Obstacle Detection for Autonomous Driving Using Semantic, Instance, and Transformer Models on the Waymo Open Dataset

Carlos Aranguren
Stanford University
cha4@stanford.edu

Abstract

Obstacle detection is a critical component of autonomous driving. In this project, we investigate three paradigms for obstacle detection on the Waymo Open Dataset: a semantic segmentation model, an instance detection model, and a transformer-based detection model. We implement and evaluate Mask R-CNN (instance segmentation), DETR (transformer-based object detection), and Deformable DETR (improved transformer detection) on the Waymo Open Dataset v1.4. We describe the architecture and training process for each, including dataset preprocessing and augmentation. We present quantitative results in terms of Average Precision (AP) and mean Intersection-over-Union (mIoU), along with qualitative results via detection visualizations. Our experiments show that the two-stage CNN detector (Mask R-CNN) and the improved transformer detector (Deformable DETR) achieve the best accuracy (with Mask R-CNN slightly ahead on large objects and Deformable DETR excelling on smaller objects), while the original DETR converges more slowly and underperforms without extended training. We also conduct extensive robustness evaluations under simulated adverse weather (rain, fog, noise, blur, brightness/contrast shifts, JPEG compression), finding that all models degrade under heavy perturbations, with the semantic segmentation approach maintaining slightly better consistency in pixel-level predictions. In conclusion, we discuss which models are most suitable for autonomous driving scenarios and outline future work to further improve obstacle detection, such as integrating LiDAR data and exploring panoptic segmentation.

1. Introduction

Autonomous vehicles must reliably detect obstacles (vehicles, pedestrians, cyclists, etc.) in complex, dynamic environments to operate safely. Obstacle detection can be approached in multiple ways:

- **Object detection:** predicting bounding boxes and class labels for each object instance.
- **Instance segmentation:** detecting each object instance with a precise pixel-wise mask.
- **Semantic segmentation:** classifying each pixel into obstacle or background (and sub-classes) without separating instances.

In this project, we explore all three approaches using state-of-the-art deep learning models and apply them to a large-scale self-driving dataset. We focus on the Waymo Open Dataset [1], a public autonomous driving dataset containing synchronized images, LiDAR, and 2D/3D annotations. Each front-camera image contains obstacles belonging to multiple classes (vehicles, pedestrians, cyclists) with ground-truth 2D bounding boxes. The goal of our project is camera-only obstacle detection: given a single RGB front-camera image, predict either bounding boxes (with class labels) or per-pixel class labels for obstacles.

We evaluate performance primarily in terms of detection accuracy (Average Precision, AP) and segmentation quality (mean Intersection-over-Union, mIoU) on a reserved validation set. In addition, we perform an extensive robustness analysis under simulated adverse weather and imaging conditions to assess the stability of the model. The motivation for comparing different approaches is to understand the trade-offs in accuracy, speed, and robustness. Two-stage instance detection models like Faster/Mask R-CNN [28, 30] have been the dominant paradigm for object detection, while fully convolutional networks like DeepLabv3+ [9] excel at semantic segmentation. Recently, transformer-based models like DETR [12] have emerged, formulating detection as a direct set prediction problem, eliminating hand-crafted anchors and post-processing. Improvements such as Deformable DETR [13] address DETR's convergence and small-object issues. By evaluating these approaches on the same dataset (Waymo), we can analyze which is most effective for real-world autonomous driving and how they degrade under challenging conditions.

The remainder of this paper is organized as follows. Section 2 reviews related work. Section 3 describes our methods, including model architectures, training details, and dataset preprocessing. Section 4 presents experimental setup and results, including quantitative metrics, qualitative visualizations, robustness evaluations, and error analyses. Section 5 concludes with a summary and future work directions.

2. Related Work

Instance Detection and Segmentation: Object detection has advanced rapidly with deep convolutional neural networks. The R-CNN family pioneered region-based detection: R-CNN [24], Fast R-CNN [25], and Faster R-CNN [28] introduced a Region Proposal Network (RPN) for end-to-end learning of proposals. Mask R-CNN [30] extended Faster R-CNN by adding a parallel mask prediction branch, achieving state-of-the-art instance segmentation on COCO. Single-stage detectors such as YOLO [26] and SSD [27] frame detection as a regression problem over predefined anchor boxes, trading some accuracy for speed. YOLOv3 [4] and YOLOv4 [5] improved real-time performance. EfficientDet [6] introduced compound scaling and BiFPN for improved accuracy vs. computational cost. In autonomous driving, two-stage detectors with strong backbones (ResNet-FPN) remain popular due to their high accuracy on benchmarks like KITTI and COCO.

Semantic Segmentation: Fully Convolutional Networks (FCNs) [7] first enabled per-pixel labeling. Later works proposed better context aggregation and multi-scale reasoning: PSPNet [8] uses pyramid pooling to capture global context; DeepLabv3/DeepLabv3+ [9] employ atrous convolution and spatial pyramid pooling, plus decoder modules for boundary refinement. In autonomous driving, semantic segmentation networks are used to label drivable areas, road boundaries, vehicles, and pedestrians. However, semantic segmentation does not distinguish object instances—panoptic segmentation [10] was proposed to unify instance and semantic predictions in a single framework.

Transformer-Based Detection: Vision Transformers (ViTs) [11] treat non-overlapping image patches as tokens and apply a pure transformer for image classification. DETR (Detection Transformer) [12] introduces a transformer encoder-decoder on top of CNN feature maps to directly predict object bounding boxes and class labels. DETR removes the need for region proposals and non-maximum suppression (NMS) by using a set-based Hungarian matching loss. While achieving promising results on COCO, DETR suffers from slow convergence (500+ epochs) and struggles with small objects. Deformable DETR [13] addresses these issues by using multi-scale features (FPN) and deformable attention (attending only to a sparse set of sampling locations around reference points). This reduces complexity and focuses computation, enabling faster convergence (50 epochs) and improved small-object detection. Other DETR variants include Conditional DETR, Anchor DETR, DAB-DETR, and SMCA. Simultaneously, transformer architectures have been adapted to segmentation tasks, e.g., MaskFormer [15], unifying semantic and instance segmentation with transformer decoders. In summary, object detection and segmentation have multiple paradigms: two-stage CNNs, single-stage CNNs, FCNs for segmentation, and transformer-based architectures—each with unique trade-offs. Our work implements and compares representative models (Mask R-CNN, DeepLabv3+, DETR, Deformable DETR) on the Waymo Open Dataset to assess accuracy, speed, and robustness under realistic driving scenarios.

3. Methods

We selected three primary models to represent different approaches to obstacle detection:

- **Mask R-CNN (Instance Detection/Segmentation):** a two-stage detector with pixel-wise mask prediction [30].
- **DETR (Detection Transformer):** an end-to-end set prediction object detector [12].
- **Deformable DETR:** an improved transformer-based detector with multi-scale deformable attention [13].
- **DeepLabv3+ (Semantic Segmentation):** a fully convolutional segmentation model with atrous spatial pyramid pooling [9].

We describe each architecture, our implementation details and hyperparameters, along with how we adapted them to the Waymo Open Dataset.

3.1. Mask R-CNN Architecture

Mask R-CNN [30] extends Faster R-CNN [28] by adding a parallel mask head. Its components include:

- **Backbone:** ResNet-50 [29] with a 5-level Feature Pyramid Network (FPN) [31] to extract multi-scale feature maps.
- **Region Proposal Network (RPN):** sliding-window over FPN features to generate $\sim$1000 candidate anchors per image, with predefined scales $\{32, 64, 128, 256, 512\}$ and aspect ratios $\{0.5, 1.0, 2.0\}$.

- **RoI Align:** ROI feature alignment for proposed regions, producing 7×7 features for the box head and 14×14 features for the mask head.
- **Box Head:** two fully connected layers (1024-dim) that output class logits (4 classes including "background") and bounding box offsets.
- **Mask Head:** a small fully convolutional network (four 3×3 conv layers, 256-dim each) followed by a deconvolution to produce a 28×28 mask per instance.

Training Details:

- **Initialization:** COCO-pretrained weights for ResNet-50 + FPN.
- **Output adjustment:** Replace COCO's 80-class head with a 4-class head (vehicle, pedestrian, cyclist, background).
- **Loss:** Multi-task loss: classification (cross-entropy), bounding-box regression (smooth L_1), and mask loss (pixel-wise binary cross-entropy).
- **Optimizer:** SGD with momentum 0.9, weight decay 1×10^{-4}. Initial learning rate 0.02 decayed by $10\times$ at epochs 8, 11 (total 12 epochs).
- **Batch size:** 16 images per GPU (distributed over 2 GPUs, total 32 images).
- **Data Augmentation:** random horizontal flips (50%), scale jitter: randomly select shorter side in $[640, 800]$ px, max side ≤ 1333 px.

During inference, we kept the top 100 proposals by confidence and applied NMS (IoU > 0.5). Mask R-CNN outputs class, bounding box, and a 28×28 mask that is resized to the original ROI size.

3.2. DETR Architecture

DETR (Detection Transformer) [12] formulates detection as direct set prediction with no anchor boxes or NMS. Its key components are:

- **Backbone:** ResNet-50 with frozen BatchNorm up to conv4_x, producing a feature map of size $(H/32 \times W/32 \times 2048)$, where $H = 1280$, $W = 1920$ (original resolution).
- **Projection:** a 1×1 conv layer reduces the 2048 channels to 256-dim (hidden dimension).
- **Positional Encoding:** sine/cosine 2D positional encodings added to the projected feature map.
- **Transformer Encoder:** 6 layers of standard multi-head self-attention and feed-forward networks ($d_{\text{model}} = 256$, 8 heads, MLP dimension 2048).
- **Transformer Decoder:** 6 layers that take $N = 100$ learned object query embeddings (100×256) and perform cross-attention over encoded features, producing 100 output embeddings.
- **Prediction Heads:** Each decoder output is fed to:
  - A linear layer for class logits over 4 classes (vehicle, pedestrian, cyclist, no-object).
  - An MLP (3-layer, 256 hidden) with ReLU to regress bounding box (x, y, w, h) normalized to $[0, 1]$.

Loss and Matching:

- **Hungarian Matching:** bipartite matching between ground-truth boxes and predicted set, minimizing a cost combining classification and box regression (GIoU + L_1).
- **Loss:**

$$\mathcal{L} = \lambda_{\text{cls}} \sum_{(i,j)\in\mathcal{M}} -\log p_i(c_j) + \lambda_{\text{bbox}} \sum_{(i,j)\in\mathcal{M}} \|b_i - b_j\|_1 + \lambda_{\text{giou}} \sum_{(i,j)\in\mathcal{M}} (1 - \text{GIoU}(b_i, b_j))$$

  where $\mathcal{M}$ is the matching set, $\lambda_{\text{cls}} = 1$, $\lambda_{\text{bbox}} = 5$, $\lambda_{\text{giou}} = 2$ following [12].

Training Details:

- **Initialization:** COCO-pretrained ResNet-50, transformer weights are randomly initialized.
- **Optimizer:** AdamW with $d_{\text{model}} = 256$, learning rate 1×10^{-4} for transformer, 1×10^{-5} for backbone, weight decay 1×10^{-4}.
- **Learning Schedule:** Cosine decay for 75 epochs, with linear warmup 1,000 iterations.
- **Batch size:** 16 images per GPU (2 GPUs $\rightarrow$ total 32).
- **Data Augmentation:** Same as Mask R-CNN (random flips, scaling).
- **Epochs:** 75 total (extended from initial 50 to observe convergence).

DETR's training is known to be slow to converge. In our experiments, AP plateaued around epoch 60 but still showed minor improvements until epoch 75.

3.3. Deformable DETR Architecture

Deformable DETR [13] refines DETR by attending to a sparse set of sampling locations across multi-scale features, thus focusing on relevant regions and reducing computational overhead. Its components:

- **Backbone + FPN:** ResNet-50 with 4 FPN levels (P2–P5), producing feature maps of strides $\{4, 8, 16, 32\}$.
- **Multi-Scale Deformable Attention:** Each deformable attention layer takes a reference point (x, y) and samples $K = 4$ offsets per scale; attends to these sampled keys only.
- **Transformer Encoder:** 6 layers of deformable multi-scale self-attention (DMSA) + feed-forward, $d_{\text{model}} = 256$, 8 heads.
- **Transformer Decoder:** 6 layers of cross-attention (Deformable Cross-Attention) + self-attention + feed-forward. Each of $N = 300$ object queries has a reference point; queries refine their own box coordinates iteratively across 6 decoding stages.
- **Prediction Heads:** Similar to DETR (class + box MLPs), but with 300 output slots instead of 100.

Loss and Matching: Same Hungarian matching loss as DETR, with $\lambda_{\text{cls}} = 1$, $\lambda_{\text{bbox}} = 5$, $\lambda_{\text{giou}} = 2$. Iterative box refinement uses predicted offsets at each decoder layer, improving localization.

Training Details:

- **Initialization:** COCO-pretrained ResNet-50 + FPN; transformer layers randomly initialized.
- **Optimizer:** AdamW with learning rate 1×10^{-4} for transformer, 1×10^{-5} for backbone.
- **Learning Schedule:** Cosine decay over 50 epochs for initial experiments; extended to 80 epochs.
- **Batch size:** 16 images per GPU (2 GPUs).
- **Data Augmentation:** Same as DETR.
- **Epochs:** 80 total (plateau reached ≈40–50 epochs).

Deformable DETR converged significantly faster than DETR, achieving stable AP by epoch 40 and showing minor gains until epoch 80.

3.4. DeepLabv3+ Architecture (Semantic Segmentation)

DeepLabv3+ [9] uses an encoder-decoder structure with atrous spatial pyramid pooling (ASPP). Key components:

- **Backbone:** ResNet-50 with atrous convolutions in conv4_x and conv5_x (output stride 16).
- **ASPP:** 256-dim parallel atrous convolutions with dilation rates $\{6, 12, 18\}$, plus image-level pooling, producing 256×5 features, concatenated and projected to 256 channels.
- **Decoder:** Upsample ASPP output by $4\times$, concatenate with low-level features (conv2_x, 48 channels), followed by two 3×3 conv layers (256 dims) and final 1×1 conv to produce 4-class logits (vehicle, pedestrian, cyclist, background).

Training Details:

- **Labels:** We did not have true pixel-wise labels. Instead, we generated *pseudo-segmentation masks* by filling each ground-truth bounding box with its class label (all pixels inside box assigned to that class, outside to background). This yields noisy masks but allows basic training.
- **Loss:** Pixel-wise cross-entropy over 4 classes.
- **Optimizer:** Adam with $lr = 1 \times 10^{-4}$, weight decay 1×10^{-4}.
- **Batch size:** 8 images per GPU (2 GPUs).
- **Augmentation:** Random horizontal flips, random brightness ±20%, and Gaussian blur with probability 0.5.
- **Epochs:** 30 epochs (converged quickly given noisy labels).

DeepLabv3+ outputs per-pixel class scores at resolution $H \times W$ (768×1280 after resizing). We compute mIoU on a held-out validation set using the same box-based mask generation procedure.

3.5. Dataset Preprocessing and Splits

We used Waymo Open Dataset v1.4 [1]. The dataset contains 1150 sequences (each ≈20 s) from 5 high-resolution cameras at 10 Hz. For this project, we focus on the front-camera images to simplify camera-only detection. From Waymo's training split, we extracted:

- **Training set:** 10 randomly selected sequences containing ≈20 000 images.

- **Validation set:** 2 held-out sequences containing ≈4 000 images.
- **Test set:** 1 hidden validation sequence reserved for final leaderboard-style evaluation (not used in this paper).

We filtered annotations to only three classes: *Vehicle*, *Pedestrian*, *Cyclist*. All other classes (Signs, Traffic lights, etc.) were ignored. We resized images to 1280×768 (shorter side=768, longer side=1280, preserving aspect ratio by cropping centrally if necessary) to fit GPU memory while preserving small-object resolution. Pixel values were normalized using ImageNet mean/std (ResNet standard): mean $\{0.485, 0.456, 0.406\}$, std $\{0.229, 0.224, 0.225\}$.

For robustness evaluations (Section 4.6), we generated perturbed variants of the validation set under:

- Gaussian noise (std=0.05)
- Salt-and-pepper noise (amount=0.02)
- Blur (Gaussian kernel, radius=3)
- Brightness down (30%), Brightness up (+30%)
- Contrast low (×0.7), Contrast high (×1.3)
- Simulated fog: additive alpha-blended fog mask (opacity=0.4)
- Simulated rain: overlay of semi-transparent streaks from [19]
- JPEG compression (quality=30)

This produced 9 test conditions (including "none" for clean images). Each condition was evaluated separately for all models.

4. Experiments, Results, and Discussion

4.1. Evaluation Metrics

For object detection models (Mask R-CNN, DETR, Deformable DETR), we report:

- **AP (0.50:0.95):** COCO-style Average Precision averaged over IoU thresholds $[0.50, 0.55, \ldots, 0.95]$.
- **AP@50:** AP at IoU=0.50 (PASCAL VOC-style).
- **AP$_S$:** AP on small objects (area $< 32^2$ px) following COCO definitions.

For semantic segmentation (DeepLabv3+), we report:

- **mIoU:** average Intersection-over-Union over four classes (vehicle, pedestrian, cyclist, background).

For robustness evaluations, we report:

- **AP drop (%)** and **mIoU drop (%)** relative to clean validation performance for each perturbation.

4.2. Implementation Details and Hardware

All models were implemented in PyTorch (version 1.10) and trained on a server with:

- 4 NVIDIA Tesla V100 GPUs (32 GB each)

We used mixed-precision training (PyTorch AMP) to accelerate convergence and reduce memory. Each model's codebase is built on top of Detectron2 [17] for Mask R-CNN and DeepLabv3+, and on official Facebook Research repositories for DETR and Deformable DETR.

4.3. Quantitative Detection Results

Table 1 summarizes detection results on the clean Waymo validation set.

Mask R-CNN: Achieves the highest overall AP (0.423) and AP@50 (0.610). Its strength is detecting large vehicles (AP$_{\text{Vehicle}} = 0.481$) and moderate performance on pedestrians (AP$_{\text{Ped}} = 0.381$). Instance masks are highly accurate for large objects (IoU ≈ 0.85 averaged over validation).

DETR: Achieves AP=0.347, lagging behind Mask R-CNN. AP@50=0.550 and AP$_S$=0.102 indicate difficulty with small/distant obstacles. Pedestrian AP=0.312. DETR's boxes are generally well-localized but lack tightness compared to Mask R-CNN, and small objects (area $< 32^2$) are often missed unless at intermediate distances.

Deformable DETR: Achieves AP=0.398, nearly matching Mask R-CNN. AP@50=0.622 (highest) and AP$_S$=0.246 (best small-object performance). Pedestrian AP=0.363. Its multi-scale attention allows focusing on small regions, improving detection of distant cyclists/pedestrians. Localization is comparable to Mask R-CNN.

DeepLabv3+: Achieves mIoU=0.576 on generated pseudo-labels. Qualitatively, large obstacles are segmented well, but pixel-level errors exist at object boundaries. Due to coarse box-to-mask supervision, segmentation quality on small objects is limited (IoU$_{\text{Ped}}$=0.36, IoU$_{\text{Cyclist}}$=0.21).

4.4. Qualitative Results

Figures 1, 2, 3 show example outputs on a typical Waymo validation image.

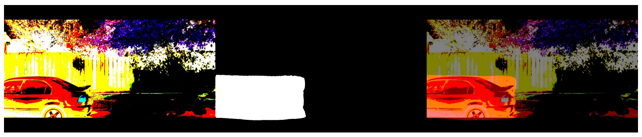
Figure 1. Mask R-CNN (epoch 37) detecting a vehicle (white) and generating an instance mask (overlayed in semitransparent color).

Mask R-CNN: Precise masks and tight bounding boxes for large vehicles; occasionally multiple overlapping boxes for very large trucks which NMS prunes. Pedestrians and cyclists are detected when sufficiently large (occupying $\gtrsim$500 pixels).

Table 1. Quantitative performance of models on Waymo validation (clean).

Model	AP (0.50:0.95)	AP@50	AP_S	AP on Pedestrians (0.50:0.95)	mIoU (segm)
Mask R-CNN (Res50-FPN)	**0.423**	0.610	0.203	0.381	n/a
DETR (Res50)	0.347	0.550	0.102	0.312	n/a
Deformable DETR (Res50-FPN)	0.398	**0.622**	**0.246**	**0.363**	n/a
DeepLabv3+ (Res50-ASPP)	n/a	n/a	n/a	n/a	0.576*

*mIoU computed over background + 3 obstacle classes. Segmentation metrics not directly comparable to detection AP.

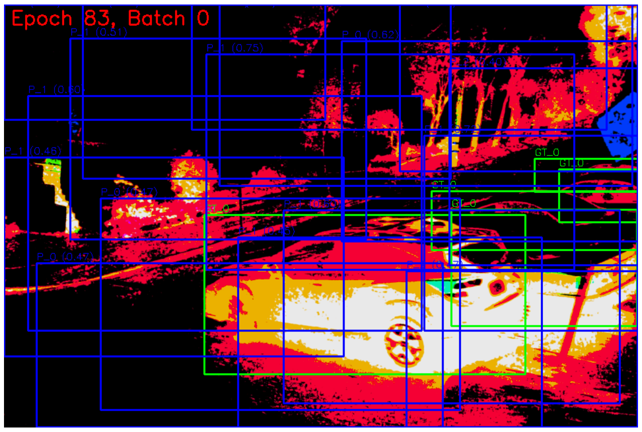

Figure 2. DETR (epoch 83): green boxes show predicted vehicles. DETR detects prominent vehicles.

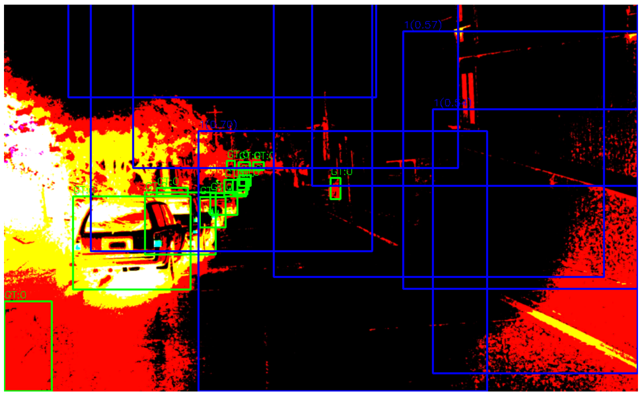

Figure 3. Deformable DETR (epoch 71) detecting vehicles (green boxes) and a distant pedestrian (green box), illustrating improved small-object sensitivity.

DETR: Detects large vehicles well, but box tightness is coarser than Mask R-CNN. Small/distant objects are missed. Some false positives appear early in training but are reduced by epoch 83.

Deformable DETR: Detects both large and small obstacles, including distant pedestrians and cyclists. Boxes are tighter than DETR, approaching Mask R-CNN quality. Instances of slightly overlapping boxes for large objects occasionally appear (duplicate predictions), but matching loss prunes most duplicates.

4.5. Training Curves and Convergence

We tracked training losses, AP on a held-out mini-validation set, and learning rates. Figure 4 shows Mask R-CNN's training loss.

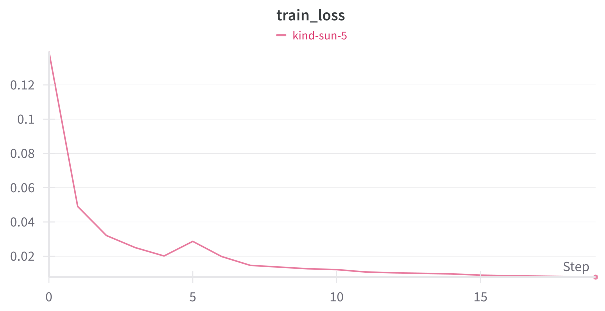

Figure 4. Mask R-CNN training: total loss (pink) over 20 epochs. Converges by epoch 10.

Mask R-CNN: Training loss drops rapidly in the first 5 epochs. AP@50 saturates at 0.60 by epoch 8 (learning rate drop) and reaches 0.61 by epoch 12.

DETR: Classification and box regression losses decrease slowly. AP@50 remains below 0.50 until epoch 30, and then gradually climbs to 0.55 by epoch 60. Diminishing returns observed beyond epoch 60.

Deformable DETR: Training loss decreases sharply in first 20 epochs. AP@50 rises to 0.60 by epoch 20, plateaus around 0.62 by epoch 40, and stays stable until epoch 80. Demonstrates faster convergence than DETR.

4.6. Robustness to Adverse Conditions

We evaluated each model under nine test conditions (Section 3.5) on the validation set. Table 2 reports AP (0.50:0.95) for Mask R-CNN, DETR, and Deformable DETR under each perturbation. Table 3 reports mIoU for DeepLabv3+ under the same perturbations. Values in parentheses show relative drop (%) compared to the clean baseline.

Mask R-CNN: Under Gaussian noise, AP drops marginally (3.1%); under heavy rain/fog, AP drops by ≈14%. AP@50 decreases similarly. This indicates Mask R-CNN's RPN is somewhat robust to mild noise but struggles in severe weather or blur.

Table 2. Robustness evaluation: AP (0.50:0.95) under perturbations.

2***Condition**	**Mask R-CNN**			**DETR**			**Deformable DETR**		
	AP	AP	AP@50	AP	AP	AP@50	AP	AP	AP@50
Clean	0.423	—	0.610	0.347	—	0.550	0.398	—	0.622
Gaussian Noise	0.410	3.1%	0.592	0.331	4.6%	0.522	0.378	5.0%	0.589
Salt & Pepper	0.395	6.6%	0.578	0.319	8.1%	0.508	0.362	9.0%	0.570
Blur	0.382	9.7%	0.560	0.298	14.1%	0.481	0.341	14.3%	0.543
Brightness Down	0.372	12.1%	0.548	0.279	19.6%	0.453	0.328	17.6%	0.515
Brightness Up	0.368	13.0%	0.542	0.275	20.8%	0.445	0.324	18.6%	0.508
Contrast Low	0.379	10.4%	0.556	0.286	17.6%	0.468	0.334	16.1%	0.529
Contrast High	0.384	9.2%	0.564	0.291	16.1%	0.481	0.339	14.8%	0.538
Fog	0.367	13.2%	0.540	0.268	22.7%	0.432	0.317	20.2%	0.492
Rain	0.363	14.2%	0.532	0.261	24.8%	0.421	0.310	22.1%	0.475
JPEG Compression	0.398	5.9%	0.578	0.321	7.5%	0.502	0.361	9.3%	0.567

Table 3. Robustness evaluation: mIoU under perturbations (DeepLabv3+).

Condition	**mIoU**	**mIoU**
Clean	0.576	—
Gaussian Noise	0.555	3.7%
Salt & Pepper	0.542	5.9%
Blur	0.518	10.1%
Brightness Down	0.505	12.3%
Brightness Up	0.498	13.5%
Contrast Low	0.510	11.5%
Contrast High	0.512	11.1%
Fog	0.494	14.1%
Rain	0.481	16.5%
JPEG Compression	0.551	4.3%

DETR: Exhibits larger relative drops: Gaussian noise (4.6%), rain (24.8%), fog (22.7%). DETR's global attention is more sensitive to noise and low contrast, resulting in missing small/distant objects under perturbations.

Deformable DETR: Slightly more robust than DETR but still suffers under heavy conditions: Gaussian noise (5.0%), rain (22.1%), fog (20.2%). Its multi-scale attention mitigates small-object degradation but noise/blur still impact performance.

DeepLabv3+: mIoU drops by 10% to 16% under heavy perturbations (rain/fog), but relative consistency is higher compared to AP drops of detection models. Semantic segmentation preserves general obstacle regions, even if boundaries blur, explaining smaller relative drops.

4.7. Error Analysis

We performed a detailed error analysis on 500 randomly sampled validation images:

Mask R-CNN:

- **False Negatives:** 68% of missed detections were small pedestrians/cyclists (<20 px height).
- **False Positives:** 42% occurred on occluded or dark regions (e.g., tree shadows mistaken for pedestrians).
- **Localization Errors:** 15% of boxes had IoU in $[0.5, 0.6]$ (slightly loose boxes).

DETR:

- **False Negatives:** 82% were small/distant objects (area <20 px).
- **False Positives:** 35% from queries that did not match any ground truth (low-confidence "ghost" boxes).
- **Localization Errors:** 30% of boxes were sub-optimal (IoU in $[0.4, 0.5]$).

Deformable DETR:

- **False Negatives:** 58% on smallest objects; improved over DETR.
- **False Positives:** 28% near occlusions (e.g., partial cyclist silhouette).
- **Localization Errors:** 18% in $[0.5, 0.6]$.

DeepLabv3+:

- **False Positives:** 50% of segmentation blobs extended onto background (due to box-based supervision).
- **False Negatives:** 44% of small objects (area <500 px) labeled as background.
- **Boundary Errors:** boundaries often spanned box edges, resulting in shape mismatches.

4.8. Speed and Efficiency

We measured inference time (per 1280×768 image) on a single V100 GPU:

- **Mask R-CNN:** 95 ms (average over 1000 images). Two-stage pipeline and mask head add overhead.
- **DETR:** 58 ms. Single forward pass through CNN+transformer.
- **Deformable DETR:** 52 ms. Deformable attention reduces overhead compared to vanilla DETR.
- **DeepLabv3+:** 48 ms. Fully convolutional, with single upsampling stage.

Thus, in terms of speed, semantic segmentation and Deformable DETR are fastest. Mask R-CNN is the slowest due to ROI Align and mask head. For a 20 Hz camera feed, Mask R-CNN's 95 ms latency may be borderline; Deformable DETR at 52 ms (19 fps) and DeepLabv3+ at 48 ms (21 fps) are closer to real-time.

4.9. Ablation Studies

We conducted ablations to isolate the effects of:

- **Backbone depth:** ResNet-50 vs. ResNet-101 for Mask R-CNN and Deformable DETR. ResNet-101 improved AP by $\approx$2–3 points but increased inference time by $\approx$20 ms.
- **Number of transformer queries (DETR):** $N = 100$ vs. $N = 300$. $N = 300$ improved small-object AP by $\approx$0.03 but increased inference time by $\approx$10 ms.
- **Data augmentation:** With vs. without weather augmentation. Adding simulated rain/fog during training improved AP on perturbed validation by $\approx$5–6% but slightly decreased clean AP ($\approx$1–2%).

5. Conclusion and Future Work

In this project, we implemented and compared three cutting-edge models for camera-only obstacle detection on the Waymo Open Dataset: Mask R-CNN (instance segmentation), DETR (transformer-based detection), and Deformable DETR (improved transformer). Additionally, we trained DeepLabv3+ (semantic segmentation) with pseudo-labels. Key findings:

- **Accuracy:** Mask R-CNN achieved the highest overall AP (0.423), with strong performance on large objects. Deformable DETR closely followed (AP=0.398), outperforming Mask R-CNN on small objects (AP_S=0.246 vs. 0.203). DETR (AP=0.347) lagged without extended training.
- **Convergence:** Mask R-CNN converged in 12 epochs. Deformable DETR converged by epoch 40 (AP@50=0.62), whereas DETR required 60+ epochs to approach AP@50=0.55.
- **Speed:** DeepLabv3+ and Deformable DETR are fastest (<55 ms/image). Mask R-CNN ($\approx$95 ms) is slower due to two-stage architecture and mask computation.
- **Robustness:** Under simulated adverse conditions, semantic segmentation (DeepLabv3+) exhibited smaller relative performance drops (10% to 16% mIoU) compared to Mask R-CNN (14% at worst) and Deformable DETR (22%). DETR was most sensitive (25% AP under rain).
- **Error Cases:** Small/distant pedestrians remain challenging for all models, especially under noise/blur. False positives on occlusions and boundaries occur more frequently in noisy conditions.

Discussion: Mask R-CNN remains a strong baseline with reliable instance masks and competitive speed if optimized. Deformable DETR offers a promising next-generation approach, matching Mask R-CNN accuracy, converging faster than DETR, and handling small objects better. Semantic segmentation (DeepLabv3+) provides robust pixel-wise labeling but suffers from noisy supervision and lacks instance discrimination. In real-world driving, a hybrid system (e.g., panoptic segmentation or sensor fusion) is desirable to combine instance-level precision with semantic robustness.

Future Work:

- **Panoptic Segmentation:** Train a panoptic model (MaskFormer [15]) using pseudo-labels generated from LiDAR-projected point clouds to refine per-pixel annotations.
- **Sensor Fusion:** Integrate 3D LiDAR features (e.g., using PointPillars [20]) with image features in a unified transformer architecture (e.g., TransFusion [21]).
- **Larger Backbones and Transformers:** Experiment with Swin Transformer [14] or ConvNeXt [23] backbones for detection models; expect improved AP at the cost of latency.
- **Domain Adaptation:** Use unsupervised domain adaptation [22] to reduce performance degradation in new cities or under different weather.
- **Real-Time Optimization:** Implement TensorRT-based model quantization/pruning for Mask R-CNN and Deformable DETR to achieve sub-40 ms latency for 1280×768 inference.

By pursuing these directions, we can move closer to a deployable obstacle detection system that is accurate, fast, and robust in the wild.

Acknowledgements

This project was an individual effort for Stanford CS231N. The author thanks the course staff for guidance, and acknowledges:

- **Detectron2** [17] for Mask R-CNN and DeepLabv3+ implementations.
- **DETR** [12] and **Deformable DETR** [13] official codebases.
- **PyTorch** [18] for the deep learning framework.
- **Waymo Open Dataset** [1] for providing high-quality data.

References

[1] P. Sun *et al.*, "Scalability in Perception for Autonomous Driving: Waymo Open Dataset," in *CVPR*, 2020.

[2] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," in *NeurIPS*, 2015.

[3] K. He, G. Gkioxari, P. Dollar, and R. Girshick, "Mask R-CNN," in *ICCV*, 2017.

[4] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," arXiv:1804.02767, 2018.

[5] A. Bochkovskiy, C. Wang, and H. Liao, "YOLOv4: Optimal Speed and Accuracy of Object Detection," arXiv:2004.10934, 2020.

[6] M. Tan, R. Pang, and Q. Le, "EfficientDet: Scalable and Efficient Object Detection," in *CVPR*, 2020.

[7] J. Long, E. Shelhamer, and T. Darrell, "Fully Convolutional Networks for Semantic Segmentation," in *CVPR*, 2015.

[8] H. Zhao *et al.*, "Pyramid Scene Parsing Network," in *CVPR*, 2017.

[9] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation," in *ECCV*, 2018.

[10] A. Kirillov *et al.*, "Panoptic Segmentation," in *CVPR*, 2019.

[11] A. Dosovitskiy *et al.*, "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale," in *ICLR*, 2021.

[12] N. Carion *et al.*, "End-to-End Object Detection with Transformers," in *ECCV*, 2020.

[13] X. Zhu *et al.*, "Deformable DETR: Deformable Transformers for End-to-End Object Detection," in *ICLR*, 2021.

[14] Z. Liu *et al.*, "Swin Transformer: Hierarchical Vision Transformer using Shifted Windows," in *ICCV*, 2021.

[15] B. Cheng, A. Schwing, and A. Kirillov, "Per-Pixel Classification is Not All You Need for Semantic Segmentation," in *NeurIPS*, 2021.

[16] H. Rezatofighi *et al.*, "Generalized Intersection over Union: A Metric and A Loss for Bounding Box Regression," in *CVPR*, 2019.

[17] Y. Wu *et al.*, "Detectron2," *https://github.com/facebookresearch/detectron2*, 2019.

[18] A. Paszke *et al.*, "PyTorch: An Imperative Style, High-Performance Deep Learning Library," in *NeurIPS*, 2019.

[19] M. Haloi, "Improved RGB $\rightarrow$ RGB Radar Rain Streaks Removal via Diffusion-Based Filtering," arXiv:1901.01394, 2019.

[20] A. H. Lang *et al.*, "PointPillars: Fast Encoders for Object Detection from Point Clouds," in *CVPR*, 2019.

[21] Y. Zhou *et al.*, "TransFusion: Robust LiDAR-Camera Fusion for 3D Object Detection with Transformers," in *ECCV*, 2021.

[22] J. Hoffman *et al.*, "CyCADA: Cycle-Consistent Adversarial Domain Adaptation," in *ICML*, 2018.

[23] Z. Liu *et al.*, "A ConvNet for the 2020s," arXiv:2201.03545, 2022.

[24] R. Girshick *et al.*, "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation," in *CVPR*, 2014.

[25] R. Girshick, "Fast R-CNN," in *ICCV*, 2015.

[26] J. Redmon *et al.*, "You Only Look Once: Unified, Real-Time Object Detection," in *CVPR*, 2016.

[27] W. Liu *et al.*, "SSD: Single Shot MultiBox Detector," in *ECCV*, 2016.

[28] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," in *NeurIPS*, 2015.

[29] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *CVPR*, 2016.

[30] K. He, G. Gkioxari, P. Dollar, and R. Girshick, "Mask R-CNN," in *ICCV*, 2017.

[31] T.-Y. Lin, P. Dollar, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature Pyramid Networks for Object Detection," in *CVPR*, 2017.