COL780 A3:- Foggy Cityscapes Object Detection

Yash Bansal (2022CS51133)

Contents

1 Introduction

The Foggy Cityscapes dataset is a synthetic extension of the Cityscapes dataset, designed to evaluate object detection performance under adverse weather conditions, specifically fog. It consists of around 3475 finely annotated images with dense fog, generated by simulating fog effects on the original Cityscapes images. Each image includes pixel-level annotations for various urban scene objects such as cars, pedestrians, trains, and more. The primary task is object detection—identifying and localizing objects in challenging, low-visibility environments.

Figure 1: Sample images from the Foggy Cityscapes dataset

For our study, we perform extensive ablation experiments and comparative evaluations using several state-of-the-art object detection models. The primary model investigated is **Deformable DETR**, for which we conduct targeted ablations, including full model training, encoder-only training, decoder-only training, and inference-time evaluations. We also explore **zero-shot object grounding** using **Grounding DINO**, further experimenting with **soft prompt tuning** techniques to adapt it to the foggy domain.

For competitive benchmarking, we perform hyperparameter optimization and evaluate additional models such as **Grounding DINO Tiny**, **DETR**, **DAB-DETR**, and **Faster R-CNN**.

All models are implemented using **Python** with the **PyTorch** framework, and many leverage pre-trained weights and utilities from **Hugging Face's Transformers** library. The goal is to assess robustness, generalization, and fine-tuning strategies for object detection under degraded visual conditions.

The trained models are saved in this link.

Trained Models Link

2 Deformable DETR

2.1 Inference on Pretrained Model

This section presents the inference results obtained using Hugging Face's SenseTime Deformable DETR model with pretrained weights.

2.1.1 Quantitative Analysis

The model was evaluated on both the training and validation datasets. The mean Average Precision (mAP) values observed are:

- **Training mAP**: 0.117
- **Validation mAP**: 0.126

The detailed analysis below is based on the evaluation metrics computed on the validation dataset using the COCO evaluation function provided by `pycocotools`.

- **mAP and Recall at Different IoU Thresholds**:
  - **AP@[IoU=0.50:0.95]**: 0.159
  - **AP@0.50**: 0.261
  - **AP@0.75**: 0.165
  - **AR@[IoU=0.50:0.95]**: 0.270

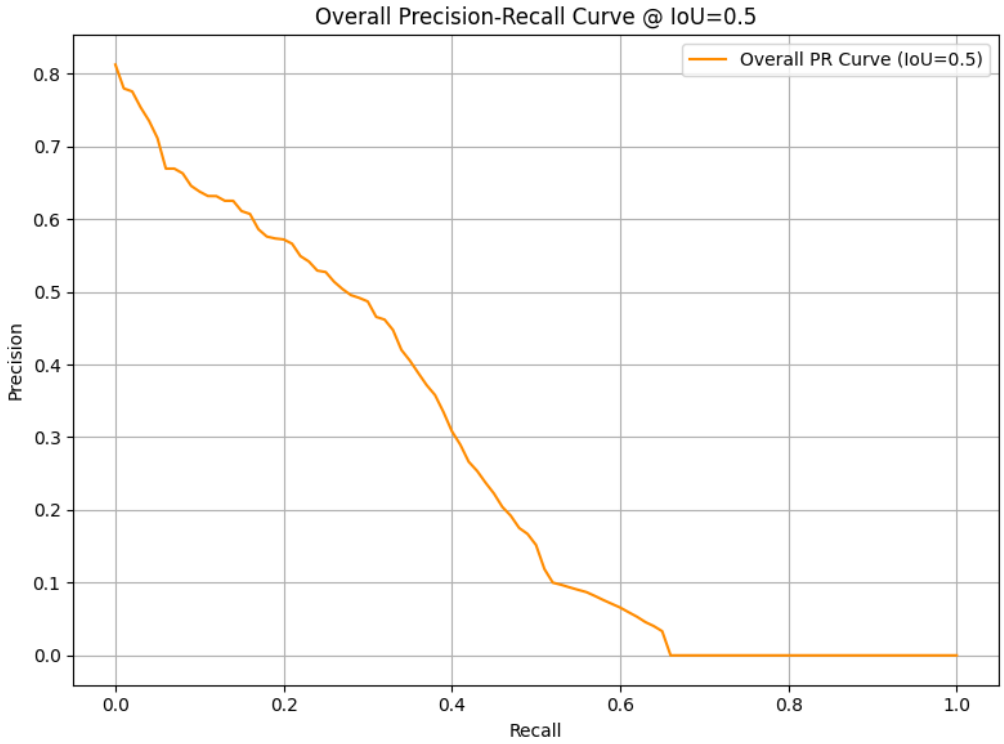


Figure 2: Precision-Recall (PR) curve at IoU threshold 0.5

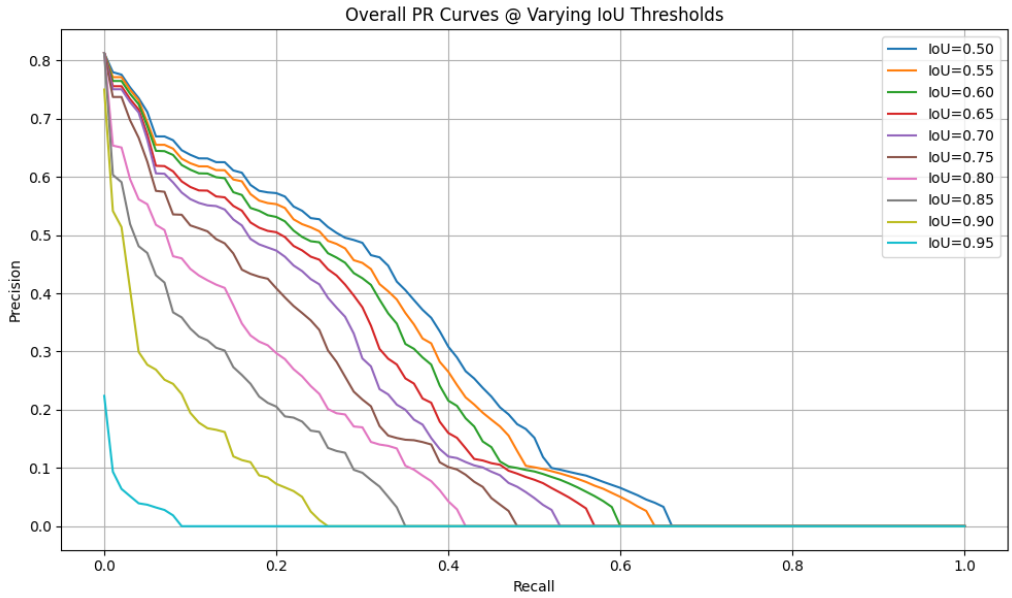


Figure 3: PR curves at different IoU thresholds

As seen in the plots, precision and recall decrease with increasing IoU thresholds. This trend occurs because stricter IoU thresholds demand tighter alignment between predicted and ground truth bounding boxes. Hence, fewer detections qualify as true positives under higher thresholds.

- **Per-Class mAP and Detection Performance**

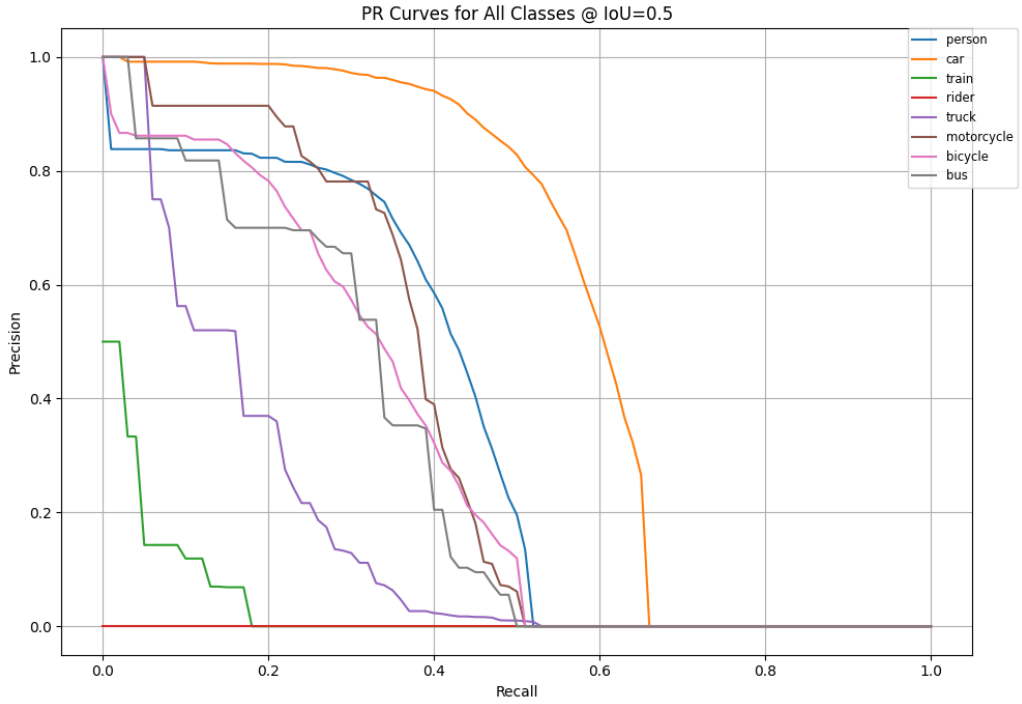


Figure 4: PR curves for each class at IoU thresholds 0.5 and 0.75

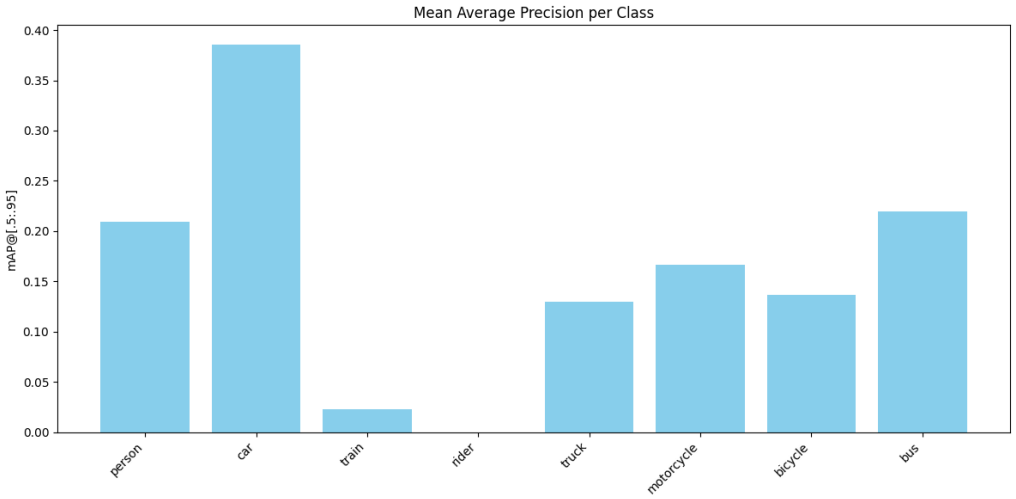


Figure 5: mAP values per class at varying IoU thresholds

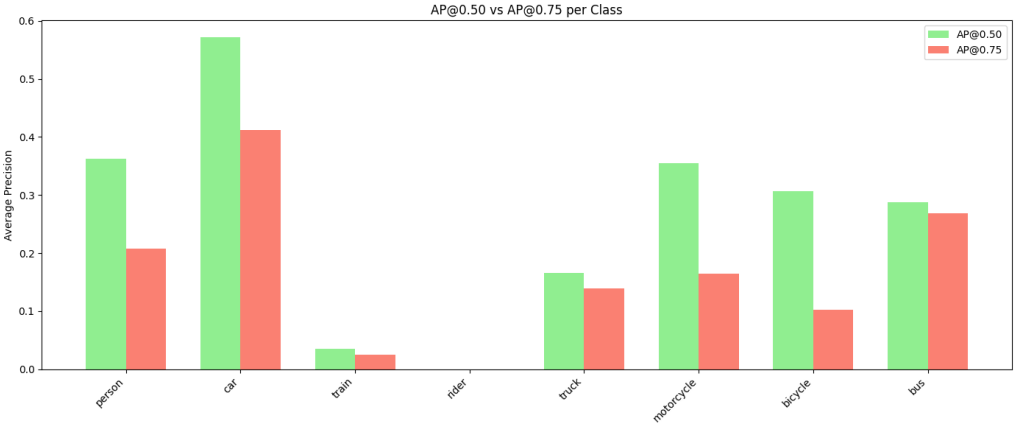


Figure 6: Comparison of AP@0.50 and AP@0.75 for each class

From the plots, it's evident that the model fails to detect the *rider* class effectively. This is likely because the pretrained Deformable DETR model was not initially trained to recognize the *rider* class, as it is absent from the standard COCO dataset. Detection performance for the *train* class is also poor, suggesting insufficient representation or visual variance in the training data.

- **Average Precision by Object Size**

  The model performs significantly better on large objects compared to small ones. This is expected, as small objects—often located far from the camera—tend to appear blurry, especially in challenging weather conditions such as fog. Consequently, the model struggles to detect these distant, low-contrast features.

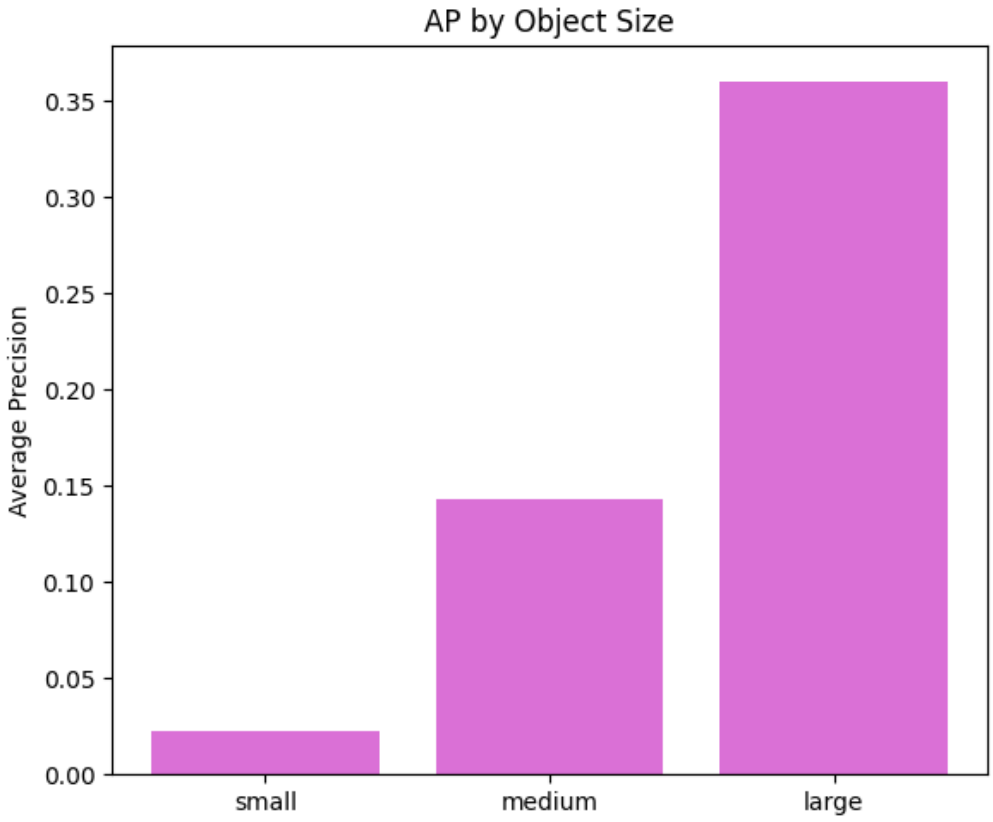


Figure 7: Average Precision for small, medium, and large objects

- **Average Precision by Maximum Detections**

  As seen from the plot, average recall improves as the maximum number of detections per image increases. Allowing more detections enables the model to identify additional objects that might otherwise be missed due to suppression or thresholding constraints. However, beyond a certain point, the gain in performance diminishes, indicating a trade-off between detection quality and computational efficiency.

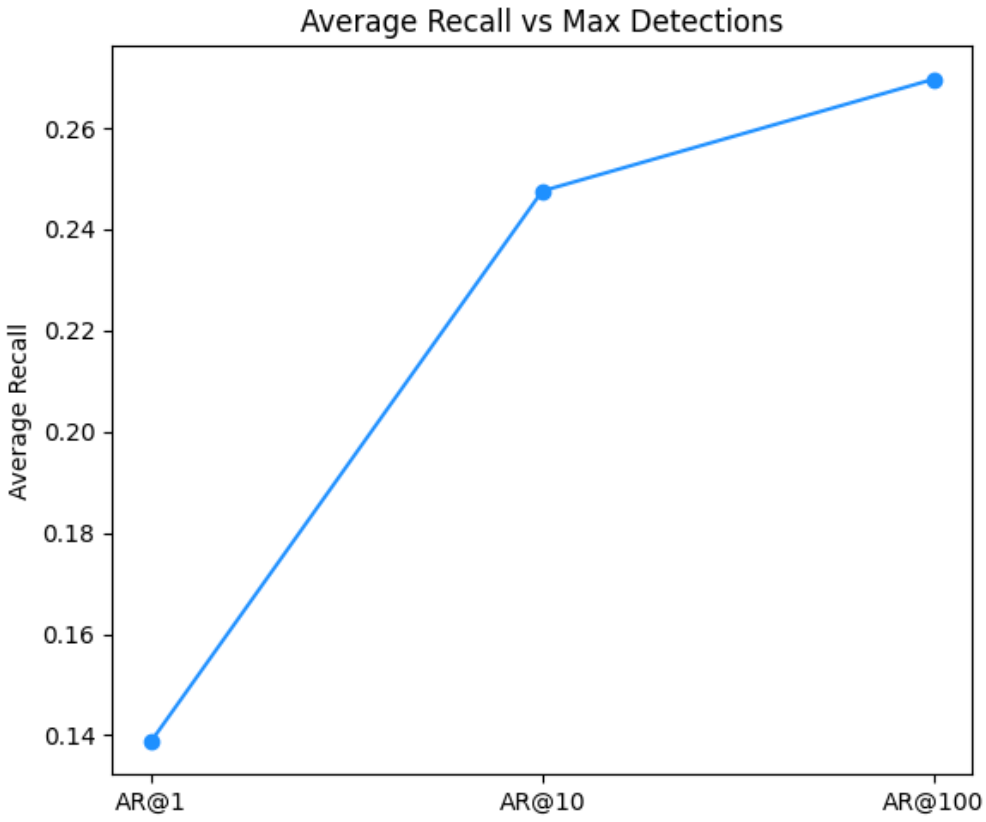


Figure 8: Average Recall versus Maximum Detections Allowed

Overall, the inference results suggest that the pretrained Deformable DETR model demonstrates modest performance out-of-the-box on this dataset. The mAP scores are relatively low compared to typical benchmarks, particularly for small and uncommon object categories. However, the results are consistent with expectations given that the model has not been fine-tuned on this specific dataset and was originally trained on the COCO dataset.

Notably:

- The model performs better on larger objects, which are more prominent and easier to detect.
- Certain classes such as *rider* and *train* are poorly detected due to their absence or underrepresentation in the COCO dataset.
- As the IoU threshold increases, the model's precision and recall drop, indicating less accurate bounding box localization.
- Increasing the number of allowed detections improves recall, but the benefit plateaus, reflecting a trade-off between prediction count and detection quality.

These findings indicate that while the pretrained Deformable DETR model serves as a strong baseline, its performance on this custom dataset can be substantially improved through fine-tuning. In particular, fine-tuning will enable the model to better adapt to the domain-specific distribution of objects, improve detection of smaller objects, and learn to recognize classes such as *rider*, which were not part of the original training dataset. These improvements and the associated training procedures are discussed in detail in the following sections.

2.1.2 Qualitative analysis

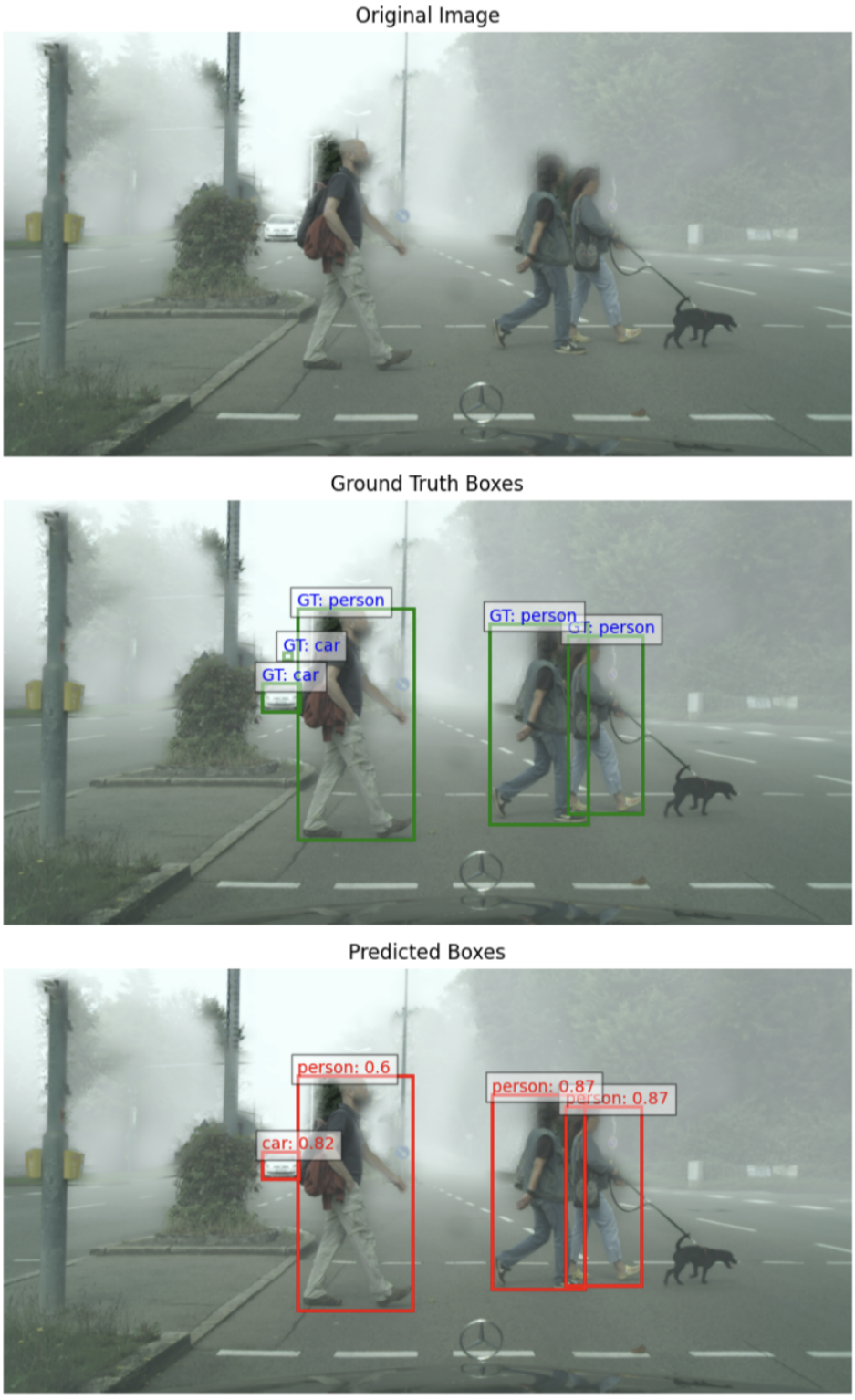


Figure 9: Qualitative visualization of deformable detr performance

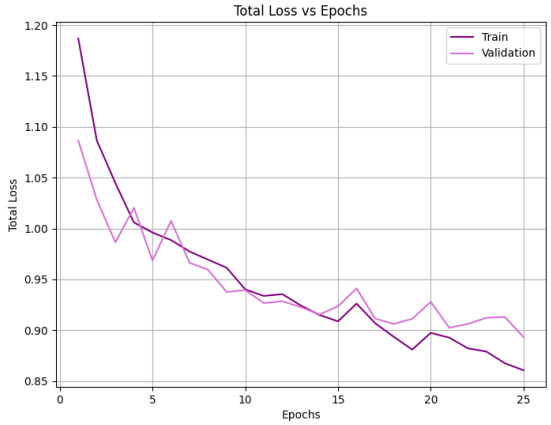
2.2 Fine-Tuning the Pretrained Model

To improve performance on the Foggy Cityscapes dataset, we fine-tune the pretrained Deformable DETR model. Several training configurations and ablations were explored, including learning rate adjustments, weight decay, learning rate scheduling, and layer freezing. These are detailed in the subsections below.

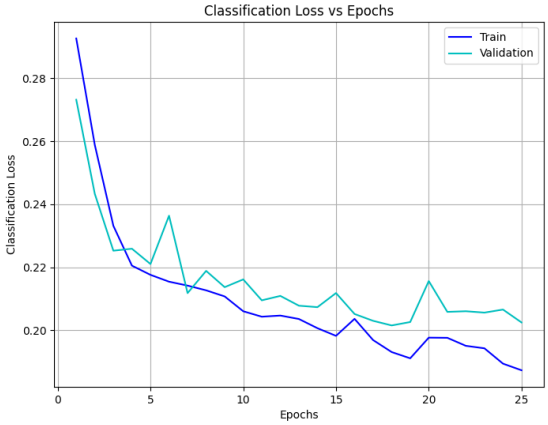
2.2.1 Fine-Tuning the Entire Model

In this configuration, all layers of the model are updated during training. The model is fine-tuned for 25 epochs using the AdamW optimizer, with a learning rate of 1×10^{-5} and a weight decay of 1×10^{-4}.

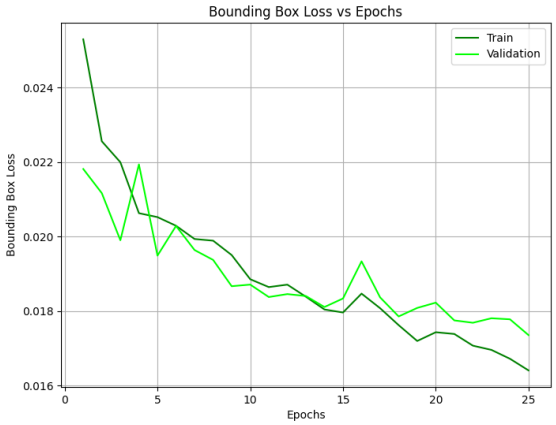
The following plots illustrate the training and validation losses across epochs:



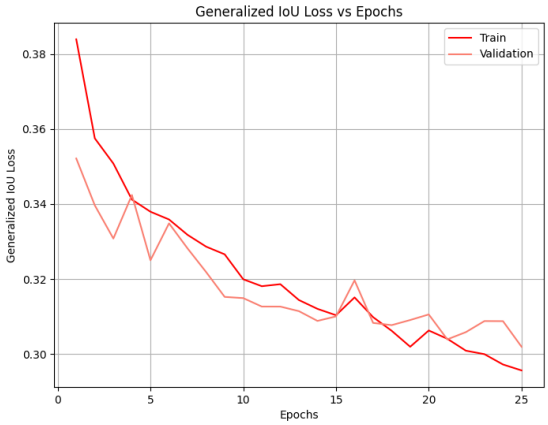
(a) Total loss vs. epochs



(b) Classification loss vs. epochs



(c) Bounding box regression loss vs. epochs



(d) Generalized IoU loss vs. epochs

Figure 10: Training and validation losses during full model fine-tuning

After fine-tuning, the following performance metrics were observed:

- **Training mAP**: 0.226

- **Validation mAP**: 0.180

COCO-style evaluation metrics on the validation set:

- **AP@[IoU=0.50:0.95]**: 0.180
- **AP@0.50**: 0.254
- **AP@0.75**: 0.195

These results show a significant improvement over the initial pretrained model. Notably, the model is now able to detect previously missed classes, such as the *rider* class. The overall mAP values have improved across all IoU thresholds, and class-wise detection accuracy has become more balanced.

The following plots illustrate improvements in per-class performance and precision-recall curves:

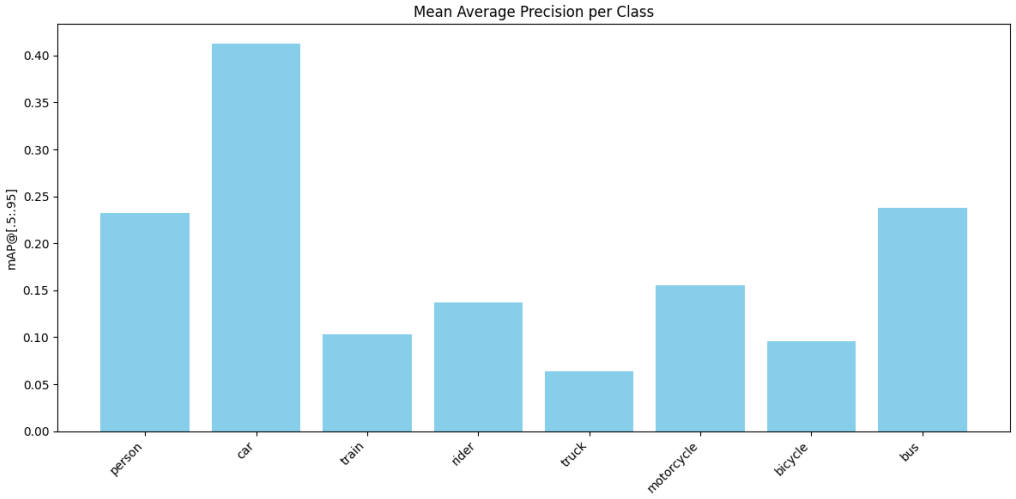


Figure 11: Per-class mAP after full fine-tuning

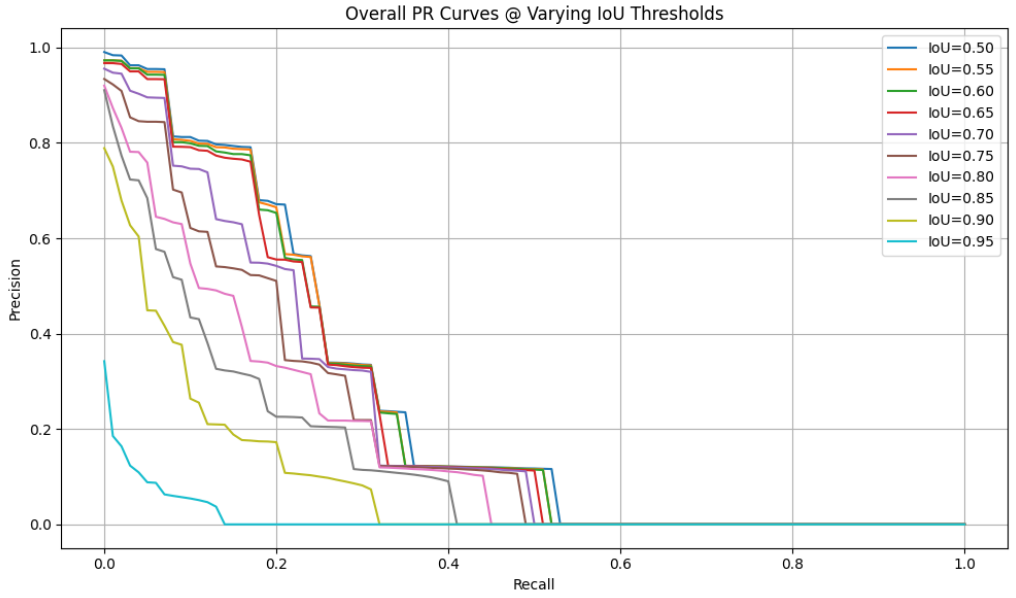


Figure 12: PR curves at multiple IoU thresholds after fine-tuning

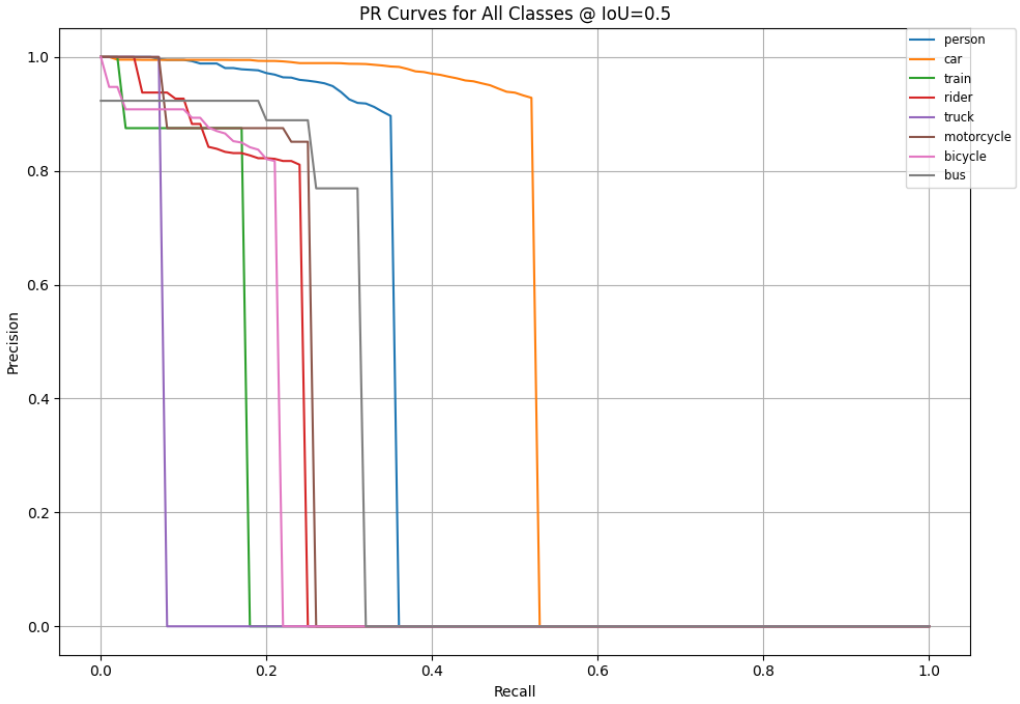


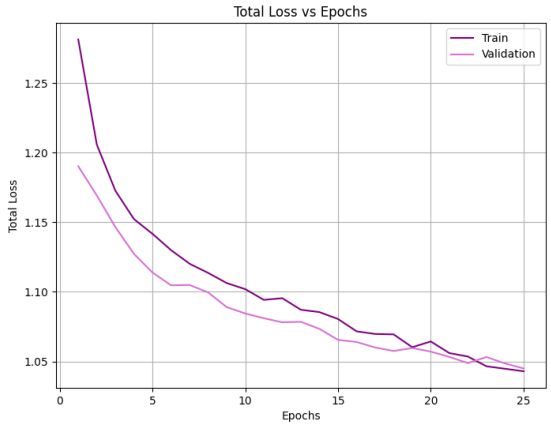
Figure 13: Per-class PR curves at IoU=0.5 and IoU=0.75

In summary, fine-tuning the full model led to noticeable improvements in detection performance, particularly for underrepresented and previously undetected classes. These results highlight the importance of domain adaptation and targeted training on custom datasets.

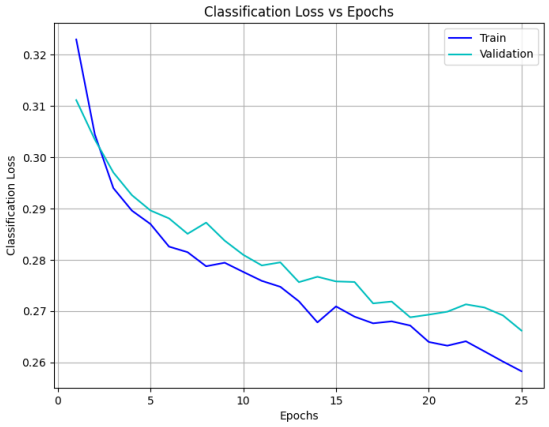
2.2.2 Fine-Tuning the Encoder Layer

In this experiment, we fine-tune only the input projection layers and the transformer encoder layers of the Deformable DETR model. The ResNet backbone and the transformer decoder are kept frozen throughout training.

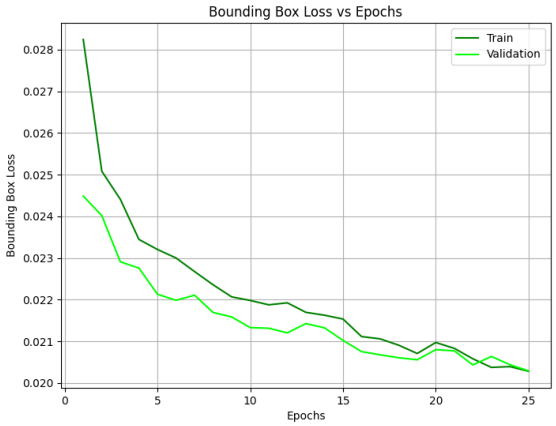
The following plots illustrate the training and validation losses across epochs:



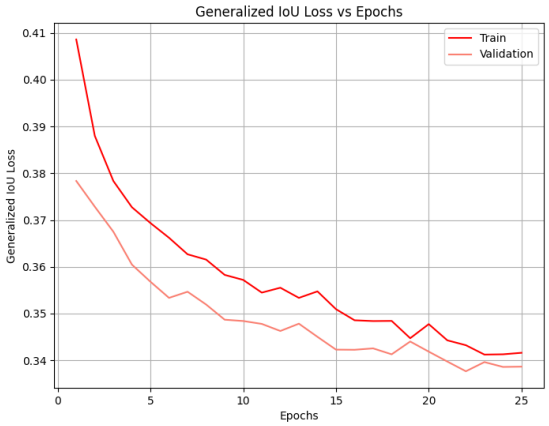
(a) Total loss vs. epochs



(b) Classification loss vs. epochs



(c) Bounding box regression loss vs. epochs



(d) Generalized IoU loss vs. epochs

Figure 14: Training and validation losses during encoder-only fine-tuning

After fine-tuning, the following performance metrics were observed:

- **Training mAP**: 0.226
- **Validation mAP**: 0.197

COCO-style evaluation metrics on the validation set:

- **AP@[IoU=0.50:0.95]**: 0.197
- **AP@0.50**: 0.318

- **AP@0.75**: 0.198

Fine-tuning only the encoder layers leads to an improvement in overall detection performance. However, the model still fails to accurately detect the *rider* class. This limitation is likely due to the frozen decoder layers, which are responsible for generating bounding boxes and classifying objects. Without updating these layers, the model cannot effectively adapt to new object categories such as *rider*.

The following plots show the per-class average precision and the precision-recall curves across different IoU thresholds:

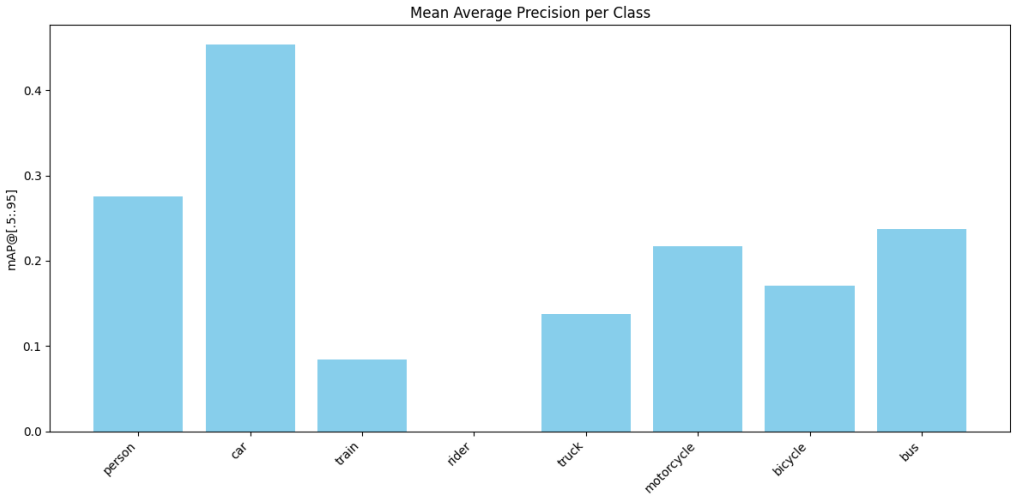


Figure 15: Per-class mAP after encoder-only fine-tuning

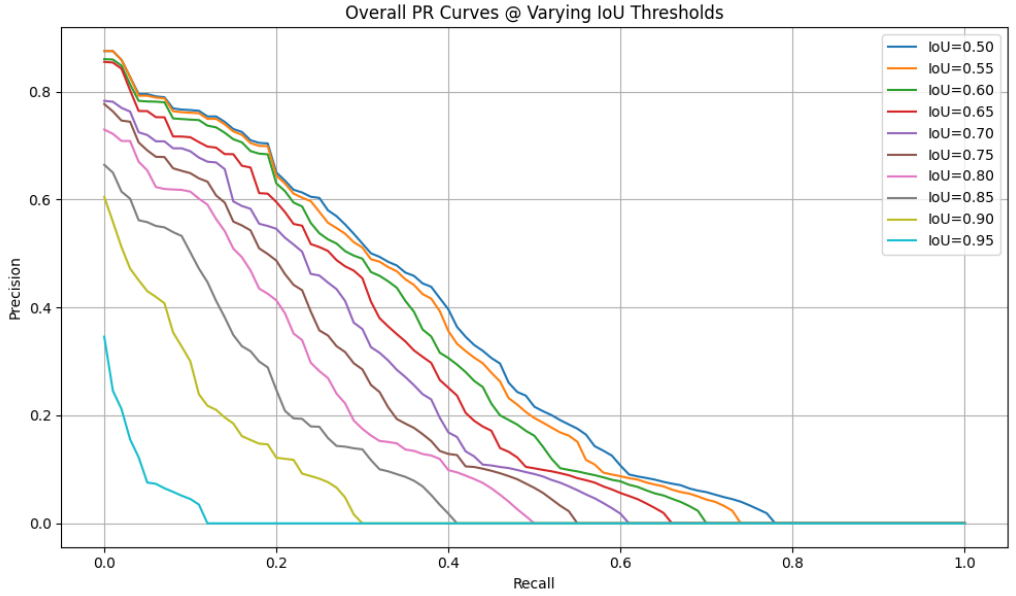


Figure 16: PR curves at multiple IoU thresholds after encoder-only fine-tuning

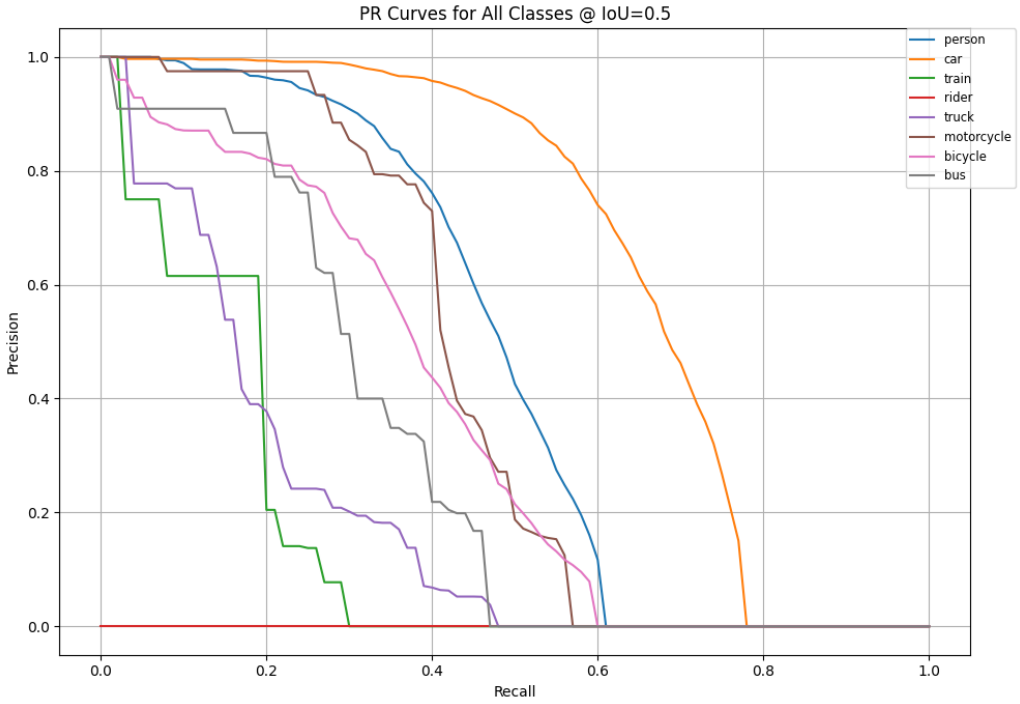


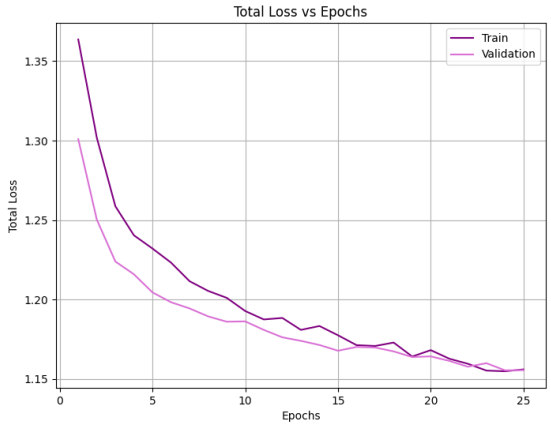
Figure 17: Per-class PR curves at IoU=0.5 and IoU=0.75

In summary, encoder-only fine-tuning yields a modest increase in detection performance, especially for classes already partially learned during pretraining. However, adapting the model to detect new classes—such as *rider*—requires decoder fine-tuning. These results highlight the importance of updating both encoder and decoder components when transferring detection models to a new domain or dataset.

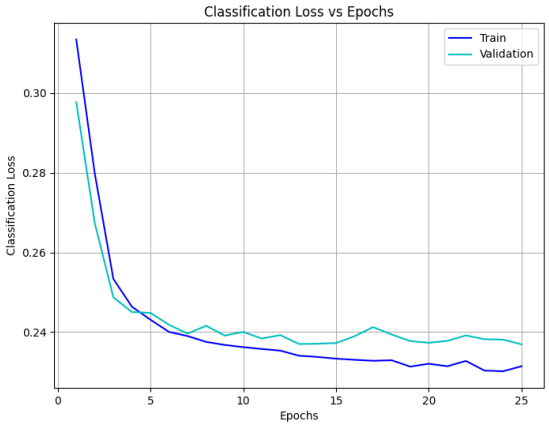
2.2.3 Fine-Tuning the Decoder Layer

In this experiment, we fine-tune only the transformer decoder layers and the final classification and bounding box prediction layers. The ResNet backbone and the transformer encoder are kept frozen throughout training.

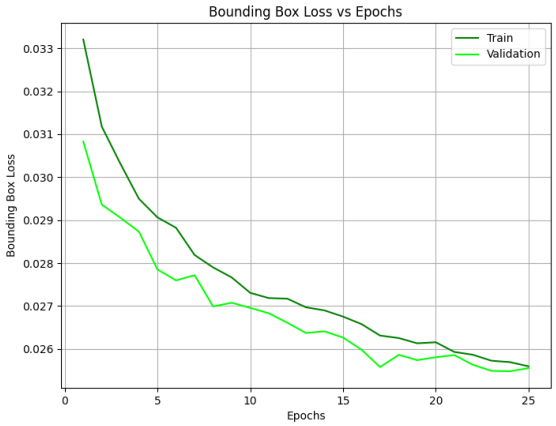
The following plots illustrate the training and validation losses across epochs:



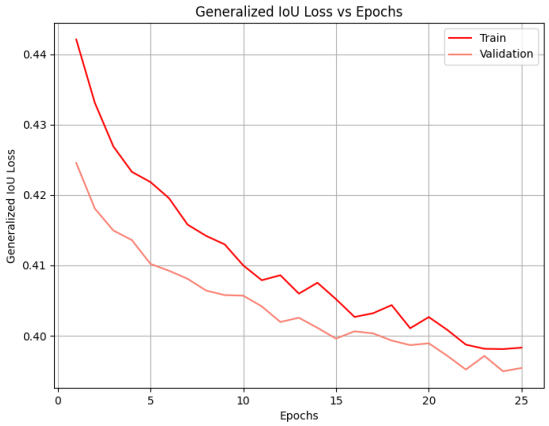
(a) Total loss vs. epochs



(b) Classification loss vs. epochs



(c) Bounding box regression loss vs. epochs



(d) Generalized IoU loss vs. epochs

Figure 18: Training and validation losses during decoder-only fine-tuning

After fine-tuning, the following performance metrics were observed:

- **Training mAP**: 0.242
- **Validation mAP**: 0.211

COCO-style evaluation metrics on the validation set:
AP@[IoU=0.50:0.95] : 0.231 AP@0.50 : 0.326 AP@0.75 : 0.249

Fine-tuning the decoder significantly improves overall detection performance and allows the model to correctly learn and detect the *rider* class. Unlike encoder-only fine-tuning, this approach directly

updates the model components responsible for object classification and bounding box regression, making it more effective for learning new classes and improving precision.

The following plots show the per-class average precision and precision-recall curves at different IoU thresholds:

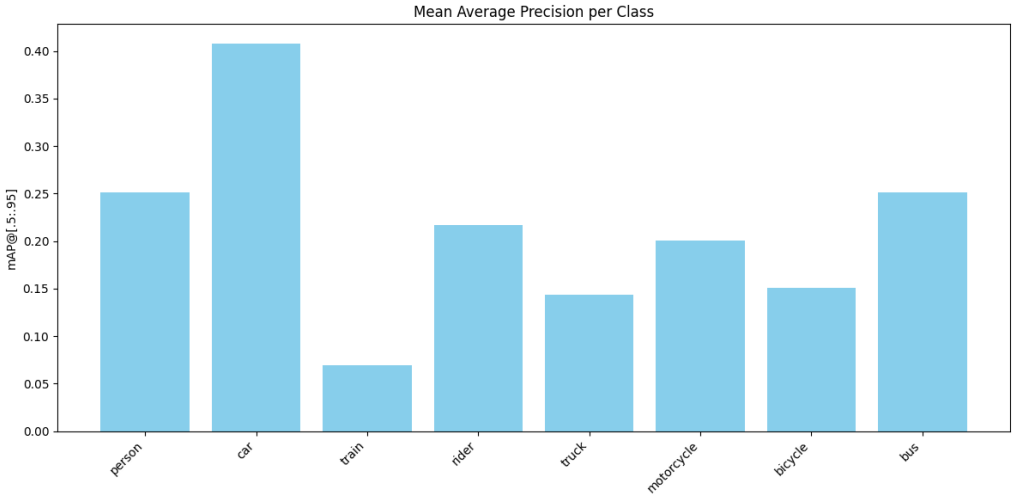


Figure 19: Per-class mAP after decoder-only fine-tuning

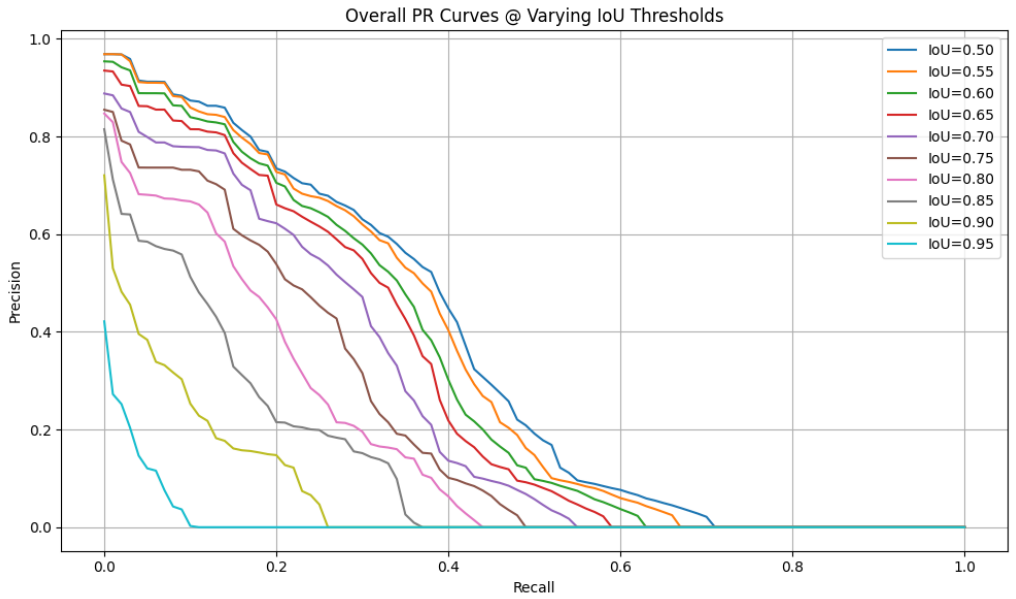


Figure 20: PR curves at multiple IoU thresholds after decoder-only fine-tuning

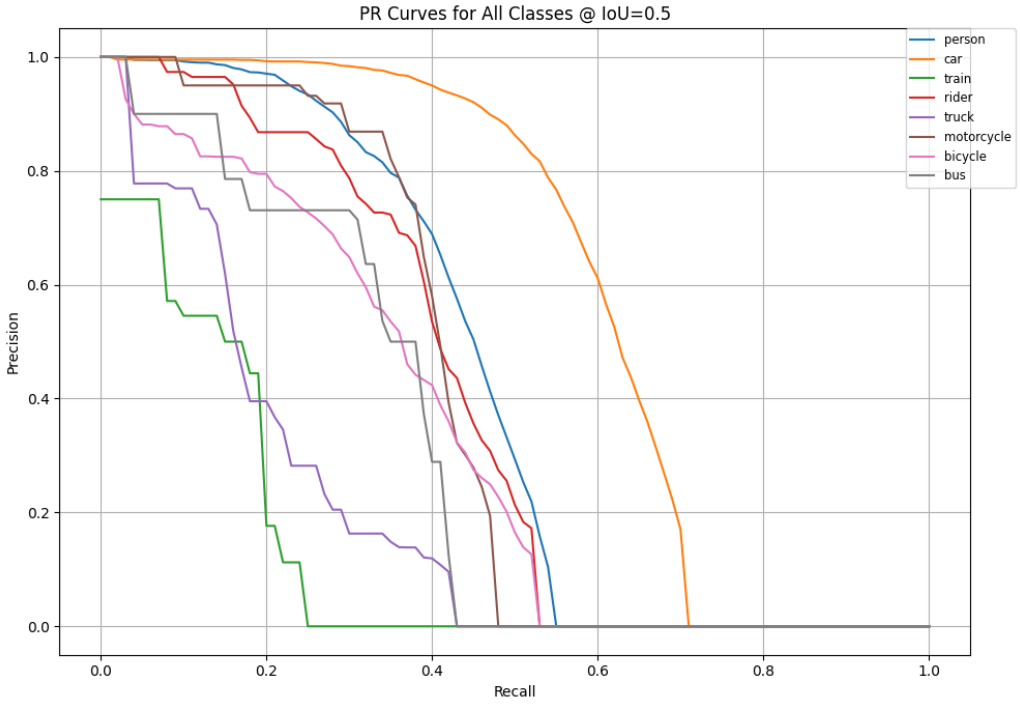


Figure 21: Per-class PR curves at IoU=0.5 and IoU=0.75

In summary, decoder-only fine-tuning leads to notable improvements in both overall detection metrics and per-class performance. Crucially, it enables the model to detect previously unseen classes like *rider*, which were not included in the original COCO pretraining. This suggests that fine-tuning decoder components is essential when transferring pretrained detection models to new domains or when introducing new object categories.

3 Grounding DINO

3.1 Inference on Pretrained Model

This section presents the inference results obtained using Hugging Face's idea research grorunding dino with pretrained weights.

The base prompt used iinitally is:

"person.car.train.rider.truck.motorcycle.bicycle.bus."

3.1.1 Quantitative Analysis

The model was evaluated on both the training and validation datasets. The mean Average Precision (mAP) values observed are:

- **Training mAP**: 0.242
- **Validation mAP**: 0.126

The detailed analysis below is based on the evaluation metrics computed on the validation dataset using the COCO evaluation function provided by `pycocotools`.

- **mAP and Recall at Different IoU Thresholds**:
  - **AP@[IoU=0.50:0.95]**: 0.159
  - **AP@0.50**: 0.261
  - **AP@0.75**: 0.165
  - **AR@[IoU=0.50:0.95]**: 0.270

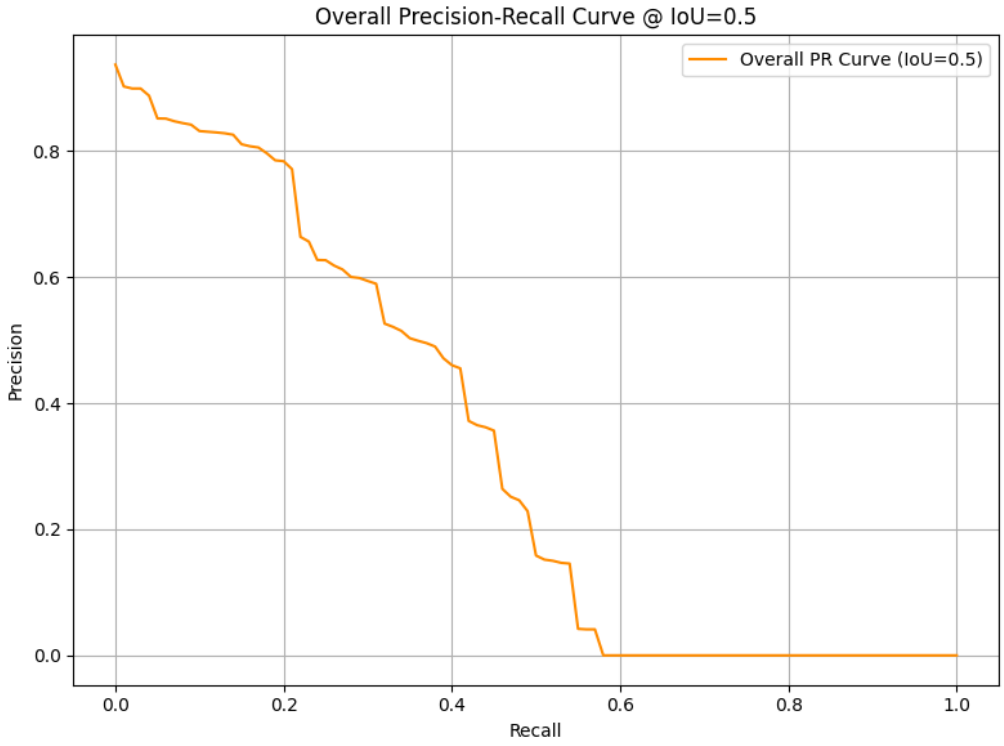


Figure 22: Precision-Recall (PR) curve at IoU threshold 0.5

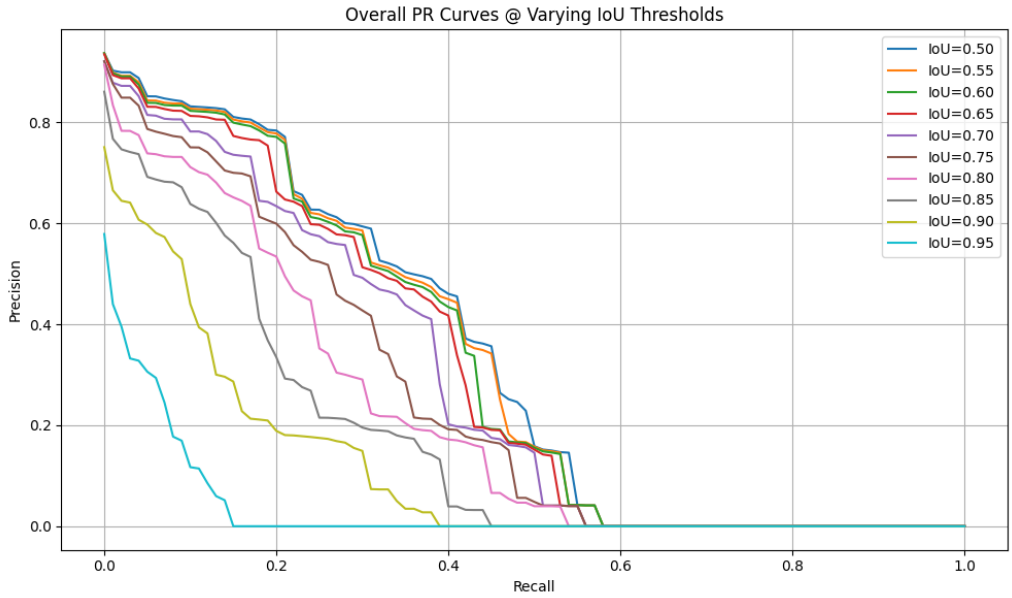


Figure 23: PR curves at different IoU thresholds

As seen in the plots, precision and recall decrease with increasing IoU thresholds. This trend occurs because stricter IoU thresholds demand tighter alignment between predicted and ground truth bounding boxes. Hence, fewer detections qualify as true positives under higher thresholds.

- **Per-Class mAP and Detection Performance**

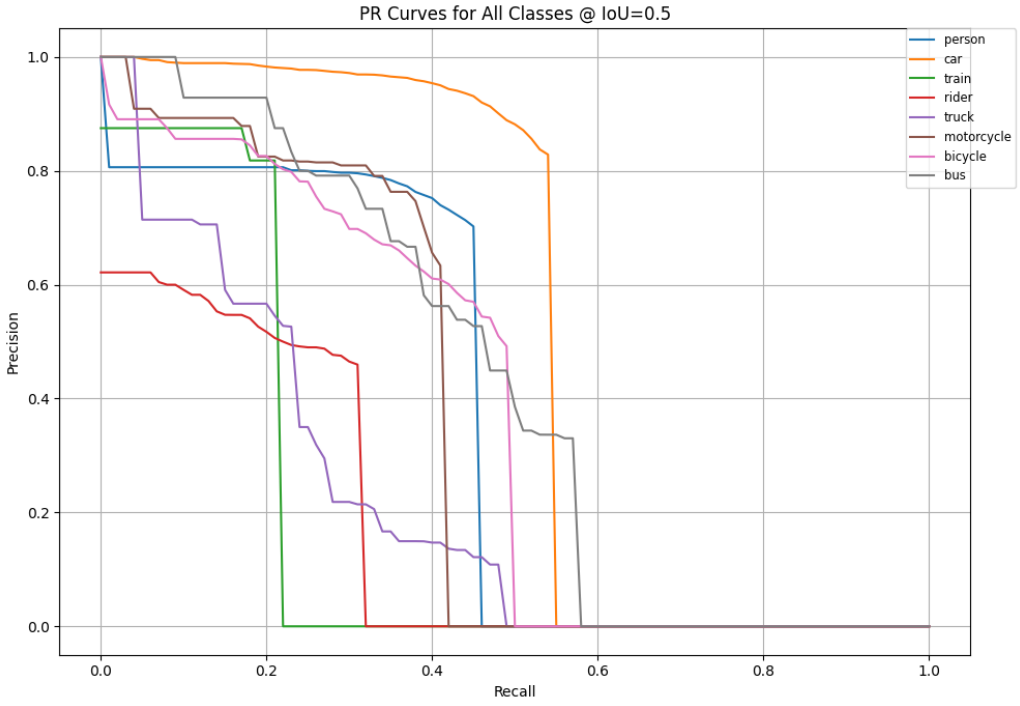


Figure 24: PR curves for each class at IoU thresholds 0.5 and 0.75

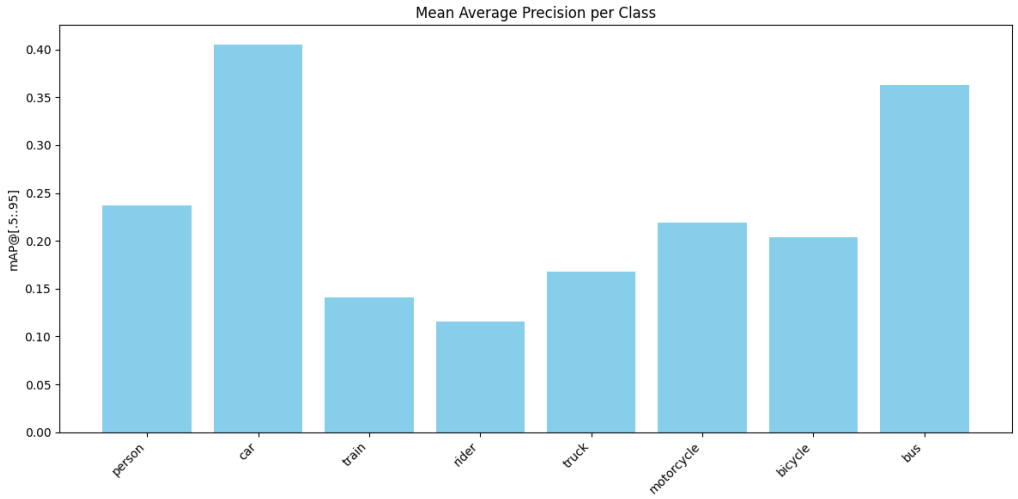


Figure 25: mAP values per class at varying IoU thresholds

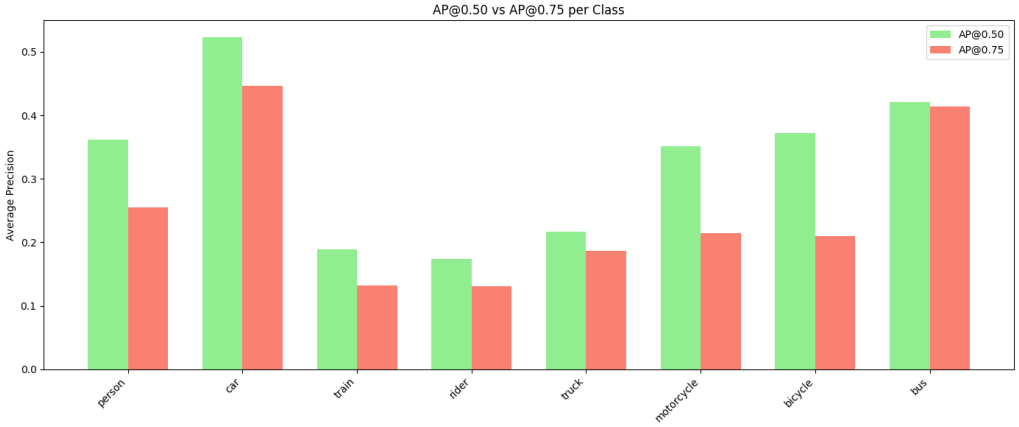


Figure 26: Comparison of AP@0.50 and AP@0.75 for each class

This pretrained model is able to detect all the classes very effectively, This is becuase it is a grounded model. It is not trained to r3cognized just a limited number of classes. It detects all the classes based on the prompt.

- **Average Precision by Object Size**

  The model performs significantly better on large objects compared to small ones. This is expected, as small objects—often located far from the camera—tend to appear blurry, especially in challenging weather conditions such as fog. Consequently, the model struggles to detect these distant, low-contrast features.

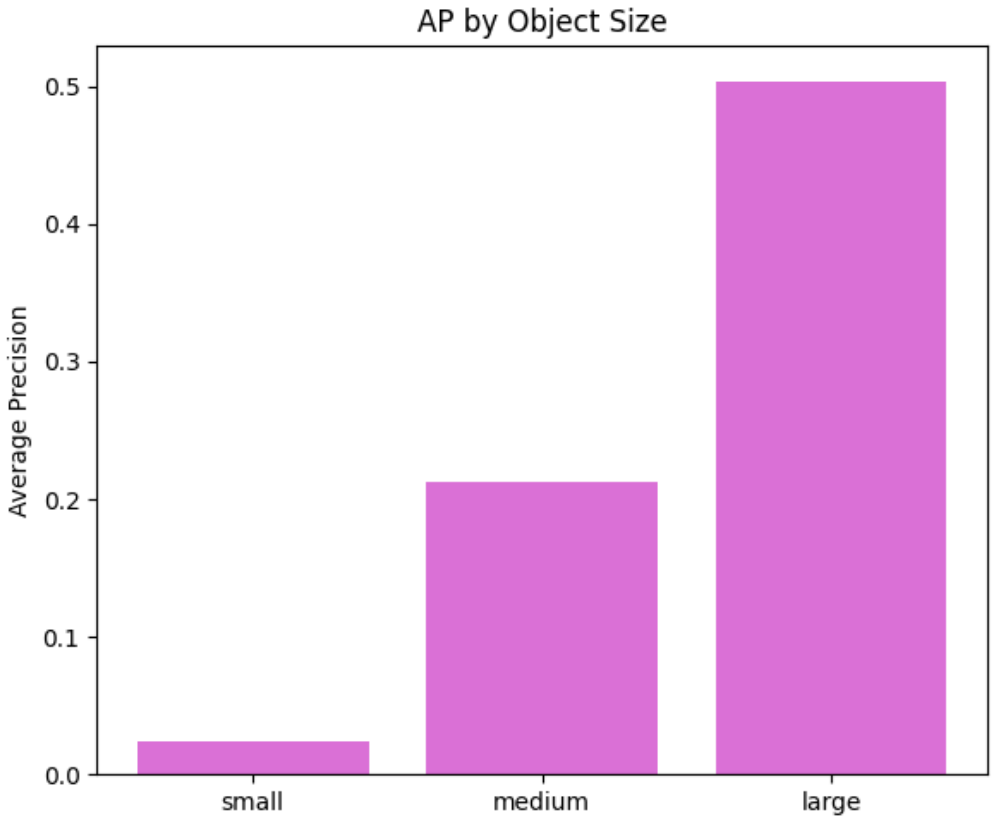


Figure 27: Average Precision for small, medium, and large objects

- **Average Precision by Maximum Detections**

  As seen from the plot, average recall improves as the maximum number of detections per image increases. Allowing more detections enables the model to identify additional objects that might otherwise be missed due to suppression or thresholding constraints. However, beyond a certain point, the gain in performance diminishes, indicating a trade-off between detection quality and computational efficiency.

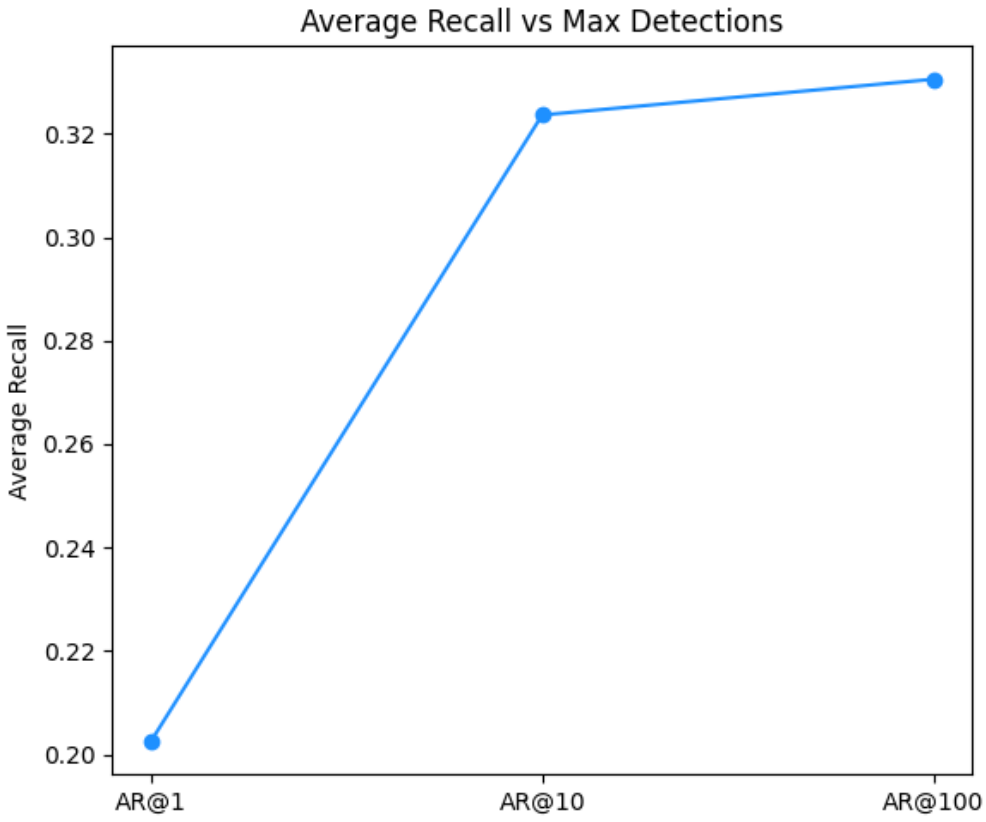


Figure 28: Average Recall versus Maximum Detections Allowed

Overall, the inference results suggest that the pretrained Deformable DETR model demonstrates modest performance out-of-the-box on this dataset. The mAP scores are relatively low compared to typical benchmarks, particularly for small and uncommon object categories. However, the results are consistent with expectations given that the model has not been fine-tuned on this specific dataset and was originally trained on the COCO dataset.

Notably:

- The model performs better on larger objects, which are more prominent and easier to detect.
- Certain classes such as *rider* and *train* are poorly detected due to their absence or underrepresentation in the COCO dataset.
- As the IoU threshold increases, the model's precision and recall drop, indicating less accurate bounding box localization.
- Increasing the number of allowed detections improves recall, but the benefit plateaus, reflecting a trade-off between prediction count and detection quality.

These findings indicate that while the pretrained Deformable DETR model serves as a strong baseline, its performance on this custom dataset can be substantially improved through fine-tuning. In particular, fine-tuning will enable the model to better adapt to the domain-specific distribution of objects, improve detection of smaller objects, and learn to recognize classes such as *rider*, which were not part of the original training dataset. These improvements and the associated training procedures are discussed in detail in the following sections.

3.1.2 Qualitative analysis

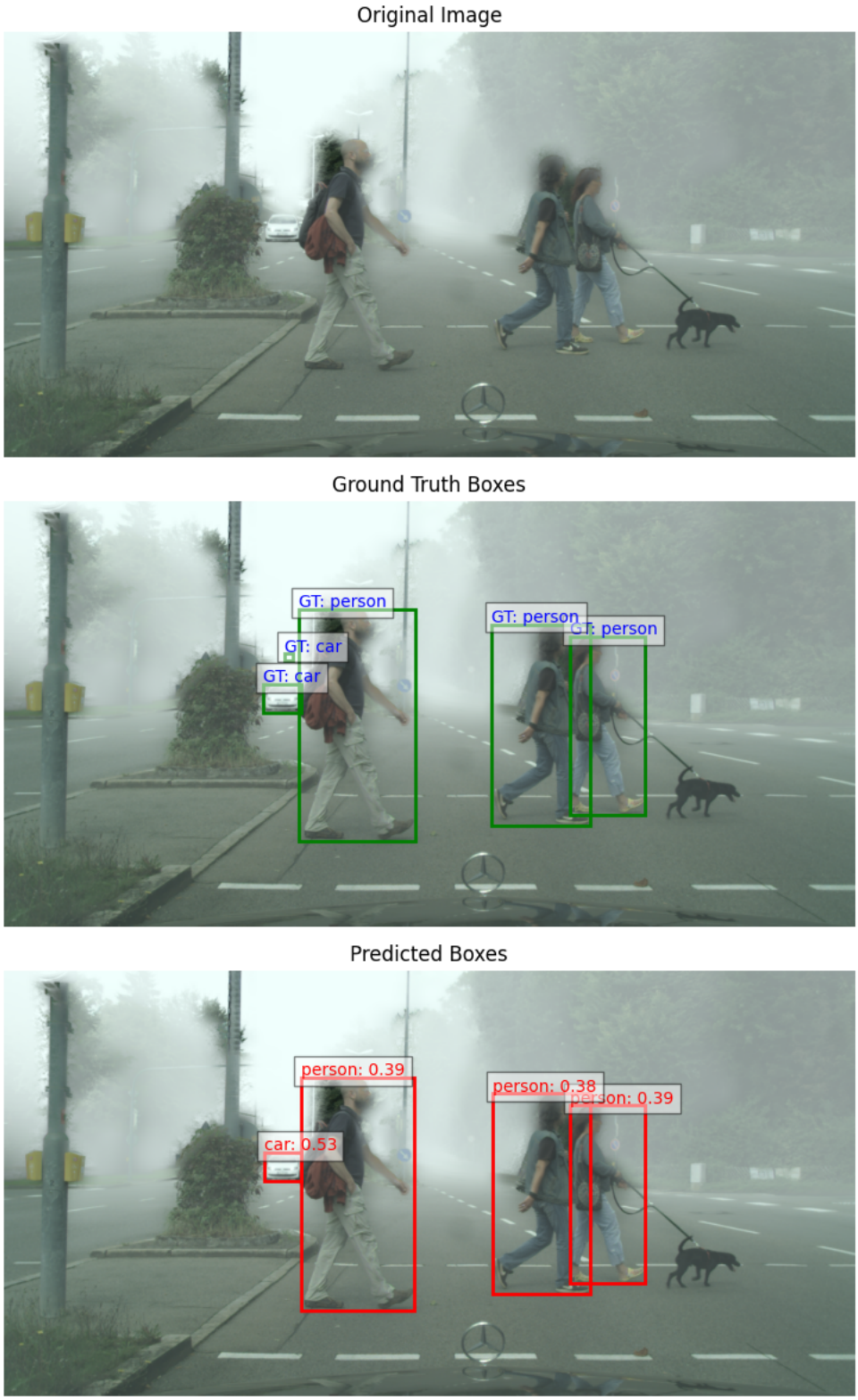


Figure 29: Qualitative visualization of deformable detr performance

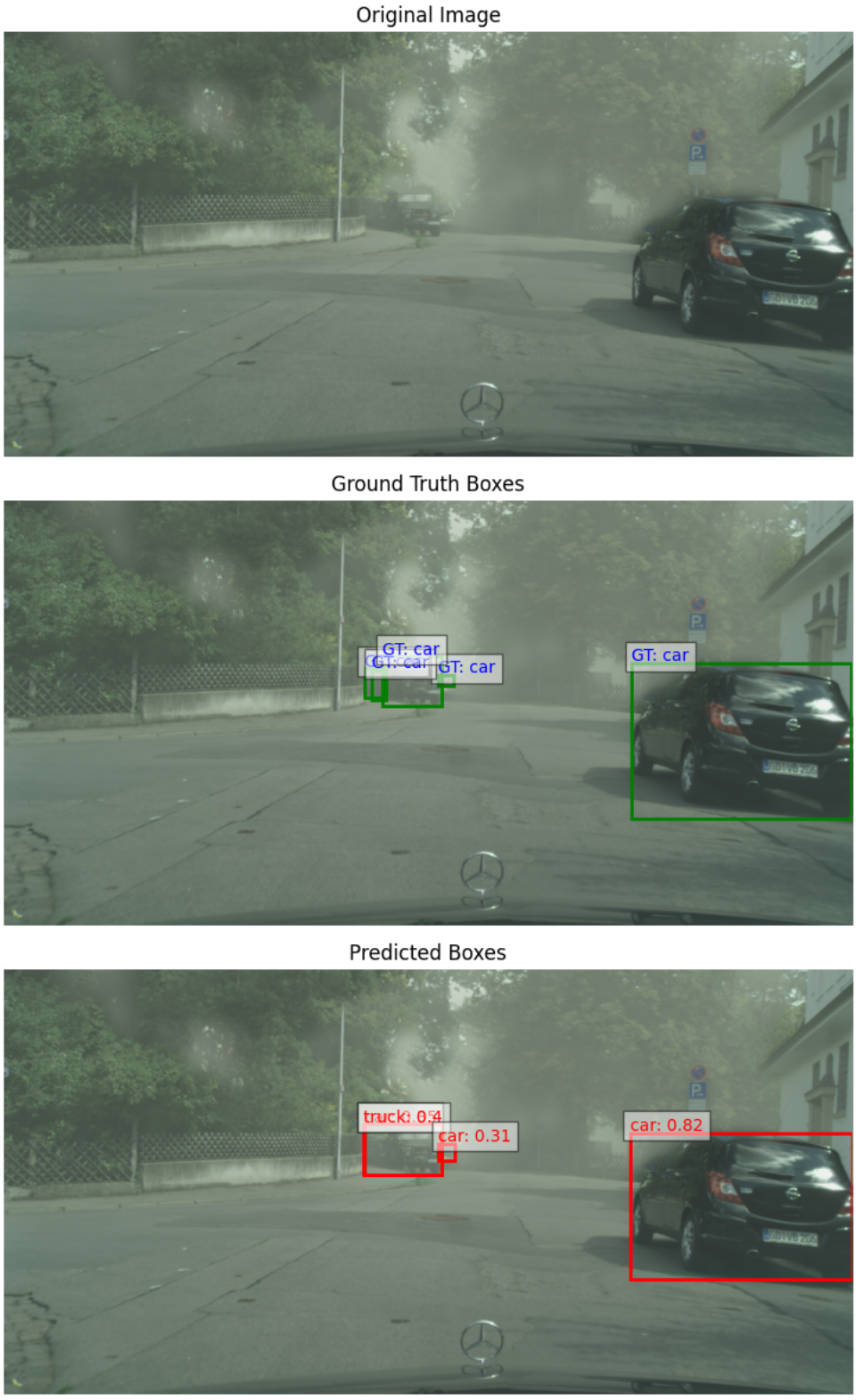


Figure 30: Qualitative visualization of deformable detr performance

4 Grounding DINO Prompt Tuning

5 Competitive Part