Comparative Evaluation of Object Detection Models: YOLO, DETR, ViT-DETR, and OWL-ViT2

Mohammadkazem Rajabi
University of Padova

Abstract

Object detection remains a key challenge in computer vision, with increasing interest in both high-performance closed-set models and flexible open-vocabulary systems. This study presents a comparative evaluation of modern architectures, including ***YOLOv8****(7),* ***DETR****(2), and the vision-language model* ***OWL-ViT2****(5), on the PASCAL VOC 2012 dataset(3).*

We fine-tune convolutional and transformer-based models—including YOLOv8 variants and DETR with ResNet50 and Resnet101 backbones—to assess how architecture and scale affect detection accuracy, speed, and robustness. All models use standardized splits and augmentations for fair comparison.

We also evaluate zero-shot detection with OWL-ViT2, which recognizes unseen categories via vision-language alignment, enabling a contrast with supervised models in terms of flexibility and real-world use.

Evaluations include ***mAP@0.5****,* ***precision****,* ***recall****, and inference time, along with qualitative analysis. Our results highlight tradeoffs between CNN-based and transformer-based detectors, offering guidance for model selection in practical applications.*

1. Introduction

1.1. Object Detection

Object detection is one of those core tasks in computer vision that's easy to describe—finding and labeling objects in images—but deceptively complex to do well. It's not just about recognizing what's in an image; it's about saying where each thing is, and doing it reliably, even when objects are small, overlapping, or half-hidden.

There are two main types of detectors people use today:

One-stage detectors, like YOLO or SSD, are built for speed. They predict object locations and classes in one go, making them ideal for real-time applications like self-driving cars or live video analysis.

In recent years, the game has shifted again with Transformer-based models. DETR, for example, uses the same architecture that revolutionized NLP and applies it to detection—treating object detection as a set prediction problem. It skips the hand-crafted parts like anchor boxes and post-processing, and just learns everything end-to-end. Vision Transformers (ViT) take this even further by treating image patches like tokens in a sentence and processing them with attention mechanisms, often resulting in strong performance, especially on complex or cluttered scenes.

2. Dataset

We used the **PASCAL VOC 2012** dataset, obtained in a curated version from Roboflow, to train and evaluate the object detection models. This dataset is a widely used benchmark in the object detection community, offering a diverse set of annotated real-world images across 20 categories.

2.1. Overview

Each object in the dataset is labeled with a bounding box and class name. Common categories include *person*, *car*, *dog*, and *bicycle*. We used the following Roboflow-hosted version:

- **Dataset Source:** `https://universe.roboflow.com/jacob-solawetz/pascal-voc-2012/dataset/13`
- **Total Images:** 17,000 (pre-split into train/val/test)
- **Number of Classes:** 20

2.2. Data Preparation

Roboflow's export tools were used to convert the dataset into formats suitable for each model:

- **YOLOv8:** Exported in YOLO format (TXT files with normalized coordinates). The Ultralytics training interface handled label parsing, resizing, and augmentation.
- **DETR:** The original XML annotations were converted to COCO-style format using a custom PyTorch loader.

Standard augmentations such as resizing, flipping, and normalization were applied.

2.3. Class Mapping

We applied a consistent mapping from class names to integer indices for all models. The background class was handled implicitly by the training framework and is not listed.

2.4. Why PASCAL VOC 2012?

We selected this dataset for the following reasons:

- It is a well-known benchmark widely used in object detection research.
- It includes clean annotations and multiple object instances per image.
- It is lightweight enough for fast experimentation while offering sufficient diversity for evaluation.
- Roboflow allows easy export to formats compatible with both YOLO and DETR.

3. Model Architectures

This section provides an overview of the object detection architectures evaluated in this study, focusing on three major paradigms in object detection: single-stage detectors, transformer-based methods, and two-stage region proposal frameworks.

3.1. YOLOv8

YOLO (You Only Look Once) revolutionized object detection by treating it as a single regression problem. Unlike traditional region-based approaches such as R-CNN and Faster R-CNN, which rely on multi-stage pipelines, YOLO applies a single neural network to the entire image to simultaneously predict bounding boxes and class probabilities. This significantly improves inference speed.

YOLOv8 incorporates several key advancements, including anchor-free detection, decoupled head architectures, and improved feature extraction. In our study, we evaluate four YOLOv8 variants (small, medium, large,and extra-large) to analyze the impact of model size on performance. The results of these evaluations are discussed in Section 5.

3.1.1 YOLO Architecture

Grid-Based Approach:
The input image is divided into an $S \times S$ grid. Each grid cell is responsible for detecting objects whose center falls within that cell. Each grid cell predicts:

- B bounding boxes, each defined by:

$$(x, y, w, h, c) \tag{1}$$

  where:

  - x, y - Center coordinates of the bounding box (relative to the grid cell).
  - w, h - Width and height of the bounding box (relative to the entire image).
  - c - Confidence score, defined as:

$$c = P(\text{Object}) \times IOU_{\text{pred, truth}} \tag{2}$$

- C class probabilities for each object category.

Thus, the final output tensor size is:

$$S \times S \times (B \times 5 + C) \tag{3}$$

3.1.2 Network Design

The YOLO network consists of two main parts:

- **Feature Extractor:** The first part is a deep convolutional neural network (CNN) that extracts features from the input image.
- **Detector:** The second part consists of fully connected layers that predict bounding boxes and class probabilities.

Convolutional Layers YOLO uses 24 convolutional layers inspired by the GoogLeNet architecture. Instead of inception modules , it uses 1×1 convolutions followed by 3×3 convolutions for dimensionality reduction. Fully Connected Layers After feature extraction, YOLO uses two fully connected layers to predict:

- Bounding box coordinates (x, y, w, h)
- Confidence scores (c)
- Class probabilities $(P(\text{Class}|\text{Object}))$

we can see the architecure

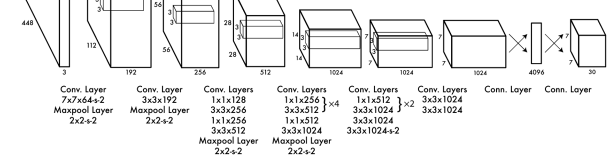

Figure 1. Yolo architecture.

3.1.3 YOLO-NAS: Neural Architecture Search-Based Detection

YOLO-NAS (You Only Look Once with Neural Architecture Search) is a recent object detection model developed by Deci AI (1). Unlike traditional YOLO variants, which are designed manually, YOLO-NAS leverages Neural Architecture Search (NAS) to automatically discover the most efficient architecture for object detection tasks.

This automated design process evaluates thousands of architectural configurations to find an optimal balance between accuracy, latency, and computational efficiency. As a result, YOLO-NAS achieves higher mAP and lower latency compared to earlier YOLO models like YOLOv5.

In addition to architectural improvements, YOLO-NAS integrates post-training quantization and pruning techniques to optimize model size and inference speed, making it suitable for deployment on resource-constrained devices.

According to Roboflow's evaluation (9), YOLO-NAS models are available in various sizes (S, M, L), and surpass previous YOLO variants on benchmarks such as COCO and Pascal VOC, often without requiring Non-Maximum Suppression (NMS).

3.2. DETR

DETR(2) presents a paradigm shift by treating object detection as a direct set prediction problem. Instead of relying on heuristics such as anchor boxes, DETR employs a transformer-based architecture that predicts all objects in a single pass. This end-to-end approach eliminates the need for region proposals and post-processing techniques, making the model both conceptually simpler and more generalizable.

3.2.1 DETR Architecture

The DETR model is composed of three main components:

1. **CNN Backbone:** Extracts feature maps from the input image.
2. **Transformer Encoder-Decoder:** Processes and refines features using self-attention mechanisms.
3. **Prediction feed-forward networks (FFNs):** Produces the final object classes and bounding boxes.

3.2.2 CNN Backbone

The CNN backbone processes the input image $x_{\text{img}} \in \mathbb{R}^{3\times H_0\times W_0}$, where H_0 and W_0 are the original height and width of the image. The backbone extracts feature representations and produces a lower-resolution feature map:

$$f \in \mathbb{R}^{C\times H\times W}, \tag{4}$$

where C is (typically 2048) and

$$H = \frac{H_0}{32}, \quad W = \frac{W_0}{32}. \tag{5}$$

This feature map is then passed into the transformer encoder.

3.2.3 Transformer Encoder

Before passing the feature map into the transformer encoder, a 1×1 convolution is applied to reduce the number of channels from C to a smaller dimension d. This results in a new feature map:

$$z_0 \in \mathbb{R}^{d\times H\times W}. \tag{6}$$

Since transformers operate on sequences rather than spatial feature maps, the spatial dimensions $H \times W$ are flattened into a single sequence of length HW, forming an input tensor of shape $d \times HW$.

3.2.4 Transformer Decoder

The transformer decoder in DETR follows the standard architecture of transformers and is responsible for converting the encoded image features into object predictions. Unlike conventional transformer decoders, which generate sequences in an autoregressive manner (one token at a time), DETR's decoder operates in parallel, predicting all objects simultaneously.

3.2.5 Parallel Decoding of Object Queries

The input to the decoder consists of:

- The feature representations output by the transformer encoder.
- A set of N learned embeddings, referred to as **object queries**.

Each object query is a fixed-length vector that represents a potential object in the image. These object queries are not associated with specific regions in the image beforehand; instead, the model learns to associate them with objects during training. Since the decoder operates on all N object queries simultaneously, it can leverage global relationships between all predicted objects.

3.2.6 Final Prediction

The output of the transformer decoder is a set of N learned embeddings, each of which corresponds to a potential object. These embeddings are then passed through a small prediction module consisting of:

- A feed-forward network (FFN) that predicts the class label using a softmax function.
- A separate linear layer that predicts bounding box coordinates in a normalized format.

Since DETR predicts a fixed number of objects, some object queries will not correspond to actual objects in the image. To handle this, an additional ”no object” class is introduced. The model learns to assign unused queries to this category, allowing it to effectively filter out empty detections.

3.3. Comparison Between Facebook DETR and Roboflow RF-DETR

To understand the evolution of Transformer-based object detectors in practice, we compare the original DETR model from Facebook AI Research (2) with the recently released RF-DETR model by Roboflow (8). While both models share the end-to-end detection formulation based on set prediction with Transformers, their design goals, usability, and performance characteristics differ substantially.

Architectural Core

Both models are inspired by the DETR paradigm: they frame object detection as a direct set prediction problem using a Transformer encoder-decoder and bipartite Hungarian matching loss. However, RF-DETR integrates refinements from later developments such as Deformable DETR (11) and Lightweight DETR (LW-DETR), and adopts a pre-trained DINOv2(6) ViT backbone. According to Roboflow’s official release (8), “RF-DETR is the first real-time object detection model to surpass 60 mAP on COCO with no need for Non-Maximum Suppression (NMS).”

Training Pipeline and Usability

The original DETR codebase is designed for research and requires significant customization. It relies on COCO-style datasets, manual environment setup, and verbose data preprocessing. In contrast, RF-DETR wraps its training pipeline in a streamlined PyTorch Lightning implementation, supporting Roboflow JSON export format natively. This allows practitioners to “train a state-of-the-art object detector on custom datasets in minutes” (8).

Performance and Inference

While DETR is known for its elegant formulation and competitive performance, it suffers from slow convergence and large-scale training requirements. RF-DETR addresses this by adopting deformable attention modules and lightweight architectures. The RF-DETR Base model has 29M parameters, while the Large model contains 128M parameters. According to Roboflow, inference time on a T4 GPU is as low as 6ms per image, making it suitable for real-time deployment.

3.3.1 Architectural and Practical Differences Between DETR and RF-DETR

Although RF-DETR builds upon the original DETR framework (2), it introduces several targeted improvements that enhance efficiency, accuracy, and usability.

Deformable Attention Mechanism

RF-DETR replaces the global attention mechanism used in DETR with *deformable attention*, as proposed in Deformable DETR (11). This approach allows each query to attend to a sparse set of key positions rather than the full feature map, leading to significantly faster convergence and improved performance on small and occluded objects.

Backbone Architecture

While the original DETR uses convolutional backbones such as ResNet-50(4) or ResNet-101, RF-DETR supports Vision Transformer (ViT) backbones, including options such as DINOv2 (6). This enables RF-DETR to leverage powerful semantic representations learned from large-scale self-supervised pretraining, improving its ability to generalize across domains. According to Roboflow’s official documentation (8), “RF-DETR is the first real-time object detection model to achieve over 60 mAP on COCO with no NMS, using a DINOv2 backbone and deformable attention.”

Training and Usability

The original Facebook DETR implementation is designed for research and expects COCO-style datasets, requiring manual configuration and custom scripts. In contrast, RF-DETR simplifies training and deployment by using PyTorch Lightning and supporting Roboflow-formatted JSON datasets out of the box. This allows practitioners to train models on custom datasets in minutes with minimal setup, making it highly accessible for production use.

Real-Time Performance

RF-DETR is optimized for real-time inference. The base model contains only 29 million parameters and runs at approximately 6 milliseconds per image on an NVIDIA T4 GPU, while still achieving high detection accuracy. This makes RF-DETR suitable for deployment in latency-sensitive applications.

Summary

In summary, RF-DETR preserves the core idea of end-to-end set prediction from DETR but improves on it with deformable attention, ViT-based backbones, faster convergence, and ease of integration. These changes make RF-DETR a production-ready evolution of DETR that maintains accuracy while improving usability and efficiency.

3.4. OWL-ViT2: Zero-Shot Object Detection

OWL-ViT2 is a transformer-based object detector designed for zero-shot recognition using textual prompts (5). Unlike conventional detectors trained on a fixed label set, it generalizes to unseen categories by leveraging large-scale vision-language pretraining.

We evaluated the `owlvit-base-patch32` variant on the full PASCAL VOC 2012 dataset (approx. 17,000 images). Each of the 20 object classes was queried using simple prompts (e.g., "a cat", "a person"). Outputs were evaluated using precision, recall, and mAP@0.5. Prompt ensembling (e.g., "a photo of a bird", "an image of a bird") was applied to improve robustness.

Across all classes, the model achieved a final mAP@0.5 of 0.446. High-performing categories included *cat* (AP 0.795), *aeroplane* (0.765), and *dog* (0.711), while *pottedplant*, *chair*, and *bottle* were comparatively weaker. These results highlight OWL-ViT2's sensitivity to prompt formulation and its difficulty with small or ambiguous objects in complex scenes.

Although the overall performance is lower than fully supervised detectors like YOLOv8 and DETR (typically 0.75 on VOC), OWL-ViT2 outperforms earlier zero-shot methods (mAP 0.30–0.42) (10), validating its ability to recognize unseen categories without VOC-specific training.

OWL-ViT2 offers a flexible alternative to fixed-class detectors, particularly suitable for applications where labeled data is limited or class sets change over time—such as in robotics, remote sensing, and wildlife monitoring.

4. Training and Fine-Tuning

This section describes the training procedures used for the fully supervised models (YOLOv8 and DETR) as well as the evaluation setup for the zero-shot model OWL-ViT2. Each model required distinct preparation depending on its architecture and training paradigm. We include OWL-ViT2 to examine the performance of open-vocabulary detection in contrast to conventional supervised detectors.

4.1. Fine-Tuning YOLOv8

We fine-tuned YOLOv8 using the Ultralytics framework, which supports training on custom datasets formatted in the YOLO annotation style. Our dataset, originally sourced in VOC format, was converted to YOLO format using Roboflow.

The fine-tuning process involved loading a pre-trained YOLOv8 model and replacing its classification head to match the number of classes in our dataset (20). This new head was randomly initialized, while the backbone and detection layers retained their pre-trained weights. The model was then trained on our dataset using the standard Ultralytics training pipeline.

This transfer learning approach allows the model to quickly adapt to new object categories while preserving the feature representations learned from large-scale datasets.

4.2. Fine-Tuning DETR on Pascal VOC

To adapt the DEtection TRansformer (DETR) model for our object detection task, we fine-tuned a pre-trained DETR model on the PASCAL VOC 2012 dataset. The original DETR model is trained on the COCO dataset with 91 classes, requiring modification of the classification head to match our dataset's 20 object categories plus one background class.

We replaced the original classification layer with a new randomly initialized head containing 21 output neurons. During training, this layer was learned from scratch, while the rest of the model retained its pre-trained weights. This allowed DETR to transfer its rich visual representations while adapting to the new domain.

A lower learning rate was used for the backbone and encoder-decoder layers, and a slightly higher rate for the new classification head, ensuring stable convergence. The model was trained using a combination of classification loss, bounding box regression loss, and generalized IoU loss. Standard data augmentations such as resizing, flipping, and normalization were applied to enhance generalization.

This fine-tuning strategy enabled DETR to effectively learn the object categories of the VOC dataset while maintaining its transformer-based detection capabilities.

5. Experiments

We evaluated a range of object detection models on the PASCAL VOC 2012 dataset to compare performance across conventional fully supervised and zero-shot detection paradigms. All supervised models were fine-tuned on the VOC training set and evaluated using standard metrics: mAP@0.5, precision, and recall. We also included **OWL-ViT2** as a zero-shot baseline, which relies entirely on vision-language pretraining and textual prompts without any VOC-specific training.

As shown in Table 1, larger YOLOv8 variants consistently perform better. YOLOv8-X achieves the highest mAP@50, precision, and recall, indicating that increasing model size benefits VOC performance.

Table 1. YOLOv8 Fine-Tuning Results on VOC Dataset

Model	mAP@50	Precision	Recall
YOLOv8-N	0.70	0.71	0.60
YOLOv8-M	0.71	0.73	0.61
YOLOv8-L	0.74	0.77	0.64
YOLOv8-X	**0.76**	**0.79**	**0.67**

Table 2. Evaluation Metrics for DETR on VOC Dataset

Model	mAP@50	Average Recall
DETR-ResNet50	0.720	0.710
DETR-ResNet101	**0.721**	**0.730**

DETR models also show competitive performance. The deeper ResNet101 backbone slightly improves recall compared to ResNet50, although gains in mAP are marginal.

Table 3. Fine-Tuned RF-DETR Results on VOC Dataset

Model	mAP@50	Precision	Recall
RF-DETR	0.74	0.75	0.72

Compared to standard DETR, RF-DETR shows a better balance between precision and recall, possibly due to its refinement module.

Table 4. Zero-Shot OWL-ViT2 Results on VOC Dataset

Model	mAP@50	Precision	Recall
OWL-ViT2 (zero-shot)	0.446	0.70	0.62

Finally, OWL-ViT2, despite being zero-shot, achieves promising results without any VOC-specific training, it relies entirely on vision-language pretraining and textual prompts for cate- gory specificatio This highlights the generalization power of vision-language models.
The model checkpoints used in our experiments can be downloaded from the following links:

- **ResNet-101**: Download ResNet-101 checkpoint
- **YOLOv8x**: Download YOLOv8x checkpoint

6. Conclusion

This study conducted a comparative evaluation of state-of-the-art object detection models across both supervised and zero-shot paradigms using the PASCAL VOC 2012 dataset. Specifically, we benchmarked YOLOv8 (in four variants), DETR (ResNet-50 and ResNet-101), RF-DETR, and OWL-ViT2. The results reveal clear tradeoffs between model types.

Among supervised models, YOLOv8-X achieved the highest mAP@0.5 (0.76), showing strong performance with low latency and compact design. DETR-ResNet101 followed closely with an mAP of 0.721, reflecting the strength of transformer-based models for spatial reasoning, albeit with higher computational cost.

OWL-ViT2, operating in a zero-shot setting without any VOC-specific training, attained a mAP@0.5 of 0.446. While its performance lags behind supervised counterparts, it significantly outperformed prior open-vocabulary detectors and proved effective for categories like "cat" and "aeroplane." Its generalization to unseen categories demonstrates the potential of vision-language models for open-world object detection.

These findings underscore a fundamental design choice in object detection systems: the balance between performance on fixed class sets and the adaptability to novel concepts. YOLOv8 and DETR excel in precision and recall under tight supervision, while OWL-ViT2 offers flexibility when labeled data is unavailable or constantly changing.

Future directions could involve hybrid models that combine vision-language priors with fine-tuning, or use larger, more diverse datasets such as LVIS or Objects365 to explore cross-domain robustness. Additionally, investigating prompt sensitivity and multimodal grounding could enhance the zero-shot capabilities of models like OWL-ViT2.

Ultimately, the best model depends on application requirements. Supervised detectors remain dominant where high accuracy is needed and class sets are stable. Zero-shot detectors, however, hold promise for scalable, real-world deployments in dynamic or data-scarce environments.

References

[1] Deci AI. Yolo-nas: A new state-of-the-art for object detection, 2023.

[2] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. *ECCV*, 2020.

[3] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision (IJCV)*, 88(2):303–338, 2010.

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pages 770–778, 2016.

[5] Matthias Minderer, Krishan Patil, Cordelia Schmid, Xiaohua Zhai, Konstantinos Drossos, Alexander Kolesnikov, Alexey Dosovitskiy, and Neil Houlsby. Simple open-vocabulary object detection with vision transformers. In *European Conference on Computer Vision (ECCV)*, 2022.

[6] Maxime Oquab, Théo Darcet, Theo Moutakanni, Alexandre Ramé, Ishan Misra, Julien Mairal, Piotr Bojanowski, Ivan Laptev, Cordelia Schmid, and Mathilde Caron. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

[7] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pages 779–788, 2016.

[8] Peter Robicheaux, James Gallagher, Joseph Nelson, and Isaac Robinson. RF-DETR: A sota real-time object detection model. `https://blog.roboflow.com/rf-detr`, 2025.

[9] Roboflow. Yolo-nas: Next-generation object detection from deci, 2023.

[10] Alireza Zareian, Ronghang You, Shih-Fu Chang, and Yue Wang. Open-vocabulary object detection using captions. In *CVPR*, 2021.

[11] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable detr: Deformable transformers for end-to-end object detection. *ICLR*, 2021.