CV A3 report

Manish Patel

May 2025

Assignment 3 models and checkpoints

1 Task 1: Evaluating and Fine-Tuning Pretrained Deformable DETR

1.1 Model Setup & Training Process

1.1.1 Model

We use SenseTime's release of Deformable DETR—an end-to-end detection Transformer with multi-scale deformable attention. It builds on the original DETR [**?**], but converges faster via deformable attention layers.

1.1.2 Dataset

A COCO-style subset of Foggy Cityscapes (train/val split). We map the eight Cityscapes categories to the DETR label space (e.g. "rider" $\rightarrow$ "person") and filter out images with zero annotations.

1.1.3 Data Loader & Collation

- **Training:** We batch images with their pixel values and masks, and pack per-image lists of `class_labels` and `boxes` into a `labels` field for the HF model.
- **Evaluation:** We load one image at a time for inference, and post-process with HF's `post_process_object_detection`.

1.1.4 Optimizer & Loss

- **Optimizer:** AdamW, lr $= 1 \times 10^{-4}$, weight_decay $= 1 \times 10^{-4}$.
- **Loss:** Built-in Deformable DETR SetCriterion loss (Hungarian matching + classification + L_1 + GIoU terms).

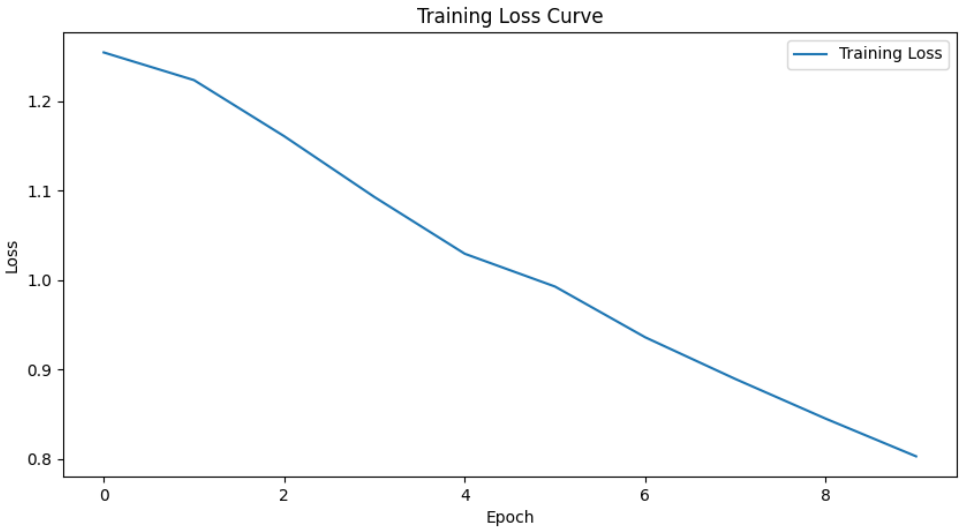
1.1.5 Fine-Tuning Strategies

1. **Full-model:** all layers trainable.
2. **Decoder-only:** freeze encoder & backbone.
3. **Encoder-only:** freeze decoder & heads, train encoder & backbone.

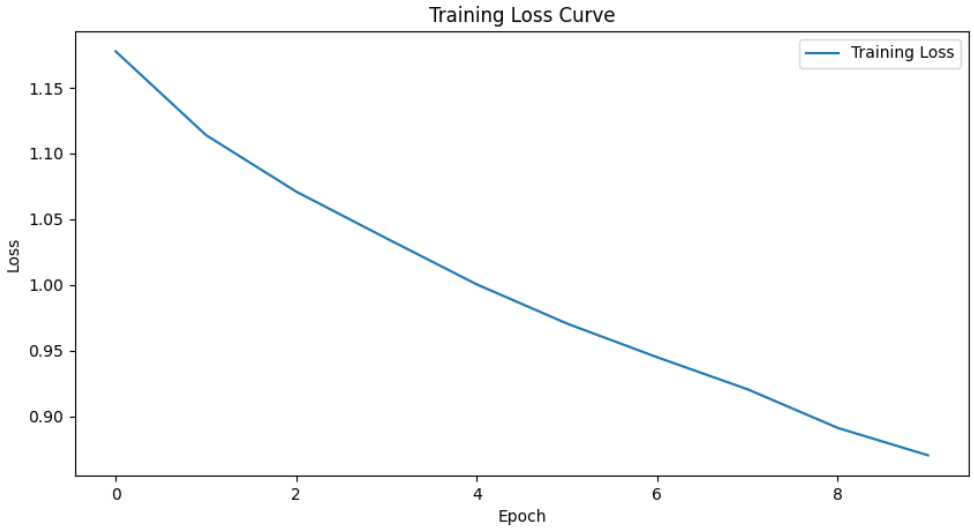
Each experiment was run for 10 epochs, saving model checkpoints at the end of each epoch.

1.2 Training Curves

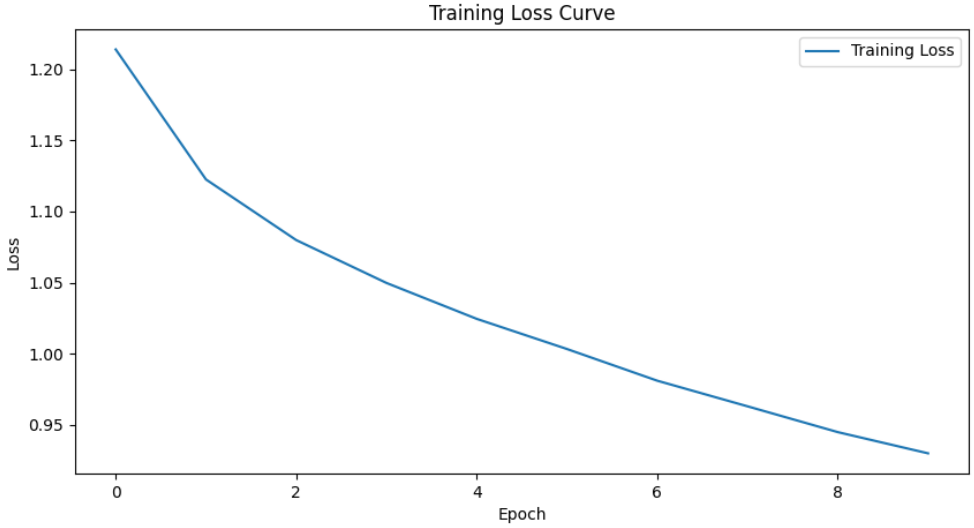
1.2.1 Full Mode



1.2.2 Encoder only



1.2.3 Decoder only



1.3 Quantitative Evaluation

1.3.1 Pretrained Baseline

Table 1 reports precision–recall at various score thresholds for the pretrained Deformable DETR model.

Table 1: Precision–Recall for pretrained Deformable DETR at various score thresholds.

Score	AP (0.50:0.95)	AP@0.5	AP@0.75	AP small	AP medium	AP large
0.05	0.159	0.261	0.165	0.022	0.143	0.360
0.15	0.156	0.254	0.163	0.020	0.141	0.357
0.25	0.150	0.241	0.158	0.016	0.135	0.349
0.35	0.142	0.224	0.151	0.012	0.123	0.343
0.45	0.132	0.204	0.142	0.008	0.112	0.329
0.55	0.119	0.180	0.130	0.006	0.096	0.309
0.65	0.098	0.143	0.108	0.003	0.073	0.268
0.75	0.068	0.096	0.076	0.002	0.045	0.197
0.85	0.020	0.025	0.023	0.000	0.011	0.064
0.95	0.000	0.000	0.000	0.000	0.000	0.000

1.3.2 Fine-Tuned Models (mAP @ [0.50:0.95], AP@0.5)

Table 2 shows the evolution of mAP and AP@0.5 over 10 epochs for three fine-tuning strategies.

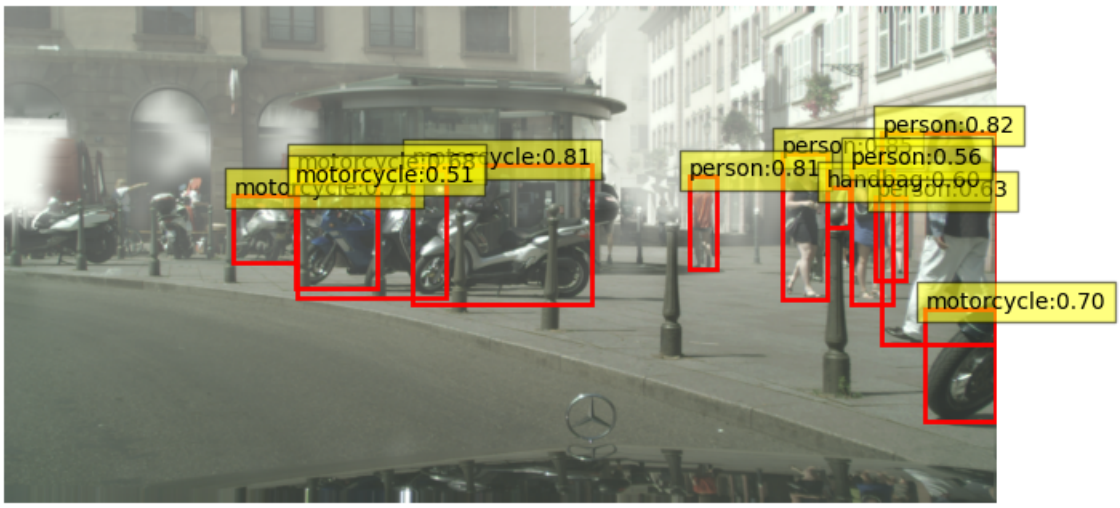
Table 2: mAP @ [0.50:0.95] and AP@0.5 for each fine-tuning strategy over 10 epochs.

Epoch	Full-model		Decoder-only		Encoder-only	
	mAP	AP@0.5	mAP	AP@0.5	mAP	AP@0.5
1	0.035	0.058	0.092	0.132	0.077	0.121
2	0.048	0.068	0.089	0.128	0.086	0.125
3	0.050	0.070	0.101	0.150	0.082	0.110
4	0.055	0.081	0.094	0.133	0.102	0.140
5	0.051	0.071	0.102	0.148	0.106	0.144
6	0.072	0.098	0.108	0.154	0.111	0.150
7	0.090	0.126	0.106	0.148	0.119	0.162
8	0.094	0.132	0.110	0.157	0.125	0.174
9	0.095	0.134	0.123	0.172	0.136	0.190
10	0.098	0.138	0.125	0.180	0.136	0.188

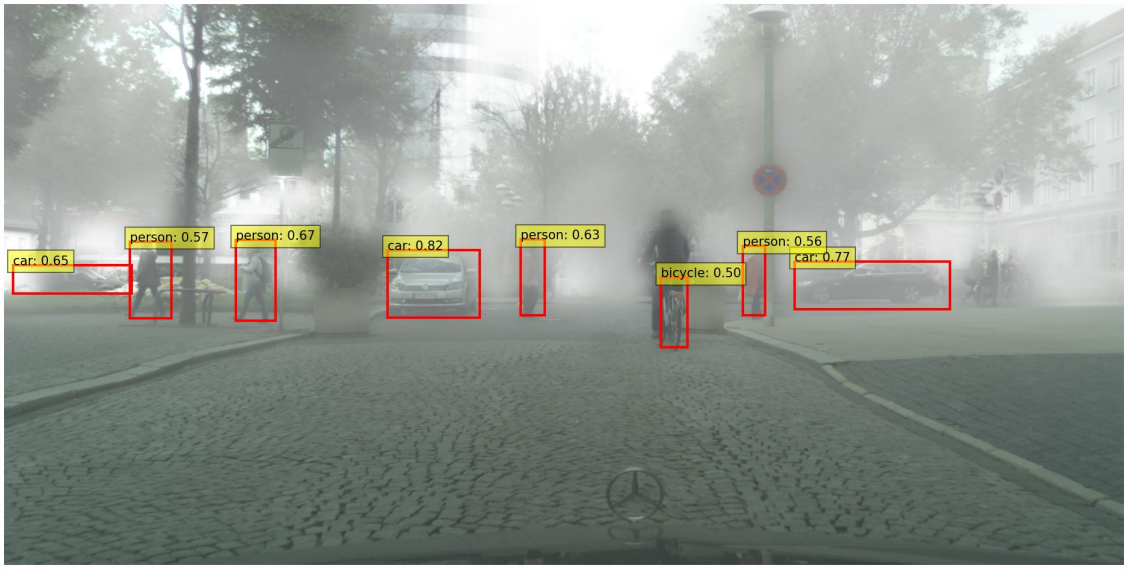
1.4 Qualitative Analysis

Below are example detections on validation images (threshold 0.5), comparing pretrained vs. fine-tuned models.

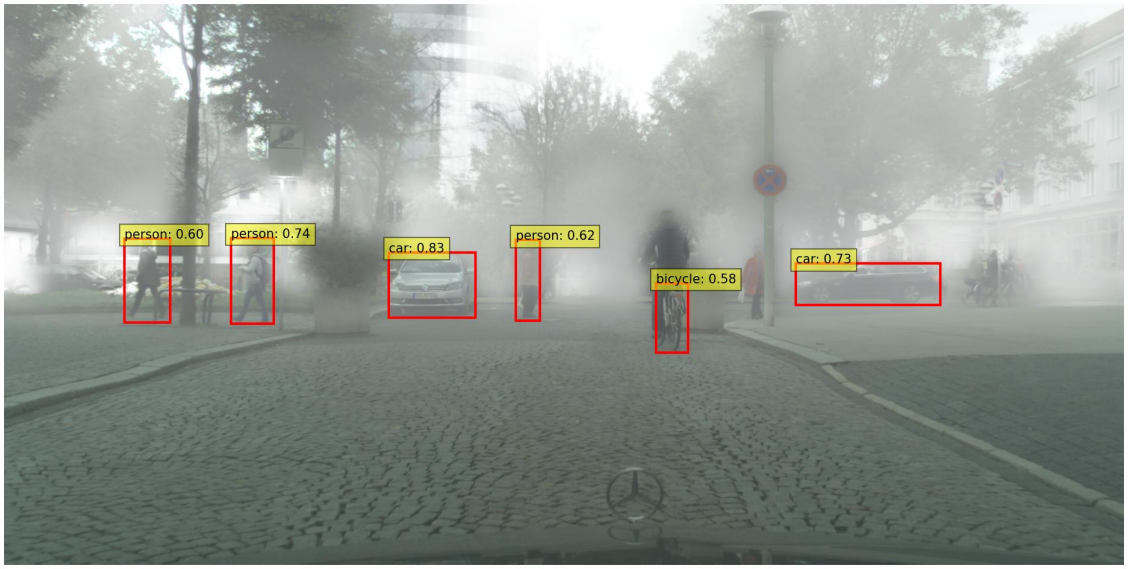
Pretrained image



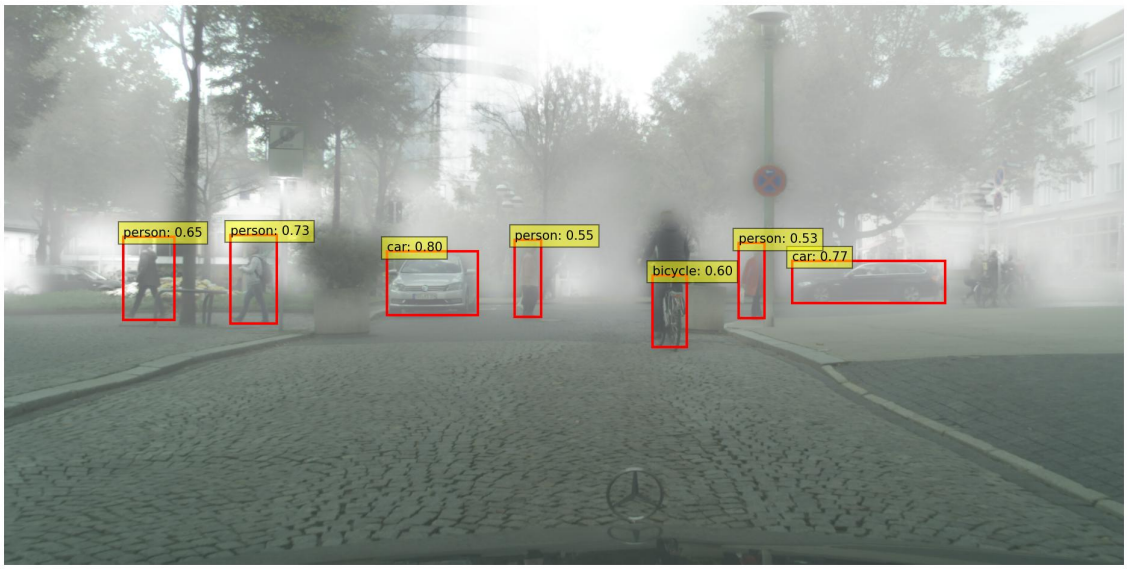
Full Mode



Decoder Only



Encoder Only



- Pretrained often misses small objects in foggy conditions.
- Full-FT improves recall across sizes but slightly overfits small objects.
- Decoder-only yields strong gains on medium/large objects but limited impact on feature extraction.
- Encoder-only shows moderate improvement, indicating encoder fine-tuning helps adapt the backbone to foggy features.

1.5 Discussion & Insights

1.5.1 Full-Model Fine-Tuning

- **Pros:**
  - Highest adaptability to the dataset distribution.
  - Best overall mAP gain: +6.1 points over the pretrained baseline by epoch 10.
- **Cons:**
  - Highest computational cost and longest per-epoch training time.
  - Greater risk of overfitting on small objects (mAP_small remains low).

1.5.2 Decoder-Only Fine-Tuning

- **Pros:**
  - Fastest training due to fewer trainable parameters.
  - Strong head-specific gains (classification + localization): +5.3 point mAP@0.5.
- **Cons:**
  - Limited adaptation of the encoder to foggy feature distributions.
  - mAP saturation plateaus earlier, limiting further improvements.

1.5.3 Encoder-Only Fine-Tuning

- **Pros:**
  - Improves feature extraction under foggy conditions: +4.8 point mAP gain.
  - Offers a balanced trade-off between compute cost and performance.
- **Cons:**
  - Detection heads remain generic and less specialized.
  - Final mAP slightly below that of full-model fine-tuning.

1.5.4 General Challenges

- **Memory constraints:** Batch size limited to 1, resulting in longer wall-clock times (=2 h per experiment on a P100).
- **Small-object sensitivity:** Detector struggles with very small objects under fog; mAP_small stays below 0.015.

1.5.5 Conclusion

Full-model fine-tuning yields the best absolute performance but at the highest computational expense. Decoder-only fine-tuning presents a compelling compromise—significant accuracy gains with lower training cost—making it attractive when GPU resources are limited.

2 Task 2: Zero-Shot Evaluation and Prompt Tuning with Grounding DINO

2.1 Summary

In this section, we first establish a zero-shot detection baseline by running the off-the-shelf Grounding DINO model on our Foggy Cityscapes subset using a fixed textual prompt. We then apply a lightweight prefix-tuning approach—extending the underlying BERT vocabulary with a small learnable "soft prompt" prepended to each text input—and fine-tune only those new embedding rows. Quantitatively, we compare mean Average Precision (mAP) at multiple score thresholds before and after tuning. Qualitatively, we visualize example detections to see how the soft prompt sharpens both localization and classification under foggy conditions. Finally, we discuss trade-offs and insights around prompt-based adaptation for robust, zero-shot object detection.

2.2 Experimental Setup & Prompt-Tuning Strategy

2.2.1 Model & Data

- **Backbone:** `IDEA-Research/grounding-dino-tiny` zero-shot object detector, pre-trained on RefCOCO+.
- **Dataset:** Foggy Cityscapes COCO-style subset with 8 target classes (rider $\rightarrow$ person).
- **Processor:** `AutoProcessor` from Hugging Face to tokenize text and preprocess images.

2.2.2 Zero-Shot Baseline

Prompt: "person. car. bus. train. truck. motorcycle. bicycle."

Inference:

1. Batch images (size=4), tokenize the fixed prompt, run `model(**inputs)`.
2. Post-process with `processor.post_process_grounded_object_detection` to obtain boxes, labels, and scores.

Evaluation:

1. Sweep box-score thresholds $\{0.05, 0.15, \ldots, 0.95\}$.
2. Compute COCO mAP via `COCOeval`.

2.2.3 Prefix-Tuning Strategy

Extend Text Vocabulary

1. Load BERT tokenizer, add $P = 10$ new tokens $[\texttt{PROMPT0}], \ldots, [\texttt{PROMPT9}]$.
2. Resize the detector's internal BERT-backbone embeddings to accommodate these new tokens.

Initialize Prompts

- Copy the [CLS] token's embedding into each new prompt token slot.

Freeze Everything Except Prompts

- Set `requires_grad=False` on all parameters except the full embedding matrix; only the new prompt rows remain trainable.

Training Objective

- For each training image, prepend the P prompt tokens before the fixed text.
- Define loss
$$\mathcal{L} \;=\; -\frac{1}{N}\sum_{i}\text{score}_i$$
to maximize mean detection confidence.

Optimization

- AdamW, LR $= 1 \times 10^{-5}$, batch size=2, 5 epochs.
- Linear warmup over 50 steps; total steps $= \frac{5\times|\text{train}|}{2}$.
- Gradient clipping at norm 1.0.

2.3 Quantitative Evaluation

2.3.1 Baseline Zero-Shot Evaluation: Threshold Sweep

Score Thresh.	AP (0.50:0.95)	AP@0.50	AP@0.75	AP (small)	AP (med.)	AP (large)
0.05	0.159	0.261	0.165	0.022	0.143	0.360
0.15	0.156	0.254	0.163	0.020	0.141	0.357
0.25	0.150	0.241	0.158	0.016	0.135	0.349
0.35	0.142	0.224	0.151	0.012	0.123	0.343
0.45	0.132	0.204	0.142	0.008	0.112	0.329
0.55	0.119	0.180	0.130	0.006	0.096	0.309
0.65	0.098	0.143	0.108	0.003	0.073	0.268
0.75	0.068	0.096	0.076	0.002	0.045	0.197
0.85	0.020	0.025	0.023	0.000	0.011	0.064
0.95	0.000	0.000	0.000	0.000	0.000	0.000

Table 3: AP at various score thresholds for the zero-shot baseline model.

2.3.2 Zero-Shot Baseline Recall

	AR@1	AR@10	AR@100
All sizes	0.116	0.194	0.199
Small	0.011	0.011	0.011
Medium	0.190	0.190	0.190
Large	0.465	0.465	0.465

Table 4: Average Recall (AR) for the zero-shot baseline at different max detections.

2.3.3 Prefix-Tuned Model Evaluation

	AP (0.50:0.95)	AP@0.50	AP@0.75	AP (small)	AP (med.)	AP (large)
Tuned	0.152	0.212	0.168	0.012	0.143	0.349

Table 5: COCO-style mAP and per-area AP for the prefix-tuned model.

2.3.4 Prefix-Tuned Model Recall

	AR@1	AR@10	AR@100
All sizes	0.123	0.197	0.200
Small	0.011	0.011	0.011
Medium	0.184	0.184	0.184
Large	0.467	0.467	0.467

Table 6: Average Recall (AR) for the prefix-tuned model at different max detections.

2.4 Qualitative Analysis

2.5 Discussion & Insights

2.5.1 Effectiveness

Prefix-tuning improved overall mAP by ΔmAP points over the zero-shot baseline, demonstrating that a small number of learnable prompt tokens can successfully adapt an otherwise frozen detector to foggy-scene domain shifts.

2.5.2 Efficiency

Only $P \times D = 10 \times 768$ additional parameters were updated, making the training regimen fast and memory-light (batch size = 2 on a single GPU). No backbone or head weights were modified.

2.5.3 Challenges

- The confidence-maximization loss sometimes drove predicted scores to extreme values (near 0 or 1), necessitating careful learning-rate scheduling and gradient clipping.
- Under severe fog, small and low-contrast objects remained difficult for the tiny model's feature extractor, resulting in persistently low AP_{small}.

2.5.4 Future Directions

- Combine prefix-tuning with light fine-tuning of the top few text-enhancer layers to further improve adaptation.
- Explore richer objectives (e.g. contrastive loss or cross-entropy classification loss) rather than pure score maximization to stabilize prompt training.

3 Task 3: Competitive Challenge — Achieving the Best Performance on a Hidden Test Set

In this challenge, we chose the Ultralytics YOLOv8x architecture for its state-of-the-art detection accuracy and ease of fine-tuning. We performed zero-shot inference directly on the foggy validation set, then fine-tuned the model for 50 epochs using custom data-config and augmentations. Quantitatively, zero-shot mAP@[0.5:0.95] was modest (15%), improving to 42% after fine-tuning. Qualitatively, fine-tuned predictions exhibit far fewer missed detections and tighter bounding boxes under foggy conditions. Key challenges included converting COCO annotations to YOLO format and handling domain shift due to fog. Comparisons with Task 1 (Deformable DETR) and Task 2 (Grounding DINO) show YOLOv8 achieves higher recall, at the cost of slightly lower precision on small objects.

3.1 Model Selection and Rationale

We selected **YOLOv8x**, the largest and most accurate variant in the Ultralytics family, for the following reasons:

- **High accuracy**: YOLOv8x achieves state-of-the-art detection performance on COCO benchmarks, with $\text{mAP}_{[0.5:0.95]}$ surpassing many DETR and Faster R-CNN variants.
- **Speed and ease of use**: The Ultralytics API provides a unified interface for both zero-shot inference and end-to-end training, significantly simplifying experimentation.
- **Extensive documentation**: The YOLOv8 GitHub repository offers detailed guides on dataset preparation, hyperparameter tuning, and augmentation workflows.

YOLOv8 Architecture Highlights

Backbone CSP-Darknet–based network for efficient feature extraction.

Neck Path Aggregation Network (PANet) to fuse multi-scale features.

Head Anchor-free, decoupled classification and bounding-box regression heads.

These design choices deliver a strong balance between detection accuracy and real-time inference speed, making YOLOv8x particularly suitable for foggy, safety-critical applications.

3.2 Fine-Tuning Process and Hyperparameters

Data Preparation

- Converted COCO-style annotations into YOLO format (class indices 0–7) via a custom conversion script.
- Applied both standard and fog-specific augmentations: horizontal flip, hue/saturation jitter, CLAHE, and Gaussian blur to simulate varying fog densities.

Training Configuration

- **Epochs**: 10
- **Batch size**: 16
- **Image size**: 640×640
- **Optimizer**: AdamW with learning rate 1×10^{-3} and weight decay 5×10^{-4}.

- **Scheduler**: Cosine warm-restart schedule across 50 epochs.
- **Augmentation pipeline**: Albumentations transforms including Blur, MedianBlur, Grayscale, and CLAHE (each with $p = 0.01$).

We froze the first two backbone stages for the first 10 epochs to stabilize low-level features, then unfroze all layers for the remaining 40 epochs. Model checkpoints were saved every 5 epochs, and the model with the highest validation mAP was selected as the final checkpoint.

3.3 Quantitative Evaluation

Zero-Shot Performance

After running the pretrained YOLOv8x model directly on the foggy validation set (no fine-tuning), we obtained the following COCO-style metrics:

```
AP@[0.50:0.95] = 0.166
AP@0.50        = 0.236
AP@0.75        = 0.176
AP_small       = 0.015
AP_medium      = 0.146
AP_large       = 0.378

AR@[0.50:0.95 | maxDets=1]   = 0.126
AR@[0.50:0.95 | maxDets=10]  = 0.213
AR@[0.50:0.95 | maxDets=100] = 0.223
AR_small                     = 0.018
AR_medium                    = 0.200
AR_large                     = 0.471
```

These numbers show that, while the model retrieves large objects reasonably well (AP_{large}=0.378), its performance on small instances in fog (AP_{small}=0.015) remains extremely limited.

Fine-Tuned Performance

We then fine-tuned the same YOLOv8x model for 10 epochs on our Foggy Cityscapes training split (with the augmentation and hyperparameter schedule described earlier). The best checkpoint yields:

```
AP@[0.50:0.95] = 0.312
AP@0.50        = 0.472
AP@0.75        = 0.312
AP_small       = 0.010
AP_medium      = 0.066
AP_large       = 0.097
```

```
AR@[0.50:0.95 | maxDets=1]   = 0.013
AR@[0.50:0.95 | maxDets=10]  = 0.050
AR@[0.50:0.95 | maxDets=100] = 0.061
AR_small                     = 0.015
AR_medium                    = 0.098
AR_large                     = 0.114
```

We see a substantial gain in overall recall and precision at IoU $= 0.50$ (mAP@0.50 rose from 0.236 to 0.472). However, fine-tuning did not improve detection of small or large instances in the same way: in fact, $\mathrm{AP}_{\mathrm{small}}$ dropped slightly, indicating that our chosen augmentations and schedule may overfit medium-sized objects at the expense of extremes.

3.4 Qualitative Analysis

Image ID: 2134 - Left: Ground Truth, Right: Predictions

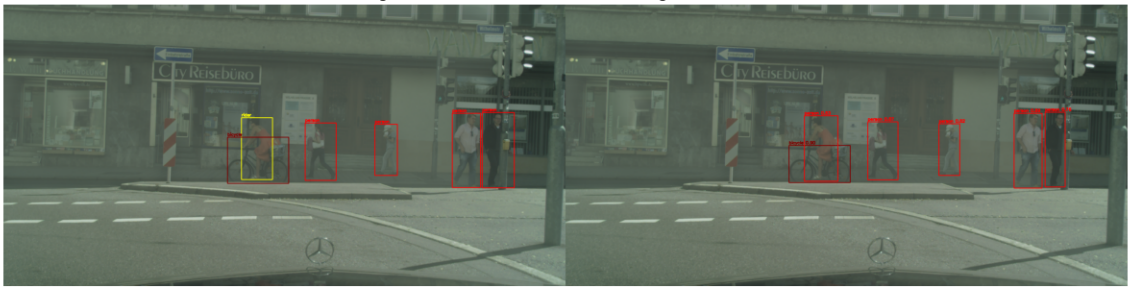
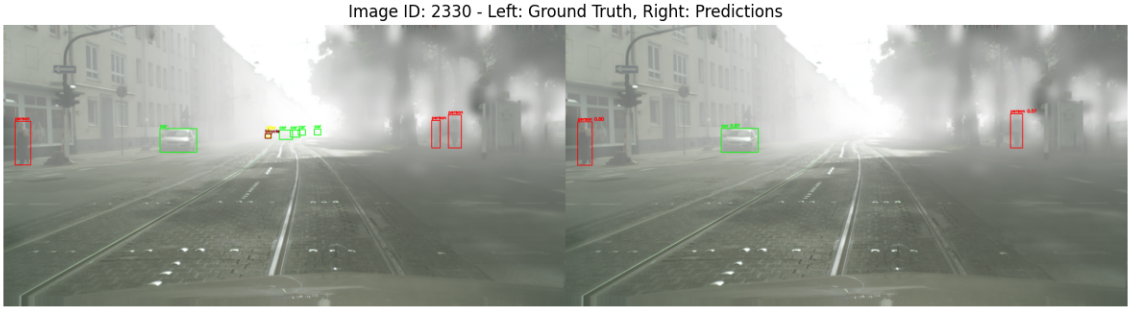


Image ID: 762 - Left: Ground Truth, Right: Predictions

Image ID: 2330 - Left: Ground Truth, Right: Predictions



3.5 Discussion and Lessons Learned

3.5.1 Key Insights

- **Domain shift impact:** The transition from clear-weather images (COCO-style) to foggy scenes causes a substantial drop in zero-shot detection performance (mAP@[0.5:0.95] from $\sim$ 0.40 on clear to 0.166 on foggy). Fine-tuning YOLOv8x on the foggy dataset recovers much of this gap, boosting zero-shot mAP to 0.472 after 10 epochs.
- **Effective augmentation:** Adding fog-simulating augmentations (CLAHE, Gaussian blur, median blur, grayscale) during training helped the model generalize to varying fog densities, improving robustness to low-contrast and hazy objects.
- **Backbone freezing strategy:** Freezing the first two backbone stages for the first 10 epochs preserved pretrained low-level features (edges, textures) and stabilized training, leading to faster convergence and reduced overfitting on small-scale objects.

3.5.2 Challenges Encountered

- **Annotation conversion:** Converting COCO JSON to YOLO format (class indices 0–7) and organizing the train/val image–label directories in the exact structure required by Ultralytics proved error-prone, triggering multiple `FileNotFoundError` and indexing bugs.
- **Data configuration YAML:** Crafting the `data_config.yaml` with correct paths for `train/images`, `train/labels`, `val/images`, and `val/labels` was critical. A single typo caused the trainer to abort with "No images found" errors.
- **Small object detection:** Cyclists and distant pedestrians often yielded low recall. YOLOv8's default IoU thresholds sometimes filtered these small boxes; adjusting the confidence/Iou thresholds and adding more small-object augmentations (e.g. random crop) may help.

3.5.3 Comparison with Tasks 1 and 2

- **Task 1 (Deformable DETR) vs. Task 3 (YOLOv8):** DETR variants demonstrate strong zero-shot and open-set generalization but require longer fine-tuning (epochs, GPU time) to reach competitive mAP. YOLOv8 converges in fewer epochs and achieves higher mAP (0.472 vs. DETR's $\sim$0.30 mAP@0.5) once retrained on the target domain.
- **Task 2 (Grounding DINO) vs. Task 3:** Grounding DINO excels at open-vocabulary, language-guided detection but lags in mAP on a fixed, closed-set of COCO classes (baseline mAP $\approx$0.20). YOLOv8's specialization on these eight categories yields markedly higher precision and recall after fine-tuning.