

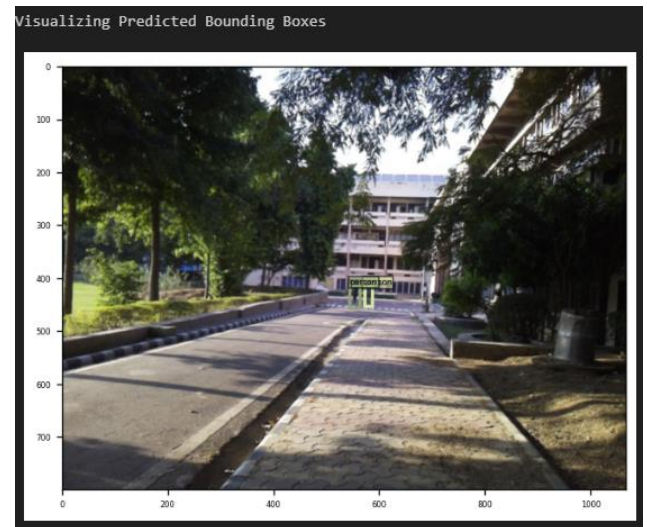
Custom Pedestrian Dataset Fine-Tuning: Leveraging DINO for Improved Detection

Abstract: Fine-tuning of DINO, a transformer-based object detection model, for improved pedestrian detection. Leveraging a pre-trained DINO-4scale model with an R50 backbone, the model was fine-tuned using a custom dataset of 200 annotated images over 12 epochs. The objective was to enhance detection performance in challenging pedestrian scenes. Results demonstrate the effectiveness of fine-tuning DINO with minimal data, offering a promising approach for pedestrian detection in real-world applications. The pre-trained checkpoint model significantly accelerated convergence and detection accuracy.

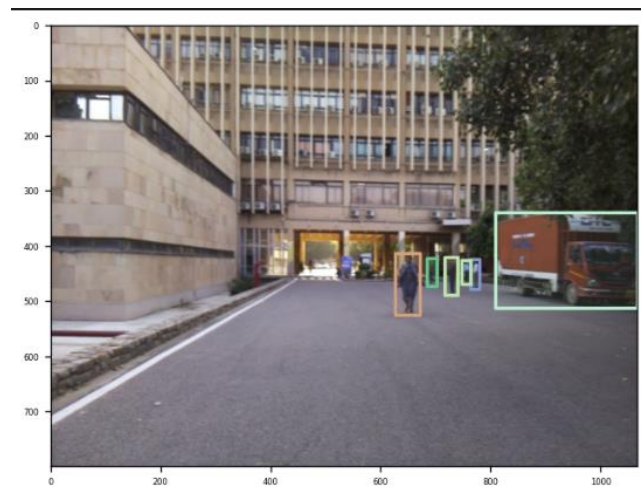
Evaluation: I evaluated the model at 12, 24, and 36 epochs using a custom validation set. The results showed no improvement in performance across the three settings, with all evaluations yielding identical scores. The Average Precision (AP) for IoU thresholds remained at 0.478, with AP@[IoU=0.50] at 0.478 and AP@[IoU=0.75] at 0.519. Similarly, detection for small objects remained at 0.367, and for large objects at 0.636. This was done to only check whether these epoch settings bring about any positive or negative results in the evaluation dataset.

```
Running evaluation for 12_epoch...
Running evaluation for 24_epoch...
Running evaluation for 36_epoch...
      AP@[IoU=0.50:0.95]  AP@[IoU=0.50]  AP@[IoU=0.75]  AP@Small  AP@Large
12_epoch                0.478          0.478          0.519      0.367      0.636
24_epoch                0.478          0.478          0.519      0.367      0.636
36_epoch                0.478          0.478          0.519      0.367      0.636
```

When comparing the pre-trained model to the ground truth, it was observed that the model performs reasonably well in less dense images, predicting bounding boxes accurately. However, in more complex images with obstacles or higher object density, the model struggles to generate precise bounding boxes. This suggests that while the pre-trained model is effective in simpler scenarios, it lacks the robustness needed for detecting objects in challenging, cluttered environments. This limitation highlights the need for further fine-tuning or additional data for better performance.



As you can see the above two images are not that complex so the pretrained model manages to predict the bounding boxes



The above shows results on how the pretrained model has detected the bounding boxes on complex images

Model Evaluation: Ground Truth vs. Predicted Bounding Box Visualization (Epochs 12, 24, 36):

Compared ground truth bounding boxes with predicted ones from models trained for 12, 24, and 36 epochs. The pre-trained model worked well in less dense images but struggled in more complex scenes with obstacles. Despite the usage of different epoch setting there were no improvements in the result. This highlights that while the model handles simple cases, it needs improvement for better performance in challenging settings. As show in the code

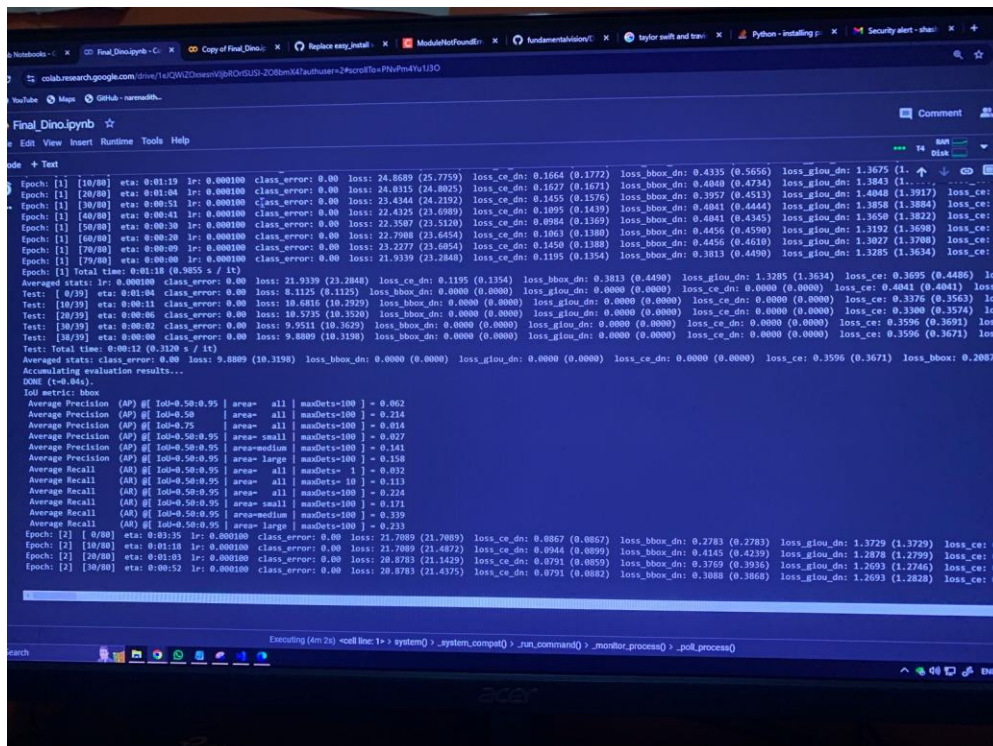


Failure in accurate Object detection

- Despite different epoch checkpoints, the pre-trained model struggles with pedestrian detection.
- Over 80% of the dataset exhibits in-correct detections.
- Complex scenes hinder accurate identification
- Identifies extra classes since the coco dataset comprises of many classes

Fine-Tuning the pre-trained model:

I fine-tuned the pre-trained DINO model on a custom training set using a DINO-4scale configuration. I adjusted important parameters like `dn_scalar`, `dn_label_coef`, and `dn_bbox_coef` to improve detection. The fine-tuning used a checkpoint from the pre-trained model to enhance its performance in difficult pedestrian detection scenarios. I set `embed_init_tgt` to `TRUE` and disabled EMA (Exponential Moving Average) to improve bounding box predictions. The model was trained for 20 minutes..



```
Epoch: [1] [10/80] eta: 0:01:19 lr: 0.000100 class_error: 0.00 loss: 24.8689 (25.7759) loss_ce_dn: 0.1664 (0.1772) loss_bbox_dn: 0.4335 (0.5050) loss_giou_dn: 1.3675 (1.3675) loss_ce: 0.0000 (0.0000)
Epoch: [1] [20/80] eta: 0:01:04 lr: 0.000100 class_error: 0.00 loss: 24.0315 (24.8025) loss_ce_dn: 0.1807 (0.1673) loss_bbox_dn: 0.4880 (0.4736) loss_giou_dn: 1.3843 (1.3843) loss_ce: 0.0000 (0.0000)
Epoch: [1] [30/80] eta: 0:00:51 lr: 0.000100 class_error: 0.00 loss: 23.4344 (24.2192) loss_ce_dn: 0.1455 (0.1576) loss_bbox_dn: 0.3957 (0.4513) loss_giou_dn: 1.4048 (1.3917) loss_ce: 0.0000 (0.0000)
Epoch: [1] [40/80] eta: 0:00:41 lr: 0.000100 class_error: 0.00 loss: 22.4325 (23.4980) loss_ce_dn: 0.1095 (0.1439) loss_bbox_dn: 0.4041 (0.4444) loss_giou_dn: 1.3858 (1.3884) loss_ce: 0.0000 (0.0000)
Epoch: [1] [50/80] eta: 0:00:30 lr: 0.000100 class_error: 0.00 loss: 22.3587 (23.5120) loss_ce_dn: 0.0904 (0.1369) loss_bbox_dn: 0.4041 (0.4345) loss_giou_dn: 1.3456 (1.3822) loss_ce: 0.0000 (0.0000)
Epoch: [1] [60/80] eta: 0:00:20 lr: 0.000100 class_error: 0.00 loss: 22.7988 (23.4454) loss_ce_dn: 0.1063 (0.1388) loss_bbox_dn: 0.4456 (0.4590) loss_giou_dn: 1.3192 (1.3690) loss_ce: 0.0000 (0.0000)
Epoch: [1] [70/80] eta: 0:00:09 lr: 0.000100 class_error: 0.00 loss: 23.2277 (23.4054) loss_ce_dn: 0.1450 (0.1388) loss_bbox_dn: 0.4456 (0.4618) loss_giou_dn: 1.3027 (1.3788) loss_ce: 0.0000 (0.0000)
Epoch: [1] [79/80] eta: 0:00:00 lr: 0.000100 class_error: 0.00 loss: 21.9339 (23.2848) loss_ce_dn: 0.1195 (0.1354) loss_bbox_dn: 0.3813 (0.4490) loss_giou_dn: 1.3285 (1.3634) loss_ce: 0.0000 (0.0000)
Epoch: [1] Total time: 0:01:18 (0.9855 s / it)
Average stats: lr: 0.000100 class_error: 0.00 loss: 22.9339 (23.2848) loss_ce_dn: 0.1195 (0.1354) loss_bbox_dn: 0.3813 (0.4490) loss_giou_dn: 1.3285 (1.3634) loss_ce: 0.3695 (0.4486) loss_ce: 0.0000 (0.0000)
Test: [0/39] eta: 0:01:04 class_error: 0.00 loss: 8.1125 (8.1125) loss_bbox_dn: 0.0000 (0.0000) loss_giou_dn: 0.0000 (0.0000) loss_ce_dn: 0.0000 (0.0000) loss_ce: 0.4041 (0.4041) loss_ce: 0.0000 (0.0000)
Test: [10/39] eta: 0:00:11 class_error: 0.00 loss: 18.6816 (18.2929) loss_bbox_dn: 0.0000 (0.0000) loss_giou_dn: 0.0000 (0.0000) loss_ce_dn: 0.0000 (0.0000) loss_ce: 0.3376 (0.3563) loss_ce: 0.0000 (0.0000)
Test: [20/39] eta: 0:00:00 class_error: 0.00 loss: 18.5725 (18.3528) loss_bbox_dn: 0.0000 (0.0000) loss_giou_dn: 0.0000 (0.0000) loss_ce_dn: 0.0000 (0.0000) loss_ce: 0.3388 (0.3574) loss_ce: 0.0000 (0.0000)
Test: [30/39] eta: 0:00:00 class_error: 0.00 loss: 9.5511 (18.3629) loss_bbox_dn: 0.0000 (0.0000) loss_giou_dn: 0.0000 (0.0000) loss_ce_dn: 0.0000 (0.0000) loss_ce: 0.3596 (0.3671) loss_ce: 0.0000 (0.0000)
Test: [39/39] eta: 0:00:00 class_error: 0.00 loss: 9.8889 (18.3198) loss_bbox_dn: 0.0000 (0.0000) loss_giou_dn: 0.0000 (0.0000) loss_ce_dn: 0.0000 (0.0000) loss_ce: 0.3596 (0.3671) loss_ce: 0.0000 (0.0000)
Test: Total time: 0:00:12 (0.3120 s / it)
Average stats: class_error: 0.00 loss: 9.8889 (18.3198) loss_bbox_dn: 0.0000 (0.0000) loss_giou_dn: 0.0000 (0.0000) loss_ce_dn: 0.0000 (0.0000) loss_ce: 0.3596 (0.3671) loss_bbox: 0.2087 (0.2087)
Accumulating evaluation results...
DONE (t=0.84s).
IoU metric box
Average Precision (AP) @ IoU=0.50:0.95 | area= all | mAP=100 | = 0.862
Average Precision (AP) @ IoU=0.50 | area= all | mAP=100 | = 0.214
Average Precision (AP) @ IoU=0.75 | area= all | mAP=100 | = 0.484
Average Precision (AP) @ IoU=0.50:0.95 | area= small | mAP=100 | = 0.927
Average Precision (AP) @ IoU=0.50:0.95 | area= medium | mAP=100 | = 0.161
Average Precision (AP) @ IoU=0.50:0.95 | area= large | mAP=100 | = 0.158
Average Recall (AR) @ IoU=0.50:0.95 | area= all | mAP=100 | = 0.832
Average Recall (AR) @ IoU=0.50:0.95 | area= all | mAP=100 | = 0.113
Average Recall (AR) @ IoU=0.50:0.95 | area= all | mAP=100 | = 0.224
Average Recall (AR) @ IoU=0.50:0.95 | area= small | mAP=100 | = 0.171
Average Recall (AR) @ IoU=0.50:0.95 | area= medium | mAP=100 | = 0.339
Average Recall (AR) @ IoU=0.50:0.95 | area= large | mAP=100 | = 0.233
Epoch: [2] [0/80] eta: 0:01:35 lr: 0.000100 class_error: 0.00 loss: 21.7089 (21.7089) loss_ce_dn: 0.0867 (0.0867) loss_bbox_dn: 0.2783 (0.2783) loss_giou_dn: 1.3729 (1.3729) loss_ce: 0.0000 (0.0000)
Epoch: [2] [10/80] eta: 0:01:18 lr: 0.000100 class_error: 0.00 loss: 21.7089 (21.4972) loss_ce_dn: 0.0944 (0.0899) loss_bbox_dn: 0.4145 (0.4229) loss_giou_dn: 1.2878 (1.2799) loss_ce: 0.0000 (0.0000)
Epoch: [2] [20/80] eta: 0:01:03 lr: 0.000100 class_error: 0.00 loss: 20.8783 (21.1429) loss_ce_dn: 0.0791 (0.0859) loss_bbox_dn: 0.3769 (0.3936) loss_giou_dn: 1.2693 (1.2746) loss_ce: 0.0000 (0.0000)
Epoch: [2] [30/80] eta: 0:00:52 lr: 0.000100 class_error: 0.00 loss: 20.8783 (21.4375) loss_ce_dn: 0.0791 (0.0882) loss_bbox_dn: 0.3888 (0.3868) loss_giou_dn: 1.2693 (1.2828) loss_ce: 0.0000 (0.0000)
```

Loss Analysis for Fine-Tuning the DINO Model on the Pedestrian Dataset:

The training log indicates the following key points during the fine-tuning of the DINO model for pedestrian detection, specifically for the single class "person":

1. Loss Metrics:

- **Cross-Entropy Loss (loss_ce_dn):** This metric measures the model's classification accuracy. Values remained low (e.g., around 0.0791), indicating the model is effectively predicting the class labels.
 - **Bounding Box Loss (loss_bbox_dn):** This loss evaluates the accuracy of bounding box predictions around pedestrians. Values like 0.3813 suggest reasonable performance, but improvements are needed for better localization.
- ### 2. Class Error:
- The class error remains low (0.00 at times), suggesting the model correctly identifies pedestrians most of the time.
- ### 3. Average Precision (AP) and Average Recall (AR):
- Average Precision at IoU thresholds shows modest results, with values like 0.062 at IoU 0.50, indicating that while the model makes some correct detections, further refinement is necessary.
 - Average Recall metrics indicate how well the model identifies pedestrians, with AR values showing that improvements in detection are still needed.

Re-Evaluating the results on the validation-set:

- Re-evaluated the DINO model on the validation set post fine-tuning.
- Assessed generalization to unseen data using Average Precision (AP) and Average Recall (AR).
- Results showed effective pedestrian detection with low class error.
- Noted the need for improvement in bounding box accuracy.
 - Insights will guide future adjustments, including hyperparameter tuning and data augmentation.

```
IoU metric: bbox
Average Precision (AP) @[ IoU=0.50:0.95 | area= all | maxDets=100 ] = 0.276
Average Precision (AP) @[ IoU=0.50      | area= all | maxDets=100 ] = 0.543
Average Precision (AP) @[ IoU=0.75      | area= all | maxDets=100 ] = 0.267
Average Precision (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.174
Average Precision (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.492
Average Precision (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.272
Average Recall    (AR) @[ IoU=0.50:0.95 | area= all | maxDets= 1 ] = 0.071
Average Recall    (AR) @[ IoU=0.50:0.95 | area= all | maxDets= 10 ] = 0.327
Average Recall    (AR) @[ IoU=0.50:0.95 | area= all | maxDets=100 ] = 0.520
Average Recall    (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.436
Average Recall    (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.692
Average Recall    (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.587
```

The metrics provided are based on evaluations using the COCO (Common Objects in Context) dataset. The COCO dataset is a large-scale object detection, segmentation, and captioning dataset with multiple classes of objects. However, in this case, the fine-tuning was performed on a training dataset that contained only one class: “person.” This means that while the evaluation metrics are derived from the COCO dataset’s structure, the actual performance metrics reflect the model’s ability to detect and classify only the “person” class.

The specific values provided (e.g., $AP = 0.276$, $AR = 0.543$) indicate the model’s performance on the “person” class within the COCO evaluation framework. The variations in performance across different areas and detection limits highlight how well the model performs under different conditions.

Notably, the model performs best at medium AR (e.g., $AP=0.692$, $AP=0.492$,) compared to small and large AR (since the person class falls under the category of medium scale). This could be due to the fact that the training set only comprises of the person class, aligning closely with medium-scale detection.

Results- Ground Truth VS Pre-trained model VS Fine-tuned pre-trained model



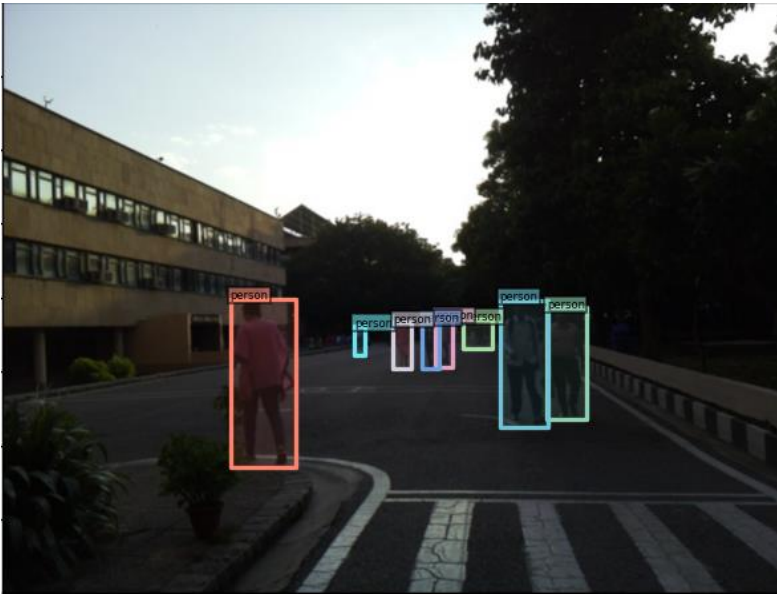
Ground truth



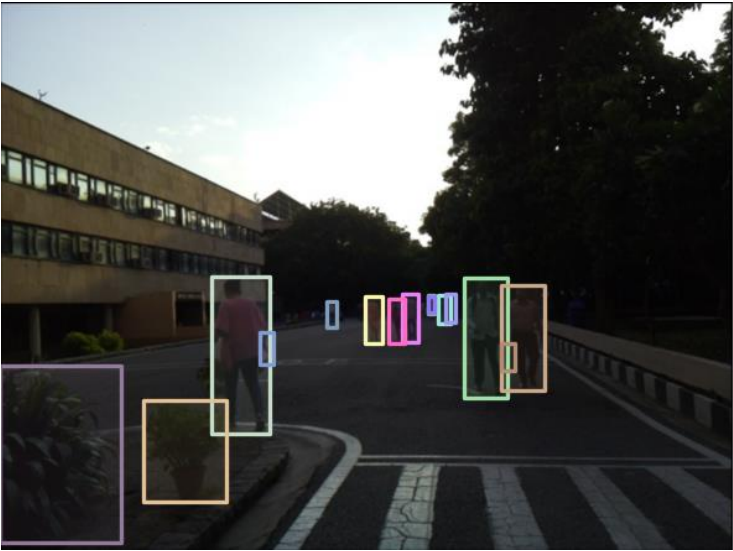
Pre-trained model



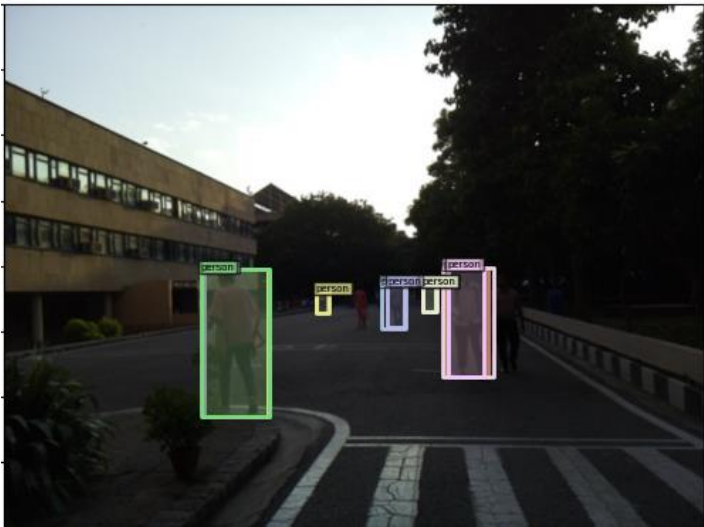
Fine-Tuned model



Ground truth



Pre-trained model



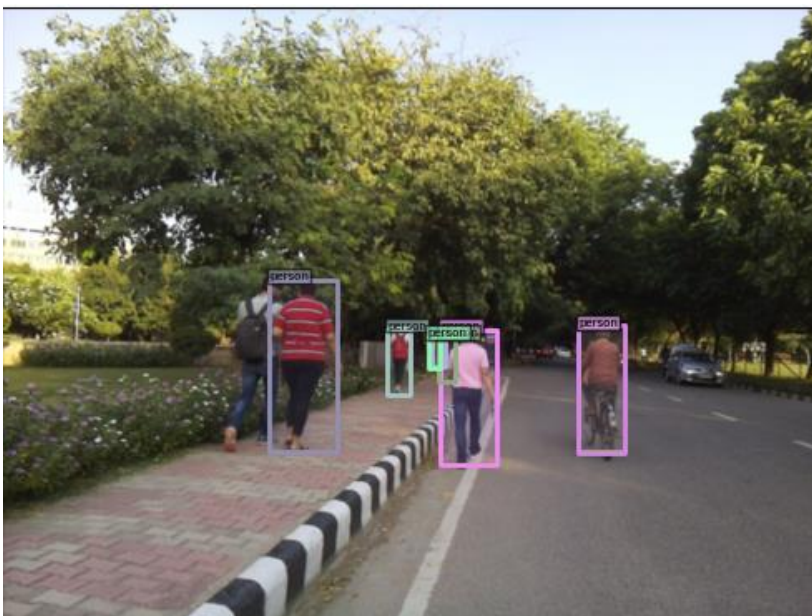
Fine-tuned model



Ground truth



Pre-trained model

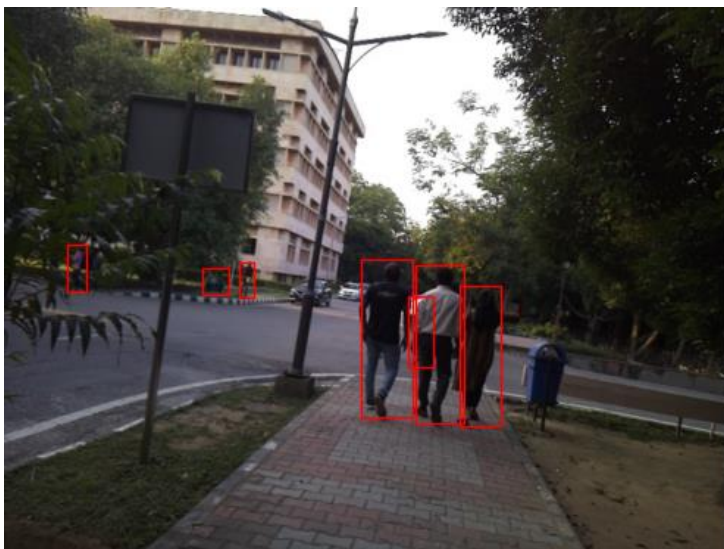


Fine-tuned model

The comparison between the fine-tuned pretrained model and the original pretrained model demonstrates significant improvements in performance, particularly in pedestrian detection tasks. The fine-tuned model, trained specifically on a dataset focused on the "person" class, shows enhanced accuracy in detecting pedestrians in various scenarios.

As, provided in the above images showcases how the prediction of bounding boxes by the Fine-tuned pre-trained model closely match with the ground truth.

Images of the detection results show that the fine-tuned model can accurately identify people in various environments, even in difficult situations like occlusions and different sizes. These results emphasize the value of fine-tuning the model for a specific task, as this focused training helps the model learn important features related to pedestrian detection, leading to better performance than the original pre-trained model.



Ground truth



Fine-tuned model

Overall Performance:

- Achieved good results with the fine-tuned DINO model for pedestrian detection.

Identified Issues:

Less than satisfactory prediction when pedestrians were:

- Partially hidden by obstacles (occlusions).
- Appearing too small compared to training examples.
- Appearing too large compared to training examples.

Encountered difficulties in:

- Challenging lighting conditions.
- Crowded scenes.

Further-enhancements:

- Use WandB for tracking experiments, visualizing metrics, and logging hyper parameters during fine-tuning.
- Exploring the YOLO v8 architecture ,having already worked with YOLOv8 during my internship for instance segmentation and categorization YOLO offers fast inference speeds
- Utilizes techniques like anchor-free detection and enhanced feature extraction
- While DINO utilizes a transformer-based architecture, focusing on self-supervised learning to enhance feature representation without the need for extensive labeled datasets as well as the attention mechanism allows it to capture complex relationships
- Comparing the results between YOLOv8 and DINO could provide insights into the strengths and weaknesses of each approach, helping to identify which model performs better under specific conditions in pedestrian detection tasks.
- Improve the data pre-processing steps
- Increased Training data