VIETNAM GENERAL CONFEDERATION OF LABOUR

**TON DUC THANG UNIVERSITY**

**FACULTY OF INFORMATION TECHNOLOGY**



**TANG DAI – 520H0523**

**NGO QUOC TUAN – 520H0190**

**COMPARATIVE ANALYSIS OF RT-DETR VS YOLOV8 IN REAL-TIME OBJECT DETECTION**

**INFORMATION TECHNOLOGY PROJECT**

**SOFTWARE ENGINEERING**

**HO CHI MINH CITY, 2025**

VIETNAM GENERAL CONFEDERATION OF LABOUR

**TON DUC THANG UNIVERSITY**

**FACULTY OF INFORMATION TECHNOLOGY**



**TANG DAI – 520H0523**

**NGO QUOC TUAN – 520H0190**

**COMPARATIVE ANALYSIS OF RT-DETR VS YOLOV8 IN REAL-TIME OBJECT DETECTION**

**INFORMATION TECHNOLOGY PROJECT**

**SOFTWARE ENGINEERING**

Instructor

**PhD. PHAM VAN HUY**

**HO CHI MINH CITY, 2025**

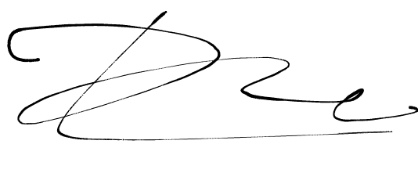
**ACKNOWLEDGEMENT**

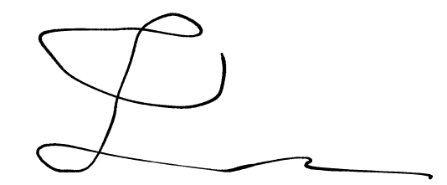
We would like to express our sincere gratitude to PhD. Pham Van Huy for generously dedicating your time to guide us in completing this important report, a critical component of our learning journey. Your extensive expertise in Computer Vision has profoundly deepened our understanding of the subject and significantly enhanced the quality of our work. We are truly grateful for your unwavering support, patience, and guidance, and we believe that the knowledge and experience gained from your mentorship will be immensely valuable in our future studies and careers.

*Ho Chi Minh City, 10.2.2025*

*Author*

*(Sign and Write full name)*

****

****

**Tăng Đại**

**Ngô Quốc Tuấn**

**DECLARATION OF AUTHORSHIP**

I hereby declare that this thesis was carried out by myself under the guidance and supervision of PhD. Pham Van Huy; and that the work and the results contained in it are original and have not been submitted anywhere for any previous purposes. The data and figures presented in this thesis are for analysis, comments, and evaluations from various resources by my own work and have been duly acknowledged in the reference part.

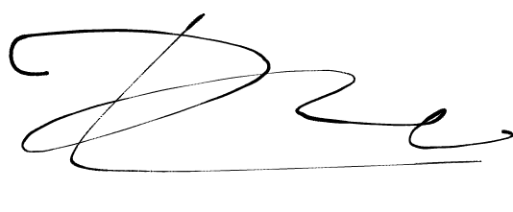
In addition, other comments, reviews and data used by other authors, and organizations have been acknowledged, and explicitly cited.

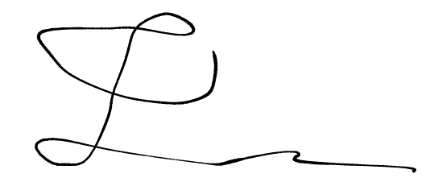
**I will take full responsibility for any fraud detected in my thesis**. Ton Duc Thang University is unrelated to any copyright infringement caused on my work (if any).

*Ho Chi Minh City, 10.2.2025*

*Author*

*(Sign and Write full name)*

**

**

**Tăng Đại**

**Ngô Quốc Tuấn**

**COMPARATIVE ANALYSIS OF RT-DETR VS YOLOV8 IN REAL-TIME OBJECT DETECTION**

**ABSTRACT**

This study presents a comparative analysis of two state-of-the-art real-time object detection models, YOLOv8 and RT-DETR, highlighting their foundational concepts, architectural distinctions, and performance trade-offs in a practical application scenario. YOLOv8, an evolution of the popular “You Only Look Once” series developed by Ultralytics, employs an efficient CNN-based backbone with advanced components such as CSPNet, SPPF, and multi-scale detection heads, offering rapid inference and low computational cost. In contrast, RT-DETR leverages a transformer-based architecture with an efficient hybrid encoder and IoU-aware query selection, designed to enhance detection accuracy, particularly in complex scenes with occlusion.

This report also has provided a comprehensive comparison between YOLOv8 and RT-DETR in the context of object detection for football players. YOLOv8 demonstrates strong performance with high accuracy (mAP50 of 69.3% and mAP50-95 of 43.6%), rapid inference, and low computational cost, making it suitable for real-time applications. In contrast, RT-DETR offers competitive accuracy (mAP50 of 66%, mAP50-95 of 46%, and mAP75 of 57%), particularly excelling in complex scenes due to its transformer-based design. These findings contribute valuable insights into the ongoing evolution of object detection technologies and offer guidance for selecting the most appropriate model based on specific operational contexts.

**TABLE OF CONTENT**

[LIST OF FIGURES vi](#_Toc190355054)

[LIST OF TABLES vii](#_Toc190355055)

[DANH MỤC CÁC CHỮ VIẾT TẮT viii](#_Toc190355056)

[CHAPTER 1. INTRODUCTION 1](#_Toc190355057)

[1.1 Problem 1](#_Toc190355058)

[1.2 Objective 1](#_Toc190355059)

[1.3 Report strucure 1](#_Toc190355060)

[CHAPTER 2. BACKGROUND 2](#_Toc190355061)

[2.1 Overview of YOLOv8 2](#_Toc190355062)

[2.1.1 Key Innovations and Enhancements 2](#_Toc190355063)

[2.1.2 Development Timeline 3](#_Toc190355064)

[2.1.3 Architecture 4](#_Toc190355065)

[2.1.4 Architecture Blocks 6](#_Toc190355066)

[2.1.5 Performance Metrics 11](#_Toc190355067)

[2.1.6 YOLOv8 Models 11](#_Toc190355068)

[2.1.7 YOLOv8 Annotation Format 12](#_Toc190355069)

[2.2 Overview of RT-DETR 13](#_Toc190355070)

[2.2.1 Key features 13](#_Toc190355071)

[2.2.2 RT-DETR Structure 14](#_Toc190355072)

[2.2.3 IoU-driven Query Selection 19](#_Toc190355073)

[2.2.4 Dynamic Inference Speed Adjustment 20](#_Toc190355074)

[2.2.5 RT-DETR Performance Metrics 21](#_Toc190355075)

[2.2.6 RT-DETR models 22](#_Toc190355076)

[CHAPTER 3. METHODOLOGY 22](#_Toc190355077)

[3.1 Data Collection 22](#_Toc190355078)

[3.2 Experimental Setup 24](#_Toc190355079)

[3.3 Quantitative Metrics 24](#_Toc190355080)

[3.4 Limitations 24](#_Toc190355081)

[CHAPTER 4. RESULTS 25](#_Toc190355082)

[4.1 Quantitive Analysis 25](#_Toc190355083)

[4.2 Qualitavie Observations 26](#_Toc190355084)

[CHAPTER 5. CONCLUSION & DISCUSSION 27](#_Toc190355085)

[5.1 Conclusion 27](#_Toc190355086)

[5.2 Further work 28](#_Toc190355087)

[REFERNCES 29](#_Toc190355088)

# LIST OF FIGURES

[Figure 1 YOLOv8 Architecture 6](#_Toc190355089)

[Figure 2 Image Resizing 7](#_Toc190355090)

[Figure 3 Standard YOLOv8 convolutional block 8](#_Toc190355091)

[Figure 4 Bottleneck blocks. 8](#_Toc190355092)

[Figure 5 C2f Block 9](#_Toc190355093)

[Figure 6 SPPF 10](#_Toc190355094)

[Figure 7 YOLOv8 Detect Block. 10](#_Toc190355095)

[Figure 8 RT-DETR Architecture. 14](#_Toc190355096)

[Figure 9 The Encoder of RT-DETR 16](file:///C:\\Users\\ACER\\Desktop\\BAOCAO%20(1).docx" \l "_Toc190355097)

[Figure 10 Decoder of the RT-DETR model. 18](file:///C:\\Users\\ACER\\Desktop\\BAOCAO%20(1).docx" \l "_Toc190355098)

[Figure 11 Data example. 23](#_Toc190355099)

[Figure 12 Label of the image shown in Fig. 10. 23](#_Toc190355100)

[Figure 13 YOLOv8 Validation output. 25](#_Toc190355101)

[Figure 14 RT-DETR Validation output. 25](#_Toc190355102)

# LIST OF TABLES

[Table 1: Performance metrics for the YOLOv8 models. 12](#_Toc190355247)

[Table 2: Comparison with traditional Query Selection. 20](#_Toc190355248)

# LIST OF ABBREVIATES

|  |  |
| --- | --- |
| AI | Artificial Intelligence |
| CNN | Convolutional Neural Network |
| COCO | Common Objects in Context |
| FPS | Frames Per Second |
| GFLOPs | Giga Floating-Point Operations |
| GPU | Graphics Processing Unit |
| IoU | Intersection over Union |
| mAP | Mean Average Precision |
| PANet | Path Aggregation Network |
| RT-DETR | Real-Time Detection Transformer |
| SILU | Sigmoid Linear Unit |
| YOLO | You Only Look Once |

# INTRODUCTION

## Problem

In recent years, the demand for real-time object detection has surged across various fields, including autonomous driving, robotics, surveillance, and augmented reality. This surge is driven by the need to process and analyze vast volumes of visual data both swiftly and accurately. At the heart of these applications lie advanced deep learning architectures designed to identify and localize objects within images or videos in a fraction of a second.

In the sphere of high-performing models, RT-DETR and YOLOv8 stand out as key contenders for real-time object detection. RT-DETR (Real-Time Detection Transformer) relies on the Transformer architecture’s attention mechanism to deliver robust detection accuracy at high inference speeds. Meanwhile, YOLOv8 expands on the well-established “You Only Look Once” series, featuring an enhanced backbone and head layers that strike a balance between precision and efficiency.

## Objective

The main this report is analyzing their foundational concepts, architectural distinctions, training requirements, and performance benchmarks, we aim to determine which model excels across varied real-world applications.

## Report strucure

This report is organized as follows:

* Chapter 1 Introduction: Provides an overview of the problem, outlines the objectives, and introduces the report structure.
* Chapter 2 Background: Reviews the evolution of object detection technologies with a focus on YOLOv8 and RT-DETR, including their key innovations and architectural features.
* Chapter 3 Methodology: Details the research methods, experimental setup, dataset preparation, and the key quantitative metrics used for evaluating both models.
* Chapter 4 Results: Presents a comparative quantitative and qualitative analysis of the performance of YOLOv8 and RT-DETR, based on metrics such as mAP, inference speed, model complexity, and computational cost.
* Chapter 5 Conclusion & Discussion: Interprets the findings, discusses the implications for real-world applications, and outlines further work to expand on the current study.

# BACKGROUND

## Overview of YOLOv8

YOLOv8, developed by Ultralytics and released in 2023, builds on the ideas introduced in earlier YOLO versions (YOLOv5) while incorporating significant architectural and methodological improvements. These updates are aimed at enhancing detection accuracy and simplifying the training and deployment processes for real-time object detection tasks.

### Key Innovations and Enhancements

YOLOv8 introduces several notable innovations:

* *Simplified Model Training:* The model adopts an anchor-free approach, which streamlines the training process by eliminating the need for predefined anchor boxes and associated hyperparameters. This change not only reduces complexity but also improves the model’s adaptability to objects with varying aspect ratios and scales.
* *Improved Detection Accuracy:* Architectural enhancements such as the integration of a CSPNet backbone and a hybrid FPN+PAN neck enable YOLOv8 to capture and process multi-scale features more effectively. The CSPNet backbone enhances feature extraction efficiency, while the hybrid FPN+PAN neck improves the fusion of deep and shallow features, which is crucial for detecting objects of different sizes.
* *Advanced Data Augmentation:* Techniques such as mosaic and mixup augmentation are employed to expose the model to diverse spatial arrangements and object appearances. These augmentations help the model generalize better across different datasets and real-world scenarios.
* *Optimized Training Strategies:* YOLOv8 leverages mixed-precision training using PyTorch, which accelerates both training and inference while reducing memory consumption. This is particularly beneficial for deployment on resource-constrained devices.

### Development Timeline

* January 10, 2023: *Official Release*. YOLOv8 is launched with an anchor-free architecture designed to simplify training and improve detection accuracy.
* February 15, 2023: *Developer Tools Introduced.* A dedicated Python package and command-line interface (CLI) are released, streamlining the processes of training, validating, and deploying models.
* March 5, 2023: *Advanced Augmentation Techniques.* Mosaic and mixup augmentation techniques are implemented, helping the model to learn from more varied and complex image compositions.
* April 20, 2023: *Integration of Advanced Architectural Elements.* The introduction of a CSPNet backbone and a hybrid FPN+PAN neck optimizes multi-scale feature extraction and fusion, enhancing overall detection performance.
* June 1, 2023: *Deployment Support* *Expanded.* Support for ONNX and TensorRT formats is added, facilitating model deployment on a broader range of hardware, including edge devices.

### Architecture

YOLOv8 maintains the unified, end-to-end design idea of earlier YOLO versions, performing both object localization and classification within a single neural network. Its architecture is organized into three primary components:

1. *Backbone:* The backbone is responsible for extracting rich, multi-scale features from the raw input image. In YOLOv8, the backbone begins with a set of two convolutional blocks (each using a 3×3 kernel with stride 2) that both process the input data and reduce its spatial resolution. This is followed by several stages composed of specialized building blocks (primarily the C2f blocks) that progressively abstract and encode the image features. The architecture is organized into eight stages corresponding to specific blocks (blocks 2, 4, 6, and 8 serve as the backbone with shortcut connections, while blocks 12, 15, 18, and 21 form part of the neck and do not use shortcuts). Downsampling is consistently applied using convolution blocks with a stride of 2 to halve the spatial resolution at key points, thereby increasing the network’s receptive field and computational efficiency.
2. *Neck:* The neck aggregates and refines the multi-scale features extracted by the backbone. A critical component in YOLOv8’s neck is the **SPPF (Spatial Pyramid Pooling Fast) block**. Positioned immediately after the backbone, the SPPF block pools the feature maps at multiple scales (using a fixed kernel size of 5), which helps capture both fine and coarse contextual information with reduced computational overhead compared to earlier spatial pyramid pooling methods. In addition, the neck employs upsampling (using the nearest neighbor method) and concatenation operations to fuse feature maps from different levels—preserving resolution while increasing the number of channels—to effectively prepare the data for object prediction.
3. *Head:* The head takes the aggregated feature maps from the neck and produces the final object detection predictions. In YOLOv8, this is achieved through three separate detection heads, each tailored to objects of a different scale:

* The first head (connected to block 15) is optimized for detecting small objects.
* The second head (connected to block 18) is aimed at medium-sized objects.
* The third head (connected to block 21) targets large objects.

Each head outputs bounding are box coordinates and class probabilities.

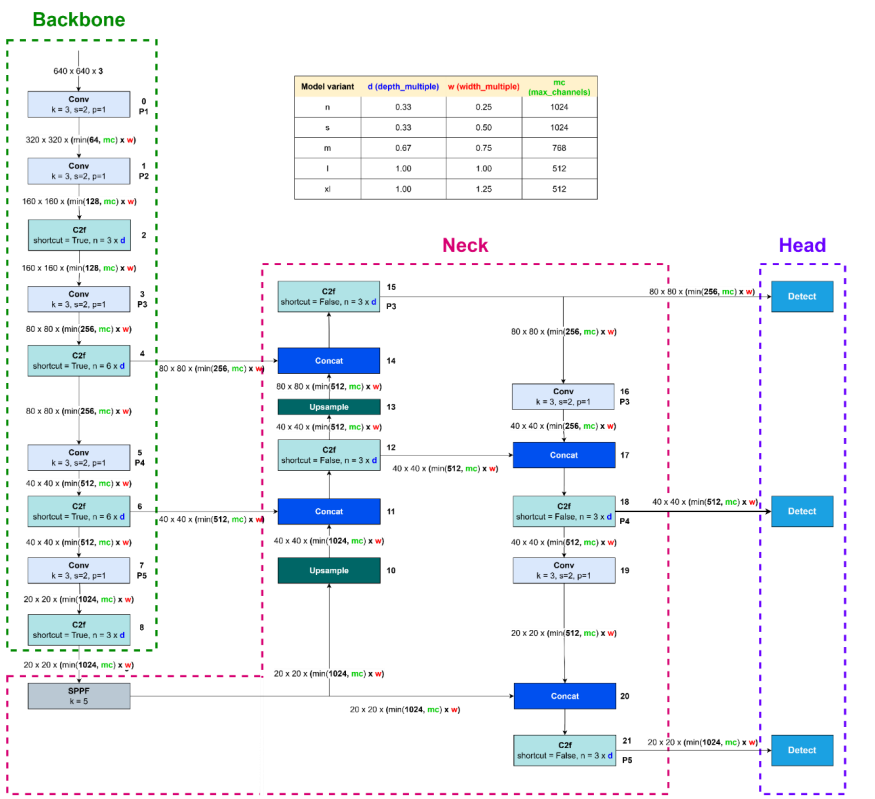


Figure 1 YOLOv8 Architecture

### Architecture Blocks

*Input Image Resizing:* In YOLOv8, the input image is resized while preserving its aspect ratio. To maintain this ratio, images that do not fit the target dimensions are padded with gray pixels. If the input image is already square, it is resized without padding.

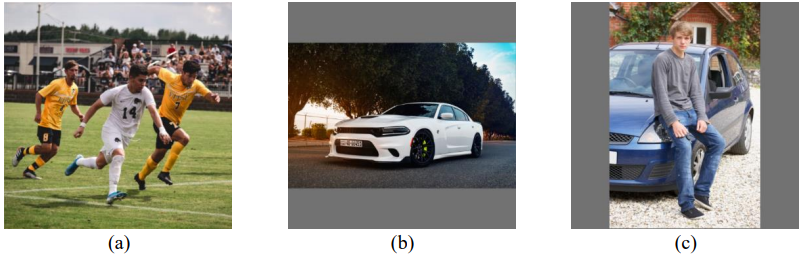


Figure 2 Image Resizing

1. Square Image, (b) Landscape Image (c) Portrait Image

*Convolutional (Conv) Block:* The fundamental building unit in YOLOv8 is the convolutional block. Each Conv block fuses a 2-dimensional convolutional layer with 2-dimensional batch normalization and a SILU activation function into a single operation. In certain modules, such as the feed-forward network (FFN) within the Position-Sensitive Attention Block (PSABlock), the Conv block is used without the activation function. Additionally, YOLOv8 employs depthwise convolution (DWConv) where a single kernel operates on each channel separately, reducing computational cost while still capturing diverse features. (See Figure 5 for an illustration of the YOLOv8 convolutional block)

*Downsampling Block*: YOLOv8 uses a standard 3×3 convolution with a stride of 2 to perform downsampling. This operation reduces the spatial resolution of the feature maps by half, thereby lowering computational requirements while preserving critical information.

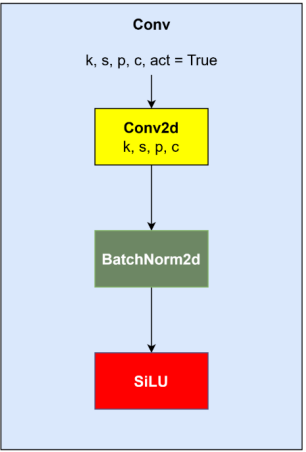


Figure 3 Standard YOLOv8 convolutional block

*Bottleneck:* The bottleneck block in YOLOv8, is a block that is similar to the ResNet block, allows for the construction of deeper networks by stacking additional layers with minimal extra computational cost. Some bottlenecks include shortcut (residual) connections to facilitate gradient flow and improve training stability, while others do not—this design choice is based on empirical performance evaluations.

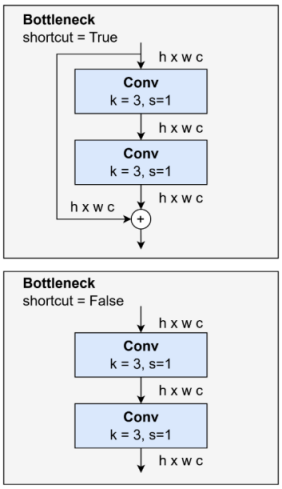


Figure 4 Bottleneck blocks.

*C2f Block:* YOLOv8 employs the C2f block as an enhanced implementation of the CSP bottleneck, utilizing two convolutional layers. The C2f block is used across various stages of the network for effective feature extraction. It efficiently learns multiscale features and expands the receptive field by leveraging feature vector switching and multilayer convolution.

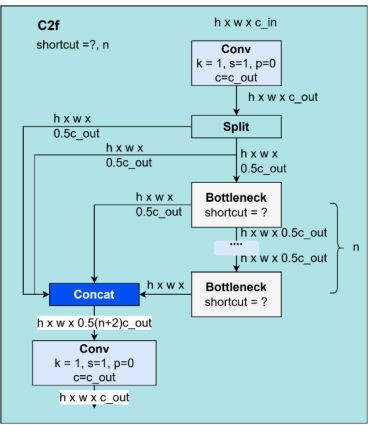


Figure 5 C2f Block

*SPPF Block:* The Spatial Pyramid Pooling – Fast (SPPF) block in YOLOv8 enables the model to capture features at multiple scales by pooling at different resolutions. SPPF is a refined version of the earlier Spatial Pyramid Pooling (SPP) block which is available in the YOLO model before; while SPP typically uses max pooling with kernel sizes of 3, 5, and 9, SPPF simplifies the process by employing a kernel size of 5. This modification reduces the number of floating-point operations (FLOPs) while maintaining the ability to extract multi-scale contextual information.

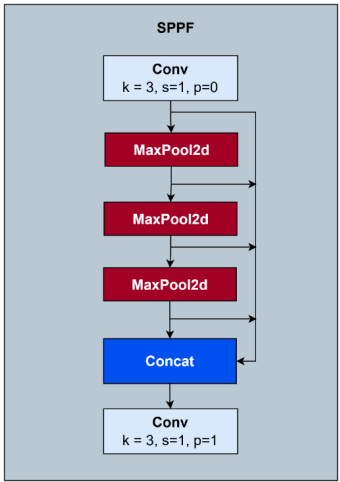


Figure 6 SPPF

*Detect Block*: In YOLOv8, the detect block is responsible for generating the final predictions for object detection. It comprises two main components:

* Classification Head: Generates class probability scores.
* Regression Head: Predicts the bounding box coordinates.

Each head typically consists of two convolutional blocks followed by a final 2-dimensional convolutional layer. To optimize the architecture and reduce the number of model parameters, YOLOv8 may incorporate depthwise convolutions within these blocks, particularly in the classification head.

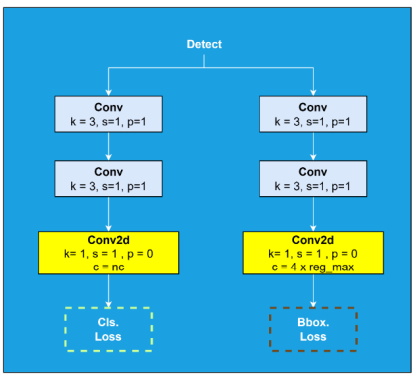


Figure 7 YOLOv8 Detect Block.

### Performance Metrics

To validate the architectural and methodological improvements introduced in YOLOv8, its performance is evaluated using several key metrics. These metrics offer a quantitative basis for comparing YOLOv8 with previous models and for assessing its efficiency and effectiveness in real-world applications.

The following metrics are typically used to assess object detection models such as YOLOv8:

* Mean Average Precision (mAP): Measures detection accuracy across different object classes. Higher mAP values indicate better performance.
* Inference Time: Evaluates the speed at which the model processes images, which is crucial for real-time applications.
* Training Time: Assesses the efficiency of the training process by measuring how quickly the model reaches optimal performance.
* Model Size: Indicates the computational resources required for deployment. Smaller models are advantageous for devices with limited memory and processing power.

### YOLOv8 Models

* **YOLOv8n**: provides maximum efficiency and speed for resource-constrained devices, with a model size of approximately 2 MB in INT8 and 3.8 MB in FP32, making it ideal for edge, IoT, and mobile applications through support for ONNX Runtime and TensorRT.
* **YOLOv8s**: with around 9 million parameters, serves as a balanced baseline suitable for both CPUs and GPUs, leveraging enhanced spatial pyramid pooling and PANet for improved small object detection.
* **YOLOv8m**: comprising about 25 million parameters, delivers a mid-tier performance trade-off for real-time applications requiring higher precision.
* **YOLOv8l**: with roughly 55 million parameters, is designed for high-precision tasks such as medical imaging and autonomous driving through more complex feature extraction and refined attention mechanisms.
* **YOLOv8x**: the most powerful with approximately 90 million parameters, achieves the highest accuracy for applications like surveillance and detailed industrial inspections, though it necessitates high-end GPUs for real-time inference.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Model | Params (Million) | Accuracy (mAP@0.5) | CPU Time (ms) | GPU Time (ms) |
| YOLOv8n | 2.0 | 47.2 | 42 | 5.8 |
| YOLOv8s | 9.0 | 58.5 | 90 | 6.0 |
| YOLOv8m | 25.0 | 66.3 | 210 | 7.8 |
| YOLOv8l | 55.0 | 69.8 | 400 | 9.8 |
| YOLOv8x | 90.0 | 71.5 | 720 | 11.5 |

Table 1: Performance metrics for the YOLOv8 models.

### YOLOv8 Annotation Format

YOLOv8 uses an annotation format derived from the YOLOv5 PyTorch TXT format. Annotations are stored in a text file where each line corresponds to one object in the image. Each line contains the class label followed by the normalized coordinates of the bounding box, relative to the image dimensions. The format is as follows:

<class> <center\_x> <center\_y> <width> <height>

For example, an annotation may look like this:

0 0.635 0.478 0.157 0.286

This annotation format is supported by a YAML configuration file that specifies the model’s architecture and the class labels. The format ensures that YOLOv8 can be easily adapted to different datasets and tasks. For compatibility, annotations generated by tools such as Roboflow, VOTT, LabelImg, and CVAT may need to be converted to match the YOLOv8 format. These tools typically provide export options or conversion utilities to facilitate this process.

## Overview of RT-DETR

**Real-Time Detection Transformer (RT-DETR)**, developed by **Baidu**, is a cutting-edge end-to-end object detector that provides real-time performance while maintaining high [accuracy](https://www.ultralytics.com/glossary/accuracy). It is based on the idea of DETR (the NMS-free framework), meanwhile introducing conv-based backbone and an efficient hybrid encoder to gain real-time speed. RT-DETR efficiently processes multiscale features by decoupling intra-scale interaction and cross-scale fusion. The model is highly adaptable, supporting flexible adjustment of inference speed using different decoder layers without retraining. RT-DETR excels on accelerated backends like CUDA with TensorRT, outperforming many other real-time object detectors.

### Key features

* **Efficient Hybrid Encoder:** Baidu's RT-DETR uses an efficient hybrid encoder that processes multiscale features by decoupling intra-scale interaction and cross-scale fusion. This unique Vision Transformers-based design reduces computational costs and allows for real-time [object detection](https://www.ultralytics.com/glossary/object-detection).
* **IoU-aware Query Selection:** Baidu's RT-DETR improves object query initialization by utilizing IoU-aware query selection. This allows the model to focus on the most relevant objects in the scene, enhancing the detection accuracy.
* **Adaptable Inference Speed:** Baidu's RT-DETR supports flexible adjustments of inference speed by using different decoder layers without the need for retraining. This adaptability facilitates practical application in various real-time object detection scenarios.

### RT-DETR Structure

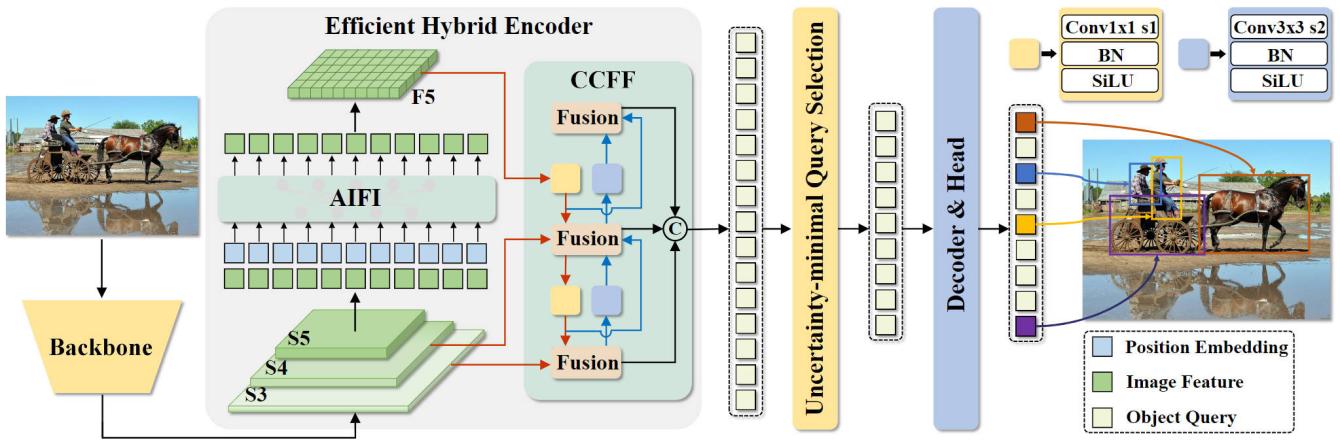


Figure 8 RT-DETR Architecture.

The RT-DETR model architecture diagram shows the last three stages of the backbone {S3, S4, S5} as the input to the encoder. The efficient hybrid encoder transforms multiscale features into a sequence of image features through intrascale feature interaction (AIFI) and cross-scale feature-fusion module (CCFM). The IoU-aware query selection is employed to select a fixed number of image features to serve as initial object queries for the decoder. Finally, the decoder with auxiliary prediction heads iteratively optimizes object queries to generate boxes and confidence scores

#### Backbone Network

Extracts rich, multi-scale feature maps from the input image. Features are extracted from the last three stages of the backbone, denoted as {S3, S4, S5}. These multi-scale features are crucial because they provide both fine details (from lower-level features) and strong semantic representations (from higher-level features).

Multi-Scale Feature:

* S3: Lower-level features with high spatial resolution but relatively weak semantic content.
* S4: Intermediate-level features that balance spatial details and semantic abstraction.
* S5: High-level features that contain strong semantic information (e.g., object-level concepts) but have coarser spatial resolution.

#### Efficient Hybrid Encoder

The encoder is designed to process the concatenated multi-scale features efficiently while reducing the heavy computational cost of applying full self-attention across all scales.

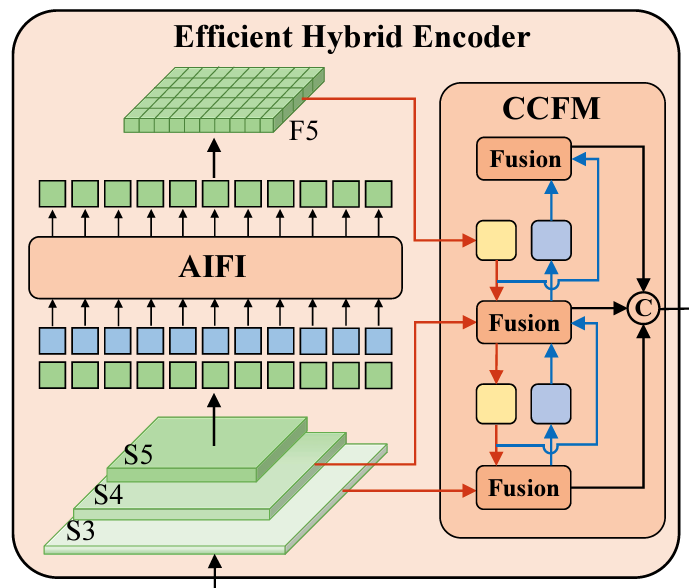


Figure 9 The Encoder of RT-DETR

This Efficient Hybrid Encoder consist of two key modules:

* Attentionbased Intra-Scale Feature Interaction (AIFI): Capture semantic relationships and dependencies **within the high-level features (S5)**. Implementation details:
* Flattening: S5 is flattened into a sequence of tokens (each token representing a spatial location) so that it can be fed into a self-attention module.
* Self-Attention Processing: A single-layer Transformer block (or similar attention mechanism) is applied to this sequence, allowing tokens to interact and share context.
* Reshape: The output is reshaped back into the spatial dimensions of S5, yielding an enhanced feature map, referred to as F5.
* CNNbased Cross-scale Feature Fusion (CCFF): Fuse the enhanced high-level features (F5) from AIFI with the lower-level features (S3 and S4) to create a unified representation that captures both detailed spatial and robust semantic information.
* 1×1 Convolutions: Adjust the number of channels for each input (S3, S4, and F5) to ensure compatibility.
* RepConv Blocks: A series of replication-based convolution blocks (RepBlocks) are used for effective feature integration. These blocks are lightweight yet powerful for feature fusion.
* Element-wise Addition: The outputs of two parallel paths (often representing different fusion strategies) are merged via element-wise addition.

The calculation of the hybrid encoder formulated as:

* (Query): In self-attention, queries are used to determine which parts of the input should attend to others.
* (Key): Keys are compared with queries to compute attention scores.
* (Value): Values are the information that is aggregated based on the attention scores.

All three Q, K, V are set to the flattened version of S5. This means that the self-attention mechanism will operate on the high-level features from S5.

#### Transformer Decoder

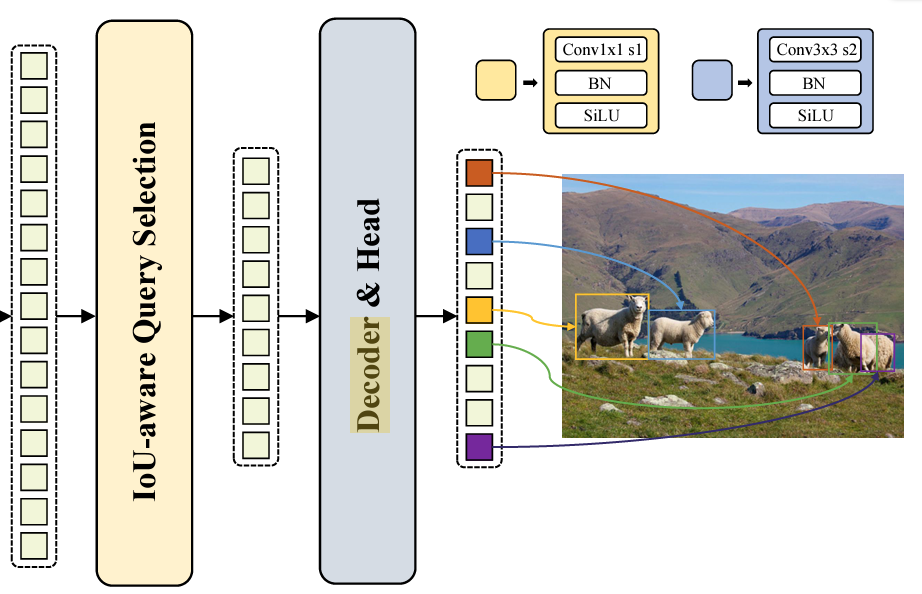


Figure 10 Decoder of the RT-DETR model.

The decoder does not receive all of the visual features from the hybrid encoder as input. Instead, a set of picture features is chosen by the IoU Aware Query Selection module to serve as the Transformer Decoder's initial input.

The boxes and the confidence scores are produced by the Decoder, which is made up of auxiliary prediction heads.

### IoU-driven Query Selection

The overlap between a predicted bounding box and the actual ground- truth box is measuared by the intersection over union (IoU), a crucial metric in computer vision. In object detection, where the objective is to locate things as precisely as possible, it is essential.

IoU-driven Query Selection: deliberately gives preference to proposals with higher IoU scores rather than evaluating all object proposals equally, which improves accuracy and reduces false detections. This guarantees that the final forecasts are based on only the most accurate and pertinent object inquiries.

The perform steps:

* **Producing Several Proposals**: For every object that is spotted, the model predicts several bounding boxes.
* **Calculating IoU Scores:** The IoU score of each projected box is determined by comparing it to the ground truth.
* **Choosing the Best Queries:** The model prioritizes bounding boxes that surpass a predetermined threshold and ranks them according to their IoU scores.
* **Refining Predictions:** The model assigns greater weight to high-IoU queries rather than averaging all proposals, which results in more accurate object localization.

The way makes it vital:

* **Reduces False Positives**: Low-IoU proposals that could result in inaccurate object detections are ignored.
* **Increases Bounding Box Accuracy:** improved localization is achieved by concentrating on queries that match ground-truth items well.
* **Improves Training Stability:** By eliminating erroneous object proposals, the model is kept from being misled, resulting in quicker convergence and improved learning.

|  |  |  |
| --- | --- | --- |
| Feature | Traditional Query Selection | IoU-driven Query Selection |
| Selection Criteria | Random or confidence-based | Prioritizes high-IoU proposals |
| Accuracy | Moderate | Higher accuracy due to precise selection |
| Computational Overhead | Higher (due to redundant low-quality queries) | Lower (by filtering out weak proposals early) |

Table 2: Comparison with traditional Query Selection.

### Dynamic Inference Speed Adjustment

Inference speed is the time it takes for a model to process incoming data and make predictions. In many real-world settings, a fixed-speed model may be inefficient. Dynamic inference speed adjustment allows a model to adapt its computational complexity based on real-time conditions, balancing accuracy and efficiency.

A model with adaptive inference speed can adjust its processing pipeline in response to changing conditions. It achieves this by:

* **Selective Computation Scaling:** Changing the number of layers, attention heads, or convolutional operations according to available resources.
* **Early-Exit Mechanisms:** If the model is confident in its prediction at an intermediate level, it can halt processing and reduce delay.
* Resolution Adaptation: Lowering input resolution for less crucial operations to minimize calculation time and preserve acceptable accuracy.

Benefits of Adaptive Inference Speed:

* **Balances speed and precision:** by adjusting computing effort dynamically to match real-time restrictions.
* **Optimizes Resource Usage:** Reduces superfluous processing to save energy and computational power.
* **Enhances Real-Time Performance:** Ideal for applications needing fast response and high accuracy, such video analytics and robotics.

### RT-DETR Performance Metrics

RT-DTER (Real-Time Dynamic Traffic Equilibrium Routing) is evaluated using a comprehensive set of performance measures that consider efficiency, reliability, safety, and environmental effect.

Traffic efficiency metrics: Average Travel Time (ATT) and Travel Time Index (TTI) are traffic efficiency indicators that quantify congestion levels, whereas Traffic Flow Rate (Q) and Queue Length (QL) examine movement smoothness. Congestion and delay measurements, such as Vehicle Delay Time (VDT) and Level of Service (LOS), indicate how well traffic is managed. Environmental parameters such as Fuel Consumption (FC) and Emission Factor (EF) measure the system's impact on sustainability.

Congestion and delay metrics: such as Vehicle Delay Time (VDT) and Level of Service (LOS), assist in determining how successfully RT-DTER eliminates bottlenecks and balances demand across alternate routes, resulting in smoother vehicular movement and fewer disturbances. Environmental sustainability is critical, with measurements like Fuel Consumption (FC) and Emission Factor (EF) measuring carbon footprint reductions through optimized driving patterns and less stop-and-go traffic. The system also improves road safety, as evidenced by Accident Rate (AR) and Incident Response Time (IRT), which evaluate decreases in road incidents and emergency response times, respectively. A lower accident rate indicates better driving conditions, and a quicker incident response time guarantees that disturbances are dealt quickly to avoid secondary congestion.

System performance metrics: Processing Latency (L) and Network Uptime (%) are system performance indicators that measure the speed and reliability of data processing, ensuring that real-time updates are communicated without delay and allowing for immediate route alterations. A low latency value assures minimum delays in traffic optimization, whilst a high network uptime ensures continuous system availability and uninterrupted operation. Together, these metrics provide a comprehensive assessment of RT-DTER's effectiveness in dynamically optimizing traffic flow, reducing congestion, improving road safety, minimizing environmental impact, and ensuring high system reliability, resulting in a more intelligent and sustainable urban transportation network.

### RT-DETR models

RT-DEtR include 2 models:

* RT-DETR-L (Large): Achieves 53.0% Average Precision (AP) on the COCO validation dataset and operates at 114 frames per second (FPS) on an NVIDIA T4 GPU.
* RT-DETR-X (Extra Large): Offers 54.8% AP on the COCO validation dataset with a processing speed of 74 FPS on a T4 GPU.

# METHODOLOGY

## Data Collection

Experiments were carried out using publicly available datasets to ensure reproducibility and generality. For our specific domain, we used the football-players-detection dataset from Roboflow. This dataset, derived from images of an actual football match, consists of 298 training images, 49 validation images, and 25 test images, all originally captured at a resolution of 1200 pixels. The dataset is provided in a YOLO-compatible format along with a data.yaml file that defines the training, validation, and test image paths and class labels (e.g., ['football', 'player', 'referee']).

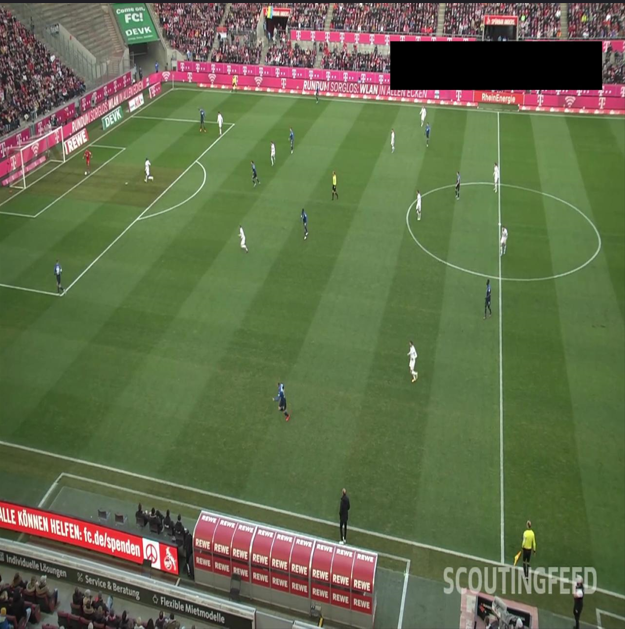


Figure 11 Data example.

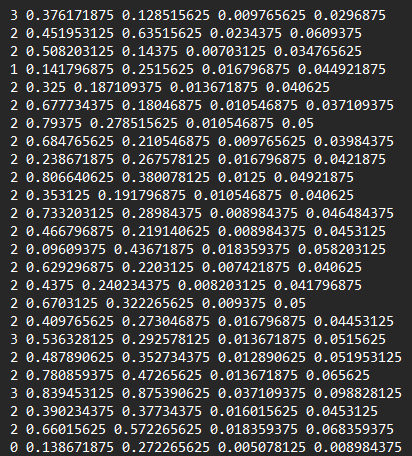


Figure 12 Label of the image shown in Fig. 10.

## Experimental Setup

All experiments were conducted on Google Colab using a Tesla T4 GPU (15,095 MiB memory). This environment ensured consistent performance comparisons and simulated real-time deployment scenarios.

Both YOLOv8 and RT-DETR were implemented using PyTorch (torch-2.5.1+cu124) on Python 3.11.11. The pre-trained YOLOv8 model (Ultralytics version 8.3.75, e.g., yolov8s.pt) and the corresponding pre-trained RT-DETR weights were loaded for evaluation.

Although our primary focus was on comparative evaluation using pre-trained weights, both models were optionally fine-tuned on the football-players-detection dataset. For YOLOv8, training was conducted with a batch size of 12 over 40 epochs (image size: 480), while RT-DETR was trained with a batch size of 8 over 20 epochs.

## Quantitative Metrics

* Mean Average Precision (mAP): Assesses detection accuracy across classes.
* Inference Speed (FPS): Measures the speed of image processing, critical for real-time applications.
* Model Complexity: Quantified by the number of parameters
* Computational Cost (GFLOPs)

## Limitations

* Hardware Constraints: Performance metrics such as FPS may vary across different hardware configurations. The results reported here are based on evaluations conducted on Google Colab’s Tesla T4 GPU.
* Dataset Bias: The football-players-detection dataset is domain-specific and may exhibit biases in object scales, scene complexity, or lighting conditions. Such factors could favor one model architecture over another and might limit the generalizability of the findings.
* Model Maturity: YOLOv8 is a mature, widely adopted model with extensive industry support, while RT-DETR is relatively newer. Differences in implementation maturity may influence aspects such as deployment ease, inference stability, and overall performance.
* Annotation Format Consistency: It is assumed that the annotations in the dataset are consistent and correctly formatted as specified in the data.yaml file. Any discrepancies could affect the evaluation of detection metrics.

.

# RESULTS

## Quantitive Analysis

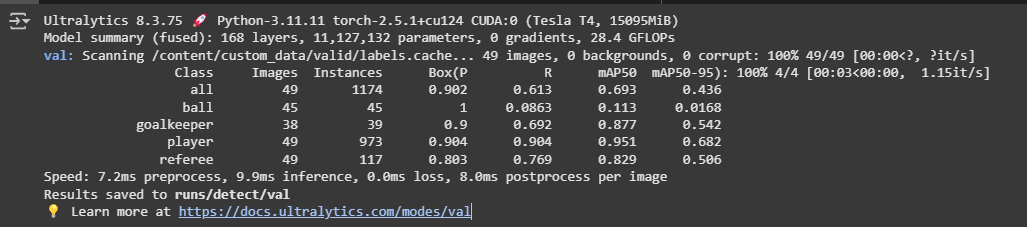


Figure 13 YOLOv8 Validation output.

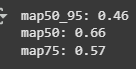


Figure 14 RT-DETR Validation output.

*Detection Accuracy:*

* YOLOv8: On the validation set (49 images, 1,174 instances), YOLOv8 achieved an mAP50 of 69.3% and an mAP50-95 of 43.6%, demonstrating robust detection performance across the defined classes.
* RT-DETR: The model recorded an mAP50 of 66%, an mAP50-95 of 46%, and an mAP75 of 57%. These results indicate competitive accuracy, particularly in detecting smaller or overlapping objects, thanks to its transformer-based attention mechanism.

*Inference Speed*: YOLOv8 is known for its high inference speed owing to its efficient CNN-based architecture. RT-DETR, although slightly slower due to the additional computational overhead of transformer components, still meets real-time constraints.

*Model Complexity and Computational Cost:*

* YOLOv8: Features a lightweight architecture with approximately 11.1 million parameters and requires about 28.4 GFLOPs, resulting in lower computational overhead.
* RT-DETR: Incorporates transformer components that increase the parameter count and computational cost (32 million parameters and 110 GFLOPs); however, optimization techniques mitigate some of this overhead.

## Qualitavie Observations

Detection in Crowded Scenes: RT-DETR exhibits improved performance in scenarios with heavy occlusion and overlapping objects, likely due to its global attention mechanism, which better captures contextual relationships.

Robustness to Variations: YOLOv8 demonstrates consistent performance across different lighting conditions and object scales, reflecting its mature architecture and extensive training on diverse datasets.

# CONCLUSION & DISCUSSION

## Conclusion

Overall, the results indicate that while YOLOv8 excels in applications requiring rapid inference and low computational overhead, RT-DETR offers enhanced accuracy in complex detection scenarios. The trade-off between inference speed and detection precision underscores the importance of selecting a model based on the specific operational requirements of the target application.

The comparative analysis reveals that YOLOv8’s superior inference speed and lower computational cost make it ideal for real-time applications where rapid decision-making is crucial. Conversely, RT-DETR’s transformer-based design, despite a higher computational burden, delivers a more holistic scene understanding that enhances accuracy in cluttered or occluded environments. YOLOv8 leverages an efficient CNN-based backbone to achieve high speed and low complexity. In contrast, RT-DETR’s integration of transformers, while more computationally demanding provides significant benefits in handling complex visual contexts.

The choice between YOLOv8 and RT-DETR should be based on the specific needs of the application:

* YOLOv8 ideal use cases: Real-time Surveillance, Industrial Automation, Mobile and Edge Deployments, Webcam based applications, …
* RT-DETR ideal use cases: Autonomous Driving, Robotics, Advanced Security Systems, …

## Further work

While this study provided a comprehensive comparison of YOLOv8 and RT-DETR for real-time object detection in the context of football match imagery, its findings are limited by the scope of the dataset, which focused on a specific sports domain with 298 training images, 49 validation images, and 25 test images at a fixed resolution. Future work should extend this evaluation to a broader range of datasets that encompass diverse scenes, environments, and object types. This would help determine the models’ generalizability and robustness across different real-world applications. Additionally, further investigations could explore domain adaptation techniques, hybrid approaches that combine the efficiency of CNN-based methods with the contextual strengths of transformer architectures, and fine-tuning strategies for different operational contexts. Finally, integrating additional tasks such as object tracking and segmentation may provide a more complete understanding of the models’ capabilities in dynamic, complex scenarios.

# REFERNCES

1. Achit, M., Yassa, N., Bouzida, A., Mazari, A., & Fergani, O. (2024). Enhancing PV panel diagnosis: RT-DETR vs YOLO v8 for real-time hotspots detection. STUDIES IN ENGINEERING AND EXACT SCIENCES, 5(2), e8390. <https://doi.org/10.54021/seesv5n2-268>
2. Hidayatullah, P., Syakrani, N., Sholahuddin, M. R., Gelar, T., & Tubagus, R. (2025, January 23). YOLOV8 to YOLO11: A Comprehensive Architecture In-depth Comparative Review. arXiv.org. <https://arxiv.org/abs/2501.13400>
3. Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2015, June 8). *You only look once: Unified, Real-Time Object Detection*. arXiv.org. <https://arxiv.org/abs/1506.02640>
4. Ultralytics. (2025a, January 4). RTDETRv2 vs YOLOv8: A Detailed Model Comparison. Ultralytics YOLO Docs. [https://docs.ultralytics.com/compare/rtdetr-vs-yolov8/#rtdetrv2-real-time-detection-transformer-v2](https://docs.ultralytics.com/compare/rtdetr-vs-yolov8/" \l "rtdetrv2-real-time-detection-transformer-v2)
5. Ultralytics. (2025c, January 26). *RT-DETR (Realtime Detection Transformer)*. Ultralytics YOLO Docs. <https://docs.ultralytics.com/models/rtdetr/>
6. Yaseen, M. (2024, August 28). *What is YOLOv8: An In-Depth Exploration of the Internal Features of the Next-Generation Object Detector*. arXiv.org. <https://arxiv.org/abs/2408.15857>
7. Zhao, Y., Lv, W., Xu, S., Wei, J., Wang, G., Dang, Q., Liu, Y., & Chen, J. (2023, April 17). *DETRs beat YOLOs on real-time object detection*. arXiv.org. <https://arxiv.org/abs/2304.08069>
8. Ultralytics. (2025d, February 13). *YOLOV8*. Ultralytics YOLO Docs. <https://docs.ultralytics.com/models/yolov8/>
9. Skalski, P. (2024, July 11). *How to Train RT-DETR on a Custom Dataset with Transformers*. Roboflow Blog. [https://blog.roboflow.com/train-rt-detr-custom-dataset-transformers/#load-pre-trained-rt-detr-model](https://blog.roboflow.com/train-rt-detr-custom-dataset-transformers/" \l "load-pre-trained-rt-detr-model)
10. Skalski, P. (2024b, December 5). *How to train YOLOV8 object detection on a custom dataset*. Roboflow Blog. <https://blog.roboflow.com/how-to-train-yolov8-on-a-custom-dataset/>