www.editada.org

International Journal of Combinatorial Optimization Problems and Informatics, 16(3), May-Aug 2025, 360-377. ISSN: 2007-1558. https://doi.org/10.61467/2007.1558.2025.v16i3.778

An analysis of YOLO models versus RT-DETR applied to multi-object detection in images

Alan J. González-Hernández[1], Juan P. Sánchez-Hernández [2], Deny L. Hernández-Rabadán[2], Juan Frausto-Solis[1], Juan J. González-Barbosa[1]

[1] Graduate & Research Division, Tecnológico Nacional de México, Instituto Tecnológico de Ciudad Madero, Ciudad Madero 89440, México
[2] Information of Technology Division, Universidad Politécnica del Estado de Morelos, Boulevard Cuauhnáhuac 566, Jiutepec 62574, México
D09070523@cdmadero.tecnm.mx, juan.paulosh@upemor.edu.mx, dhernandezr@upemor.edu.mx, juan.frausto@gmail.com, jjgonzalezbarbosa@hotmail.com
Corresponding: Juan Paulo Sánchez Hernández (juan.paulosh@upemor.edu.mx)

Abstract. Object detection is one of the most critical and essential tasks in computer vision, with applications ranging from surveillance and industrial control to robotics and image analysis. This research presents a performance analysis of different versions of YOLO (You Only Look Once) and a transformer-based model. The study evaluates YOLOv8, YOLOv9, YOLOv10, YOLOv11, and the RT-DETR (Real-Time Detection Transformer) model for object detection. The experiment uses a dataset comprising 1,730 images classified into five categories: birds, dogs, cats, plants, and fruits, each with its own subtypes. In addition, we evaluate a frog dataset consisting of 613 images, which are considered complex due to occlusion, intricate backgrounds, and variations in lighting. Performance is assessed using standard metrics such as precision, recall, mAP50, and mAP50–95.
Keywords: Object Detection, Computer Vision, Transformer Model, RT-DETR.

Article Info
Received 31 January 2025
Accepted 11 March 2025

1 Introduction

Object detection and classification in images are fundamental tasks in computer vision with a wide range of applications in recent years. For instance, the relevant applications are involved in surveillance control (Zhao et al., 2019), industrial processes (Ren et al., 2020), object tracking (Bertinetto et al., 2016), medical image analysis (Yang & Yu, 2021), and robotics (Levine et al., 2016).

The research of new models and applications in this area can bring improvements in the performance of new intelligence systems. On the other hand, Deep Learning is a set of algorithms, which apply novel techniques to solve object detection problems successfully. One of the Deep Learning algorithms is YOLO (You Only Look Once) proposed by (Redmon et al., 2016), which has been applied successfully in object detection. YOLO has evolved significantly in recent years, and each new version offers improvements in performance and efficiency. Alternatively, the transformer models have been implemented in object detection, in particular, the RT-DETR (Real Time Detection Transformer) model in one of these models with applications in object detection proposed by (Zhao et al., 2023). RT-DETR is designed to improve real-time performance while maintaining a balance between accuracy and speed of object detection.

In this work, we evaluate the performance of recent YOLO versions such as YOLOv8 (Jocher et al., 2023), YOLOv9 (Wang et al., 2024), YOLOv10 (Wang et al., 2024), and YOLOv11 (Jocher, G., & Qiu, J. 2024), as well as the RT-DETR model (Zhao et al., 2023). The datasets used in this analysis include images of dogs, cats, plants, fruits, birds, and frogs, which are categorized into specific subtypes and complex backgrounds. The ability to classify into types and subtypes adds complexity, which in turn

is important in many applications, obtaining a more appropriate and specific response to the object being recognized. The metrics used to evaluate the performance are Precision, Recall, mAP50, and mAP50-95.

This work is organized as follows. In the introduction section, we present the context of the problem. In the analysis of deep learning models section, we describe the models of YOLO and RT-DETR. In the section YOLO and RT-DETR applications, we review the researchers where the models of YOLO and RT-DETR have been applied. In the performance metrics sections, we describe the main metrics used in this work. In the Methodology section, we explain the steps used in our methodology. In the results section, we present all the results obtained in the experimental phase. Finally, the conclusions are presented in the last section.

2 Background of deep learning architectures

The present study aims to analyze the impact of YOLO and RT-DETR models in the field of real-time object detection. The study will examine the versions of YOLO ranging from 8 to 11, as well as the application of transformer models such as RT-DETR. The models will be analyzed to ascertain their ability to address the challenges in object detection. The objectives of the study are as follows:

- Evaluation of YOLO versions versus RT-DETR: The progression of YOLO versions, from YOLOv8 to YOLOv11, is analyzed, as well as the transition towards transformer-based models, such as RT-DETR.
- Exploration of practical applications: Specific applications are identified where YOLO and RT-DETR models have demonstrated high efficiency in classification and segmentation. These examples highlight the versatility of these architectures in uncontrolled environment scenarios.
- Critical performance analysis: A detailed performance evaluation of YOLO versions and RT-DETR is performed, considering key metrics such as accuracy, recall, mAP50, mAP5095, Training time, and Latency. This analysis allows identifying the strengths and limitations of the datasets used in this work.

In summary, the objective of this paper is to provide a comprehensive overview of the evolution and impact of the YOLO versions and the RT-DETR model. Furthermore, we are undertaking a new exploration of the capabilities of the deep learning architectures reviewed in this study.

2.1 YOLO model

The YOLO model was a revolutionary development in the field of real-time object detection when Joseph Redmon presented the work (Redmon et al., 2016). This model represented a radical departure from conventional region-based methods, such as R-CNN and Fast R-CNN, by integrating detection and classification tasks into a unified step. The main goal of YOLO was to achieve a balance between accuracy and speed. Its design allowed complete images to be processed in real time with a single pass through the network, achieving fast and efficient detection without significantly compromising performance. The original YOLO model is based on dividing the input image into an S×S grid, where each cell predicts bounding boxes, the confidence of each prediction, and the associated classes. The YOLO architecture incorporates a straightforward convolutional neural network that processes the entire image, thereby obviating the necessity to generate region proposals, as had been the case in previous architectures. This innovation yielded a faster architecture with reduced vulnerability to duplicate detection errors. The initial YOLO architecture incorporated 24 convolutional layers, succeeded by 2 fully connected layers. The convolutional layers were pre-trained using a set of ImageNet images of size 224x224.

2.1.1 YOLO versions evolution

The YOLO model has evolved through multiple versions developed by distinct research teams and technology communities. Each iteration has introduced significant innovations aimed at enhancing the efficiency, accuracy, and generalizability of the

model for various object detection tasks. Although there are important intermediate versions, this study will describe the main features and advances that have emerged from YOLOv1 to YOLOv11 (Figure 1). These versions represent a technological advancement within the YOLO family, which makes them relevant for modern applications and challenges in computer vision.

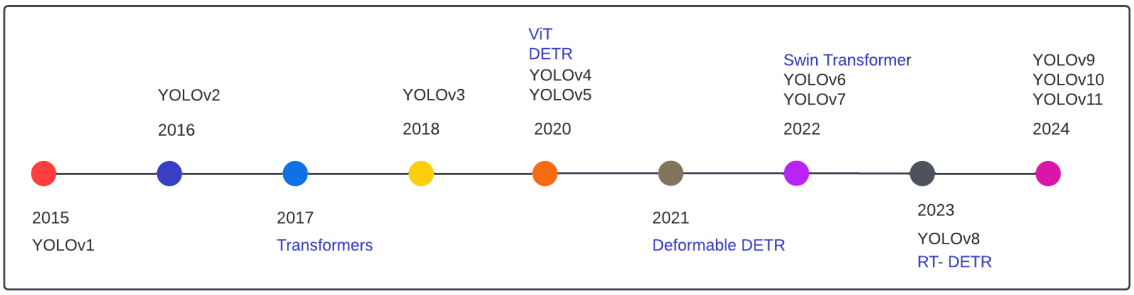

Fig. 1. The YOLO evolution

2.1.2 YOLOv5

In 2020, Glen Jocher introduced the YOLOv5 model (Jocher, 2020). A salient feature of YOLOv5 is the introduction of scalable versions, which are designated as nano (n), small (s), medium (m), large (l), and extra-large (x). These versions vary in size and computational capacity, allowing them to be adapted to different needs and devices, from those with limited resources to high-precision applications.

In terms of architecture, YOLOv5 implements a backbone based on CSP-Darknet53, a modification of the original Darknet53, which improves efficiency and reduces redundancy of extracted features. In the neck, it uses SPPF (Spatial Pyramid Pooling Fast) and CSP-PAN modules for better feature fusion. Finally, in the head, it maintains the YOLOv3 Head structure, facilitating the prediction of bounding boxes and labeled classes.

YOLOv5 also incorporates several advanced data augmentation techniques that improve model generalization. These techniques include: Mosaic Augmentation, MixUp, Copy-Paste, random transformations, HSV augmentation, and random horizontal flipping. Additionally, it employs sophisticated training strategies, such as multiscale training, AutoAnchor for automatic anchor optimization, Warm-Up, Learning Rate Scheduling (Cosine LR), Exponential Moving Average (EMA), Mixed Precision Training, and Hyperparameter Evolution.

Concerning the calculation of loss, YOLOv5 employs three primary components: Class Loss (L_{cls}), Loss of Objectness (L_{obj}), and Location Loss (L_{loc}). The latter evaluates the prediction accuracy of bounding boxes. The resulting global loss function (Loss) is shown in Equation (1):

$$Loss = \lambda_1 L_{cls} + \lambda_2 L_{obj} + \lambda_3 L_{loc} \quad (1)$$

Where:

$\lambda_1 L_{cls}$: classification Loss

$\lambda_2 L_{obj}$: objectness Loss

$\lambda_3 L_{loc}$: localization Loss

2.1.3 YOLOv6

In 2022, Meituan introduced YOLOv6 as an optimized solution for industrial and mobile applications (Li et al., 2023). This iteration of YOLO was meticulously engineered to optimize efficiency on devices with limited resources while ensuring the retention of accuracy levels and the capacity to adapt to real-time object detection tasks.

In terms of its architectural design, YOLOv6 uses a backbone based on RepVGG, a structure that combines computational efficiency and compression capacity through quantization. This makes it ideal for low-latency applications. In the neck, it implements an enhanced CSPNet (Cross Stage Partial Network) module to optimize feature fusion, while its head incorporates improvements that enable accurate and fast detection.

YOLOv6 stands out for introducing advanced techniques to optimize both training and model performance. Among these techniques are the bi-directional Concatenation (BiC) module, Anchor-Assisted Training (AAT) strategy, improved spine and neck design, and Self-distillation strategy. Particularly, YOLOv6 marks a significant advancement in the YOLO family by introducing optimizations that maximize performance in low-latency scenarios, making it especially useful for industries that require lightweight and accurate solutions. These innovations consolidate YOLOv6 as a state-of-the-art tool for real-time object detection.

2.1.4 YOLOv7

YOLOv7, released in 2022 (Wang et al., 2022), introduces several improvements to enhance its performance, including model re-parameterization and dynamic label assignment, which optimize the training process and inference accuracy. One of its key innovations is model re-parameterization, a strategy that adjusts network layers to optimize gradient propagation during training. This approach not only accelerates learning but also improves the accuracy of final predictions.

Another notable innovation is dynamic label assignment, a method designed to address the challenge of dynamically assigning targets in networks with multiple output layers. Additionally, YOLOv7 incorporates extended and compound scaling methods, optimizing parameter usage and computational efficiency. These techniques enable the model to maintain high-precision performance even in lightweight configurations, making it suitable for resource-constrained devices.

In terms of efficiency, YOLOv7 reduces the number of parameters by 40% and the computational cost by 50% compared to other advanced object detectors such as YOLOv6 and YOLOv5, while simultaneously improving inference speed and accuracy. The YOLOv7 architecture also features an optimized backbone based on CSPNet, which enhances gradient flow and minimizes computational redundancy. Moreover, it integrates E-ELAN blocks, which expand the model's learning capacity by enabling more efficient multi-scale feature extraction. Its advanced neck, which utilizes a Path Aggregation Network (PAN), enhances small object detection, while auxiliary training modules stabilize the training process and accelerate convergence.

2.1.5 YOLOv8

The YOLOv8 model was first introduced in 2023 (Jocher, 2023). This version focused on optimizing the flexibility of the model, allowing for easier integration for different computer vision tasks. YOLOv8 was specifically designed to offer significant improvements in both accuracy and speed, with a particular focus on adaptability for various scenarios and devices. In terms of architecture, YOLOv8 introduced key innovations, such as an Anchor-Free detection approach, which eliminates the need to predefine anchor boxes, simplifying training and reducing computational complexity. Its backbone is based on CSPNet, with modifications to better handle multi-scale features. The neck uses advanced PAN structures for efficient feature fusion, while the head is completely redesigned to take advantage of anchorless prediction capabilities.

YOLOv8 incorporates a wide range of data augmentation techniques that improve the robustness and generalization of the model. These techniques include Mosaic V2, MixUp, Random Scaling, and other advanced methods from the package, such as color and geometry transformations (Buslaev et al., 2020). In addition, improvements in the loss functions were implemented to optimize the quality of predictions and ensure stability during the training process.

2.1.6 YOLOv9

The YOLOv9 model, which was introduced in 2024 (Wang et al., 2024), and represents a significant advance in the YOLO family of models. This version addresses critical problems such as information loss in deep networks by introducing the concept of Programmable Gradient Information (PGI). PGI allows deep networks to adapt to multiple targets by providing

comprehensive input information to compute objective functions, thereby ensuring more reliable gradients for updating network weights.

YOLOv9 incorporates a novel structure, termed the Generalized Efficient Layer Aggregation Network (GELAN), which has been developed based on gradient path planning. GELAN employs conventional convolutional operators to achieve superior parameter utilization in comparison to contemporary methods that utilize depth convolutions. This architecture has exhibited superior performance in lightweight models, optimizing feature extraction and aggregation. A comparison of GELAN with traditional architectures such as PlainNet, ResNet, and CSPNet reveals its superiority in preserving the most relevant information necessary for the computation of the objective function.

2.1.7 YOLOv10

The YOLOv10 model, developed in 2024 by researchers at Tsinghua University in collaboration with Ultralytics (Wang et al., 2024), represents a significant advance in the YOLO family by addressing key deficiencies in the post-processing and architecture of previous models. This version introduces an innovative approach that eliminates Non-Maximum Suppression (NMS) and optimizes several model components, achieving an outstanding balance between accuracy and latency.

The YOLOv10 architecture combines the strengths of previous versions with key performance-enhancing innovations. The backbone of YOLOv10 is based on an optimized version of CSPNet, improves gradient flow and reduces computational redundancy, while its neck uses PAN layers to efficiently fuse multi-scale features and facilitate small object detection. The new YOLOv10 model includes a dual head, which eliminates the need for NMS during inference, thus reducing latency and improving efficiency.

YOLOv10 adopts an approach that aligns with the end-to-end object detector paradigm, eliminating the reliance on manual components such as NMS. Inspired by modern trends such as DETR and its variants, YOLOv10 introduces a dual head with “one-to-many” predictions during training and “one-to-one” in inference. Besides, YOLOv10 not only simplifies the detection process but also significantly improves model latency and accuracy. Moreover, as other end-to-end detectors alternatives, such as RT-DETR and OneNet, this new Yolo version demonstrates that it is possible to achieve state-of-the-art performance without additional post-processing steps.

In terms of overall improvements, YOLOv10 employs dual coherent mappings for NMS-free training. In addition. enhance components such as lightweight sorting heads and space-channel down-sampling, maximizing efficiency. YOLOv10 also incorporates large kernel convolutions and partial self-attenuation modules, which expand the receptive field and improve overall learning, consolidating YOLOv10 as an innovative and efficient tool.

2.1.8 YOLOv11

The most recent version of the Ultralytics YOLO series, YOLOv11 (Jocher & Qiu, 2024), has been developed. This version introduces significant advancements in its architecture and training methods, establishing itself as a versatile solution for a wide range of computer vision tasks. This model redefines the standards in accuracy, speed, and efficiency, making it a state-of-the-art tool for researchers and engineers.

In terms of architecture, YOLOv11 features an improved backbone and neck, which is improved by accurate feature extraction and robust performance on complex tasks. These enhancements enable the model to handle high object density scenarios and multi-scale tasks more effectively. In addition, YOLOv11 has been engineered to strike a balance between accuracy and computational efficiency, through the introduction of optimized training pipelines that expedite processing without compromising the quality of predictions.

2.2 DETR model

The DETR (DEtection TRansformer) model, introduced by Facebook AI (Carion et al., 2020), signified a paradigm shift in object detection by eliminating the need for manual components such as NMS. DETR employs a transformer-based architecture,

which enables end-to-end detection by directly matching predictions with real objects using the Hungarian loss. This approach simplifies the workflow and improves the generalizability of the model.

However, although DETR offers significant advantages in terms of simplicity and accuracy, it has important limitations, such as high computational cost and reduced inference speed. These constraints have made it less practical for real-time applications, especially in scenarios where latency is critical. These limitations have led to the development of variants such as Deformable-DETR and, more recently, RT-DETR, which seek to overcome these obstacles without compromising the fundamental advantages of DETR.

2.2.1 RT-DETR model

The Real-Time DEtection TRansformer (RT-DETR), developed by Baidu (Zhao et al., 2023), redefines the real-time object detection landscape by combining unprecedented accuracy and speed. This model, based on the DETR architecture, introduces significant advances that optimize both computational efficiency and accuracy and overcome the traditional limitations of object detectors.

RT-DETR is distinguished by its efficient and adaptive architecture, which integrates several significant innovations. Its hybrid encoder decouples intrascale interaction and interscale fusion, allowing it to process multiscale features efficiently and significantly reducing computational costs, making it ideal for environments with high object density. Furthermore, the incorporation of an IoU-based query selection mechanism enhances the accuracy of the model by ensuring that it focuses on the most relevant objects from the outset, without compromising speed. Finally, its adaptive inference speed allows the number of decoder layers to be dynamically adjusted to meet different requirements without the need to retrain the model, making it highly versatile in a variety of scenarios.

RT-DETR demonstrates remarkable efficiency in speed and accuracy. Furthermore, RT-DETR's compatibility with CUDA and TensorRT technologies permitting to perform efficient implementations. Thus, it could become a practical device for industrial applicability.

RT-DETR is a significant advance in the evolution of real-time object detectors. Its efficient hybrid architecture, combined with NMS elimination and IoU-based query selection, makes it a highly accurate, efficient, and adaptable model. These advances not only set new standards in object detection but also position it as an ideal solution for modern applications requiring exceptional speed and accuracy.

2.3 YOLO and RT-DETR

Real-time object detection has experienced significant advances thanks to the evolution of models such as YOLO and RT-DETR. These models have redefined the way of approaching computer vision tasks, offering solutions that combine high accuracy, speed, and adaptability. Both models developed and promoted by Ultralytics, have proven to be powerful tools for a variety of applications that require efficient real-time object detection.

2.3.1 YOLO applications

Since its introduction, YOLO has established itself as a benchmark in real-time object detection due to its ability to process complete images in a single forward pass through the network (Redmon et al., 2016). This approach not only reduces computational complexity, but also enables extremely low inference times, making it ideal for applications in security, transportation, and agriculture.

Recent studies have demonstrated the efficacy of YOLO models in various fields, including vehicle identification and traffic monitoring. These models have shown significant potential in enhancing management and safety measures on urban roadways and highways (Kunekar et al., 2024; Bakirci, 2024). In the agricultural sector, YOLO models have exhibited remarkable efficiency in tasks such as pest detection, fruit sorting, and crop monitoring. This contributes to the optimization of agricultural processes (Badgujar et al., 2024).

In the domain of medicine, the YOLO algorithm has been utilized for the detection of lung lesions, cancer, and other abnormalities in X-ray and CT images. The YOLO capacity to process substantial volumes of data in real time renders it a pivotal instrument for expeditious and precise diagnoses (Ali et al., 2024). YOLOv8 and YOLOv11, have introduced key improvements to the architecture, including self-attenuation modules, advanced data augmentation strategies, and large kernel convolutions. These innovations have made it possible to achieve Mean Average Precision (mAP) close to 55% on the MS COCO dataset (Jocher et al., 2023; Wang et al., 2024).

2.3.2 The RT-DETR applications

RT-DETR represents a modern approach to transformer-based object detection, eliminating the dependence on NMS and improving both speed and accuracy compared to traditional architectures such as YOLO (Zhao et al., 2023). RT-DETR design is based on an efficient hybrid encoder, which decouples intrascale interaction and interscale fusion, optimizing multiscale feature processing.

One of the main innovations of RT-DETR is its IoU-based query selection, which improves the quality of initial predictions by focusing on the most relevant objects in the scene. This model also allows dynamically adjusting the inference rate by varying the number of decoder layers without retraining, which makes it highly adaptable to different applications (Zhao et al., 2023). RT-DETR has exhibited remarkable efficacy in autonomous navigation, industrial monitoring, and logistics management tasks. For instance, on the MS COCO dataset, RT-DETR-R50 attained a mAP of 53.1% with a speed of 108 FPS on T4 GPUs, surpassing conventional detectors such as YOLO in terms of accuracy and speed (Zhao et al., 2023).

2.3.3 YOLO and RT-DETR applications

YOLO and RT-DETR are two approaches to optimizing real-time object detection. Their technical approaches are complementary. YOLO, based on convolutional architectures, is designed to maximize speed and adaptability. RT-DETR, on the other hand, adopts end-to-end transformers, eliminating manual components such as NMS and improving scalability. This duality allows both models to address specific challenges in a wide range of applications. The relevance of these models has been highlighted in multiple studies comparing their capabilities in various scenarios. The subsequent examples are indicative of the model's effectiveness:

- Liquid Leak Detection Using Thermal Images (Bansod et al. 2023). This study, employed YOLO and RT-DETR to address the challenge of detecting liquid leaks in the oil and gas industry using thermal images. YOLO demonstrated its speed in recognizing leaks and related patterns in real time, while RT-DETR excelled in identifying specific features in the thermal images, improving overall accuracy. This approach combined both models to optimize continuous monitoring and analysis, reducing environmental and financial risks.
- Real-Time Detection and Analysis of Vehicles and Pedestrians using Deep Learning (Sadik et al. 2023) evaluated the ability of YOLOv8 and RT-DETR to identify vehicles and pedestrians in complex urban environments. In this work, it was highlighted that YOLOv8 (large version) showed high accuracy and robustness for real-time pedestrian detection, while RT-DETR provided advantages in scenarios with small objects and challenging environmental conditions. The results showed significant improvements in mAP and Recall rates, positioning these models as key tools for traffic management and safety.
- First Qualitative Observations on Deep Learning Vision Model YOLO and DETR for Automated Driving in Austria (Schoder, 2023), were reported by Schoder. The study examined the utilization of YOLO and RT-DETR in advanced driver assistance systems (ADAS) and autonomous vehicles on Austrian roads. The study evaluated specific sensing challenges in rural, urban, and alpine environments. In these environments, YOLO demonstrated its capacity to identify and track objects with high speed. RT-DETR, on the other hand, exhibited enhanced accuracy in complex scenarios, such as adverse weather conditions. These results underscore the significance of the technology in the realm of automated driving.
- Utilizing RT-DETR Model for Fruit Calorie Estimation from Digital Images (Tang & Yan, 2024). In this paper, Tang and Yan explored the use of the RT-DETR model for the estimation of fruit calorie content from digital images. The fruit dataset was created, and data augmentation techniques were employed. The RT-DETR model integrated into the Ultralytics framework outperformed YOLOv10 on metrics such as accuracy (99.01%) and mAP50-95 (94.45%). This approach highlighted RT-DETR's ability to improve nutritional monitoring by accurately detecting fruits and estimating their caloric content.

- Enhanced Weed Detection Using YOLOv9 on Open-Source Datasets for Precise Weed Management (Malik & Mahmud, 2024). In a weed detection study, Malik and Mahmud evaluated YOLOv9, RT-DETR, and other models on two open datasets with crop and weed images. The analysis revealed that YOLOv9 outperformed the other models, achieving a mAP of 0.94 and 0.85 for sets A and B, respectively. While RT-DETR demonstrated competitive accuracy, its inference speed was significantly slower. This study underscores the significance of selecting an appropriate model that aligns with the demands of speed and accuracy in the context of agricultural weed management.

These examples underscore the complementary nature of YOLO and RT-DETR in addressing particular sensing challenges. While YOLO is ideal for applications that require rapid responses, such as traffic and surveillance systems, RT-DETR demonstrates particular aptitude in scenarios where accuracy and robustness are paramount, including thermal imaging and complex environments. The ability of RT-DETR to address contemporary issues and its potential for future applications underscores its position at the forefront of the field of computer vision.

3. Performance metrics

In this section, we describe the metrics employed to assess the performance of models on the HORD and frog datasets. These metrics, widely used in the computer vision community, have their roots in standards such as those defined by the COCO (Lin et al., 2014) and PASCAL VOC (Everingham et al., 2010) datasets.

3.1 Precision

Accuracy measures the proportion of correct predictions among all predictions made for a specific class. It is particularly important in applications where false positive errors can have significant consequences, such as anomaly detection. The equation of the accuracy metric is shown in Equation (2).

$$Precision = \frac{TP}{TP+FP} \quad (2)$$

where TP: True positives
FP: False positives

3.2 Recall

Recall metric measures the ability of the model to correctly detect the objects present in the images, considering the true positives concerning the total number of relevant objects. The Recall metric is shown in Equation (3).

$$Recall = \frac{TP}{TP + FN} \quad (3)$$

where, FN: False negatives
TP: True positives

3.3 Mean Average Precision

Mean average precision (mAP) is a standard metric for the evaluation of object detection models. It represents the average of the calculated accuracy at different Intersection over Union (IoU) thresholds. The IoU metric is shown in Equation (4) for the mAP50, and mAP50-95 alternatives: a) mAP50: this metric measures how accurate a detection model is, by assessing the overlap between the predicted and actual bounding boxes using the threshold IoU=0.5; b) mAP50-95: this metric is similar to the previous one. However, the average over an IoU is obtained using the threshold IoU=0.5 to 0.95.

$$mAP = \frac{1}{N}\sum_{n=1}^{N} Precision\ (IoU) \quad (4)$$

4. Methodology

This section delineates the methodology employed in the research endeavor, which was meticulously divided into several stages: image input, labeling, distribution in datasets, training, and evaluation. The methodology was designed to compare the

performance of different object detection models in scenarios with different levels of difficulty, using two complementary datasets.

4.1 Datasets

Two primary datasets were utilized in the present work: the HORD dataset (Hybrid Object Recognition Dataset) and the frog's dataset.

HORD dataset contains 1,730 images categorized into five major object types: birds, dogs, cats, plants, and fruits. Each object type is further subdivided into various subtypes, thereby offering a diverse range of classes for object recognition. For instance, the category of dogs includes the subtypes Boxer and Beagle; the category of cats includes Birman and Abyssinian; the category of birds includes Goose and Thrush; the category of plants includes Potato leaf, tomato, and bell pepper; and finally, the category of fruits includes Papaya, pineapple, and watermelon. HORD images are generally centered and present easily detectable objects, allowing the performance of the models to be evaluated in a controlled scenario.

The frog dataset contains 613 images in uncontrolled environments, which are used to evaluate the performance of the models in more challenging scenarios, including occlusion conditions and uncontrolled environments. This dataset consists of 513 frog images for training and 100 images for validation, all labeled with a single class named frogs. The images present challenges such as partially visible objects, illumination variations, and complex backgrounds, allowing the ability of the models to generalize in difficult situations to be evaluated.

4.2 Methodology steps

The methodology used is presented in Figure 2, which summarizes the steps followed during the training process. The five steps in the methodology are Image input, Image labeling, Image distribution, Training and Configurations models, and Model evaluation.

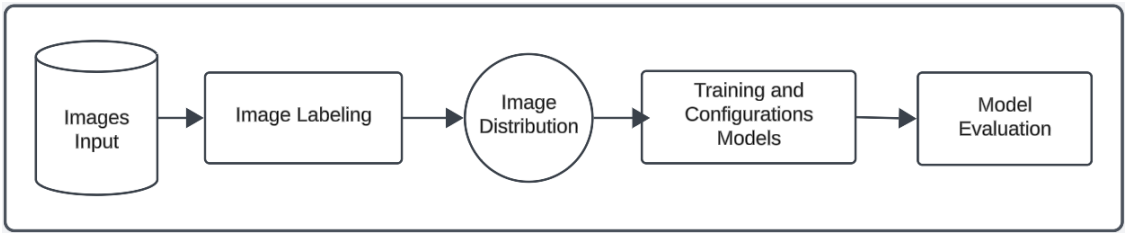

Fig. 2. Methodology for the training process

The steps are described in more detail below:

- Input of Images: The HORD dataset is composed of 1,730 images, which are distributed across five distinct classes with multiple subtypes. In contrast, the frog dataset contains 613 images, which are classified into a single class.
- Image labeling: The labels of both datasets were converted to the format compatible with the models used. This included transforming the coordinates of the bounding boxes and object classes into a format interpretable by models such as YOLO and RT-DETR.
- Image distribution: The labeled HORD images were divided into two sets: training (84.4%) and validation (15.6%). For the frog set, the distribution was training (83.7%) and validation (16.3%). This distribution ensures that the models can be trained correctly and that their performance can be evaluated fairly.
- Training and Configurations models: YOLOv8, YOLOv9, YOLOv10, YOLOv11, and RT-DETR models were trained under the same conditions to ensure a fair comparison. The configuration parameters were as follows (Table 1):

Table 1. Training parameters for the models evaluated

YOLO versions	RT-DETR
Epochs: 100	Epochs: 100
Batch size: 4	Batch size: 4
Images size: 640 x 640 pixels	Images size: 640 x 640 pixels
Initial learning rate: 0.002	Initial learning rate: 0.0005
Final learning rate:0.01	Final learning rate:0.01
Momentum: 0.9	Momentum: 0.937
Optimizer: AdamW	Optimizer: AdamW

- Model evaluation: Models were evaluated using standard metrics, such as accuracy, recall, mean average precision 50 (mAP50), mean average precision 50-95 mAP50-95, and training speed, on both datasets. This allowed analyzing the performance of the models in controlled (HORD) and challenging (Frog images) scenarios.
- Computational infrastructure: The experiments were conducted on a machine equipped with an Nvidia RTX 4050 GPU, which provided adequate performance for the expected training times.

The combined use of HORD and the frog image datasets allowed us to evaluate not only the precision of the models under ideal conditions but also their capability in more complex scenarios. This hybrid approach provides a comprehensive view of the models' performance, highlighting their strengths and limitations.

5. Results

In this section, the results obtained from the training and evaluation of the YOLOv8 to YOLOv11 and RT-DETR object detection models using the HORD dataset and the dataset of frog images in uncontrolled environments are presented and analyzed. The analysis focuses on key metrics such as mAP50, mAP50-95, Precision, Recall, Latency, and Training time, to evaluate the performance of the models in controlled and complex scenarios. In addition, a visual analysis of the predictions made by the models is included.

5.1 Results with HORD dataset

The HORD dataset, consisting of centered images and well-defined objects, allowed the performance of the models to be evaluated in a controlled environment. Two comparison tables (Tables 2 and 3) are presented below: one for the 'l' versions of the YOLO and RT-DETR models, and one for the 'x' versions.

In Table 2, YOLOv8l stands out as the best model in the HORD dataset, achieving the best results in key metrics such as mAP50 (0.918), Precision (0.91), and mAP50-95 (0.703), obtaining the three best results (marked with the symbol * in Table 2) and the second place in result (in training time, marked with the symbol °). In addition, the model maintains an efficient training time (2.43 hours) and a competitive latency (14.7 ms).

Table 1. Results of metrics obtained on the HORD dataset with large models.

Models	*Epochs*	*Precision*	*Recall*	*mAP50*	*mAP50-95*	*Latency (ms)*	*Training time (hr)*
YOLOv8l	100	***0.91**	**°0.827**	***0.918**	***0.703**	14.7	**°2.43**
YOLOv9c	100	0.86	***0.854**	**°0.911**	**°0.684**	**°12.4**	2.76
YOLOv10l	100	0.801	0.689	0.811	0.611	13.9	3.7
YOLOv11l	100	**°0.883**	0.822	0.889	0.671	***12**	***2.14**
RT-DETRl	100	0.801	0.742	0.783	0.55	14.9	3.67

Note: *A symbol * is added for the best results for each metric and the symbol ° for second places.*

The best second model, YOLOv9c, particularly excelled in Recall (0.854), obtaining the best result in this metric. In the mAP50, mAP50-95, and latency metrics, it obtained the best second results (marked with the symbol °) as shown in Table 2, consolidating itself as a solid option for controlled scenarios.

On the other hand, YOLOv11l, which recorded the shortest training time (2.14 hours) and the lowest latency (12 ms), is positioned as an efficient alternative, although its overall performance was slightly lower than that of YOLOv8l and YOLOv9l. In contrast, RT-DETRl, with a latency of 14.9 ms and a training time of 3.67 hours, lagged behind in its metrics, Precision (0.801), mAP50 (0.783), and mAP50-95 (0.55), suggesting that its design may be less effective compared to YOLO in controlled scenarios such as HORD. Table 3, presents the results for the 'x' versions of YOLO, compared to RT-DETR. These models are designed to achieve maximum performance with higher computational requirements.

Table 3. Results for metrics obtained on the HORD dataset with extra-large (x) models.

Models	*Epochs*	*Precision*	*Recall*	*mAP50*	*mAP50-95*	*Latency (ms)*	*Training time (hr)*
YOLOv8x	100	***0.88**	***0.868**	***0.924**	***0.693**	23	5.77
YOLOv9e	100	0.767	0.789	**°0.827**	**°0.623**	26.3	6.1
YOLOv10x	100	0.798	0.699	0.794	0.603	***19.3**	°4.421
YOLOv11x	100	0.766	**°0.804**	**°0.827**	0.609	**°20.5**	***3.8**
RT-DETRx	100	**°0.827**	0.728	0.753	0.540	23.1	7.826

Note: A symbol * is added for the best results for each metric and the symbol ° for second places.

In Table 3, which compares the extra-large ('x') versions, YOLOv8x leads again, achieving the best results in all major metrics, with a mAP50 (0.924), mAP50-95 (0.693), Precision (0.88), and a Latency (23 ms), all marked with the symbol *. Its Training time (5.77 hr) is efficient relative to its superior performance. In this case, RT-DETRx showed remarkable Precision (0.827), indicated with the symbol °, while the Recall (0.728), mAP50 (0.753), and mAP50-95 (0.540) were inferior compared to the YOLO models. Also, YOLOv11x, which obtained the shortest training time (3.8 hr) and a competitive Latency (20.5 ms), stands out for its efficiency, although its performance in mAP50 (0.827) and mAP50-95 (0.609) puts it behind YOLOv8x.

5.2 Results with frog's dataset

The frog's dataset, characterized by images with occlusion, complex backgrounds, and variations in illumination, presented a greater challenge for the models evaluated. To analyze their performance, two comparisons were made: one between the 'l' versions of YOLO and RT-DETR, and one between the 'x' versions. The metrics included are Precision, Recall, mAP50, mAP50-95, Latency, and Training time. Table 4 shows the results of the models trained on the frog dataset.

As can be observed in Table 4, RT-DETRl excelled in Precision (0.74), outperforming all YOLO models evaluated. This indicates that its transformer architecture handles image complexity better, albeit at the cost of a longer training time (1.476 hr) and a higher Latency (15.1 ms). Among the YOLO models, YOLOv8l achieved the best Recall performance (0.521), mAP50 (0.498), and mAP50-95 (0.241). YOLOv11l showed an acceptable balance between accuracy and speed, having the best Latency and Training time (marked with an *), with intermediate metrics compared to RT-DETR and YOLOv8l. YOLOv9c and

YOLOv10l had lower metrics overall, which may indicate that their architecture is not as optimized to handle complex images as the frog dataset.

Table 4. Comparison between the YOLO "l" and RT-DETRl models with images from the Frog dataset

Models	*Epochs*	*Precision*	*Recall*	*mAP50*	*mAP50-95*	*Latency (ms)*	*Training time (hr)*
YOLOv8l	100	0.633	***0.485**	***0.498**	***0.241**	13.8	1.03
YOLOv9c	100	0.472	0.427	0.403	0.182	**°12.9**	**°0.94**
YOLOv10l	100	0.493	0.418	0.384	0.184	13.2	1.025
YOLOv11l	100	**°0.661**	0.405	0.478	**°0.224**	***11**	***0.855**
RT-DETRl	100	***0.74**	**°0.456**	**°0.48**	0.223	15.1	1.476

Note: A symbol * is added for the best results for each metric and a symbol ° for second places.

Table 5 shows the results of the 'x' models trained on the frog dataset. In Table 5, YOLOv8x leads in Precision (0.638), but its low Recall (0.305) suggests that it has difficulty detecting certain objects in challenging conditions. RT-DETRx, with a Precision (0.637) and a Recall (0.432), showed balanced performance in this complex environment, albeit with a longer Training time (2.781 hours).

On the other hand, YOLOv10x excelled in mAP50 (0.49), mAP50-95 (0.22), and Latency (19.1 ms), consolidating its position as the most balanced model in this category. YOLOv11x, while competitive in overall metrics, was slightly behind YOLOv10x in mAP50-95 (0.189) and Training time (1.491 hr). Overall, the 'x' models presented a mixed performance, with YOLOv10x and RT-DETRx showing a better balance between key metrics.

Table 2. Comparison between the YOLO "x" and RT-DETRx models with images from the Frog dataset

Models	*Epochs*	*Precision*	*Recall*	*mAP50*	*mAP50-95*	*Latency (ms)*	*Training Time (hr)*
YOLOv8x	100	***0.638**	0.305	0.372	0.164	23.7	1.517
YOLOv9e	100	0.627	0.288	0.314	0.157	24.7	2.204
YOLOv10x	100	0.582	***0.465**	***0.49**	***0.22**	***19.1**	***1.475**
YOLOv11x	100	0.573	0.412	0.414	0.189	**°20.2**	**°1.491**
RT-DETRx	100	**°0.637**	**°0.432**	**°0.441**	**°0.202**	23.7	2.781

Note: A symbol * is added for the best results for each metric and a symbol ° for second places.

In the subsequent analysis, the performance of the models with the HORD dataset is presented. The HORD dataset is characterized by centered images and well-defined objects. This controlled environment allows us to observe how the models achieve their optimal performance under favorable conditions. For this analysis, graphical visualizations are included to illustrate the relationship between key metrics such as Precision, Recall, mAP50, mAP50-95, Latency, and Training time.

Figure 3, presents a dispersion analysis relating latency to the main metrics: Precision, Recall, mAP50, and mAP50-95. This analysis provides a clear view of how Latency affects the quality of the model and allows for identifying the best trade-offs between accuracy and response time:

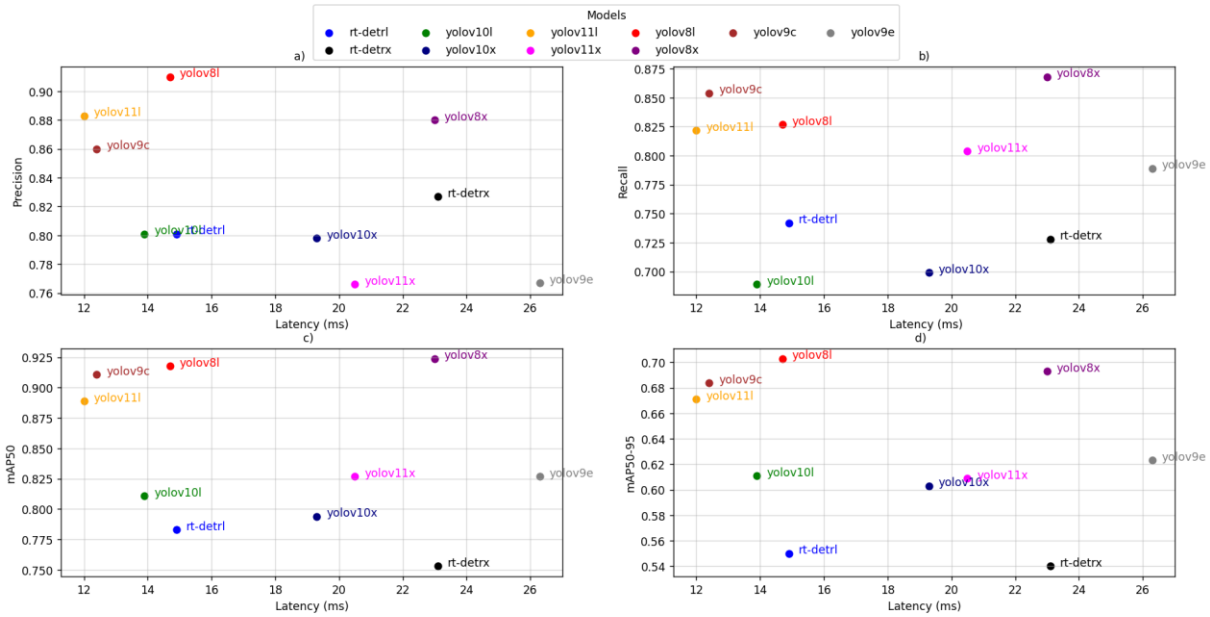

Fig. 3. Scatter plots of Latency vs Precision(a), Recall(b), mAP50(c), and mAP50-95(d) for the models with the HORD dataset.

a) Latency vs Accuracy: In Figure 3a, YOLOv8l leads with the highest Precision (0.91) at a Latency (14.7 ms), while YOLOv11l offers a balance between Precision (0.883) and the lowest Latency (12 ms).
b) Latency vs Recall: In Figure 3b, YOLOv8x stands out with the highest Recall (0.868) and Latency (23 ms), however, YOLOv9c obtained the smaller Latency (12.4 ms), and Recall (0.54), consolidating itself as the best balanced.
c) Latency vs mAP50: In Figure 3c, YOLOv8x stands out again with the best value of mAP50 (0.924), however in terms of Latency it is far above its competitors, considering YOLOv8l the best option, having mAP50 (0.918) and a moderate Latency (14.7 ms), followed by YOLOv9l with a mAP50 (0.911) and Latency (12.4 ms).
d) Latency vs mAP50-95: In Figure 3d, YOLOv8l also leads in mAP50-95 (0.703), consolidating itself as the model with the best balance.

Figure 4, shows all the metrics Precision, Recall, mAP50, and mAP50-95 for each version of YOLO and RT-DETR in function of Latency. Figure 4 allows us to observe clear performance patterns and shows that YOLOv8l maintains an excellent balance between metrics and latency, while RT-DETRl shows average accuracy compared to the other YOLO models, but with a higher computational cost.

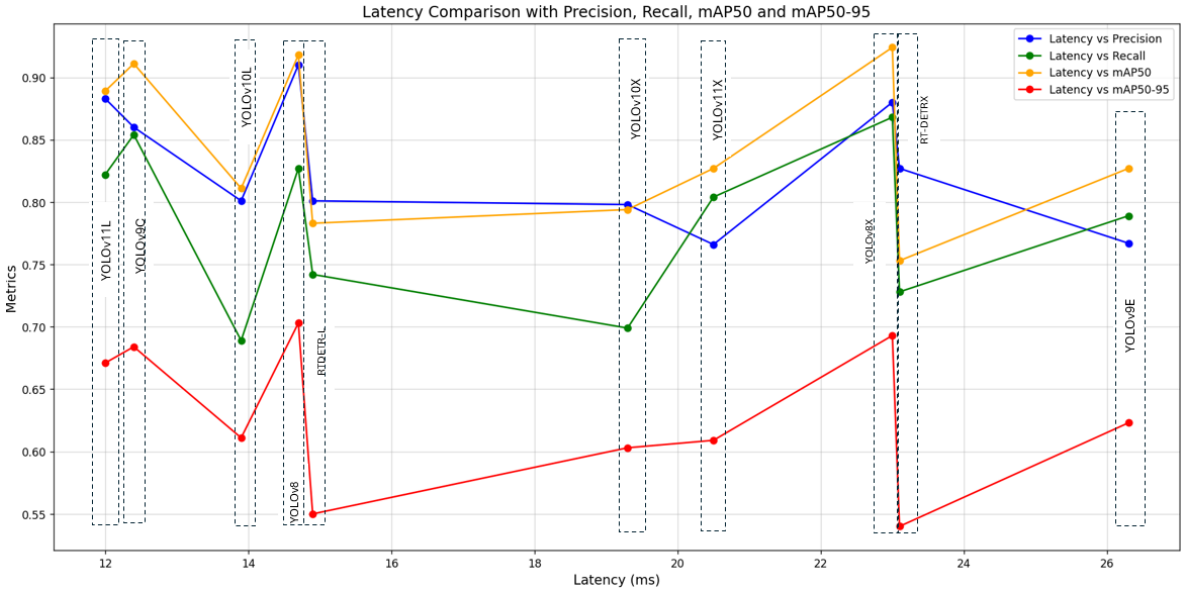

Fig. 4. Line graphs of Latency and key metrics with the HORD dataset

Figure 5, presents a bar chart detailing the training time duration for each model. The results show that YOLOv11l is the most efficient model, with a Training time (2.14 hr), followed closely by YOLOv8l with 2.43 hours. In contrast, RT-DETRl records the highest Training time (3.67 hr), reflecting the additional computational costs of its transformer architecture.

The analysis highlights that YOLOv8l is the most balanced and effective model using the HORD dataset, leading in key metrics such as mAP50, mAP50-95, and Precision, while maintaining competitive Training time and Latency. YOLOv11l, meanwhile, is positioned as the most efficient model in training time, with performance close to that of YOLOv8l. Although RT-DETRl achieves good results in the analyzed metrics, its longer Training time and Latency limit its applicability in scenarios that require high computational efficiency. Overall, YOLO models excel in this controlled environment, establishing themselves as the preferred choice for real-time applications with favorable conditions.

In the next analysis, we present the performance of the models evaluated on the frog's dataset, which is characterized by its complexity due to occlusion, uncontrolled backgrounds, and illumination variations. For this analysis, graphical visualizations are presented to identify patterns of performance with key metrics such as Precision, Recall, mAP50, and mAP50-95, as well as Training times.

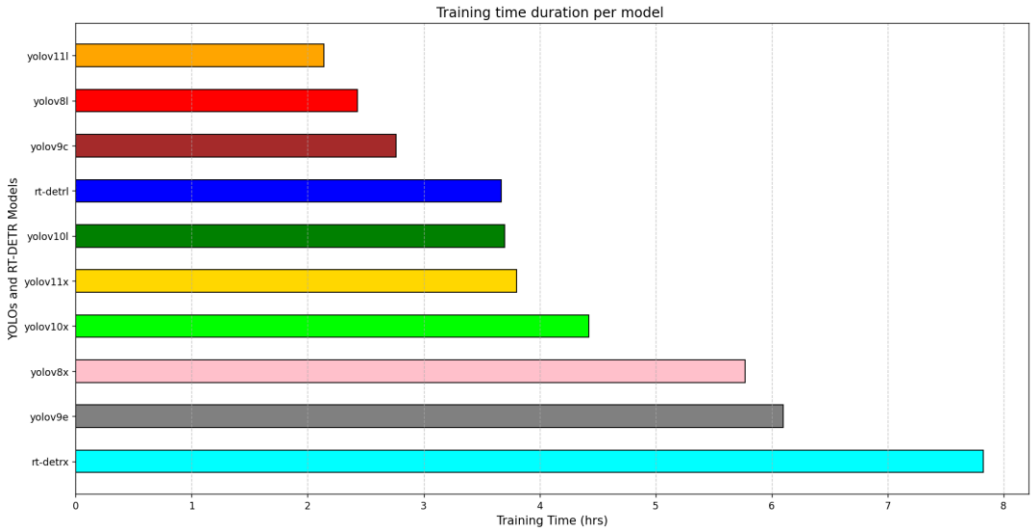

Fig. 5. The bar chart of the Training Time (hrs) for each model with the HORD dataset.

Figure 6, shows a dispersion analysis of the relationships between Latency and the key metrics: Precision, Recall, mAP50, and mAP50-95. This analysis allows us to visualize the inherent trade-offs between response time and model quality:

a) Latency vs Precision: In Figure 6a, RT-DETRl stands out with the highest Precision (0.74) at a high Latency cost (15.1 ms). On the other hand, YOLOv11l offers an acceptable balance, with an accuracy of 0.661 and a latency of 11 ms.
b) Latency vs Recall: In Figure 6b, YOLOv8l has the highest Recall (0.521) with a moderate Latency (13.8 ms), followed by YOLOv10x (0.465) with Latency (19.1 ms) and RT-DETRl (0.458 ms) with Latency (15.6).
c) Latency vs mAP50: In Figure 6c, YOLOv8l leads in mAP50 (0.498) with a Latency (13.8 ms), followed by YOLOv10x with mAP50 (0.49) and Latency (19.2 ms), and RT-DETRl with mAP50 (0.45) and Larency (15.3 ms).
d) Latency vs mAP50-95: In Figure 6d, YOLOv8l stands out with a mAP50-95 (0.241), being the most balanced model between quality and latency. The models YOLOv11l and RT-DETRl follow with mAP50-95 of 0.222 and 0.221 respectively.

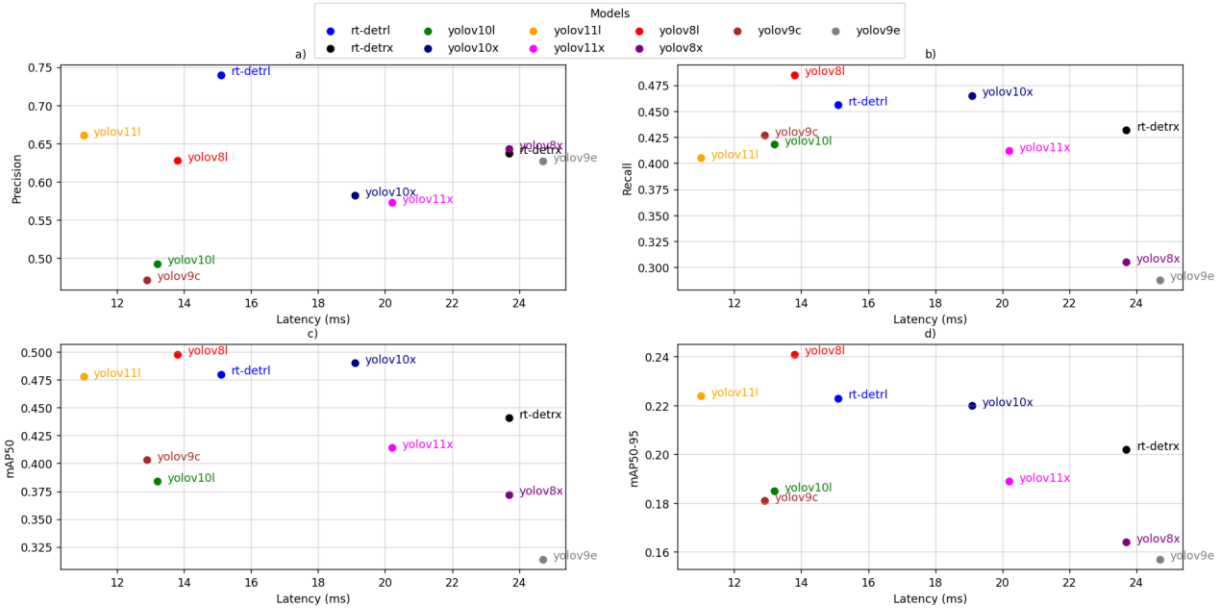

Fig. 6. Scatter plots of Latency vs Precision(a), Recall(b), mAP50(c), and mAP50-95(d) for the models with the frog's dataset.

In Figure 7, the line graphs show the performance of key metrics (Precision, Recall, mAP50, and mAP50-95) for each model. These graphs allow comparison of the overall performance of the models and highlight individual strengths and weaknesses.

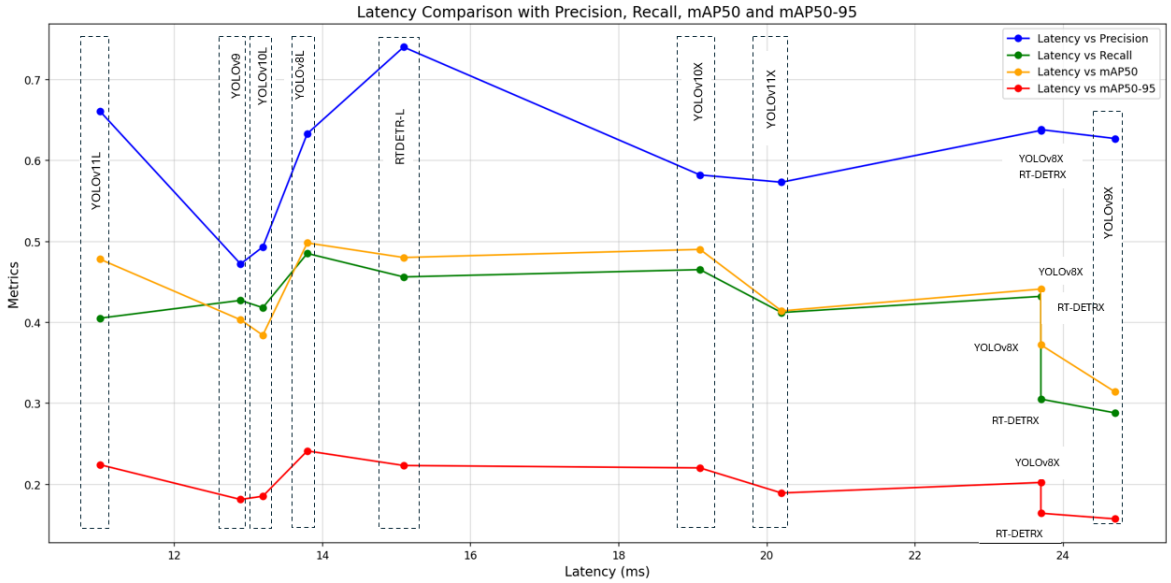

Fig. 7. Line graphs of Latency and key metrics with the frog's dataset

In Figure 8, the Bar chart shows the length of training time for each model. This visualization is useful for assessing the computational efficiency of the models and highlights how YOLOv11-L is the fastest training model, while RT-DETR-X has the highest training time.

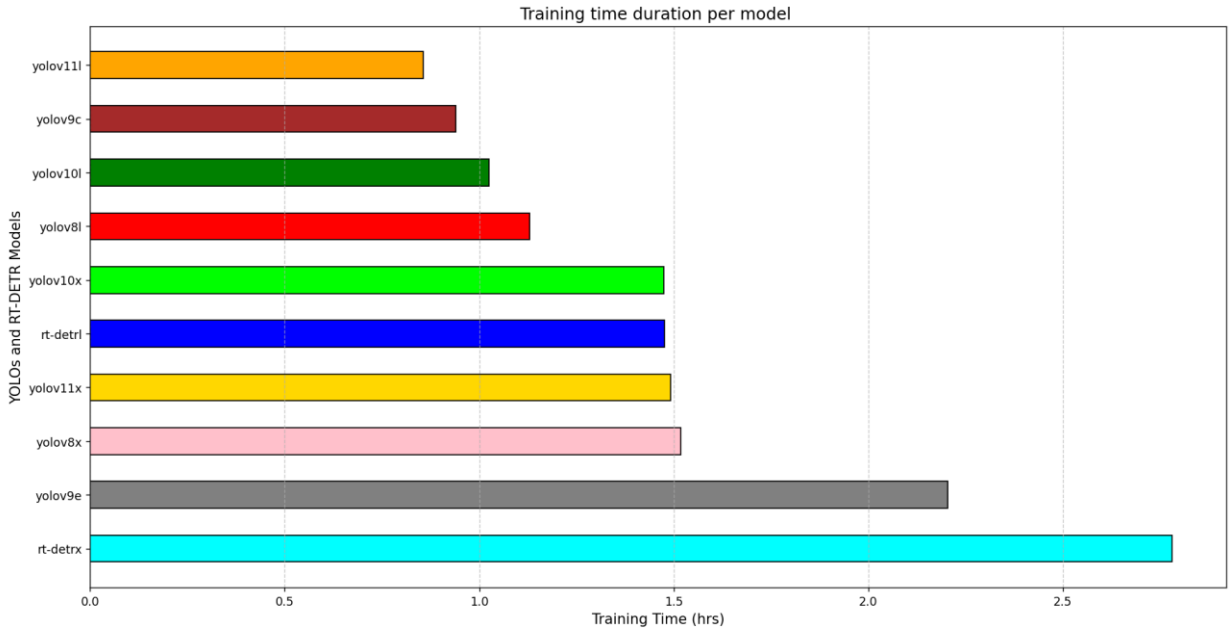

Fig. 8. The bar chart of the Training Time (hrs) for each model with the frog's dataset.

6. Conclusions

In this paper, an analysis of YOLO and RT-DETR transformer models is presented. An evaluation was executed using the common metrics of object recognition, including Precision, Recall, mAP50, and Map50-90. The experimental datasets comprised the HORD and the frog datasets. The HORD dataset contains images with centered objects that are easily identifiable, while the frog dataset contains objects in uncontrolled environments. The later poses a significant challenge for the YOLO and RT-DETR models evaluated in this work. The evaluation results indicated that eight YOLO version has superior on the HORD dataset, while the RT-DETR model exhibited superior performance on the frog dataset.

Finally, the experimentation conducted enabled the assessment of the performance of the models evaluated with different datasets. Notwithstanding, in terms of training time, YOLO demonstrated superior performance. Conversely, the RT-DETR model exhibited superior performance in datasets considered difficult. The development of this work will make it possible to decide the model with the best characteristics that can be derived from possible applications.

7 Acknowledgments

Alan Jair González Hernández gratefully acknowledges the SECIHTI (Secretariat of Science, Humanities, Technology and Innovation) for the doctoral studies grant. The authors also express their sincere appreciation for the support provided by TecNM (National Technological Institute of Mexico), the Madero City Technological Institute, and the National Laboratory of Information Technologies (LaNTI) for their collaboration, infrastructure, and resources that made this research possible.

References

Ali, Y. A., Gupta, D. D., Gupta, P., Jain, M., Saini, S., Bhinwal, A., Agarwal, P., Pandey, S., & Rawal, K. (2024). Enhancing lung cancer screening efficacy: A comparative analysis of YOLOv5 in an AI-based pipeline. *Amity Institute of Biotechnology, Amity University Uttar Pradesh*.

Badgujar, C., Poulose, A., & Gan, H. (2024). Agricultural object detection with You Only Look Once (YOLO) algorithm: A bibliometric and systematic literature review. *arXiv*, arXiv:2401.10379. https://doi.org/10.48550/arXiv.2401.10379

Bakirci, M. (2024). Real-time vehicle detection using YOLOv8-Nano for intelligent transportation systems. *Traitement du Signal*, *41*(4), 407–417. https://doi.org/10.18280/ts.410407

Bansod, K., Wan, Y., & Rai, Y. (2023). Liquid leak detection using thermal images. *arXiv*, arXiv:2312.10980. https://arxiv.org/abs/2312.10980

Bertinetto, L., Valmadre, J., Henriques, J. F., Vedaldi, A., & Torr, P. H. S. (2016). Fully-convolutional Siamese networks for object tracking. *arXiv*. https://doi.org/10.48550/arXiv.1606.09549

Buslaev, A., Iglovikov, V. I., Khvedchenya, E., Parinov, A., Druzhinin, M., & Kalinin, A. A. (2020). Albumentations: Fast and flexible image augmentations. *Information*, *11*(2), 125. https://doi.org/10.3390/info11020125

Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., & Zagoruyko, S. (2020). End-to-end object detection with transformers. *arXiv*. https://doi.org/10.48550/arXiv.2005.12872

Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J., & Zisserman, A. (2010). The Pascal visual object classes (VOC) challenge. *International Journal of Computer Vision*, *88*(2), 303–338. https://doi.org/10.1007/s11263-009-0275-4

Jocher, G. (2020). *Ultralytics YOLOv5* [Software]. GitHub. https://github.com/ultralytics/yolov5

Jocher, G., Chaurasia, A., & Qiu, J. (2023). *Ultralytics YOLOv8* (Version 8.0.0) [Software]. Ultralytics. https://github.com/autogyro/yolo-V8

Jocher, G., & Qiu, J. (2024). *Ultralytics YOLOv11* (Version 11.0.0) [Software]. Ultralytics. https://github.com/ultralytics/ultralytics

Kunekar, P., Narule, Y., Mahajan, R., Mandlapure, S., Mehendale, E., & Meshram, Y. (2024). Traffic management system using YOLO algorithm. *Engineering Proceedings*. https://doi.org/10.3390/engproc2023059210

Levine, S., Pastor, P., Krizhevsky, A., & Quillen, D. (2016). Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *The International Journal of Robotics Research*, *37*(1). https://doi.org/10.48550/arXiv.1603.02199

Li, C., et al. (2022). YOLOv6: A single-stage object detection framework for industrial applications. *arXiv*. https://doi.org/10.48550/arXiv.2209.02976

Li, C., Li, L., Geng, Y., Jiang, H., Cheng, M., Zhang, B., Ke, Z., Xu, X., & Chu, X. (2023). YOLOv6 v3.0: A full-scale reloading. *arXiv*. https://arxiv.org/abs/2301.05586

Lin, T. Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., & Zitnick, C. L. (2014). Microsoft COCO: Common objects in context. In *European Conference on Computer Vision (ECCV)*. https://doi.org/10.1007/978-3-319-10602-1_48

Malik, M. E., & Mahmud, M. S. (2024). Enhanced weed detection using YOLOv9 on open-source datasets for precise weed management. In *ASABE Annual International Meeting*. https://doi.org/10.13031/aim.202400495

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 779–788).

Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems* (Vol. 28).

Sadik, M. N., Hossain, T., & Sayeed, F. (2023). Real-time detection and analysis of vehicles and pedestrians using deep learning. *IEEE Transactions on Intelligent Transportation Systems*. https://doi.org/10.48550/arXiv.2404.08081

Schoder, S. (2023). First qualitative observations on deep learning vision model YOLO and DETR for automated driving in Austria. In *Proceedings of the International Conference on Autonomous Systems*. https://doi.org/10.48550/arXiv.2312.12314

Tang, S., & Yan, W. (2024). Utilizing RT-DETR model for fruit calorie estimation from digital images. *Information*, *15*(8), 469. https://doi.org/10.3390/info15080469

Wang, A., Chen, H., Liu, L., et al. (2024). YOLOv10: Real-time end-to-end object detection. *arXiv*, arXiv:2405.14458. https://arxiv.org/abs/2405.14458

Wang, C.-Y., Bochkovskiy, A., & Liao, H. Y. M. (2022). YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. *arXiv*. https://arxiv.org/abs/2207.02696

Wang, C.-Y., Yeh, I.-H., & Liao, H.-Y. M. (2024). YOLOv9: Learning what you want to learn using programmable gradient information. *arXiv*, arXiv:2402.13616. https://doi.org/10.48550/arXiv.2402.13616

Yang, R., & Yu, Y. (2021). Artificial convolutional neural network in object detection and semantic segmentation for medical imaging analysis. *Frontiers in Oncology*, *11*, 638182.

Zhao, Y., Lv, W., Xu, S., Wang, G., Wei, J., Cui, C., Du, Y., Dang, Q., & Liu, Y. (2023). DETRs beat YOLOs on real-time object detection. *arXiv*. https://arxiv.org/abs/2304.08069

Zhao, Z. Q., Zheng, P., Xu, S. T., & Wu, X. (2019). Object detection with deep learning: A review. *IEEE Transactions on Neural Networks and Learning Systems*, *30*(11), 3212–3232.