restricted hyperparameter tuning due to compute constraints

📌 **Model-Specific Challenges**

EfficientDet-D0 required **confidence calibration**, which was not fully explored

📌 **Scope Limitations**

No **multi-task** or joint segmentation–detection model evaluated

Future Work

Hyperparameter tuning and calibration for **EfficientDet-D0**

Explore next-generation detectors such as **YOLOv9** or transformer-based models

Develop a **multi-task model** combining segmentation + detection in one network

Test models on **edge hardware** (e.g., Jetson, Raspberry Pi, industrial robotics hardware)

Evaluate performance after optimization (quantization, pruning)

Conclusion

U-Net produced the most accurate segmentation, while **DeepLabV3+** was nearly as effective but much more efficient.

YOLOv8 was the most practical and reliable for pallet detection, while **EfficientDet-D0** needed more tuning despite being lightweight.

Our findings suggest using **U-Net** or **DeepLabV3+** based on compute resources and deploying **YOLOv8** for reliable pallet detection.

THANK YOU!