AI

MDPI

Article

Empirical Evaluation and Analysis of YOLO Models in Smart Transportation

Lan Anh Nguyen, Manh Dat Tran and Yongseok Son *

Department of Computer Science and Engineering, Chung-Ang University, Seoul 06974, Republic of Korea; loglamo@cau.ac.kr (L.A.N.); tmdat01081997@cau.ac.kr (M.D.T.)
* Correspondence: sysganda@cau.ac.kr

Abstract: You Only Look Once (YOLO) and its variants have emerged as the most popular real-time object detection algorithms. They have been widely used in real-time smart transportation applications due to their low-latency detection and high accuracy. However, because of the diverse characteristics of YOLO models, selecting the optimal model according to various applications and environments in smart transportation is critical. In this article, we conduct an empirical evaluation and analysis study for most YOLO versions to assess their performance in smart transportation. To achieve this, we first measure the average precision of YOLO models across multiple datasets (i.e., COCO and PASCAL VOC). Second, we analyze the performance of YOLO models on multiple object categories within each dataset, focusing on classes relevant to road transportation such as those commonly used in smart transportation applications. Third, multiple Intersection over Union (IoU) thresholds are considered in our performance measurement and analysis. By examining the performance of various YOLO models across datasets, IoU thresholds, and object classes, we make six observations on these three aspects while aiming to identify optimal models for road transportation scenarios. It was found that YOLOv5 and YOLOv8 outperform other models in all three aspects due to their novel performance features. For instance, YOLOv5 achieves stable performance thanks to its cross-stage partial darknet-53 (CSPDarknet53) backbone, auto-anchor mechanism, and efficient loss functions including IoU loss, complete IoU loss, focal loss, gradient harmonizing mechanism loss. Similarly, YOLOv8 outperforms others with its upgraded CSPDarknet53 backbone, anchor-free mechanism, and efficient loss functions like complete IoU loss and distribution focal loss.

Keywords: convolution neural network (CNN); real-time object detection; road transportation; smart transportation; you only look once (YOLO)

Citation: Nguyen, L.A.; Tran, M.D.; Son, Y. Empirical Evaluation and Analysis of YOLO Models in Smart Transportation. *AI* **2024**, *5*, 2518–2537. https://doi.org/10.3390/ai5040122

Academic Editor: Arslan Munir

Received: 25 September 2024
Revised: 2 November 2024
Accepted: 19 November 2024
Published: 26 November 2024

1. Introduction

Smart transportation has recently emerged as the optimal solution to address the mobility needs of people [1–3]. By leveraging modern technology, such as wireless communication, cloud–fog–edge computing, and object detection, smart transportation aims to simplify processes, enhance safety, reduce costs, and improve overall efficiency in transportation systems [4–6]. For example, smart transportation can offer optimized route suggestions, easy parking reservations, accident prevention, traffic control, and support for autonomous driving [4–6]. Among modern technology building smart transportation, object detection techniques play the most important role, as identifying and tracking specific objects is crucial for various purposes, such as traffic monitoring [7], collision avoidance [8], pedestrian safety [7], parking assistance [7], infrastructure monitoring [7], and public transport security [9].

Specifically, several object detection techniques exist, such as pioneer detection (e.g., histogram of oriented gradients (HOG) [10], deformable part model (DPM) detector [11], etc.), two-stage detection (e.g., region-based convolutional neural network (R-CNN), spatial pyramid pooling network (SPP-net) [12], Fast R-CNN [13], RepPoints [14], etc.),

transformer-based detection (e.g., detection transformer (DETR) [15], segmentation transformer (SETR) [16], transformer-based set prediction (TSP) [17], etc.), and one-stage detection (e.g., region proposal network (RPN) [18], single-shot detector (SSD) [19], You Only Look Once (YOLO) [20], RetinaNet [21], CornerNet [22], etc.). In smart transportation applications, real-time processing is essential to guarantee the safety, accuracy, and efficiency of operating vehicles. For example, in a self-driving car within a smart transportation system, a real-time object detector is required to quickly and accurately detect speed limit signs, pavement, pedestrian zone signs, and other vehicles on the street, allowing the car to take the appropriate subsequent actions in real time. Therefore, one-stage detection is the most suitable for real-time smart transportation applications due to its fast detection speed and high accuracy [23,24]. Furthermore, among recently developed one-stage detection models, YOLO is considered the leading representative.

First launched in 2015, the YOLO [20] real-time object detection model has been developed through the years with multiple versions (i.e., YOLOv2 [25], YOLOv3 [26], YOLOv4 [27], YOLOv5 [28], YOLOv6 [29], YOLOv7 [30], and YOLOv8 [31]), as depicted in Figure 1. Specifically, YOLO takes a different approach compared with traditional object detection models. Instead of dividing the image into regions and applying classification and localization separately, YOLO performs both tasks in a single pass through its network [24]. It predicts bounding boxes and class probabilities directly on a grid-based output. By doing so, YOLO and its variants have significantly contributed to object detection. Moreover, YOLO models aim to improve themselves, becoming more efficient, accurate, and scalable for real-world applications [32–34]. However, YOLO models differ in their network backbone, neck, head, loss functions, and other features, making it crucial to select the optimal models for smart transportation applications by evaluating and analyzing their performance.

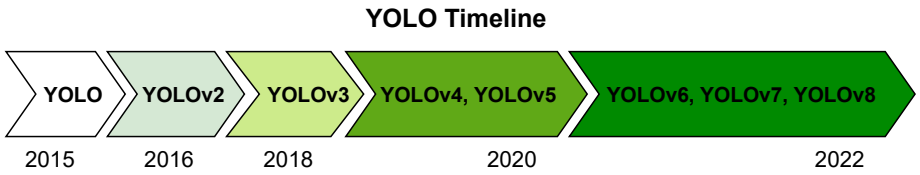


Figure 1. You Only Look Once (YOLO) development timeline.

Previous studies have investigated object detection models. Nazir et al. [35] introduced an overview of YOLO variants, including YOLOv2, YOLOv3, YOLOv4, YOLOv5, YOLOv6, and YOLOv7. In addition, Sozzi et al. [36] introduced a comparative evaluation of the accuracy and speed of various versions of YOLO object detection algorithms, such as YOLOv3, YOLOv3-tiny, YOLOv4, YOLOv4-tiny, YOLOv5x, and YOLOv5s, for grape-yield estimation. Our study is inspired by these works regarding the performance comparison and architectural analysis of various YOLO models. In contrast, this article specifically focuses on the performance of YOLO models on transportation data that affects the autonomous driving situation in smart transportation. Notably, we evaluate YOLO performance across various datasets, object classes, and Intersection over Union (IoU) thresholds.

In this article, we conduct an empirical evaluation and analysis of multiple YOLO models, including YOLO version 3 to 8 to observe their performance and determine the optimal models for smart transportation, with a focus on road transportation. To delve into empirical evaluation and analysis of YOLO models in smart transportation, we first analyze and evaluate the performance of multiple YOLO models in terms of accuracy (i.e., average precision (AP)) according to road transportation data using multiple datasets (i.e., COCO and PASCAL VOC datasets). Second, we evaluate the performance of various YOLO models on multiple object classes within the datasets, specifically focusing on those relevant to road transportation. Third, multiple IoU thresholds are also considered in the performance measurement and analysis. By doing so, we make six observations related to the following three aspects:

Datasets: The observations on datasets are obtained in the evaluation and analysis. First, the performance of YOLO models trained and evaluated on various datasets can be disparate. In particular, YOLO models perform better on PASCAL VOC than on COCO. Second, YOLOv5 and YOLOv8 are dominant models on both datasets. The reason for the performance disparity of YOLO models on the two datasets can be attributed to various factors, including differences in object classes, dataset sizes, object characteristics, architectural design, optimization techniques, and dataset biases. These factors affect model performance on various datasets, leading to variations in AP scores.

IoU thresholds: The observation on IoU thresholds indicates that as the IoU threshold increases, the mean AP (mAP) of a YOLO model decreases. YOLOv5 and YOLOv8 perform better than other YOLO models across various IoU thresholds. The first observation is related to the essence of the IoU threshold, where a higher threshold demands a higher standard for accurate predictions. For the second observation, YOLOv5 and YOLOv8 outperform other models, because of their bag of freebies (BoF) and bag of specials (BoS) techniques, such as backbone, neck, head architectures, loss functions, anchor-based mechanisms, modules for enhancing the receptive field, data augmentation strategies, feature integration, and activation functions.

Class of objects: Based on the observation of object classes, the performance of YOLO models can fluctuate or stabilize depending on the quality of the datasets and the characteristics of the objects within them. Similar to the above observations, YOLOv5 and YOLOv8 outperform most object classes in the two datasets. For the first observation related to the class of objects, we observe that differences in AP values of YOLO models across object classes come from certain factors, such as object class imbalance and object class-specific characteristics. For example, objects with smaller sizes or complex shapes can pose challenges for detection, resulting in lower AP values for those object classes. For the second observation, YOLOv5 and YOLOv8 still maintain their lead, because of their BoF and BoS techniques as analyzed in the observations related to IoU thresholds.

2. Related Works

Several studies have focused on evaluating the performance of YOLO variants in object detection. For example, Liule et al. [37] experiment on three versions: YOLOv3, YOLOv4, and YOLOv5. They evaluate and analyze the performance of the three YOLO variants regarding accuracy and inference speed by using training and predicting on the PASCAL VOC dataset. Nazir et al. [35] provide a review of the YOLO variants (YOLO version 1 to 5), focusing on their performance. They also investigate the accuracy and inference speed of various YOLO models on various datasets (i.e., COCO, KITTI, and SUN-RGBD datasets). Sozzi et al. [36] introduce a comparative evaluation of the accuracy and inference speed of various versions of YOLO object detection algorithms, such as YOLOv3, YOLOv3-tiny, YOLOv4, YOLOv4-tiny, YOLOv5x, and YOLOv5s, for grape-yield estimation. Gunduz et al. [38] carried out experiments on three versions of YOLO models such as YOLOv3, YOLOv4, and YOLOv5 for object detection. Our study is in line with their studies [35–38] in evaluating and analyzing the performance of YOLO variants. In contrast, we focus on smart transportation while evaluating and analyzing more YOLO variants, including the latest one (YOLOv8). Further, we consider their performance on various datasets (i.e., COCO and Pascal VOC) and IoU thresholds.

Feng et al. [39] analyze YOLO performance on edge devices, experimenting with YOLOv3, YOLOv3-tiny, YOLOv4, and YOLOv4-tiny. This study evaluates both accuracy (i.e., mean confidence) and device efficiency, including metrics like frames per second (fps), CPU and memory usage, and power consumption. Magalhaes et al. [40] focus on fruit detection in smart farms while comparing the YOLOv4-tiny model with multiple SSD models. Our study aligns with these works in evaluating the performance of YOLO models in specific environments. In contrast, we focus on smart transportation instead of edge devices and smart farms.

Several studies have focused on developing YOLO model variants for smart transportation in IoT. For example, Tang et al. [41] introduce the YOLO-fusion framework, which integrates the YOLO architecture with fusion techniques to improve detection accuracy and efficiency in smart transportation applications. Wu et al. [42] present a system aimed at accurately detecting, classifying, and counting vehicles in traffic environments using the YOLO object detection framework. Sindhwani et al. [43] provide a comparative analysis of the YOLO vs SSD technique for the detection of potholes. Our study aligns with these studies [41–43] in evaluating the performance of YOLO models in transportation. However, our study focuses on experimenting with diverse YOLO models on smart transportation in terms of performance on different datasets and IoU thresholds.

3. Background and Motivation

Our study presents an empirical evaluation and analysis of various YOLO models for smart transportation with a focus on road transportation. To accomplish this, we first give an overview of object detection systems in road transportation in Section 3.1. We then introduce the fundamentals of YOLO models as object detection algorithms used in such systems in Section 3.2. Finally, we provide background information on the evaluation metrics and datasets utilized in our study in Sections 3.3 and 3.4, respectively.

3.1. Object Detection in Road Transportation

To provide an overview of object detection systems in road transportation, we first introduce object-detection-based applications commonly utilized in this field in Section 3.1.1. Furthermore, an explanation of the one-stage object detection system is provided in Section 3.1.2.

3.1.1. Object-Detection-Based Applications

Object detection has become an essential element in contemporary road transportation systems, improving safety, efficiency, and decision-making processes. Four key applications of object detection in road transportation are as follows: (1) intelligent traffic management [4], (2) advanced driver assistance systems, (3) autonomous vehicles [8], and (4) traffic-related object detection and recognition [7].

Intelligent traffic management: Object detection is applied to monitor and analyze traffic conditions in real time in traffic surveillance systems. For example, sensors in these systems, combined with object detection techniques, can identify and track vehicles, pedestrians, and other road users. The data gathered from detection and tracking is then employed for traffic flow management, congestion detection, and incident identification, helping to inform decisions and enhance traffic management strategies.

Advanced driver assistance systems (ADAS): Object detection is a core technique in ADAS designed to improve driver safety and assist with driving tasks. By using sensors, object detection in ADAS can identify and track road objects, enabling features such as forward collision warnings, lane departure alerts, blind-spot detection, and adaptive cruise control.

Autonomous vehicles: Object detection using multiple sensors, such as cameras, LiDAR (light detection and ranging), radar, and ultrasonic sensors, enables vehicles to perceive and comprehend their surroundings, facilitating autonomous driving without human intervention. The data collected by these sensors allow autonomous vehicles to identify and classify objects like vehicles, pedestrians, cyclists, traffic signs, and traffic lights. This information supports decision-making and ensures safe navigation, enabling vehicles to operate independently.

Traffic sign detection and recognition: Object detection detects and recognizes traffic signs and road markings, such as traffic signs (speed limits, stop signs, yield signs, and lane markings). This information assists drivers in real-time compliance with traffic regulations and supports autonomous vehicles in understanding and obeying traffic signs.

In summary, object detection techniques are vital in road transportation systems. They enhance safety and efficiency for drivers, passengers, and pedestrians. Additionally, these techniques enable smart transportation systems to perceive and adapt to dynamic road conditions.

3.1.2. One-Stage-Based Object Detection System

Object detection systems share common components, such as video/image from a camera, image frames generated from the video, object detectors, and object bounding boxes as output. In this article, we focus on evaluating and analyzing the performance of YOLO models (i.e., a one-stage object detector) in smart transportation (i.e., one of the IoT applications). A one-stage detector (e.g., YOLO, SSD, etc.) is regarded as the object detector in an object detection system used in smart transportation, such as on autonomous vehicles and smart traffic lights. In Figure 2, we provide an overview of the one-stage object detection system in smart transportation.

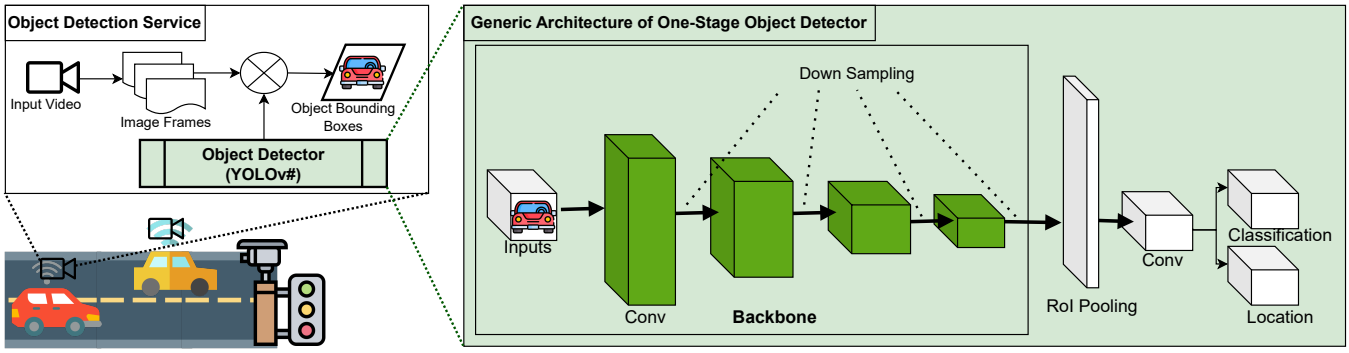


Figure 2. One-stage object detection in road transportation ("#" indicates the version number of the object detector model, such as YOLOv2, YOLOv3).

As illustrated in Figure 2, the one-stage object detection system maintains common components of a conventional object detection system. The video/image input is collected from cameras deployed in smart transportation. The output is bounding boxes of detected objects. The primary difference is which object detection mechanisms are utilized by the object detector. A one-stage object detection system employs one-stage object detection mechanisms as its object detectors. An architectural overview of one-stage object detectors is shown on the right side of the figure.

The architecture of one-stage object detectors in Figure 2 is referred from [33,44]. It typically includes a backbone network, usually pretrained models on ImageNet datasets [27], region-based analysis methods, such as region of interest (RoI) pooling, and a dense prediction with the object locations and classifications in bounding box outputs. For more details, see the following:

- **Backbone network** acts as a feature generation network, which extracts features from an image. It inputs an image and generates a feature map while preserving spatial information of the image. Popular backbone networks used in object detection methods are Darknet [20], CSPDarknet [27], ResNet [45], VGG [46], EfficientNet [47], MobileNet [48], etc.
- **RoI pooling** is a method focusing only on certain portions of the feature map generated from the backbone to output new fixed-length feature vectors. Using the feature map from the backbone network alone is ineffective for locating objects, so RoI pooling addresses this by focusing on a set number of Regions of Interest (RoIs), determined by a region proposal network (RPN), to produce a fixed-length feature vector for each RoI. Thus, RoI pooling gets two inputs, which are the feature map from the backbone and RoI proposals from an RPN.

- **Classification and location:** fixed-length feature vectors from RoI pooling are input to fully connected layers for classification and bounding box regression (i.e., location).

The YOLO models are one of the one-stage object detection algorithms (e.g., YOLO [20], SSD [19], RetinaNet [21], etc.). The following part provides an overview of the YOLO framework and its models, including key features used in different YOLO versions.

3.2. YOLO Models

Three essential components of the YOLO framework [27] are the backbone, neck, and dense prediction (or head), as presented in Figure 3.

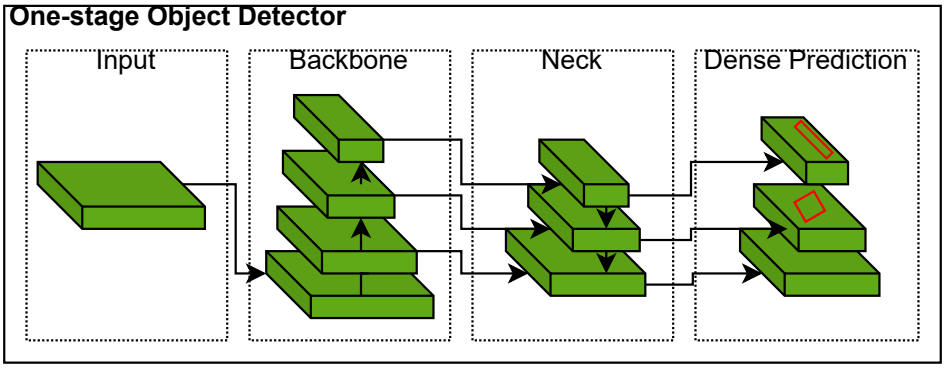


Figure 3. YOLO framework (Red frames highlight detected object regions).

- **Backbone** is a CNN that extracts features from input images, typically pretrained on a classification dataset, usually ImageNet, as explained in Section 3.1.2. All backbone models are classification models. Well-known backbones used in object detection are VGG16 [46], ResNet50 [45], EfficientNet [47], CSPDarknet53 [49], etc.
- **Neck** consists of layers of collected feature mapping from different backbone stages. Simply, it works as a feature aggregator. It is composed of several bottom-up and top-down paths as depicted in the figure. Neck networks in object detection include spatial pyramid pooling (SPP) [12], feature pyramid network (FPN) [50], path aggregation network (PANet) [51], etc.
- **Head/dense prediction** works as an object detector, which determines the object region but does not specify which object classes are presented in the region. It is equal to the RoI pooling of the one-stage object detection in Figure 2. Head networks can be classified into anchor-based or anchor-free detectors [27].

YOLO versions are developed based on the YOLO framework described above. They vary in the mechanisms used for the backbone, neck, and head. We present a summary of the primary mechanisms, such as backbone, neck, anchor box, activation function, loss function, and data augmentation, utilized in the latest YOLO models (i.e., YOLOv3, YOLOv4, YOLOv5, YOLOv6, YOLOv7, YOLOv8) in Tables 1 and 2.

Table 1. Features used in YOLO models (1).

Features	YOLOv3	YOLOv4	YOLOv5
Backbone	Darknet53	CSPDarknet53	CSPDarknet53
Neck	FPN	SPP, PANet	PANet
Anchor box	Scaled size, varied shape	Scaled size, varied shape, K-Means generating	Auto-anchor
Activation function	ReLU	Leaky ReLU	Swish
Loss function	Localization loss, confidence loss, class loss	Localization loss, confidence loss, class loss, IoU loss, CIoU loss	Localization loss, confidence loss, class loss, IoU loss, CIoU loss focal loss, GHM loss
Data augmentation	Mosaic	Mosaic	Mosaic, random shapes training

Table 2. Features used in YOLO models (2).

Features	YOLOv6	YOLOv7	YOLOv8
Backbone	EfficientRep	E-ELAN	Modified CSPDarknet53
Neck	Rep-PAN	CSPSPP, ELAN, E-ELAN	PAN
Anchor box	Anchor-free	Auto-anchor	Anchor-free
Activation function	ReLU	SiLU	Mish
Loss function	Varifocal loss, distribution focal loss	Varifocal loss, distribution focal loss	Class loss, CIoU loss, distribution focal loss
Data augmentation	Mosaic	Mosaic, mixup	Mosaic

3.3. Metrics for Evaluating Object Detection

Three popular performance metrics for object detection are frames per second (FPS), recall, and precision [23,24,33,52]. Two main approaches for object detection performance evaluations are speed and accuracy. The fps or frame rate is a performance metric focusing on the speed of the object detection procedure. This performance metric is critical in real-time applications based on object detection, such as autonomous driving, traffic monitoring, video surveillance, and medical detection in healthcare [53]. A typical fps for computer vision applications is about 15 to 30 (fps) [54]. For real-time object detection applications, a high speed of 10 (fps) or higher is required. For example, YOLO models are well-known for their accuracy and speed, on i7 CPU and the inference on 640-resolution images, YOLOv5 Nano, YOLOv6 Nano, and YOLOv6 Tiny perform more than 20 (fps) [55], satisfying the real-time requirement of object detection applications. However, this article focuses on comparing YOLO models in terms of accuracy. The following sections review two popular performance metrics related to accuracy: recall and precision.

Before considering the details of accuracy performance metrics, we discuss the same commonly used concepts. First, it is Intersection over Union (IoU), which is a principle in all state-of-the-art metrics [56]. It is defined as the intersection over a union of the detection bounding box and the ground truth bounding box.

Figure 4 visualizes the IoU with the detection and the ground truth bounding boxes.

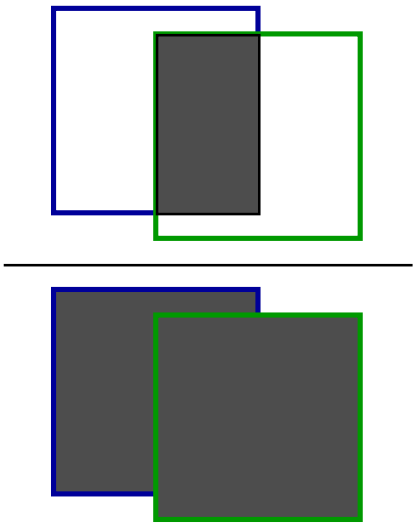


Figure 4. Intersection over Union (IoU).

The formula for calculating IoU is given below:

$$IoU = \frac{\text{Area of Overlap}}{\text{Area of Union}} \tag{1}$$

According to IoU and its score threshold definition (i.e., $IoU \geq 0.55$ is considered as a good prediction), how many objects in an image are detected correctly will be determined. Other basic concepts used in accuracy metrics are the true positive (TP), false positive (FP), and false negative (FN), which are defined as follows:

- True positive (TP): Indicates the correct detection of the ground truth bounding box [52]. It means the IoU score of the prediction is greater than a predefined threshold.
- False positive (FP): Indicates the incorrect detection of the ground truth bounding box [52]. It means the IoU score of the prediction is smaller than a predefined threshold.
- False negative (FN): indicates the undetected ground truth bounding box [52]

A true negative (TN) indicates that the bounding boxes should not be detected in an image. This concept cannot be considered in object detection, because the number of this kind of bounding box is massive in an image. Thus, metrics using the TN are not to be used to assess object detection methods, such as the receiver operating characteristic (ROC) curves, FP rate (FPR), and TP rate (TPR) [52].

Two basic metrics based on the above concepts are precision (P) and recall (R). The precision measures how accurate predictions are. In other words, this metric indicates the percentage of correct predictions.

$$P = \frac{TP}{TP + FP} = \frac{TP}{\text{all detections}} \tag{2}$$

Recall (R) measures how well the model determines all positives.

$$R = \frac{TP}{TP + FN} = \frac{TP}{\text{all ground truths}} \tag{3}$$

F1-Score is the combination of both precision and recall. This metric is known as the harmonic mean of the two metrics. The formula for the F1-Score is represented as follows:

$$F1 - Score = 2 \cdot \frac{P.R}{P + R} \tag{4}$$

Instead of using a single metric (P or R) or precision–recall curve (PR curve) in the assessment process, an area under the curve (AUC) is employed to indicate both precision and recall. This approach eliminates the need to choose a specific confidence threshold for the PR curve. A higher AUC indicates an increase in both precision and recall. However, AUC is usually a zigzag-like curve, which makes it hard to analyze the accurate measurement of AUC [52]. To address this, average precision (AP) is widely used, particularly for validating the performance of object detection and localization algorithms, as it helps overcome the limitations of AUC. The following part provides an explanation of metrics related to AP.

$AP@\alpha$ is an area under the precision–recall curve (AUC) evaluated at the α IoU threshold. It is defined as follows:

$$AP@\alpha = \int_0^1 p(r)\, dr \tag{5}$$

As detailed in Equation (5), AP is computed using additional metrics, such as the Intersection over Union (IoU) and the confusion matrix components (TP, FP, FN), as well as precision (denoted as p in the equation) and recall (denoted as r in the equation). Figure 5 illustrates the steps involved in calculating the AP for class 1.

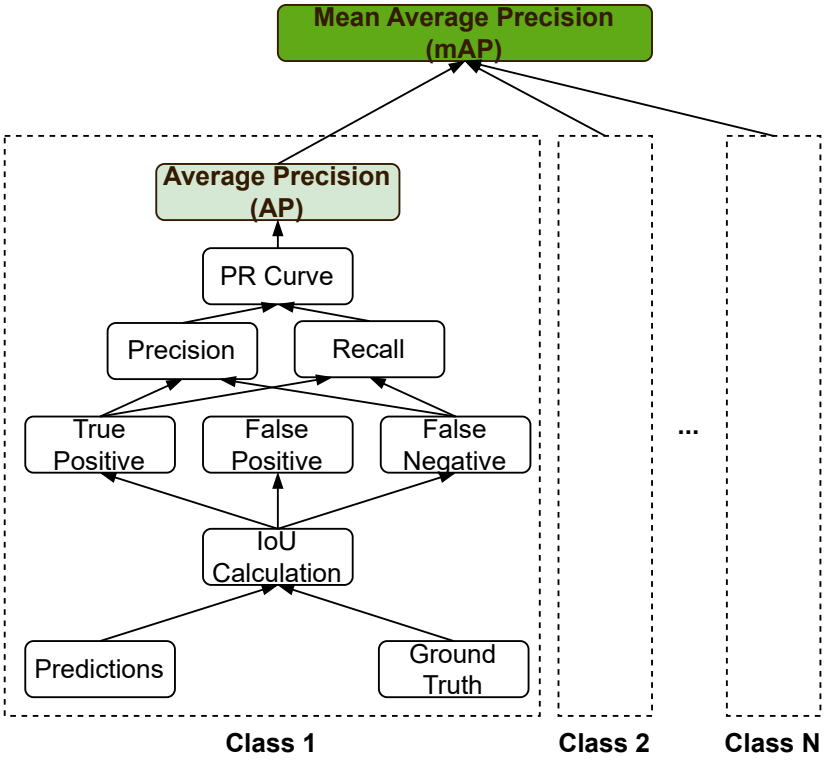


Figure 5. Steps for calculating mean average precision (mAP).

AP condenses the PR curve into a single scalar value. When AP is high, both precision and recall are also high; conversely, a low AP indicates that either precision or recall is low. The AP value ranges from 0 to 1.

One of the most popular metrics developed from $AP@\alpha$ is the mean average precision (mAP). mAP is used to assess object detection methods over all classes in a particular dataset [52]. Basically, it is the average AP over all classes:

$$mAP = \frac{1}{N} \sum_{n=1}^{N} AP_n \tag{6}$$

where AP_n denotes the AP of the nth class in the dataset, and N is the total number of classes in the dataset. Figure 5 shows the steps to obtain mAP.

Which metric should be used for measuring the performance of object detection is crucial. According to the steps in an object detection problem, mAP is suggested to be used on a validation set. Otherwise, precision, recall, and F1-Score are adequate on the test set.

In this article, we focus on analyzing and evaluating the performance of different YOLO versions, thus we consider mAP as the primary performance metric.

3.4. Datasets for Object Detection

Various datasets are used in object detection applications, such as ImageNet [57], COCO [58], PASCAL VOC [59], KITTI [60], DOTA [61], Cityscapes [62], BDD100K [63], etc. In this article, we focus on an empirical evaluation and analysis of YOLO models in smart transportation, such as road transportation. Accordingly, we select COCO and PASCAL VOC as our experimental datasets. The two datasets are capable of providing not only a number of objects in road transportation but efficient notations as well with high-resolution images. They are used in a lot of YOLO-related studies of object detection, such as YOLO [20], Single Shot MultiBox Detector (SSD) [19], You Only Look Cluster (YOLC) [64], Faster R-CNN [65], etc. Principle statistics of these datasets are provided next.

3.4.1. COCO (Common Objects in Context)

The COCO dataset is a large-scale image recognition dataset for object detection, segmentation, and captioning tasks. It contains over 330,000 images. Each image is annotated with 80 object categories and 5 captions describing the scene. Object categories are also divided into two main categories, “things” and “stuff”. While the “things” contains object classes easily handled (e.g., animals, vehicles, household items, etc.), “stuff” includes environmental items (e.g., tree, sky, road, etc.) [66]. In particular, some prevalent object classes in 80 classes are person, bicycle, car, airplane, bus, train, truck, traffic light, cat, dog, sink, etc. Moreover, COCO offers various types of annotations, such as object detection, stuff image segmentation, panoptic segmentation, and keypoint. For more details, with object detection annotation, COCO provides annotations with bounding box coordinates for 80 object classes.

3.4.2. PASCAL VOC (Visual Object Classes)

PASCAL VOC provides standardized images for object detection and segmentation tasks. The latest update contains 11,530 images with 20 object classes. Additionally, it provides 27,450 annotated objects and 6929 segmentations [67]. These object classes are grouped into person, animal, vehicle, and indoor classes. In particular, some prevalent object classes are person, bird, horse, airplane, boat, bicycle, bus, chair, sofa, etc. PASCAL VOC also provides object detection annotation with bounding box coordinates for 20 object classes.

4. Evaluation and Result Analysis

In this article, the performance of YOLO models is extensively evaluated using various benchmark datasets (i.e., COCO and PASCAL VOC) and the evaluation metric setup (i.e., IoU thresholds). Specifically, we evaluate YOLO models on the COCO and PASCAL VOC datasets, which cover a wide range of object categories on road transportation. The mAP evaluation metric is used to measure the accuracy of various YOLO models across different datasets, object classes, and IoU thresholds.

4.1. Experimental Setup

We use a server equipped with a 32-core Intel Xeon Gold CPU and 64 GB of DRAM, running with an 18.04.5 Ubuntu and 4.15.0 Linux kernel for training and evaluation. We use COCO2017 and PASCAL VOC as training and evaluation datasets. In this article, we focus on object detection in road transportation and only consider object classes related to

road transportation in two datasets. The COCO2017 datasets have about 80 classes, but only road-transportation-related object classes are used in the experiments, including ten classes: bicycle, car, motorcycle, bus, train, truck, traffic light, stop sign, parking meter, and person. The total number of images we use for the validation is 5000 and with each object class detection, the number of labels varies from 60 to 10,777. For the PASCAL VOC datasets, we use 11,530 images with 20 object classes of road transportation for training. In validation, we use 3422 images with six object classes (i.e., bicycle, bus, car, motorbike, person, and train). The number of labels varies from 142 to 7835.

In addition, for the validation step in both experiments conducted with the two datasets, we set $K = 10$ for K-fold cross-validation. A summary of the datasets' usage in the experiments is presented in Table 3. We validate various YOLO models on COCO with pretrained weights. Also, we train and validate various YOLO models on COCO and Pascal VOC with CPUs.

Table 3. Dataset description.

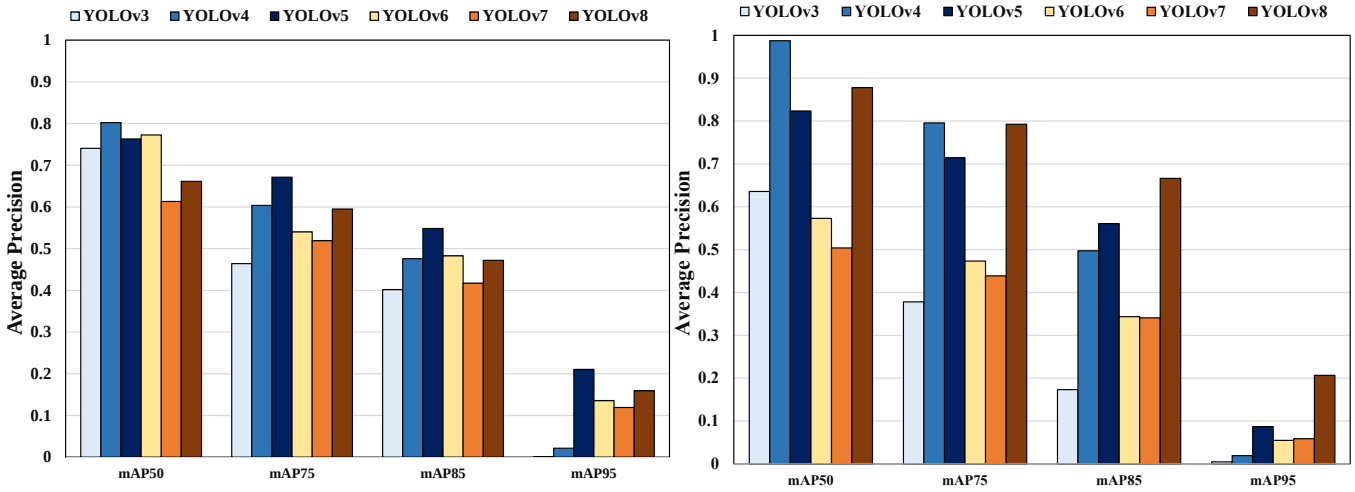
	COCO		**PASCAL VOC**	
	Training	**Validation**	**Training**	**Validation**
Num. Image	-	5000	11,530	3422
Classes	person, parking meter stop sign, traffic light, train, bus, motorcycle, car, truck, bicycle		bicycle, car, motorbike, bus, person, train	
Num. Labels	[60, 10,777]		[142, 7835]	

4.2. Evaluation Overview

We present the experiment results, comparing the performance of YOLO models in terms of the datasets (i.e., COCO and PASCAL VOC), accuracy (i.e., mAP scores at various IoU thresholds), and object classes. In experiments, we perform training and evaluation to detect objects in images from the two datasets. Figure 6 illustrates an example of object detection in images after applying YOLO models. These images contain various detected objects, including people, animals (e.g., zebra, horse, etc.), and vehicles (e.g., car, motorcycle, airplane, truck, etc.).

Figure 6. Object detection after using YOLO models.

Figure 7a,b illustrates the AP results of YOLO models on COCO and PASCAL VOC datasets, respectively. Figures 8 and 9 indicate how the IoU thresholds affect the AP of object classes in COCO and PASCAL VOC datasets, respectively. Tables 4–7 show the AP at the IoU threshold 50 and the average AP (i.e., average AP at all IoU thresholds) of YOLO models on specific object classes in COCO and PASCAL VOC datasets, respectively.



(**a**) COCO (**b**) PASCAL VOC

Figure 7. YOLO models with various IoU thresholds on COCO and PASCAL VOC.

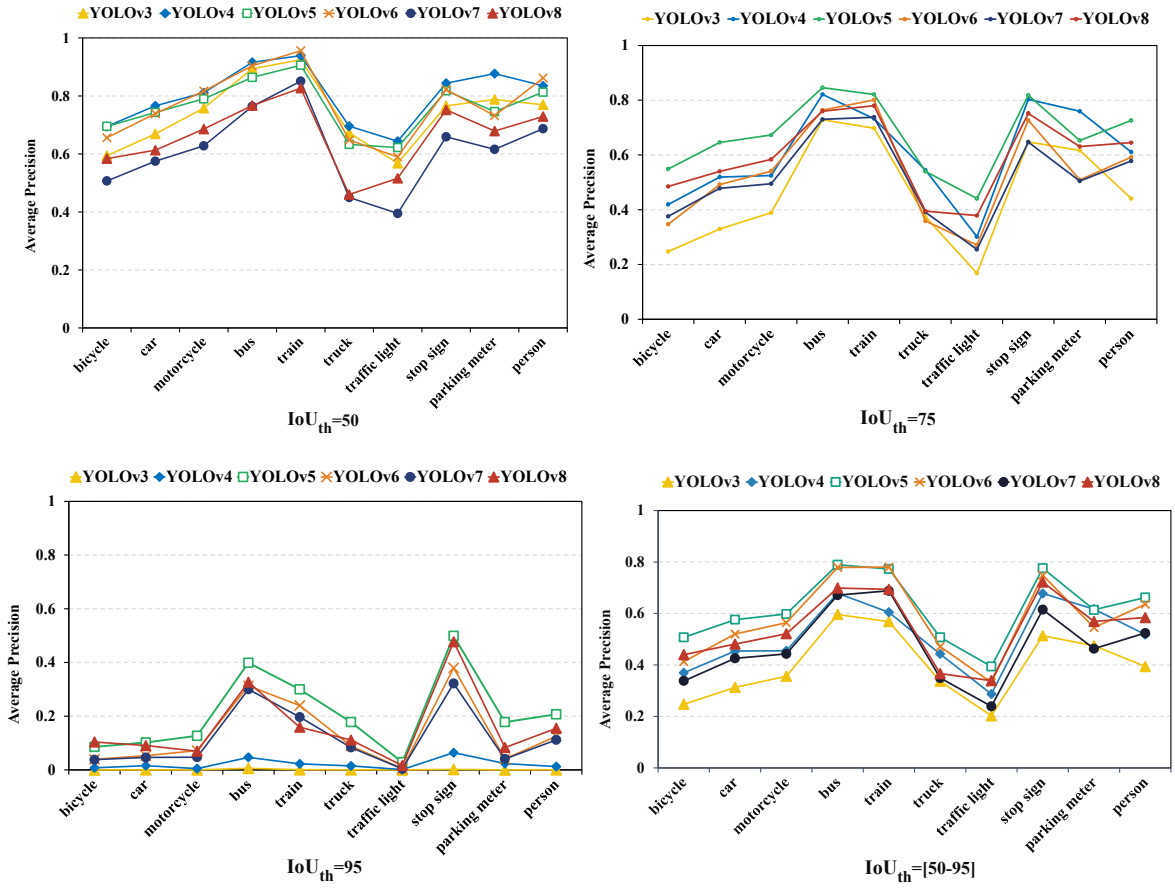


Figure 8. AP of YOLO models across object classes in COCO.

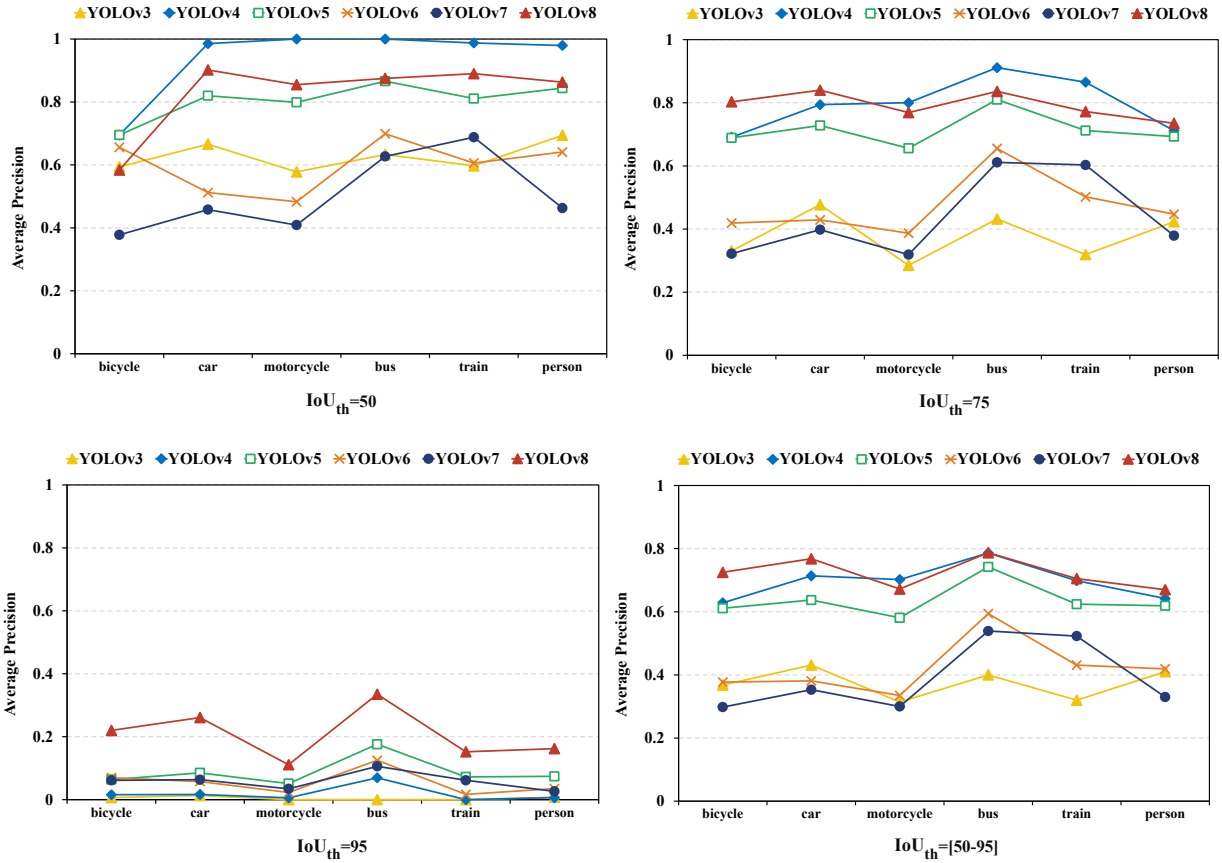


Figure 9. AP of YOLO models on object classes in PASCAL VOC.

Table 4. AP@50 of YOLO models on COCO.

Models	Bicycle	Car	Motorcycle	Bus	Train	Truck	Traffic Light	Stop Sign	Parking Meter	Person	AP
YOLOv8	0.584	0.613	0.686	0.767	0.827	0.46	0.516	0.752	0.679	0.729	0.6613
YOLOv7	0.507	0.575	0.628	0.764	0.851	0.45	0.395	0.659	0.616	0.687	0.6132
YOLOv6	0.656	0.739	**0.817**	0.904	**0.956**	0.648	0.59	0.822	0.732	**0.862**	0.7726
YOLOv5	**0.695**	0.743	0.79	0.864	0.906	0.633	0.622	0.818	0.746	0.813	0.763
YOLOv4	**0.695**	**0.766**	0.811	**0.916**	0.939	**0.696**	**0.644**	**0.844**	**0.877**	0.835	**0.8023**
YOLOv3	0.594	0.669	0.758	0.894	0.925	0.673	0.568	0.766	0.788	0.77	0.7405

Table 5. AP@50–95 of YOLO models on COCO.

Models	Bicycle	Car	Motorcycle	Bus	Train	Truck	Traffic Light	Stop Sign	Parking Meter	Person	AP
YOLOv8	0.44	0.481	0.521	0.699	0.694	0.366	0.339	0.722	0.569	0.584	0.5415
YOLOv7	0.338	0.426	0.443	0.671	0.688	0.349	0.239	0.615	0.463	0.524	0.4756
YOLOv6	0.413	0.52	0.564	0.779	**0.78**	0.471	0.332	0.748	0.545	0.636	0.5788
YOLOv5	**0.507**	**0.576**	**0.598**	**0.789**	0.773	**0.507**	**0.394**	**0.776**	0.614	**0.662**	**0.6196**
YOLOv4	0.36943	0.4536	0.4549	0.6782	0.6047	0.4425	0.2869	0.6777	**0.6169**	0.5179	0.4614
YOLOv3	0.2472	0.3126	0.356	0.5961	0.5685	0.3361	0.202	0.5133	0.4747	0.3936	0.4

Table 6. AP@50 of YOLO models on PASCAL VOC.

Models	Bicycle	Bus	Car	Motorbike	Person	Train	AP
YOLOv8	0.883	0.875	0.901	0.857	0.864	0.89	0.8783
YOLOv7	0.378	0.627	0.458	0.409	0.463	0.688	0.5038
YOLOv6	0.494	0.691	0.51	0.477	0.632	0.603	0.5678
YOLOv5	0.801	0.866	0.82	0.799	0.844	0.811	0.8235
YOLOv4	**0.9737**	**1**	**0.9854**	**1**	**0.9795**	**0.9875**	**0.9877**
YOLOv3	0.62	0.637	0.644	0.607	0.731	0.602	0.6402

Table 7. AP@50–95 of YOLO models on PASCAL VOC.

Models	Bicycle	Bus	Car	Motorbike	Person	Train	AP
YOLOv8	**0.724**	**0.788**	**0.769**	0.673	**0.669**	**0.707**	**0.7217**
YOLOv7	0.298	0.539	0.353	0.3	0.33	0.523	0.3905
YOLOv6	0.375	0.589	0.38	0.334	0.417	0.429	0.4207
YOLOv5	0.611	0.742	0.637	0.581	0.619	0.624	0.6357
YOLOv4	0.6283	0.7861	0.7136	**0.7021**	0.6418	0.6984	0.6951
YOLOv3	0.292	0.34	0.347	0.261	0.364	0.258	0.3103

4.3. Experimental Results and Observations

The next section examines the performance of YOLO from three perspectives: datasets, IoU thresholds, and object classes. Based on the performance analysis, we provide empirical observations on the performance of various YOLO models when applied to road transportation datasets (i.e., COCO and PASCAL VOC).

4.3.1. Dataset

Figure 7a,b present the AP of six YOLO models at four thresholds IoU_{th} (i.e., IoU threshold $= \{50, 75, 85, 95\}$) on the COCO and PASCAL VOC datasets, respectively. As depicted in the figures, YOLO models on the PASCAL VOC datasets can obtain a higher mAP than on the COCO datasets across various IoU thresholds.

For example, as shown in Figure 7a, dominant YOLO models (i.e., YOLOv4, YOLOv5, YOLOv8), trained on COCO datasets, raise the AP in a range $(0.65, 0.8)$ at $IoU_{th} = 50$, $(0.6, 0.68)$ at $IoU_{th} = 75$, $(0.48, 0.55)$ at $IoU_{th} = 85$, and $(0.16, 0.21)$ at $IoU_{th} = 95$. However, for the dominant YOLO models on PASCAL VOC datasets shown in Figure 7b, the AP range is greater with $(0.81, 0.99)$ at $IoU_{th} = 50$, $(0.71, 0.8)$ at $IoU_{th} = 75$, $(0.5, 0.67)$ at $IoU_{th} = 85$, and $(0.09, 0.21)$ at $IoU_{th} = 95$.

YOLO models perform better on the Pascal VOC dataset than on COCO because Pascal VOC contains fewer object classes and a less diverse range of object sizes compared with COCO. Additionally, the PASCAL VOC dataset is specifically designed for object detection tasks, whereas the COCO dataset covers a broader range of computer vision tasks, including object detection, instance segmentation, and more. Hence, images in PASCAL VOC can contain efficient and suitable objects for the training and validation processes of object detection tasks. Thus, according to the database variety, we make the following observation related to the YOLO performance.

Observation 1: The performance of YOLO models trained or evaluated on different datasets can have disparate performance.

In addition, in Figure 7a,b, we can observe that YOLOv5 and YOLOv8 are dominant models at various IoU thresholds. For example, on COCO datasets, as shown in Figure 7a, YOLOv5 outperforms the others at $IoU_{th} = \{75, 85, 95\}$. Further, YOLOv8 follows YOLOv5. At $IoU_{th} = 50$, YOLOv4 and YOLOv6 can achieve high performance, but at other IoU thresholds, they cannot maintain their performance like YOLOv5 and YOLOv8 can do.

Therefore, we cannot consider YOLOv4 and YOLOv6 as dominant YOLO models on the COCO datasets. On the PASCAL VOC datasets, as shown in Figure 7b, YOLOv8 is better than the others at $IoU_{th} = \{75, 85, 95\}$. YOLOv5 goes after YOLOv8 at $IoU_{th} = \{85, 95\}$. YOLOv4 can achieve high performance at $IoU_{th} = \{50, 75\}$ but has the lowest performance at $IoU_{th} = 95$. Therefore, we cannot consider YOLOv4 a dominant YOLO model on the PASCAL VOC datasets.

Furthermore, YOLOv5 outperforms other models because it uses the CSPDarknet53 backbone, PANet for effective multiscale feature integration, the auto-anchor technique for bounding box adjustment, and an improved loss function.

Additionally, YOLOv8 demonstrates superior performance due to its use of the modified CSPDarknet53 backbone, PAN feature fusion, anchor-free detection, class loss optimization, complete IoU loss for precise bounding boxes, and distribution focal loss to enhance accuracy across diverse object classes. Therefore, we make the following observations related to the datasets.

Observation 2: The YOLOv5/YOLOv8 model demonstrates the best performance on the COCO/PASCAL VOC datasets, respectively.

4.3.2. Intersection over Union (IoU) Thresholds

Next, we conduct experiments using various IoU thresholds to observe how YOLO models perform on the two datasets. In detail, four different IoU thresholds (i.e., 50, 75, 85, and 95) are set up in the experiment to measure the performance of the YOLO models using mAP. Figure 7a,b presents the mAP for six YOLO models on the COCO and PASCAL VOC datasets, respectively.

As shown in the two figures, as the IoU threshold increases (i.e., from 50 to 95), the mAP of all YOLO models tends to decrease. This finding is apparent according to (5) and (6), indicating that a higher IoU threshold results in a higher standard for good prediction. Thus, fewer predictions are considered good or correct. Hence, we make the first observation related to the IoU threshold as follows.

Observation 3: The mAP of the YOLO models becomes lower with a higher IoU threshold.

Also, in Figure 7a,b, we can observe that YOLOv5 and YOLOv8 outperform the rest of the models for most of the IoU threshold scales. For example, on the COCO datasets, as shown in Figure 7a, YOLOv5 and YOLOv8 obtain the highest mAP at three IoU thresholds (i.e., $75, 85, 95$). On the PASCAL VOC datasets, as shown in Figure 7b, YOLOv5 and YOLOv8 outperform the other models at two IoU thresholds (i.e., $85, 95$).

This better performance can be attributed to various architectural improvements and optimizations introduced in YOLOv5 and YOLOv8, such as CSPDarknet53, PANet, Binary Cross Entropy, Logit Loss function, and CIoU loss. The first factor is the model architecture of YOLO models. The architectural design for each YOLO model plays a crucial role in performance at various IOU thresholds. Further, YOLOv3, YOLOv4, YOLOv5, YOLOv6, YOLOv7, and YOLOv8 have distinct network structures, layer configurations, and feature fusion mechanisms. These architectural variances affect the models' ability to handle object detection tasks with varying IOU thresholds. For example, some models may prioritize precision at higher IOU thresholds, whereas others may focus on recall at lower IOU thresholds. Hence, we provide a second observation of the best YOLO models when varying IoU thresholds.

Observation 4: The YOLOv5 and YOLOv8 models outperform other YOLO models across IoU thresholds.

4.3.3. Class of Objects

Figures 8 and 9 illustrate the performance of various object classes in the two datasets. As depicted in Figure 8, the performance of YOLO models on object classes in the COCO datasets fluctuates, demonstrating the difference in performance between the traffic light and bicycle classes, which perform the worst for all YOLO models. For example, at

$IoU_{th} = 50$ and $IoU_{th} = 75$, traffic light and bicycle classes obtain the lowest performance with all YOLO models. As for the bicycle class, it achieves around 0.45 (i.e., with YOLOv7) and 0.22 (i.e., with YOLOv3) at $IoU_{th} = 50$ and $IoU_{th} = 75$, respectively. As for the traffic light class, it achieves an even lower performance of around 0.4 (i.e., with YOLOv7) and 0.19 (i.e., with YOLOv3) at $IoU_{th} = 50$ and $IoU_{th} = 75$, respectively. Considering the same YOLO models at $IoU_{th} = 50$ and $IoU_{th} = 75$, the training object classes can achieve around 0.82 and 0.7, respectively, demonstrating a large performance gap between object classes on the same dataset.

In contrast, the performance of YOLO models on object classes in the Pascal VOC dataset remains more stable than in the COCO dataset, as presented in Figure 9, because PASCAL VOC contains more efficient and suitable objects than COCO. The fluctuating and stable performance of YOLO models on various object classes in the COCO and PASCAL VOC datasets, respectively, indicate that the data quality of object classes can influence the performance of YOLO models. Thus, the first observation related to the datasets is provided below.

Observation 5: The performance of YOLO models on various object classes can fluctuate or stabilize according to the quality of the datasets.

Tables 4 and 5 detail the YOLO performance at $IoU_{th} = 50$ and the average YOLO performance at a range of IoU thresholds $IoU_{th} = [50 - 95]$ on the COCO datasets, respectively. The performance on the PASCAL VOC datasets is presented in Tables 6 and 7. As shown in Tables 4 and 6, at a low IoU threshold, $IoU_{th} = 50$, YOLOv4 outperforms other YOLO models on both datasets. For example, on the COCO datasets, shown in Table 4, YOLOv4 achieves the highest performance with 7/10 object classes (i.e., bicycle, car, bus, truck, traffic light, and parking meter), with the highest AP value, 0.8023. On the PASCAL VOC datasets, in Table 6, YOLOv4 is also dominant for 6/6 object classes (i.e., bicycle, bus, car, motorbike, person, train), with the highest AP value, 0.9877. However, at various IoU thresholds, as displayed in Tables 5 and 7, the dominant models are YOLOv5 and YOLOv8 on the COCO and PASCAL VOC datasets, respectively. For example, on the COCO datasets, as shown in Table 5, YOLOv5 outperforms on 8/10 object classes (i.e., bicycle, car, motorcycle, bus, truck, traffic light, stop sign, person), with the highest AP value: 0.6196. In contrast, on the PASCAL VOC datasets, as shown in Table 7, YOLOv8 outperforms other models on 5/6 object classes (i.e., bicycle, bus, car, person, train), with the highest AP value: 0.7217. Thus, YOLOv5 and YOLOv8 outperform other models for most object classes in the two datasets, leading to the following observation.

Observation 6: Both YOLOv5 and YOLOv8 outperform the other models on most of the object classes.

The following section summarizes the overall performance results, demonstrating that YOLOv4, YOLOv5, and YOLOv8 outperform the other YOLO models on the two datasets.

4.4. Summary

In summary, the experimental results confirm the anticipated effects of datasets, object classes, and IoU thresholds on the performance of YOLO models. Specifically, first, the performance of YOLO models trained and validated on different datasets can perform disparately. Second, the higher the IoU threshold, the lower the YOLO performance. Last, YOLO performance on various object classes stabilizes according to the dataset quality.

Moreover, the experimental results show that YOLOv5 and YOLOv8 consistently outperform others across the IoU thresholds ranging from 50 to 95 on the COCO and PASCAL VOC datasets. Additionally, at a low IoU threshold (i.e., $IoU_{th} = 50$), YOLOv4 exhibits the best performance on both datasets. These results suggest that YOLOv5 and YOLOv8 are promising choices for object detection tasks in the road transportation scenario due to their strong performance across a suitable range of IoU thresholds (i.e., from 50 to 95).

5. Conclusions

Based on the evaluation and analysis of YOLO's performance across different datasets, object classes, and IoU thresholds in the context of road transportation, it has been observed that YOLO's performance can be inconsistent due to the varying quality and characteristics of the datasets and the specific object classes within each dataset. The performance of YOLO models trained and validated on the same datasets may not always align with the expectation that the latest YOLO version will achieve higher performance. This is because, while improvements in YOLO's internal architecture enhance its capabilities as an object detector, the quality and characteristics of the datasets also significantly influence its performance. Specifically, the empirical evaluation shows that YOLOv5 and YOLOv8 outperform other YOLO models on two widely used datasets, COCO and PASCAL VOC, which are commonly employed in smart transportation applications. Based on this key observation from our empirical evaluation, these models (i.e., YOLOv5 and YOLOv8) are recommended as suitable choices for object detection in road or smart transportation systems. Our study results can aid other studies in selecting the appropriate models for smart transportation systems while reducing model selection time.

This article evaluates the performance of various YOLO models while taking into account potential factors that may affect their performance, including model internals and datasets. However, it is also essential to analyze the impact of hardware (e.g., CPU, GPU, memory, etc.) during training, validation, and testing for a comprehensive empirical evaluation and analysis. Furthermore, this article does not address inference speed (i.e., frames per second (fps)), which is another important performance metric for YOLO models. Future work should consider the effects of hardware on the training, validation, and testing of YOLO models, as well as their inference speed.

Author Contributions: L.A.N., conceptualization, data curation, formal analysis, investigation, validation, writing—original draft; M.D.T., software, writing-original draft, validation, visualization; Y.S., resources, supervision, writing—review, editing. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by the National Research Foundation of Korea (NRF) (No. NRF-2022R1A4A5034130) and Korea Institute for Advancement of Technology (KIAT) (No. KIAT-P0012724) (Corresponding Author: Yongseok Son).

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Butler, L.; Yigitcanlar, T.; Paz, A. Smart Urban Mobility Innovations: A Comprehensive Review and Evaluation. *IEEE Access* **2020**, *8*, 196034–196049. [CrossRef]
2. Golpayegani, F.; Guériau, M.; Laharotte, P.A.; Ghanadbashi, S.; Guo, J.; Geraghty, J.; Wang, S. Intelligent Shared Mobility Systems: A Survey on Whole System Design Requirements, Challenges and Future Direction. *IEEE Access* **2022**, *10*, 35302–35320. [CrossRef]
3. Longo, A.; Zappatore, M.; Navathe, S.B. The unified chart of mobility services: Towards a systemic approach to analyze service quality in smart mobility ecosystem. *J. Parallel Distrib. Comput.* **2019**, *127*, 118–133. [CrossRef]
4. Peak, E. What Makes Transportation Smart? Defining Intelligent Transportation. Available online: https://www.iotforall.com/what-makes-transportation-smart-defining-intelligent-transportation (accessed on 13 June 2023).
5. Zantalis, F.; Koulouras, G.; Karabetsos, S.; Kandris, D. A review of machine learning and IoT in smart transportation. *Future Internet* **2019**, *11*, 94. [CrossRef]
6. IBM. What Is Smart Transportation. Available online: https://www.ibm.com/blog/smart-transportation/ (accessed on 13 June 2023).
7. Chen, B.H.; Huang, S.C. An Advanced Moving Object Detection Algorithm for Automatic Traffic Monitoring in Real-World Limited Bandwidth Networks. *IEEE Trans. Multimed.* **2014**, *16*, 837–847. [CrossRef]
8. Ulmer, B. VITA-an autonomous road vehicle (ARV) for collision avoidance in traffic. In Proceedings of the Intelligent Vehicles '92 Symposium, Detroit, MI, USA, 29 June–1 July 1992; pp. 36–41. [CrossRef]
9. Vu, V.T.; Bremond, F.; Davini, G.; Thonnat, M.; Pham, Q.C.; Allezard, N.; Sayd, P.; Rouas, J.L.; Ambellouis, S.; Flancquart, A. Audio-Video Event Recognition System for Public Transport Security. In Proceedings of the 2006 IET Conference on Crime and Security, London, UK, 13–14 June 2006; pp. 414–419.

10. Dalal, N.; Triggs, B. Histograms of oriented gradients for human detection. In Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), San Diego, CA, USA, 20–25 June 2005; Volume 1, pp. 886–893. [CrossRef]
11. Felzenszwalb, P.F.; Girshick, R.B.; McAllester, D.; Ramanan, D. Object Detection with Discriminatively Trained Part-Based Models. *IEEE Trans. Pattern Anal. Mach. Intell.* **2010**, *32*, 1627–1645. [CrossRef]
12. He, K.; Zhang, X.; Ren, S.; Sun, J. Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. *IEEE Trans. Pattern Anal. Mach. Intell.* **2015**, *37*, 1904–1916. [CrossRef]
13. Girshick, R.B. Fast R-CNN. *arXiv* **2015**, arXiv:1504.08083.
14. Yang, Z.; Liu, S.; Hu, H.; Wang, L.; Lin, S. RepPoints: Point Set Representation for Object Detection. *arXiv* **2019**, arXiv:1904.11490.
15. Carion, N.; Massa, F.; Synnaeve, G.; Usunier, N.; Kirillov, A.; Zagoruyko, S. End-to-End Object Detection with Transformers. *arXiv* **2020**, arXiv:2005.12872.
16. Zheng, S.; Lu, J.; Zhao, H.; Zhu, X.; Luo, Z.; Wang, Y.; Fu, Y.; Feng, J.; Xiang, T.; Torr, P.H.S.; et al. Rethinking Semantic Segmentation from a Sequence-to-Sequence Perspective with Transformers. *arXiv* **2020**, arXiv:2012.15840.
17. Sun, Z.; Cao, S.; Yang, Y.; Kitani, K.M. Rethinking transformer-based set prediction for object detection. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Montreal, QC, Canada, 10–17 October 2021; pp. 3611–3620.
18. Girshick, R. Fast R-CNN. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), Santiago, Chile, 7–13 December 2015.
19. Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C.Y.; Berg, A.C. SSD: Single Shot MultiBox Detector. In Proceedings of the Computer Vision—ECCV 2016, Amsterdam, The Netherlands, 11–14 October 2016; Leibe, B., Matas, J., Sebe, N., Welling, M., Eds.; Springer: Cham, Germany, 2016; pp. 21–37.
20. Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You only look once: Unified, real-time object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 779–788.
21. Lin, T.Y.; Goyal, P.; Girshick, R.; He, K.; Dollár, P. Focal loss for dense object detection. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; pp. 2980–2988.
22. Law, H.; Deng, J. CornerNet: Detecting Objects as Paired Keypoints. *arXiv* **2018**, arXiv:1808.01244.
23. Zaidi, S.S.A.; Ansari, M.S.; Aslam, A.; Kanwal, N.; Asghar, M.; Lee, B. A survey of modern deep learning based object detection models. *Digit. Signal Process.* **2022**, *126*, 103514. [CrossRef]
24. Liu, L.; Ouyang, W.; Wang, X.; Fieguth, P.; Chen, J.; Liu, X.; Pietikäinen, M. Deep learning for generic object detection: A survey. *Int. J. Comput. Vis.* **2020**, *128*, 261–318. [CrossRef]
25. Redmon, J.; Farhadi, A. YOLO9000: Better, Faster, Stronger. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2016; pp. 6517–6525.
26. Redmon, J.; Farhadi, A. YOLOv3: An Incremental Improvement. *arXiv* **2018**, arXiv:1804.02767.
27. Bochkovskiy, A.; Wang, C.Y.; Liao, H.Y.M. YOLOv4: Optimal Speed and Accuracy of Object Detection. *arXiv* **2020**, arXiv:2004.10934.
28. YOLOv5: The Friendliest AI Architecture You'll Ever Use. Available online: https://ultralytics.com/yolov5 (accessed on 12 June 2023).
29. Li, C.; Li, L.; Jiang, H.; Weng, K.; Geng, Y.; Li, L.; Ke, Z.; Li, Q.; Cheng, M.; Nie, W.; et al. YOLOv6: A Single-Stage Object Detection Framework for Industrial Applications. *arXiv* **2022**, arXiv:2209.02976.
30. Wang, C.Y.; Bochkovskiy, A.; Liao, H.Y.M. YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. *arXiv* **2022**, arXiv:2207.02696.
31. Reis, D.; Kupec, J.; Hong, J.; Daoudi, A. Real-Time Flying Object Detection with YOLOv8. *arXiv* **2023**, arXiv:2305.09972.
32. Wu, X.; Sahoo, D.; Hoi, S.C.H. Recent Advances in Deep Learning for Object Detection. *arXiv* **2019**, arXiv:1908.03673. [CrossRef]
33. Jiao, L.; Zhang, F.; Liu, F.; Yang, S.; Li, L.; Feng, Z.; Qu, R. A Survey of Deep Learning-Based Object Detection. *IEEE Access* **2019**, *7*, 128837–128868. [CrossRef]
34. Tong, K.; Wu, Y.; Zhou, F. Recent advances in small object detection based on deep learning: A review. *Image Vis. Comput.* **2020**, *97*, 103910. [CrossRef]
35. Nazir, A.; Wani, M.A. You Only Look Once—Object Detection Models: A Review. In Proceedings of the 2023 10th International Conference on Computing for Sustainable Global Development (INDIACom), New Delhi, India, 15–17 March 2023; pp. 1088–1095.
36. Sozzi, M.; Cantalamessa, S.; Cogato, A.; Kayad, A.; Marinello, F. Automatic Bunch Detection in White Grape Varieties Using YOLOv3, YOLOv4, and YOLOv5 Deep Learning Algorithms. *Agronomy* **2022**, *12*, 319. [CrossRef]
37. Liu, K.; Tang, H.; He, S.; Yu, Q.; Xiong, Y.; Wang, N. Performance validation of YOLO variants for object detection. In Proceedings of the 2021 International Conference on Bioinformatics and Intelligent Computing, Harbin, China, 22–24 January 2021; pp. 239–243.
38. Gündüz, M.Ş.; Işık, G. A new YOLO-based method for real-time crowd detection from video and performance analysis of YOLO models. *J. Real Time Image Process.* **2023**, *20*, 5. [CrossRef]

39. Feng, H.; Mu, G.; Zhong, S.; Zhang, P.; Yuan, T. Benchmark analysis of yolo performance on edge intelligence devices. *Cryptography* **2022**, *6*, 16. [CrossRef]
40. Magalhães, S.A.; Castro, L.; Moreira, G.; Dos Santos, F.N.; Cunha, M.; Dias, J.; Moreira, A.P. Evaluating the single-shot multibox detector and YOLO deep learning models for the detection of tomatoes in a greenhouse. *Sensors* **2021**, *21*, 3569. [CrossRef]
41. Tang, J.; Ye, C.; Zhou, X.; Xu, L. YOLO-Fusion and Internet of Things: Advancing object detection in smart transportation. *Alex. Eng. J.* **2024**, *107*, 1–12. [CrossRef]
42. Wu, J.D.; Chen, B.Y.; Shyr, W.J.; Shih, F.Y. Vehicle Classification and Counting System Using YOLO Object Detection Technology. *Trait. Signal* **2021**, *38*, 1087. [CrossRef]
43. Sindhwani, N.; Verma, S.; Bajaj, T.; Anand, R. Comparative analysis of intelligent driving and safety assistance systems using YOLO and SSD model of deep learning. *Int. J. Inf. Syst. Model. Des. (IJISMD)* **2021**, *12*, 131–146. [CrossRef]
44. Pal, S.K.; Pramanik, A.; Maiti, J.; Mitra, P. Deep learning in multi-object detection and tracking: State of the art. *Appl. Intell.* **2021**, *51*, 6400–6429. [CrossRef]
45. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778. [CrossRef]
46. Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv* **2014**, arXiv:1409.1556.
47. Tan, M.; Le, Q. Efficientnet: Rethinking model scaling for convolutional neural networks. In Proceedings of the International Conference on Machine Learning, PMLR, Long Beach, CA, USA, 9–15 June 2019; pp. 6105–6114.
48. Howard, A.G. MobileNets: Efficient convolutional neural networks for mobile vision applications. *arXiv* **2017**, arXiv:1704.04861.
49. Wang, C.Y.; Liao, H.Y.M.; Wu, Y.H.; Chen, P.Y.; Hsieh, J.W.; Yeh, I.H. CSPNet: A new backbone that can enhance learning capability of CNN. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, Seattle, WA, USA, 17–19 June 2020; pp. 390–391.
50. Lin, T.Y.; Dollar, P.; Girshick, R.; He, K.; Hariharan, B.; Belongie, S. Feature Pyramid Networks for Object Detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017.
51. Liu, S.; Qi, L.; Qin, H.; Shi, J.; Jia, J. Path Aggregation Network for Instance Segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake City, UT, USA, 18–23 June 2018.
52. Padilla, R.; Netto, S.L.; da Silva, E.A.B. A Survey on Performance Metrics for Object-Detection Algorithms. In Proceedings of the 2020 International Conference on Systems, Signals and Image Processing (IWSSIP), Niteroi, Brazil, 1–3 July 2020; pp. 237–242. [CrossRef]
53. Two-i. What Are Some Interesting Applications of Object Detection. Available online: https://www.two-i.com/blog/what-are-some-interesting-applications-of-object-detection (accessed on 7 July 2023).
54. Lee, J.; Hwang, K.I. YOLO with adaptive frame control for real-time object detection applications. *Multimed. Tools Appl.* **2021**, *81*, 36375–36396. [CrossRef]
55. LearnopenCV.com. FPS Performance Comparison of YOLO Models. Available online: https://learnopencv.com/performance-comparison-of-yolo-models/#FPS-Performance-Comparison-of-YOLO-Models-on-CPU (accessed on 7 July 2023).
56. KDnuggets.com. Metrics Evaluate Deep Learning Object Detectors. Available online: https://www.kdnuggets.com/2020/08/metrics-evaluate-deep-learning-object-detectors.html (accessed on 11 July 2023).
57. Deng, J.; Dong, W.; Socher, R.; Li, L.J.; Li, K.; Fei-Fei, L. ImageNet: A large-scale hierarchical image database. In Proceedings of the 2009 IEEE Conference on Computer Vision and Pattern Recognition, Miami, FL, USA, 20–25 June 2009; pp. 248–255. [CrossRef]
58. Lin, T.Y.; Maire, M.; Belongie, S.; Hays, J.; Perona, P.; Ramanan, D.; Dollár, P.; Zitnick, C.L. Microsoft coco: Common objects in context. In *Proceedings of the Computer Vision—ECCV 2014: 13th European Conference, Zurich, Switzerland, 6–12 September 2014; Proceedings, Part V 13*; Springer: Berlin/Heidelberg, Germany, 2014; pp. 740–755.
59. Everingham, M.; Van Gool, L.; Williams, C.K.; Winn, J.; Zisserman, A. The pascal visual object classes (voc) challenge. *Int. J. Comput. Vis.* **2010**, *88*, 303–338. [CrossRef]
60. Geiger, A.; Lenz, P.; Urtasun, R. Are we ready for autonomous driving? the kitti vision benchmark suite. In Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition, Providence, RI, USA, 16–21 June 2012; pp. 3354–3361.
61. Xia, G.S.; Bai, X.; Ding, J.; Zhu, Z.; Belongie, S.; Luo, J.; Datcu, M.; Pelillo, M.; Zhang, L. DOTA: A large-scale dataset for object detection in aerial images. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–22 June 2018; pp. 3974–3983.
62. Cordts, M.; Omran, M.; Ramos, S.; Rehfeld, T.; Enzweiler, M.; Benenson, R.; Franke, U.; Roth, S.; Schiele, B. The cityscapes dataset for semantic urban scene understanding. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 3213–3223.
63. Yu, F.; Chen, H.; Wang, X.; Xian, W.; Chen, Y.; Liu, F.; Madhavan, V.; Darrell, T. Bdd100k: A diverse driving video database with scalable annotation tooling. *arXiv* **2018**, arXiv:1805.04687.
64. Liu, C.; Gao, G.; Huang, Z.; Hu, Z.; Liu, Q.; Wang, Y. YOLC: You Only Look Clusters for Tiny Object Detection in Aerial Images. *IEEE Trans. Intell. Transp. Syst.* **2024**, *25*, 13863–13875. [CrossRef]
65. Ren, S. Faster r-cnn: Towards real-time object detection with region proposal networks. *arXiv* **2015**, arXiv:1506.01497. [CrossRef]

66. v7labs. COCO Dataset Guide. Available online: https://www.v7labs.com/blog/coco-dataset-guide (accessed on 7 August 2023).
67. PASCAL. Pascal2. Available online: http://host.robots.ox.ac.uk/pascal/VOC/ (accessed on 7 August 2023).