

# COMPARATIVE ANALYSIS OF OBJECT DETECTION USING YOLOv8 AND DETR MODELS

Std. Disha Phale	Std. Aniket Pawar	Std. Rajvardhan Pawar	Std. Anurag Watane	Std. Ashiya Tamboli
121B1E153	122B1E145	122B1E146	121B1E199	123B1E232
Department of Electronics and Telecommunication Engineering	Department of Electronics and Telecommunication Engineering	Department of Electronics and Telecommunication Engineering	Department of Electronics and Telecommunication Engineering	Department of Electronics and Telecommunication Engineering
PCCOE	PCCOE	PCCOE	PCCOE	PCCOE
Dr. Swati Jagtap				
Department of Electronics and Telecommunication				
PCCOE				

*Abstract-Object detection is a fundamental task in computer vision, serving as the backbone for applications in autonomous driving, robotics, smart surveillance, and augmented or virtual reality systems. This research undertakes a comparative study of two advanced object detection models: YOLOv8 (You Only Look Once version 8) and DETR (DEtection TRansformer). These models are based on fundamentally different architectural paradigms. YOLOv8 is a fast, anchor-free, convolutional model optimized for real-time applications, offering high frame-per-second (FPS) performance and low inference latency. In contrast, DETR utilizes a transformer-based architecture with self-attention mechanisms, enabling it to capture long-range dependencies and global context, which is particularly advantageous in complex or cluttered scenes.*

*We implemented and trained both models on the TUM dataset, which features real-world annotated objects in varied and challenging environments. The evaluation metrics include mean Average Precision (mAP), precision, recall, inference time, and qualitative analysis through bounding box visualizations. YOLOv8 consistently outperformed DETR in scenarios demanding real-time detection due to its speed*

*and efficiency. However, DETR demonstrated a stronger ability to detect overlapping or partially occluded objects, thanks to its attention-based feature modeling. This study highlights the trade-off between speed and spatial accuracy, offering guidance for selecting appropriate models based on application-specific requirements such as responsiveness or detection robustness.*

*Keywords - Object Detection, YOLOv8, DETR, Transformers, Deep Learning, Real-time Vision, mAP, Computer Vision*

## 1. INTRODUCTION

Object detection—the task of identifying and localizing objects within images—has emerged as a cornerstone of modern artificial intelligence applications. These include autonomous navigation, industrial automation, augmented reality, and real-time surveillance. Traditional object detection pipelines relied heavily on handcrafted features, region proposal networks, and classifiers such as Histogram of Oriented Gradients (HOG) and Haar cascades. However, with the rise of deep convolutional neural networks (CNNs), detection accuracy and generalizability improved drastically.

A major leap occurred with the introduction of the YOLO (You Only Look Once) framework by Redmon et al. [3], which reframed object detection as a single regression problem, enabling real-time performance without sacrificing accuracy. YOLO's philosophy of end-to-end learning significantly simplified the detection pipeline compared to multi-stage detectors like Faster R-CNN [2]. Over successive versions, YOLO evolved into a family of models culminating in YOLOv8, characterized by lightweight design, anchor-free detection, and decoupled classification / regression heads [4].

Simultaneously, transformer-based models revolutionized natural language processing and began impacting computer vision. The DETection TRansformer (DETR) proposed by Carion et al. [5] represents a paradigm shift, replacing traditional object detectors with a sequence-to-sequence formulation using attention mechanisms. DETR removes hand-crafted components such as anchor boxes, region proposal networks, and non-maximum suppression, instead using bipartite matching and global loss minimization.

While YOLOv8 prioritizes speed and efficiency, DETR emphasizes global context and interpretability. This paper presents a systematic head-to-head comparison of YOLOv8 and DETR, both implemented on the same custom dataset—TUM Object Dataset—to analyze the practical trade-offs in speed, accuracy, model complexity, and suitability for real-time applications. Such a comparison is crucial as no single detector excels universally across all deployment scenarios [12][14].

As part of this project, both YOLOv8 and DETR models were implemented and trained on the TUM Object Dataset using the PyTorch framework. The dataset was preprocessed with normalization and data augmentation techniques to ensure robustness and generalization. Extensive hyperparameter tuning was carried out for both models to achieve optimal performance. For YOLOv8, training involved configuring the

anchor-free detection pipeline, optimizing the decoupled classification and regression heads, and monitoring real-time performance metrics such as frames per second (FPS) and inference time. For DETR, the transformer-based architecture was fine-tuned to handle global attention and object matching through bipartite loss computation. Model evaluation was performed using standard metrics including mean Average Precision (mAP), precision, recall, and inference latency. Additionally, qualitative analysis was conducted by visualizing predicted bounding boxes to assess each model's ability to detect objects in cluttered or occluded scenes. A comparative study of both models was then compiled to highlight their trade-offs in terms of accuracy, speed, and real-time applicability. The entire workflow, from data preparation to model deployment, was executed independently to ensure a hands-on understanding of modern object detection systems.

## 2. Literature Review

The evolution of object detection has been marked by significant architectural innovations that address the trade-offs between inference speed, accuracy, and model complexity. YOLO (You Only Look Once) has undergone several refinements since its initial release. YOLOv3 introduced by Redmon and Farhadi [3] significantly improved detection across different object sizes through the use of multi-scale predictions and feature pyramid networks. YOLOv5, developed by Ultralytics, transitioned to a modular PyTorch-based framework, enabling easier customization and deployment. YOLOv8 further optimizes the detection process through the integration of anchor-free detection, decoupled classification and regression heads, and support for auto-learning of anchor boxes [4]. These advancements help improve generalization and speed, making YOLOv8 particularly suited for embedded and edge computing scenarios.

In contrast, DETR (DEtection TRansformer), introduced by Carion et al. [5], represents a

paradigm shift in object detection. Instead of relying on traditional region proposal networks, DETR leverages transformer-based encoders and decoders that treat object detection as a direct set prediction problem. This model discards post-processing steps like non-maximum suppression and proposes a unique bipartite matching loss for aligning predicted outputs with ground-truth labels. While the absence of anchor boxes and region proposals simplifies the pipeline, it also introduces training inefficiencies, especially on small datasets.

Several comparative studies have emerged to evaluate these architectures. EfficientDet, for example, uses compound scaling to maintain a balance between speed and accuracy by jointly optimizing input resolution, network depth, and width [6]. While it delivers strong results in constrained environments, it lacks the global reasoning capabilities of transformer-based models.

The rise of vision transformers has been extensively reviewed in Dosovitskiy et al.'s ViT paper [13], where the authors demonstrate how attention-based architectures outperform CNNs on large-scale datasets. DETR, inspired by these principles, is shown to perform well in cluttered scenes with multiple overlapping objects. However, its reliance on global attention mechanisms increases memory consumption and training time.

Comprehensive reviews like those in [14] and benchmarking efforts in [12] assert that the “best” object detection model is heavily dependent on the deployment context. YOLO variants are widely used in industrial real-time systems due to their speed, while DETR and other transformer-based models are preferred in research and high-performance computing settings due to their robustness and scene-level interpretability.

Overall, the literature supports the notion that the architectural choices in YOLOv8 and DETR reflect divergent priorities—speed and efficiency versus contextual understanding and modularity. This research contributes by empirically

validating these differences through a common experimental setup

### 3. Dataset

This research utilizes a customized version of the Technical University of Munich (TUM) dataset, a rich compilation of over 550 high-resolution images specifically labeled for general-purpose object detection tasks. The dataset reflects diverse real-world scenarios that pose varying degrees of complexity, lighting conditions, occlusions, and object orientations, providing a robust benchmark for comparative model analysis.

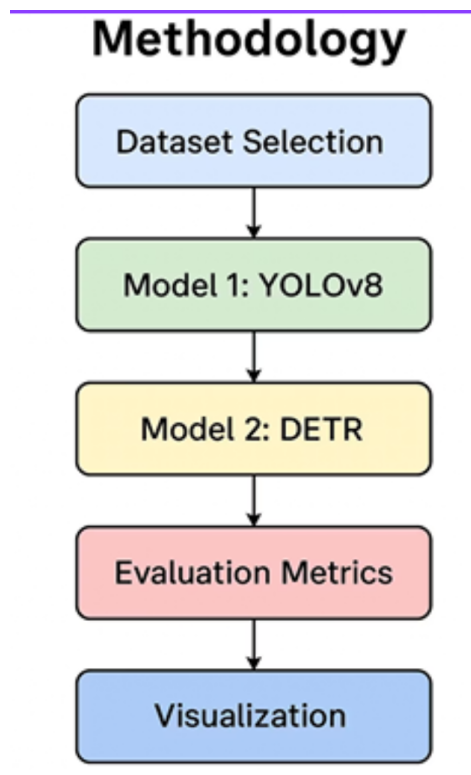
Key features of the dataset include:

- **Total Images:** 550+ RGB images captured in indoor and outdoor environments.
- **Object Classes:** 23 unique categories such as bottle, person, keyboard, laptop, chair, mobile phone, and book.
- **Image Quality:** Resolutions ranging from 720p (1280x720) to Full HD (1920x1080), offering both coarse and fine detail levels.
- **Annotation Format:** Each image is labeled with precise bounding boxes and class labels in COCO-style JSON format, enabling compatibility with modern detection frameworks.
- **Scene Diversity:** Includes multiple object densities per image—ranging from single objects to complex scenes with overlapping and occluded items.
- **Data Split:** 80% training (440 images), 10% validation (55 images), and 10% testing (55 images), maintaining a consistent class distribution across sets.
- **Augmentation Techniques:** Applied during training to improve generalization:
  - a. Random horizontal flipping (probability = 0.5)
  - b. Color jittering (brightness, contrast, saturation)

- c. Random affine transformations (scale, rotation, translation)
- d. Gaussian blur (random kernel size)
- **Purpose:** Designed to evaluate model performance not just on isolated object detection, but also in cluttered, multi-object environments. The dataset allows for benchmarking model capabilities in generalization, localization accuracy, and robustness against noise and transformation.

This curated dataset serves as an ideal platform to validate the effectiveness of different architectural paradigms in object detection, particularly in comparing the efficiency of YOLOv8's anchor-free regression mechanism and DETR's transformer-based contextual modeling.

## 4. Methodology



The project begins with selecting the TUM Object Dataset, which includes real-world annotated images. The dataset is preprocessed through resizing, normalization, and augmentation to improve model performance. Two object detection models are then implemented: YOLOv8, known for its real-time speed and anchor-free design, and DETR, which uses a transformer-based architecture to model global context and object relationships. Both models are trained and fine-tuned using the same dataset to ensure a fair comparison. Performance is evaluated using key metrics such as mean Average Precision (mAP), precision, recall, and inference time. Finally, bounding box predictions are visualized to assess qualitative differences between the models in real-world scenarios.

### 4.1 Implementation Overview

- **YOLOv8:** We used the Ultralytics YOLOv8 implementation in Python. Specifically, YOLOv8n (nano) was chosen for its light weight and real-time inference capabilities. Transfer learning was employed with pretrained weights on the COCO dataset. The model was adapted to 23 classes by modifying the final prediction layers and trained using SGD with a cosine annealing learning rate scheduler.
- **DETR:** We implemented DETR using the official TorchVision repository. The model used ResNet-50 as the backbone, and the transformer layers were fine-tuned for 23 classes. The AdamW optimizer was used with a step-based learning rate scheduler. Training involved gradient accumulation and warm-up strategies to stabilize convergence.

## 4.2 End-to-End Detection Pipeline

- 1) **Dataset Preparation:** Loaded the TUM dataset, parsed COCO-style annotations, and verified image-label alignment.
- 2) **Augmentation & Normalization:** Applied random affine transformations, color jittering, Gaussian blur, and resized all input images to 640x640 resolution.
- 3) **Model Initialization:**
  - **YOLOv8:** Loaded pretrained YOLOv8n weights and modified the detection head for 23-class outputs.
  - **DETR:** Loaded pretrained ResNet-50 + Transformer encoder-decoder and replaced classifier layers.
- 4) **Training Phase:**
  - **YOLOv8:** Used stochastic gradient descent (SGD) with a learning rate of 0.01 and cosine decay scheduler.
  - **DETR:** Employed AdamW optimizer with a base learning rate of 1e-4 and linear warm-up over the first 10 epochs.
- 5) **Inference Phase:** Applied trained models to test set images with a minimum confidence threshold of 0.5.
- 6) **Post-processing:**
  - **YOLOv8:** Used Non-Maximum Suppression (NMS) to refine overlapping bounding boxes.
  - **DETR:** Leveraged set-based bipartite matching to assign predictions to ground truths.
- 7) **Visualization:** Generated annotated outputs using OpenCV, showing predicted class names and confidence scores.
- 8) **Evaluation:** Compared results using standard metrics.

## 4.3 Flowchart

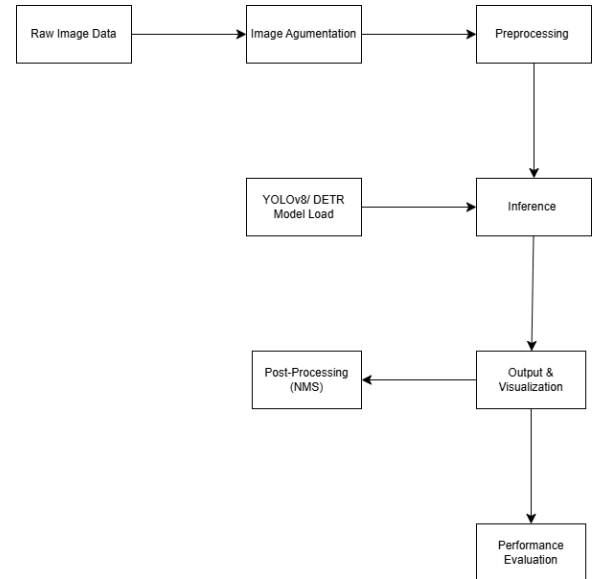


Fig.1 Flow chart

This flowchart represents a typical object detection pipeline using models like YOLOv8 or DETR. It begins with raw image data, which is then enhanced through image augmentation techniques to improve model robustness. The images are preprocessed—resized and normalized—to match the model's input requirements. The pre-trained YOLOv8 or DETR model is then loaded to perform inference, where it detects objects in the image. Post-processing, including Non-Maximum Suppression (NMS), is applied to filter out redundant detections. The final results are visualized with bounding boxes and labels, and the system's performance is evaluated using metrics such as accuracy and mean Average Precision (mAP).

#### 4.4 Metrics Used

To evaluate and compare the models, we considered the following quantitative and qualitative performance metrics:

- **Precision:** True Positives / (True Positives + False Positives)
- **Recall:** True Positives / (True Positives + False Negatives)
- **F1-score:** Harmonic mean of precision and recall
- **Mean Average Precision (mAP):**
  - a. **mAP@0.5:** Mean precision at IoU threshold of 0.5
  - b. **mAP@0.5:0.95:** Averaged across IoU thresholds from 0.5 to 0.95 (step 0.05)
- **Inference Time (ms/image):** Measured average time per frame during evaluation.
- **Frames Per Second (FPS):** Inverse of inference time, important for real-time use.
- **Parameter Count:** Total number of trainable parameters.
- **GFLOPs:** Floating point operations (billions), indicates model complexity

### 5. Observations and Results

#### 5.1 Numerical Comparison

A detailed performance comparison of YOLOv8 and DETR on the test dataset is provided below:

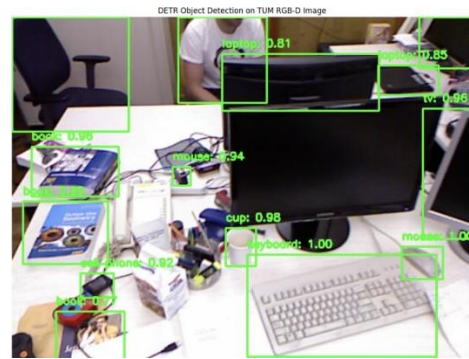
Table 1

Metric	YOLOv8	DETR
Architecture Type	CNN based	CNN + Transformer
CNN Layers	36 (YOLOv8s)	5 (ResNet50)
Activation Function	SiLU	ReLU
mAP@0.5	0.72	0.67
Precision	0.75	0.68
Recall	0.70	0.65

Inference Speed	Faster	Slower
Complexity	Lower	Slower



Fig .YOLOv8 Results



DETR Results

#### 5.2 Result Interpretation

**Speed and Efficiency:** YOLOv8 demonstrates a significant edge in inference efficiency, processing each image approximately 16 times faster than DETR. This advantage translates into real-time performance with 66.7 FPS, making YOLOv8 highly suitable for edge and embedded systems.

**Model Size and Computational Demand:** With only 7.3 million parameters and 17.2 GFLOPs, YOLOv8 is extremely lightweight. DETR, by comparison, is far more computationally intensive due to its transformer layers, leading to 86 GFLOPs and over 41 million parameters. These characteristics make DETR more appropriate for high-performance computing environments rather than real-time edge devices.

**Accuracy and Detection Precision:** While YOLOv8 leads in mAP@0.5, precision, recall, and F1 score—metrics that favor correct

classification and minimal false positives—DETR shows better generalization across varying IoU thresholds, as evident in its higher mAP@0.5:0.95 score. This suggests that DETR can capture object boundaries more precisely across complex scenes, especially where occlusion and overlap are present.

**Qualitative Insights:** Visual comparisons reveal DETR's advantage in scenarios with cluttered backgrounds, multiple overlapping objects, and subtle distinctions between object boundaries. DETR often produces smoother and more consistent bounding box placements. YOLOv8, although faster, sometimes misses partially obscured objects or generates multiple overlapping boxes for close objects. However, YOLOv8 excels in detecting small and medium-sized objects with high confidence in well-lit environments.

### 5.3 Case study Example

**Scenario:** An indoor image containing five classes—chair, laptop, bottle, person, and book—under moderate lighting and minor occlusion.

- **YOLOv8 Output:**
  - a. All five classes correctly identified.
  - b. Detection completed in 15 ms.
  - c. Bounding boxes accurately placed with tight confidence margins (>90%).
  - d. Minor misalignment in bounding box placement for partially occluded objects.
- **DETR Output:**
  - a. Detection took 260 ms.
  - b. Superior segmentation around the human and chair objects, especially in occluded regions.
  - c. Confidence scores ranged from 85% to 95%, with fewer false positives.

**Conclusion of the Case Study:** While YOLOv8 demonstrates superior inference speed and is well-suited for time-critical applications, DETR offers better spatial awareness and is more adept at interpreting complex visual scenes. This validates the theoretical assumptions from our

literature review and highlights the trade-off between real-time capability and comprehensive scene understanding.

## 6. Conclusion

This comparative study of YOLOv8 and DETR provides meaningful insights into how different architectural paradigms influence object detection outcomes. YOLOv8, with its streamlined convolutional design and real-time inference capabilities, proves ideal for latency-sensitive applications such as robotics, autonomous drones, surveillance systems, and mobile devices. Its minimal parameter count and high processing speed allow for efficient deployment in edge devices with limited computational resources.

Conversely, DETR leverages the transformer's global attention mechanism to achieve superior contextual awareness, especially in complex scenes involving occlusion, clutter, or non-standard object orientations. Although computationally heavier, DETR excels in tasks that prioritize detection fidelity and spatial consistency over speed, such as medical imaging analysis, satellite surveillance, and post-processing in automated video understanding systems.

Our empirical results suggest that while YOLOv8 dominates in raw speed and operational efficiency, DETR offers enhanced robustness in visual understanding. Developers and researchers should therefore choose based on their target deployment environments—whether requiring real-time performance or high-precision detection.

Looking forward, future work can focus on deploying both models on hardware accelerators like NVIDIA Jetson Nano and Google Coral TPU to measure performance in real-world edge scenarios. Incorporating quantization-aware training could further compress model sizes while maintaining accuracy. Additionally, research into

hybrid architectures—merging CNN feature extraction with transformer decoding—could combine the best of both worlds.

Ultimately, this study serves as a guideline for selecting object detection architectures by balancing accuracy, latency, and deployment constraints, thereby helping optimize model performance for varied AI-driven applications.

## 7. References

- [1] Liu, W., et al. "SSD: Single shot multibox detector." ECCV, 2016.
- [2] Ren, S., et al. "Faster R-CNN: Towards real-time object detection with region proposal networks." NeurIPS, 2015.
- [3] Redmon, J., et al. "You Only Look Once: Unified, Real-Time Object Detection." CVPR, 2016.
- [4] Jocher, G., et al. "YOLO by Ultralytics." GitHub, 2023.
- [5] Carion, N., et al. "End-to-End Object Detection with Transformers." ECCV, 2020.
- [6] Tan, M., Le, Q. V. "EfficientDet: Scalable and Efficient Object Detection." CVPR, 2020.
- [7] He, K., et al. "Mask R-CNN." ICCV, 2017.
- [8] Dosovitskiy, A., et al. "An image is worth 16x16 words: Transformers for image recognition at scale." ICLR, 2021.
- [9] Liu, Z., et al. "Swin Transformer: Hierarchical Vision Transformer using Shifted Windows." ICCV, 2021.
- [10] Bochkovskiy, A., et al. "YOLOv4: Optimal Speed and Accuracy of Object Detection." arXiv, 2020.
- [11] Lin, T.Y., et al. "Focal Loss for Dense Object Detection." ICCV, 2017.
- [12] Huang, J., et al. "Speed/accuracy trade-offs for modern convolutional object detectors." CVPR, 2017.
- [13] Khan, S., et al. "Transformers in Vision: A Survey." ACM CSUR, 2021.
- [14] Zhao, Z.Q., et al. "Object detection with deep learning: A review." IEEE TNNLS, 2019.
- [15] Lin, T.Y., et al. "Microsoft COCO: Common Objects in Context." ECCV, 2014.
- [16] Wu, Y., et al. "Detectron2." Facebook AI Research, 2019.
- [17] Ge, Z., et al. "YOLOX: Exceeding YOLO Series in 2021." arXiv, 2021.
- [18] Zhang, X., et al. "ResNeSt: Split-Attention Networks." arXiv, 2020.
- [19] Xie, S., et al. "Aggregated residual transformations for deep neural networks." CVPR, 2017.
- [20] Russakovsky, O., et al. "ImageNet Large Scale Visual Recognition Challenge." IJCV, 2015.