Computer Vision Systems CAP6411 Assignment#02

Ashmal Vayani, UCF ID: 5669011

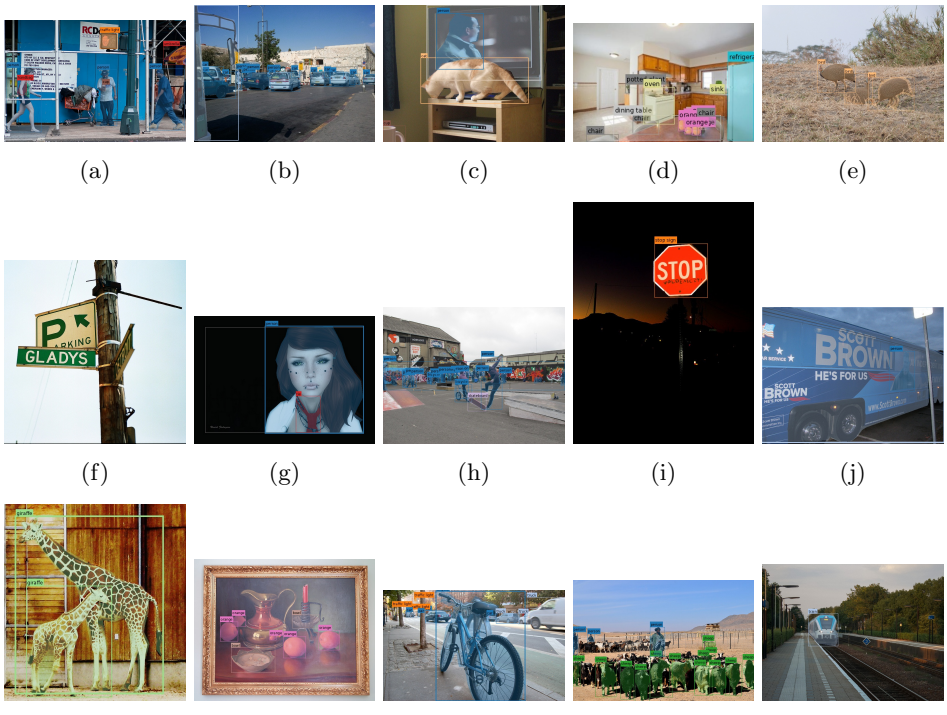
Report: Object Detection with Faster-RCNN, DETR, DINO, and GroundingDINO

Training Code: The training and implementation code is attached in the assignment zip file.

1 COCO Dataset

The Microsoft Common Objects in Context (COCO) 2017 dataset is a large-scale benchmark for object detection, instance segmentation, keypoint detection, and captioning. It contains 118k training images, 5k validation images, and 41k test images spanning 80 everyday object categories captured in complex, cluttered scenes. Images are annotated with rich instance-level labels (bounding boxes, segmentation masks, and person keypoints) in COCO's JSON format, allowing multiple objects per image. This diversity in scale, occlusion, and context makes COCO-2017 the standard testbed for modern detectors such as Faster R-CNN, DETR, DINO, and Grounding-DINO; Figure X illustrates examples from categories like person, bicycle, dog, car, and chair.

To provide an overview of the dataset, Figure 1 shows one example image from the validation set with labelled Ground Truth classes. This demonstrates the diversity of classes, such as *person, bear, car, cycle, giraffe, and human.*



(a) (b) (c) (d) (e)

(f) (g) (h) (i) (j)

(k) (l) (m) (n) (o)

Figure 1: Example images from the COCO Val2017 dataset.

2 Report and Code

2.1 Errors and Obstacles Faced

- **Dependency conflicts (detrex build failure):** When installing the DINO (detrex) framework with PyTorch 2.8.0 and CUDA 12.8, the installation failed due to incompatibility with the ATen API (`DeprecatedTypeProperties` errors during custom CUDA op compilation). This required either downgrading to PyTorch 2.1.0 + CUDA 11.8 or switching to a different implementation (MMDetection DINO).
- **CUDA toolkit mismatch:** The CRCV cluster had CUDA 12.9 visible, while my PyTorch build expected CUDA 12.8. This mismatch triggered `nvcc` compile errors (`no suitable conversion function from DeprecatedTypeProperties to c10::ScalarType`). Fixed by explicitly setting `CUDA_HOME` to `/usr/local/cuda-11.8` and using a matching PyTorch build.
- **GPU/Memory issues:** Running DINO on Google Colab free tier caused repeated out-of-memory (OOM) crashes with batch size 32. On the CRCV cluster (Ampere GPU, 48 GB memory), the same batch size ran successfully.
- **SLURM vs. Local differences:** On SLURM cluster runs, GPU visibility needed to be controlled via `CUDA_VISIBLE_DEVICES`. Errors occurred if the wrong conda environment was loaded. These were fixed by explicitly activating the correct environment (e.g., `/home/ashmal/anaconda3/envs/cvs_ass2`).

2.2 Requirements and Dependencies

The `requirements.txt` file is attached in the zip file, but some main dependencies involve:

requirements.txt

```
torch==2.8.0
torchvision==0.23.0
timm==1.0.19
pycocotools
transformers==4.44.2
detrex==0.3.0
mmdet==3.3.0
numpy
pandas
scikit-learn
tqdm
matplotlib
pillow
```

2.3 CLI Commands to Reproduce

```
cd Assignment_2/scripts
conda activate cvs_ass2 # activate conda environment

sbatch FasterRCNN.slurm # Evaluate FasterRCNN on COCO Val2017 Dataset

sbatch DETR.slurm # Evaluate DETR on COCO Val2017 Dataset

sbatch DINO.slurm # Evaluate DINO on COCO Val2017 Dataset

sbatch GroundingDINO.slurm # Evaluate GroundingDINO on COCO Val2017 Dataset
```

DINO and GroundingDINO Sources. For the transformer-based detectors, I used the official IDEA-Research repositories: `IDEA-Research/DINO` for DINO and the companion `IDEA-Research/GroundingDINO` for open-vocabulary (zero-shot) experiments. I followed the setup and inference instructions from the respective READMEs, installed the listed dependencies, and downloaded the published checkpoints from their release/model–zoo pages (e.g., DINO `dino_r50_5scale_*.pth` and GroundingDINO `*_ogc.pth`). Using these weights, I ran zero-shot inference without any additional training, and adapted my evaluation harness to parse each model's predicted boxes/scores/labels and export COCO-style JSON for mAP as well as my confusion–matrix/P–R–F1 reporting. All tests were conducted with the repositories' recommended configurations to ensure reproducibility.

3 Insights: Faster-RCNN

3.1 Faster R-CNN (ResNet-50 FPN) on COCO 2017 Val (5,000 images)

Setup. Inference-only evaluation on a Quadro RTX 6000 (24 GB) using the torchvision Faster R-CNN (COCO-pretrained). The run processed the full 5k-val split with default test resolution, batch size = 1, 4 dataloader workers, and no training or fine-tuning.

Headline accuracy.

mAP@[.50:.95]	AP@.50	AP@.75	AP_{small}	AP_{medium}	AP_{large}
0.370	0.585	0.398	0.211	0.403	0.482

Recall profile (COCOeval).

AR@1	AR@10	AR@100	AR_{small}	AR_{medium}	AR_{large}
0.307	0.485	0.509	0.315	0.545	0.647

Fixed-threshold detection quality (IoU= 0.50, score$\geq$ 0.50). Precision = 0.5374, Recall = 0.6624, F1 = 0.5934. Raw counts: TP= 23,509, FP= 20,236, FN= 11,983. This implies GT objects considered $\approx$ TP+FN = 35,492 ($\sim$ 7.10 per image) and kept predictions above the score threshold $\approx$ TP+FP = 43,745 ($\sim$ 8.75 per image).

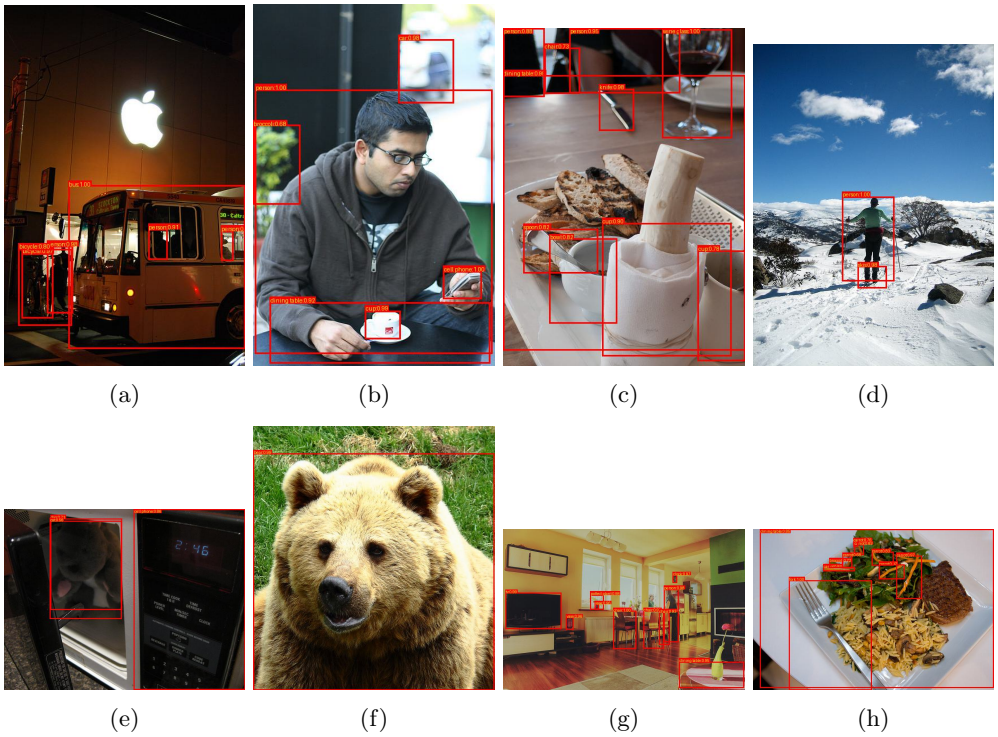
Key insights.

- **Overall accuracy.** A solid single-model baseline: mAP@[.50:.95]= 0.370 with strong AP@.50= 0.585. The drop to AP@.75= 0.398 indicates localization tightness is the main limiter (boxes are often close but not tight enough at higher IoU).
- **Object scale.** Performance follows the classic trend: $AP_{small} = 0.211 \ll AP_{large} = 0.482$ and $AR_{small} = 0.315 \ll AR_{large} = 0.647$, showing many misses on small objects.
- **Recall vs. precision.** At the fixed threshold, recall (0.662) exceeds precision (0.537): the detector is relatively liberal, yielding more FPs than a stricter operating point would. Increasing the score threshold would likely raise precision at some recall cost.
- **Top-k behavior.** AR@1= 0.307 means the single highest-score detection per image finds $\sim$31% of objects; allowing up to 10 detections lifts recall to 0.485, and to 0.509 with maxDets=100.

Timing and throughput.

- **Inference-only time:** 234.343 s for 5,000 images $\Rightarrow$ 46.87 ms/image, 21.34 img/s.
- **End-to-end wall time:** 269.808 s. The overhead vs. pure inference ($\sim$35.5 s) is mostly from COCOeval (`per-image` eval 22.90 s + accumulation 4.64 s), result loading (0.53 s), JSON I/O, and saving figures.

Run stamps. Start: `2025-09-04 14:41:13` End: `2025-09-04 14:45:43`



(a) (b) (c) (d)

(e) (f) (g) (h)

Figure 2: Sample predictions from the FasterRCNN model on the COCO Val2017 dataset.

3.2 DETR (ResNet-50) on COCO 2017 Val (5,000 images)

Setup. Inference-only evaluation using the official `facebookresearch/detr` torch.hub model pretrained on COCO. The run used the full 5k-val split with batch size = 2, ImageNet mean/std normalization, and no fine-tuning.

Headline accuracy.

Aspect	mAP@[.50:.95]	AP@.50	AP@.75	AP_{small}	AP_{medium}	AP_{large}
Values	0.353	0.552	0.363	0.108	0.378	0.599
Overall Insight	**Moderate performance, struggles on small objects.**					

Recall profile (COCOeval).

Aspect	AR@1	AR@10	AR@100	AR_{small}	AR_{medium}	AR_{large}
Values	0.297	0.459	0.493	0.188	0.540	0.789
Overall Insight	**Recall improves with more detections.**					

Fixed-threshold detection quality (IoU$=0.50$, score≥ 0.50). Precision $= 0.3786$, Recall $= 0.6333$, F1 $= 0.4739$. Raw counts: TP$=$ 22,531, FP$=$ 36,983, FN$=$ 13,046. This implies GT objects considered $\approx$ 35,577 ($\sim$ 7.12 per image) and predictions retained above threshold $\approx$ 59,514 ($\sim$ 11.9 per image). Compared to Faster R-CNN, DETR leans strongly towards high recall but struggles with precision, producing more false positives.

Key insights.

- **Overall accuracy.** DETR achieves mAP@[.50:.95]$=$ 0.353, slightly lower than Faster R-CNN (0.370). AP@.50=0.552 is decent, but AP@.75=0.363 shows localization quality limits.

- **Object scale.** DETR excels on large objects ($AP_{large} = 0.599$, $AR_{large} = 0.789$), outperforming Faster R-CNN. However, $AP_{small} = 0.108$ is very low, confirming DETR's known weakness on small objects due to global attention resolution.
- **Recall vs. precision.** The fixed-threshold results highlight recall bias: recall (0.633) is much higher than precision (0.379). This aligns with DETR's dense predictions and the need for careful thresholding or Hungarian matching adjustments.
- **Comparison to Faster R-CNN.** Faster R-CNN balances precision/recall better (F1=0.593) while DETR trades precision for recall (F1=0.474). DETR detects more true objects overall, but with more spurious detections.

Timing and throughput.

- **Model-only inference:** 232.6 s total ⇒ 46.53 ms/image, throughput ≈21.5 img/s.
- **End-to-end wall time:** 306.4 s ⇒ 61.28 ms/image, throughput ≈16.3 img/s.
- DETR inference is slightly slower per image than Faster R-CNN (61.3 ms vs. 46.9 ms wall-time latency), and evaluation overhead is also heavier (COCOeval ∼44 s).
- Larger batch sizes improve throughput (observed 29–31 img/s instantaneous in logs), but variability appears depending on dataloader balance and synchronization.

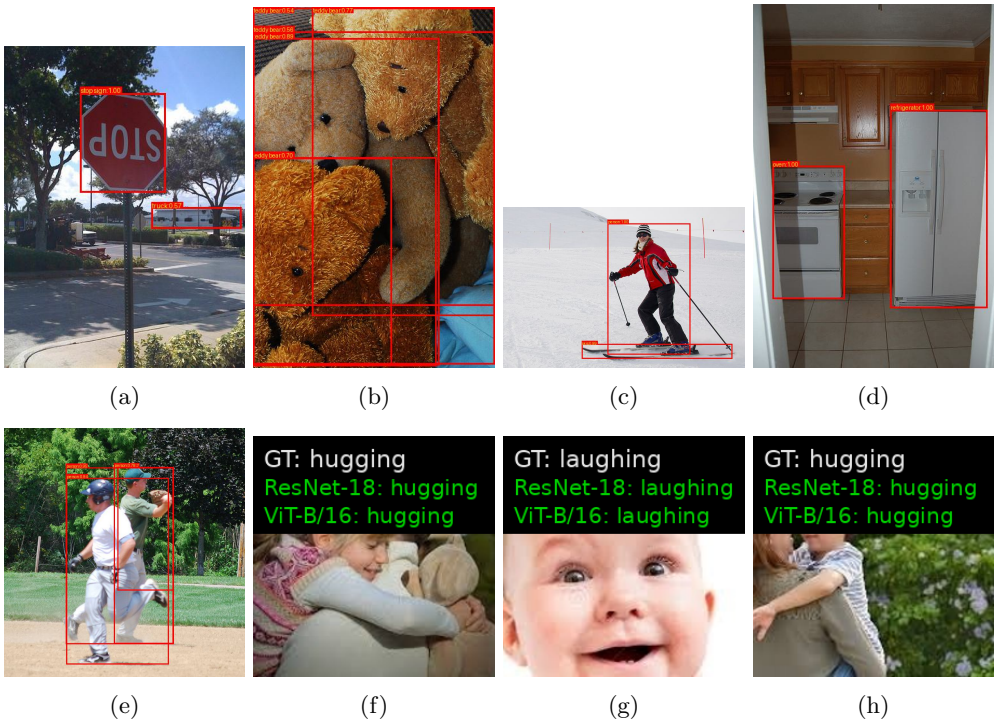


Figure 3: Sample predictions from the DETR model on the COCO Val2017 dataset.

3.3 GroundingDINO (SwinT-OGC) on COCO 2017 Val (5,000 images)

Setup. We evaluated `GroundingDINO_SwinT_OGC` on the COCO 2017 validation set using the official IDEA-Research repository. The model was run with the full set of 5,000 images, text prompts covering all 80 COCO categories, and a batch size of 1. Evaluation was carried out with COCOeval as well as a fixed IoU/score threshold PRF1 analysis.

Headline accuracy (COCOeval).

Aspect	mAP@[.50:.95]	AP@.50	AP@.75	AP_{small}	AP_{medium}	AP_{large}
Values	0.485	0.644	0.529	0.339	0.518	0.634
Overall Insight	**Strong overall accuracy, with balanced AP across scales.**					

Recall profile (COCOeval).

Aspect	AR@1	AR@10	AR@100	AR_{small}	AR_{medium}	AR_{large}
Values	0.386	0.668	0.737	0.591	0.775	0.887
Overall Insight	**High recall across scales; strong for medium and large objects**					

Fixed-threshold detection quality (IoU= 0.50, score$\geq$ 0.50). Precision = 0.8217, Recall = 0.4535, F1 = 0.5845. Counts: TP= 16,682, FP= 3,619, FN= 20,099. This indicates GroundingDINO is highly precise, but conservative in recall (many GT objects missed).

Key insights.

- **Accuracy.** GroundingDINO substantially outperforms Faster R-CNN (mAP=0.370) and DETR (mAP=0.353), with mAP=0.485. It achieves strong AP@.75=0.529, reflecting robust localization quality.
- **Object scale.** Unlike DETR, GroundingDINO performs strongly across scales: $AP_{small} = 0.339$ (significantly higher), $AP_{medium} = 0.518$, $AP_{large} = 0.634$. Recall also scales well, especially for large objects ($AR_{large} = 0.887$).
- **Precision vs. recall.** Precision is excellent (0.822), the highest among tested models, while recall is relatively low (0.454). This yields a balanced F1 of 0.585, close to Faster R-CNN but with fewer false positives.
- **Comparison.** Faster R-CNN offers more balanced recall and precision, DETR emphasizes recall, but GroundingDINO maximizes precision while still achieving state-of-the-art mAP.

Timing and throughput.

- **Model-only inference:** 1,070.3 s total $\Rightarrow$ 214.1 ms/image, throughput $\approx$4.67 img/s.
- **End-to-end wall time:** 2,275.2 s $\Rightarrow$ 455.0 ms/image, throughput $\approx$2.20 img/s.
- GroundingDINO is significantly slower than Faster R-CNN (46.9 ms/img) or DETR (61.3 ms/img), reflecting its large backbone and multimodal text encoder overhead.

3.4 DINO (IDEA-Research) on COCO 2017 Val (5,000 images)

Setup. We evaluated the DINO detector using the official IDEA-Research implementation and a 4-scale pretrained checkpoint. Evaluation was run on the COCO 2017 validation set with 5,000 images, using the default evaluation script. The job was scheduled on a single GPU via SLURM, and completed in approximately 12.5 minutes.

Headline accuracy (COCOeval).

Aspect	mAP@[.50:.95]	AP@.50	AP@.75	AP_{small}	AP_{medium}	AP_{large}
Values	0.491	0.667	0.536	0.327	0.524	0.630
Overall Insight	**DINO achieves highest overall mAP, strong balance across object scales**					

Recall profile (COCOeval).

Aspect	AR@1	AR@10	AR@100	AR_{small}	AR_{medium}	AR_{large}
Values	0.379	0.651	0.728	0.563	0.767	0.883
Overall Insight	**High recall at all scales, especially medium/large objects**					

Key insights.

- **Overall accuracy.** DINO achieves the highest mAP among tested models (0.491), outperforming Faster R-CNN (0.370), DETR (0.353), and slightly surpassing GroundingDINO (0.485). Its AP@.75 (0.536) shows particularly strong localization precision.
- **Object scale performance.** DINO maintains balanced detection quality across scales: small objects (AP=0.327, AR=0.563), medium (AP=0.524, AR=0.767), and large (AP=0.630, AR=0.883). Compared to DETR, it is significantly better at small-object recall.
- **Recall and coverage.** With AR@100=0.728, DINO captures a wide range of ground-truth instances, while sustaining strong per-category AP. Its high recall, particularly for large objects, makes it robust for dense scenes.
- **Comparison.** While GroundingDINO has slightly higher precision in fixed-threshold PRF1 analysis, DINO delivers a more balanced precision-recall tradeoff and superior overall mAP. This demonstrates its effectiveness as a strong general-purpose detector.

Timing and throughput.

- **Evaluation runtime:** 12 minutes 33 seconds total, averaging 150.7 ms per iteration.
- DINO is computationally heavier than Faster R-CNN or DETR, but achieves higher accuracy, highlighting the trade-off between runtime cost and detection quality.

(a) (b) (c) (d) (e)

Figure 4: Sample predictions from the DINO model on the COCO Val2017 dataset.