CAP6411 Assignment 2: Object Detection

Joshua Yu

September 4, 2025

1 Introduction

This is a short write-up on results obtained for Assignment 1: Object Detection. In this assignment, the author was tasked with evaluating various object detection models and comparing the results. The models used were Faster R-CNN, DETR, DINO, and grounding DINO. The models were evaluated on the COCO 2017 dataset. The video demo can be found at the following link: https://youtu.be/jwuDmlolCv8

All experiments were performed on the author's personal laptop with an NVIDIA RTX 3060 GPU.

2 Faster R-CNN

Faster R-CNN is an older but influential object detection model that utilizes a convolutional neural network as well as a region proposal network and detection head. For this assignment, the specific model used was "faster-rcnn_resnet50_fpn" from the torchvision library. The model utilizes ResNet50 as its convolutional neural network backbone. The results of the model as evaluated on the COCO2017 dataset validation split are shown in the table below.

Table 1: Faster R-CNN Performance Metrics on COCO

Metric	**Score**
AP @[IoU=0.50:0.95 — area=all]	0.369
AP @[IoU=0.50 — area=all]	0.585
AP @[IoU=0.75 — area=all]	0.398
AP @[IoU=0.50:0.95 — area=small]	0.211
AP @[IoU=0.50:0.95 — area=medium]	0.403
AP @[IoU=0.50:0.95 — area=large]	0.482
AR @[maxDets=1]	0.307
AR @[maxDets=10]	0.485
AR @[maxDets=100]	0.509
AR @[area=small]	0.315
AR @[area=medium]	0.545
AR @[area=large]	0.647
Average Inference Time (ms)	79.92
Average Memory Usage (MB)	1475.62

One notable metric is the AP (Average Precision) which was 36.9%. This is respectable but not incredible, which is expected of such an older system. Another notable metric is the AP for small objects, where it only achieved 21.1%. This shows that the model really struggles with smaller objects. Additionally, the average inference times of 79.92 ms., which comes out to about 12.5 frames per second, means that the model is quite slow and not suitable for real time applications.

3 DETR

DETR, which stands for DEtection TRansformer, is an object detection model that utilizes transformers. It was revolutionary for its simplicity; it eliminated many of the complex, hand-crafted components necessary for Faster

R-CNN. DETR is an end-to-end system and eliminated the need for a region proposal network as well as anchor boxes and non-max suppression. For this experiment, the specific model used was "detr-resnet-50" from the Huggingface transformers library. The model still utilizes ResNet50 as a convolutional neural network backbone for extracting features, but features are processed with a transformer. The results of the model as evaluated on the COCO2017 dataset validation split are shown in the table below.

Table 2: DETR Performance Metrics on COCO

Metric	**Score**
AP @[IoU=0.50:0.95 — area=all]	0.402
AP @[IoU=0.50 — area=all]	0.592
AP @[IoU=0.75 — area=all]	0.423
AP @[IoU=0.50:0.95 — area=small]	0.191
AP @[IoU=0.50:0.95 — area=medium]	0.436
AP @[IoU=0.50:0.95 — area=large]	0.588
AR @[maxDets=1]	0.319
AR @[maxDets=10]	0.475
AR @[maxDets=100]	0.488
AR @[area=small]	0.244
AR @[area=medium]	0.530
AR @[area=large]	0.693
Average Inference Time (ms)	41.93
Average Memory Usage (MB)	1499.34

One notable metric is the AP (Average Precision) which was 40.2%. This is significantly better than Faster R-CNN, which is good and expected. However, is it interesting that it actually performs worse than Faster R-CNN for small objects, with its AP for small objects coming in at 19.1%. This is possibly due to its global attention mechanism, which sometimes struggles with fine details. For speed, the average inference time was 41.93 ms., which comes out to about 23.8 frames per second. This is much faster than R-CNN and makes the model usable for some real-time applications.

4 DINO

DINO, which stands for **D**ETR with **I**mproved De-**N**oising anch**O**rs, is a model which builds on DETR for improved performance and stability. Of note is the use of contrastive denoising training, where the model is given noisy ground-trytgh boxes and is trained to reconstruct clean boxes. Through this refining process, DINO improves upon DETR with faster training convergence and better performance on small objects. For this experiment, the specific model used was "checkpoint0011_4scale.pth" from the official DINO Github repository. The results of the model as evaluated on the COCO2017 dataset validation split are shown in the table below.

DINO achieves very good performance with an average precision of 49.1 percent. Additionally, it is much better than Faster RCNN and DETR with small objects, jumping from $\tilde{2}0\%$ to 32.7%. Additionally, it has excellent average recall at 72.8%. This means that ir finds a very high percentage of the objects in an image.

5 Grounding DINO

Grounding DINO is an extension of DINO that is uniquely powerful with its understanding of natural language. Grounding DINO is highly flexible and offers "open-vocabulary" detection for any text prompt instead of being bound to a set of predefined classification categories. Grounding DINO enhances DINO by adding a text transformer which fuses an image and a text prompt, allowing the model to detect anything that can be described with words. For this experiment, the specific model used was "groundingdino_swint_ogc" from the official Grounding DINO Github repository. The results of the model as evaluated on the COCO2017 dataset validation split are shown in the table below.

Grounding DINO shows great performance with an average precision of 48.5%. Is it nearly as accurate as the regular DINO model while offering infinitely more flexibility with its open-ended text prompting system. Additionally, it is even more performant with small objects, coming in with an AP of 33.9%. It is a very powerful model.

Table 3: DINO Performance Metrics on COCO

Metric	Score
AP @[IoU=0.50:0.95 — area=all]	0.491
AP @[IoU=0.50 — area=all]	0.667
AP @[IoU=0.75 — area=all]	0.536
AP @[IoU=0.50:0.95 — area=small]	0.327
AP @[IoU=0.50:0.95 — area=medium]	0.523
AP @[IoU=0.50:0.95 — area=large]	0.630
AR @[maxDets=1]	0.379
AR @[maxDets=10]	0.651
AR @[maxDets=100]	0.728
AR @[area=small]	0.563
AR @[area=medium]	0.768
AR @[area=large]	0.884

Table 4: Grounding DINO Performance Metrics on COCO

Metric	Score
AP @[IoU=0.50:0.95 — area=all]	0.485
AP @[IoU=0.50 — area=all]	0.644
AP @[IoU=0.75 — area=all]	0.529
AP @[IoU=0.50:0.95 — area=small]	0.339
AP @[IoU=0.50:0.95 — area=medium]	0.518
AP @[IoU=0.50:0.95 — area=large]	0.634
AR @[maxDets=1]	0.386
AR @[maxDets=10]	0.668
AR @[maxDets=100]	0.737
AR @[area=small]	0.590
AR @[area=medium]	0.775
AR @[area=large]	0.887

6 Overall Results & Insights

Below is a summary tables of the models compared to each other.

Table 5: Comparison of Object Detection Model Performance

Metric	Faster R-CNN	DETR	DINO	Grounding DINO
AP @[0.50:0.95]	0.369	0.402	**0.491**	0.485
AP_{small}	0.211	0.191	0.327	**0.339**
AP_{medium}	0.403	0.436	**0.523**	0.518
AP_{large}	0.482	0.588	0.630	**0.634**
AR @[maxDets=100]	0.509	0.488	0.728	**0.737**
Inference Time (ms)	79.92	**41.93**	-	-
Memory Usage (MB)	**1475.62**	1499.34	-	-

Of all the models, Faster R-CNN performs the worst by a large margin. As mentioned in previous sections, Faster R-CNN obtained an average precision of 36.9% while all the other models score above 40%. Most of the other models also score significantly better in the detection of small objects; however, DETR is an exception. As an earlier transformer-based model, its global attention mechanism shows some deficiencies in detecting fine details. However, overall, it is clear that the huge gap between Faster R-CNN and the other models shows that transformer-based architectures are the way to go for object detection. Their simple yet powerful leverage of transformers and attention

gives them a fundamental advantage.

The best performer of all the models was DINO. DINO improved on DETR by not only improving accuracy overall, but by significantly improving on small object detection. This shows that the better training techniques used in DINO are powerful enough to warrant significant improvement.

As for grounding DINO, it is impressive how close it gets to DINO while being infinitely more flexible. The minimal drop in average precision is a worthwhile trade-off for the ability to detect almost anything. Grounding DINO is a look at the future of object detection and computer vision in general. The marriage of computer vision and natural language is very useful and especially impressive given the minimal hit to performance.

7 Challenges

By far the greatest challenge was getting the actual models to run and then packaging them up in a format that could be uploaded for this assignment. At every step of the way, there were missing modules or messed-up syntax from older libraries. The assignments so far have proven to be more of an exercise in handling Python dependencies than computer vision. DINO was particularly difficult to get running, as the repository required CUDA compilation and dependencies that could not be solved with pip or conda.

8 Conclusion

The performance of multiple object detection models was demonstrated in this experiment. It was shown that DINO was the best performer, and Faster R-CNN was the worst. The delta between Faster R-CNN and the other transformer-based models shows that transformer-based models are the future. Experimentation was surprisingly difficult as there were a lot of issues with dependencies.