Universiteit Leiden

Benchmarking Established and Recent Object Detectors for Road Scene Detection

Rapolas Bernotas (s4020545)

Supervisors:
Prof. dr. Michael Lew & Dr. Erwin M. Bakker

Bachelor Thesis– Data Science and Artificial Intelligence

Leiden Institute of Advanced Computer Science (LIACS)
liacs.leidenuniv.nl

August 17, 2026

Abstract

Object detection models are often compared using results reported under different implementation conditions. This thesis presents a controlled, practical benchmark of five publicly available COCO pretrained object detectors for road scene detection: Faster R-CNN ResNet-50-FPN, RetinaNet ResNet-50-FPN, YOLOv8s, YOLOv12s, and RT-DETR-L. All configurations were fine-tuned using the BDD100K dataset.

Performance was evaluated using mAP@50:95, class and size performance, condition evaluation, and computational efficiency measures. In addition, all ten detector pairs were evaluated using Weighted Boxes Fusion (WBF). RT-DETR-L achieved the highest individual mAP@50:95 score of 0.299, whereas YOLOv8s and YOLOv12s achieved similar aggregate accuracy. YOLOv8s was the fastest configuration at 3.45 ms per image and 289.68 FPS. Faster R-CNN outperformed RetinaNet, but both obtained lower accuracy than the three newer configurations. The best WBF pair, YOLOv12s and RT-DETR-L, improved mAP@50:95 to 0.308, but increased latency to 26.88 ms per image.

The findings show that newer practical detector configurations transferred more effectively to the evaluated road scene setting, but that newer models were not universally or decisively superior. Detector selection depends on the required balance between accuracy, recall, small object performance, latency, and resource cost.

Contents

1 Introduction

Object detection is a central computer vision task in which a model must identify object categories and determine their locations within an image. Today, it is used in various applications such as autonomous driving, traffic monitoring, robotics, and video surveillance, where a system may need to recognize relevant objects quickly and reliably [31].

Over the last decade, object detectors have been rapidly improving in this regard. In 2015, Faster R-CNN [21] introduced a highly influential two-stage detection architecture, in which candidate object regions are first generated and subsequently classified and refined. In contrast, one-stage detectors predict object classes and bounding boxes directly from dense image features. In 2017, RetinaNet [11] addressed the class imbalance problem through focal loss, while the YOLO family prioritized efficient real-time inference [20]. More recently, transformer-based detectors such as RT-DETR [30] (introduced in 2024) and attention-centric YOLO variants such as YOLOv12 [26] (introduced in 2025) have reported competitive accuracy-latency trade-offs on standard object detection benchmarks.

The most relevant benchmark for the models considered in this thesis is the Microsoft COCO dataset [12]. The original RT-DETR and YOLOv12 papers report strong COCO results relative to the models selected as their baselines [26, 30]. However, these published results should not be interpreted as evidence that a newer architecture is universally superior. Since their results are obtained under the configurations chosen by the respective authors, direct comparisons between published detector results are therefore often difficult to interpret. Many factors beyond the underlying architecture can influence reported performance. A newer model may be evaluated using a more favorable training recipe or a larger computational budget than an older model. Consequently, differences in reported benchmark scores cannot always be attributed solely to architectural progress. Unified benchmarking frameworks such as MMDetection have emphasized the importance of explicitly controlling and documenting experimental choices when comparing detection methods [3].

This motivates this thesis to investigate how selected established and newer detector configurations compare in performance, particularly after adaptation in the same road scene domain. Five COCO-pretrained detectors are fine-tuned on the BDD100K dataset: Faster R-CNN ResNet-50-FPN, RetinaNet ResNet-50-FPN, YOLOv8s, YOLOv12s, and RT-DETR-L. BDD100K is used particularly for this comparison because it contains a large and diverse collection of driving scenes with annotations for object detection and environmental attributes [29]. Furthermore, the selected models represent several detector families. These include two-stage convolutional detection, one-stage dense detection, modern YOLO-based detection, and transformer-based detection.

The aim is to provide a controlled practical comparison, where all models are fine-tuned on the same downstream dataset split, with the same image resolution, training duration, hardware environment, and evaluation procedure. The configurations use publicly available COCO pretrained weights and their respective implementations. This means that some training details necessarily remain specific to the model, and thus isolating architecture as the sole cause of observed differences is not possible to do conclusively. Nevertheless, this reflects a realistic application, where available pretrained detector configurations are adapted to a different domain such as road scene detection.

Performance is evaluated primarily using mAP@50:95, complemented by mAP@50, mAP@75, precision, recall, class AP, and AP by object size. In addition, this thesis investigates performance in different road scene conditions, including weather, time of day, scene type, occlusion, and truncation. This is interesting because a model that performs well on average may still perform relatively poorly in challenging subsets of data.

Finally, the thesis examines whether selected detectors can benefit when combined through Weighted Boxes Fusion (WBF) [23]. WBF combines overlapping bounding box predictions from multiple models using their confidence scores. The fusion experiments evaluate both the change in detection accuracy and the additional inference cost, since a more accurate ensemble may be unsuitable for applications with strict latency requirements.

The thesis is guided by the following questions:

1. How do established and newer object detection configurations compare after fine-tuning on a common dataset under a controlled training budget?
2. How do the newer detector configurations compare with established architectures in aggregate accuracy, object classes, object sizes, and road scene conditions?
3. What accuracy-efficiency trade-offs exist between the evaluated detector configurations?
4. To what extent can Weighted Boxes Fusion improve performance over the strongest individual model, and what inference overhead does this introduce?

As a result, this thesis provides several contributions. First, it provides a controlled downstream benchmark of five publicly available detector configurations spanning established and newer detectors. Second, it extends the comparison beyond a single accuracy measure by analyzing class, size, and condition performance. Third, it evaluates whether prediction-level fusion can exploit complementary model behavior while accounting for its latency cost.

1.1 Thesis overview

The remainder of this thesis is organized as follows. Section 2 establishes the theoretical foundation by introducing object detection, the principal evaluation metrics, the detectors considered, transfer learning, the BDD100K dataset, and Weighted Boxes Fusion. Section 3 reviews prior research on detector benchmarking, object detection in road scenes, robustness under challenging conditions, and prediction-level ensemble methods. Section 4 describes the complete experimental methodology, including dataset preparation, model training, individual and fused-model evaluation, efficiency measurement, and the average-scoring baseline. Section 5 presents the results for the individual detectors and fusion configurations in terms of accuracy, robustness, efficiency, and accuracy–latency trade-offs. Section 6 interprets the results, compares the relative strengths of the evaluated approaches, and considers the methodological limitations that affect their interpretation. Finally, Section 7 summarizes the main findings and addresses the research objectives. Section 8 outlines directions for future work.

2 Background

2.1 Object detection fundamentals

Object detection is a computer vision task in which objects in an image are identified, and their locations are estimated. Unlike image classification, which assigns one or multiple labels to a whole image, object detection must determine which object instances are present, where each instance is located, and to which class it belongs [31]. For example, a classifier may label an image as containing a car, but an object detector must go a step further to classify not only every object present in the image, such as a car, but also identify their location. Generally, a detector produces a set of predictions for an input image. Each prediction often consists of a bounding box, a predicted class, and a confidence score. A bounding box is usually represented by its corner coordinates or by its center coordinates together with width and height. The confidence score represents the model's estimated certainty that the predicted box contains an object of the assigned class. A traffic scene image may contain multiple instances of the same category, such as several cars or pedestrians, and therefore a detector must output a separate bounding box for each instance (Figure 1).

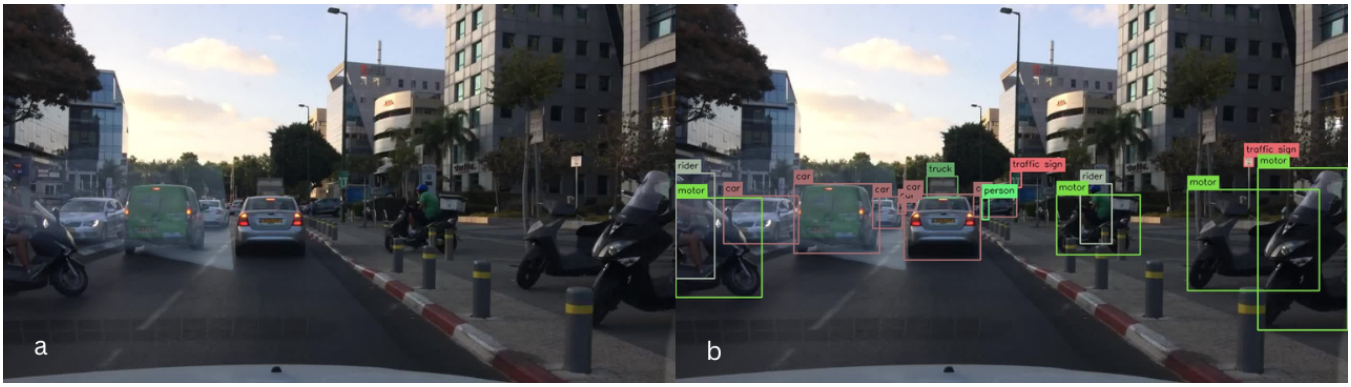

Figure 1: Example BDD100K road scene image before (a) and after (b) adding ground truth bounding box annotations. Image from BDD100K [29]; visualization created by the author.

During training, detectors learn from annotated images containing ground truth bounding boxes and class labels. A detector model is optimized to solve a classification task, which predicts the object category, and a localization task, which predicts the object bounding box position and size. Predicting only the correct label is insufficient if the predicted box is shifted away from the actual object or covers only a fraction of the object. As a result, a useful detector must recognize the object correctly and localize it accurately. Localization accuracy is commonly measured using Intersection over Union (IoU). IoU measures the overlap between a predicted bounding box and its corresponding annotated ground truth box, as shown below:

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

An IoU value of 1 indicates perfect overlap, whereas 0 indicates no overlap. Hence, a prediction is generally considered a true positive when it has the correct class and its IoU with a matched ground truth object exceeds a chosen threshold. The predictions with an incorrect class, insufficient overlap,

or no corresponding object are then counted as false positives. Lastly, ground truth objects for which no sufficiently matching prediction is made are counted as false negatives. In driving datasets such as BDD100K, detecting true positives in object detection becomes more challenging than classification because it must handle variations in object size, viewpoint, illumination, occlusion, and scene density. Objects may be partially occluded by other objects, such as a pedestrian behind a car, or affected by poor weather or nighttime lighting.

Furthermore, many detectors produce several highly overlapping predictions for the same object. Non-Maximum Suppression (NMS) is a post-processing step that is often required to retain the highest-confidence box per object and remove lower-confidence overlapping boxes [15]. Although this reduces duplicate detections, it can also discard useful predictions. Conventional YOLO, Faster R-CNN, and RetinaNet commonly use NMS, whereas RT-DETR is designed as an end-to-end detector that does not require NMS [30].

Different detector families address this task differently. Two-stage detectors, such as Faster R-CNN, first generate candidate object regions and subsequently classify and refine them [21]. In contrast, one-stage detectors, including YOLO models, predict object locations and classes directly from image features in a single detection pipeline [20]. Transformer-based detectors such as RT-DETR use attention mechanisms and object queries to generate detections end-to-end [30]. Nevertheless, although these approaches differ internally, all ultimately produce the same type of output, which is a set of labeled bounding boxes and associated confidence scores.

2.2 Evaluation metrics

The selected object detectors were evaluated across four metric groups. These are overall detection accuracy, detailed detection accuracy, model robustness, and computational efficiency. All accuracy metrics were computed using the same test dataset split images, confidence filtering procedure, and evaluator for each model. Fused models are evaluated using metrics from other groups.

Table 1: Evaluation metrics used in this thesis.

Category	**Metrics**
Overall detection	mAP@50:95, mAP@50, mAP@75, precision, recall, and F1 score
Detailed accuracy	Per-class AP@50:95, AP_{small}, AP_{medium}, and AP_{large}
Robustness	mAP@50:95 by weather, time of day, and scene type, and mAP@50:95 for occluded and truncated object subsets
Efficiency	Training time, time per epoch, best validation epoch, inference latency, FPS, checkpoint size, and peak VRAM usage
Fusion	Fused mAP@50:95 and F1, improvement over the stronger individual constituent, fusion overhead, total pair latency, and comparison with score averaging

2.2.1 Main accuracy metrics

The primary metric is mean Average Precision (AP) at IoU thresholds from 0.50 to 0.95, commonly denoted as mAP@50:95. This is the standard COCO detection metric [12]. It averages Average Precision across ten IoU thresholds:

$$\text{IoU} \in \{0.50, 0.55, 0.60, \ldots, 0.95\} \tag{2}$$

Average Precision summarizes the precision-recall curve for one object class at one IoU threshold. The mAP@50:95 metric ensures that a detector cannot obtain a high score solely by producing approximately correct boxes; it must maintain performance at stricter IoU thresholds. As a result, this metric rewards models that recognize objects correctly and localize them accurately. To complement the analysis, mAP@50 and mAP@75 are also reported. The mAP@50 and mAP@75 metrics consider a detection correct when its IoU with the ground truth object is at least 0.50 and at least 0.75, respectively. Comparing these values helps distinguish models that successfully identify objects from those that also predict tight and accurate bounding boxes. For instance, a model with a strong mAP@50 but weaker mAP@75 may recognize objects reliably while producing less precise localization.

Precision, recall, and F1 were calculated at a confidence threshold of 0.25 and an evaluation IoU threshold of 0.50. Precision measures the proportion of predicted detections that are correct:

$$\text{Precision} = \frac{TP}{TP + FP} \tag{3}$$

where TP is the number of true positive detections, and FP is the number of false positive detections. High precision therefore signifies that the model is good at avoiding identifying and locating objects mistakenly.

Recall measures the proportion of ground truth that are successfully detected:

$$\text{Recall} = \frac{TP}{TP + FN} \tag{4}$$

where FN is the number of false negative detections; therefore, high recall signifies that the model is great at identifying and locating objects. These metrics may reveal whether a model achieves its result through conservative predictions with fewer false positives, or through more permissive predictions that detect more objects but also generate more incorrect boxes.

Lastly, the F1 score summarizes the balance between precision and recall using their harmonic mean. A high F1 score requires high precision and high recall. As a result, it decreases substantially when either metric is low. The F1 score is calculated as:

$$F_1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{5}$$

2.2.2 Detailed accuracy metrics

To get a more detailed look at each model's performance, the AP@50:95 for each class is reported. This is particularly relevant for driving scenes, where strong performance for frequent large class instances such as cars may hide much poorer performance for generally smaller class instances such as traffic signs. The evaluation also reports AP for small (AP_{small}), medium (AP_{medium}), and large (AP_{large}) objects. These metrics assess whether detector performance changes with object scale. Small objects are especially challenging because they contain fewer visible pixels and are more strongly affected by factors such as image resizing, occlusion, and background interference [10]. The size definitions are chosen to match the standard COCO evaluator [12].

$$\text{Small:} \quad A < 1024\,\text{px}^2 \tag{6}$$

$$\text{Medium:} \quad 1024\,\text{px}^2 \leq A < 9216\,\text{px}^2 \tag{7}$$

$$\text{Large:} \quad A \geq 9216\,\text{px}^2 \tag{8}$$

where A is calculated from the ground truth bounding box in the original image coordinate system.

2.2.3 Robustness metrics

To examine whether performance is stable beyond the average case, mAP@50:95 is calculated separately for subsets of images grouped by weather condition, time of day, and scene type [29]. BDD100K images come with scene attributes that make it possible to compare performance, for example, in clear versus rainy conditions, or daytime versus nighttime scenes. However, because subgroups may differ in sample size, object density, class distribution, and difficulty, results are interpreted cautiously. In addition, mAP@50:95 is calculated for occluded and truncated objects. An object is occluded when another object or background element partially hides it. An object is truncated if it extends beyond the image boundary. These cases are important in road scenes because they test whether a detector can recognize incomplete visual evidence.

Image-level subsets, such as weather, time, and scene subsets, are evaluated separately. For object-level subsets, such as occlusion and truncation, only the relevant ground-truth objects are filtered and evaluated.

2.2.4 Computational efficiency metrics

Accuracy is evaluated together with operational cost. Training efficiency is described using the total training time, the average time per epoch, and the epoch selected as the best checkpoint. The best epoch indicates when the selected validation criterion reached its best value during training. However, since models use different batch sizes and framework training pipelines, total training

time should be interpreted as practical resource cost under the experimental setup rather than as a pure architecture-level speed comparison.

Inference efficiency is measured as latency in milliseconds per image and Frames Per Second (FPS), where:

$$\text{FPS} = \frac{1000}{\text{latency in milliseconds per image}} \tag{9}$$

Latency is measured on the same hardware using the same input resolution, inference precision, batch size, warm-up procedure, and number of timed images for every model. The model size is reported using the checkpoint size in megabytes. Lastly, maximum GPU memory usage during training is reported. These measurements help distinguish a model that is accurate but difficult to deploy from one that offers a more favorable accuracy-efficiency trade-off.

2.2.5 Weighted Boxes Fusion evaluation metrics

For each selected Weighted Boxes Fusion (WBF) pair, the fused predictions are evaluated using the same primary and secondary detection metrics as the individual models. The improvement over the stronger constituent is measured separately for mAP@50:95 and F1:

$$\Delta\text{mAP} = \text{mAP}_{\text{WBF}} - \max\left(\text{mAP}_{\text{Model A}}, \text{mAP}_{\text{Model B}}\right), \tag{10}$$

$$\Delta F_1 = F_{1,\text{WBF}} - \max\left(F_{1,\text{Model A}}, F_{1,\text{Model B}}\right). \tag{11}$$

A positive value indicates that fusion improves upon the stronger standalone model for the corresponding metric, whereas a negative value indicates that fusion reduces performance.

However, any improvement must be interpreted in the context of computation cost, since WBF requires running both component detectors and then performing fusion. As a result, total inference latency is expected to be higher than for either individual model. The metric used to approximate the additional cost caused is:

$$\text{Overhead} = \text{Latency}_{\text{pair}} - (\text{Latency}_{\text{A}} + \text{Latency}_{\text{B}}) \tag{12}$$

$\text{Latency}_{\text{pair}}$ means end-to-end latency for running both models and fusion.

2.3 Selected detectors

Most modern object detectors can be grouped into two-stage, one-stage, and transformer-based detectors. These families differ mainly in how image features are converted into final object predictions.

Faster R-CNN is an established two-stage object detector [21]. In the first stage, it extracts feature maps from an image using a convolutional backbone, such as ResNet-50. A Region Proposal

Network (RPN) then generates a set of candidate object regions. In the second stage, these proposals are classified, and their bounding boxes are refined. Historically, it has achieved strong detection accuracy and remains widely used as a baseline in object detection research. However, its sequential proposal and region classification stages increase computational cost compared to some one-stage detectors.

ResNet-50 is a convolutional neural network backbone derived from the Residual Network architecture [6]. ResNet uses skip connections, which allow a layer or group of layers to learn a residual transformation relative to its input. This design reduces vanishing gradients in deep networks, which makes training more effective. In object detection, a ResNet-50 backbone converts an input image into hierarchical feature maps that contain increasingly abstract visual information, while the Feature Pyramid Network (FPN) combines information across feature map scales [10]. The detection architecture then uses these feature maps to predict classes and bounding boxes. Faster R-CNN ResNet-50-FPN combines a ResNet-50 backbone with a Feature Pyramid Network and a two-stage detection head.

One-stage detectors predict object classes and bounding boxes directly from image features, without a separate region proposal stage. This design generally makes them efficient because detection is performed within one pipeline. RetinaNet is an established one-stage detector introduced by Lin et al. [11]. Like Faster R-CNN, it can use a ResNet-50 backbone, but it does not generate and process a separate set of region proposals. Instead, it predicts boxes and classes densely over feature maps at multiple scales. A major challenge for one-stage detection is extreme foreground and background class imbalance since most candidate locations correspond to background rather than real objects. RetinaNet addresses this through focal loss, which reduces the contribution of easy background examples during training and focuses learning more strongly on difficult examples. RetinaNet is therefore a useful model in this thesis because it provides an established one-stage comparison to Faster R-CNN while sharing a similar ResNet-50-FPN feature extraction backbone. Using ResNet-50-FPN configurations for both detectors reduces a source of variation.

The YOLO family is one of the most prominent examples of this approach. YOLO models come in several sizes that offer a different balance between accuracy and computation cost. Its architecture uses convolutional feature extraction and multiscale prediction layers to detect objects of different sizes [27]. YOLOv12s represents a more recent development in the YOLO family. YOLOv12 introduces an attention-centric design intended to incorporate attention mechanisms while retaining real-time detection characteristics [26]. Attention mechanisms allow a model to weigh relationships between distant visual features, which potentially improves its ability to represent complex scenes or distinguish relevant object features from background information. Comparing YOLOv8s and YOLOv12s therefore provides an opportunity to investigate whether this newer architectural direction produces measurable benefits under the shared BDD100K fine-tuning setup.

RT-DETR is a transformer-based real-time object detector proposed by Zhao et al. [30]. Unlike conventional dense one-stage detectors, it uses object queries and attention mechanisms to reason about possible object instances. It was proposed as an end-to-end real-time detector that avoids conventional Non-Maximum Suppression (NMS) during final prediction generation. The RT-DETR architecture combines convolutional image features with a transformer-based encoder and decoder. The encoder is designed to process multiscale features efficiently, while its decoder uses object queries to generate a fixed set of final detections. This differs from YOLO style dense prediction,

where many candidate boxes are predicted across image locations and subsequently filtered.

Table 2: Detector configurations evaluated in this thesis.

Model	**Year**	**Detector family**
Faster R-CNN ResNet-50-FPN	2015	Two-stage detector
RetinaNet ResNet-50-FPN	2017	One-stage detector
YOLOv8s	2023	One-stage YOLO detector
YOLOv12s	2025	Attention-centric YOLO detector
RT-DETR-L	2024	Transformer-based detector

These models cover both established and newer object detection approaches. Faster R-CNN and RetinaNet provide older but still influential proposal-based and dense prediction baselines. YOLOv8s and YOLOv12s represent newer one-stage detector configurations, while RT-DETR-L provides a recent transformer-based alternative.

2.4 Transfer learning and COCO pretraining

Training modern object detectors from randomly initialized weights can require large labeled datasets, substantial computational resources, and long training schedules. Transfer learning provides a practical alternative by initializing a model with weights learned on a source task and subsequently adapting it to a new dataset. This is efficient because the model begins with representations that already encode useful basic patterns such as edges, textures, shapes, object parts, and common spatial relationships. Early layers of visual neural networks often learn relatively general features, whereas later layers become increasingly specialized for the intended task and dataset [28]. Consequently, pretrained weights can provide a useful starting point for other related tasks.

In this study, all detectors are initialized using publicly released weights pretrained on the Microsoft Common Objects in Context (COCO) dataset. COCO is a large-scale benchmark for object detection and related computer vision tasks, containing images of everyday scenes with instance-level object annotations [12]. Because the COCO dataset contains over 330,000 images and includes 80 classes, COCO pretrained models are frequently used as starting points for downstream fine-tuning. However, the BDD100K dataset differs from COCO because it focuses on road scenes and autonomous driving-related conditions. It also differs from COCO in that the image annotations include weather conditions and times of day, which allows this thesis to evaluate the models on a more detailed level. Nevertheless, COCO and BDD100K remain sufficiently related at the visual task level, as both require detecting multiple similar objects in complex real-world images and share many classes.

2.5 BDD100K and road scene detection

BDD100K is a large-scale dataset designed to support research on perception tasks for autonomous driving [29]. It contains 100,000 driving videos collected under various real-world conditions. The dataset also includes annotations for several tasks in addition to object detection, such as semantic and instance segmentation, and object tracking. The detection subset contains images taken from driving scenes recorded in different geographical locations, weather conditions, times of day, and road environments (Figure 2). This diversity makes BDD100K useful for evaluating object detectors beyond simplified or highly controlled visual settings.

Figure 2: Examples of BDD100K road scenes under different conditions: (a) nighttime, (b) daytime, (c) rainy, and (d) foggy. Images from BDD100K [29]; visualization created by the author.

Furthermore, road scene detection comes with several challenges that make it suitable for comparing detector architectures. Objects can occur at highly variable scales, where a nearby truck may occupy a large part of an image, while a pedestrian crossing the road in the distance may cover only a small number of pixels. In addition, objects are frequently partially hidden by other objects such as vehicles, roadside infrastructure, and other scene elements. Lastly, driving images contain changing illumination and visibility conditions, including shadows, rain, and nighttime scenes. These factors can make feature extraction and accurate localization more difficult. For these reasons, BDD100K provides a relevant and challenging setting for comparing older and newer detection architectures. Its road object classes, diverse environmental conditions, and annotations for difficult cases allow the study to examine not only which model has the highest average detection score, but also where models succeed or fail under realistic driving conditions.

2.6 Weighted Boxes Fusion

Many conventional object detectors produce a large number of candidate bounding boxes around the same image. These boxes can differ slightly in location and confidence score, so post-processing is often required to obtain a final set of detections. A common initial post-processing step is confidence thresholding. Predictions with confidence scores below a chosen threshold are removed, since they are considered unlikely to correspond to true objects. The threshold affects the precision-recall trade-off, because a high threshold tends to reduce false positives but can also remove correct low-confidence detections, thus lowering recall.

The most common post-processing method used by conventional object detectors is Non-Maximum Suppression (NMS). NMS retains high-confidence detections and suppresses strongly overlapping alternatives. Overlap is generally measured using IoU. However, NMS can discard useful information. For instance, if two predictions are both plausible but have slightly different bounding box locations, retaining only one removes the localization information provided by the other. Nevertheless, some detectors, such as RT-DETR, are designed to produce a fixed set of end-to-end predictions and thus do not use conventional NMS in their standard inference procedure.

Weighted Boxes Fusion (WBF) is an ensemble post-processing method that combines matching bounding boxes from multiple models into a single fused prediction [23]. WBF calculates a weighted average of their coordinates, where higher confidence predictions contribute more strongly to the final bounding box location. WBF is most useful when models make partially complementary errors. For example, one detector may correctly identify an object but predict an imprecise box, while another may predict a better localized box with lower confidence. If both boxes overlap enough and have the same class, WBF may produce a more accurate fused prediction. Conversely, fusion is unlikely to help when models make highly similar predictions or share the same systematic errors.

3 Related Work

Object detection research has produced a large number of model benchmarks. However, direct comparison between reported results sometimes remains difficult. Chen et al. [3] address this issue through MMDetection, a unified toolbox and benchmark for many object detection approaches. Their work has highlighted the use of consistent implementation and configured training settings when comparing detectors. Similarly, Cai et al. [1] evaluate detectors from multiple families, including Faster R-CNN and DETR, under a shared training schedule on the BigDetection benchmark. These have motivated the controlled elements of this study, such as the use of a common dataset, fixed training budget, common hardware, and a shared evaluation procedure.

Several studies have compared detectors in traffic and autonomous driving contexts. More recently, Chaman et al. [2] benchmarked YOLOv8 through YOLOv12 for autonomous driving tasks. Their study used a unified traffic dataset and common evaluation metrics, including precision, recall, mAP@50, mAP@50:95, inference time, and frames per second. This work is particularly relevant to the comparison between YOLOv8s and YOLOv12s in this thesis, though their scope is restricted to the YOLO family only. The introduction of real-time detection transformers has further complicated comparisons between detector families. For instance, in the RT-DETR paper,

Zhao et al. [30] reported favorable accuracy-latency results relative to earlier YOLO detectors under their COCO evaluation settings. Likewise, Tian et al. [26] introduced YOLOv12 as an attention-centric real-time detector and reported improvements over several contemporary detector configurations, including RT-DETR.

Moreover, recent research has emphasized that detectors should be evaluated under varied image conditions. Mao et al. [13] introduced the COCO-O benchmark that replaced images from the COCO with naturally different domains. Their evaluation of more than 100 detectors showed that strong performance on the conventional benchmark did not necessarily imply robust performance when the input distribution changes. Although COCO-O is not specific to road scenes, it motivates the condition-specific analysis in this thesis.

Finally, this thesis examines prediction-level ensembling through WBF. Solovyev, Wang, and Gabruseva [23] showed that WBF may improve results when component models make complementary localization or detection errors. However, it requires every component model to run and therefore introduces additional post-processing cost. This thesis therefore evaluates both the detection improvement achieved by WBF and its end-to-end latency overhead.

In summary, existing work provides useful comparisons between similar detector families and highlights the need to test the robustness of these models further. Therefore, there remains value in a controlled downstream comparison of established and newer detector configurations. This thesis contributes such a comparison by fine-tuning the five chosen models from public COCO pretrained weights under a shared BDD100K-derived protocol. The analysis is further extended through condition-specific evaluation and selected detector-pair experiments using WBF.

4 Approach

4.1 Research objective and experimental design

The experimental design consists of three main stages. First, each detector is fine-tuned independently using the same dataset partitions, image resolution, maximum epoch budget, hardware, and checkpoint selection criterion. Second, predictions from the selected checkpoints are converted to a common format and evaluated with the same implementation of the COCO-style metrics. Third, all unique detector pairs are combined using WBF. Fusion parameters are selected using the validation set and frozen before evaluation on the test set. The validation partition is used for checkpoint selection and WBF hyperparameter selection. The test partition is used only for the final individual model and fusion evaluations.

4.2 Detector configurations

The use of public pretrained weights reflects a practical transfer setting in which a general-purpose detector is adapted to a domain-specific dataset. However, the checkpoints are not methodologically identical. They were produced using different original training recipes, augmentations, optimization schedules, and software versions. Consequently, the results were interpreted as a comparison of

practical public detector configurations rather than as an isolated comparison of architecture alone.

4.3 Dataset preparation

The experiments use the object detection portion of the BDD100K dataset [29]. Ten object categories are retained:

1. pedestrian;
2. rider;
3. car;
4. truck;
5. bus;
6. train;
7. motorcycle;
8. bicycle;
9. traffic light; and
10. traffic sign.

The BDD100K annotations used in this study were obtained through the Dataset Ninja representation of the dataset, provided in Supervisely format [4]. The source annotation schema uses several category names that differ from the detector output notations. The labels `person`, `bike`, and `motor` had to be mapped to `pedestrian`, `bicycle`, and `motorcycle`, respectively. Other annotation types, such as lane and drivable area annotations, were excluded since the experiment focuses on bounding box object detection.

Bounding boxes are represented as absolute pixel coordinates in $(x_{\min}, y_{\min}, x_{\max}, y_{\max})$ format. Model class index conventions are also normalized. In particular, Faster R-CNN internally uses an explicit background class, where the common evaluation representation uses zero for foreground, resulting in labels from 0 to 9. Predictions from all frameworks are converted to this shared representation before evaluation and fusion.

4.4 Dataset partitions

The official BDD100K detection test labels are not publicly available. Consequently, the available labeled data was divided into three partitions. The official BDD100K validation partition of 10,000 images is retained as the validation set. The original BDD100K training partition of 70,000 images is divided into a training partition of 60,000 images and a test partition of 10,000 images.

Before splitting the original BDD100K training partition, the image filenames were first sorted and then shuffled using Python's seed random number generator. The class and environmental distributions generated by the random split were verified and considered sufficiently representative for the experiment. The random split was generated using seed 0.

4.5 Controlled fine-tuning protocol

All detectors are fine-tuned for a maximum of 100 epochs at an input size of 640 pixels. A common epoch budget ensures that each configuration receives the same maximum number of passes over the training partition. The value of 100 epochs was selected as a compromise between allowing the detectors sufficient opportunity to converge and the available computational budget. Early stopping is disabled to prevent some models from receiving a smaller training budget due to their framework applying a different early stopping rule. It also allows later peaking models to complete the full training schedule. Validation performance is recorded throughout the training, and the checkpoint with the highest validation mAP@50:95 is taken for the final evaluation. Training is conducted on the Leiden University computing cluster using one NVIDIA L40S GPU per training task. Each training task is allocated four CPU cores, 32 GB of RAM, and four data loading worker processes. The same hardware is used for all five detectors. The batch size is selected separately for each detector as the largest value that trains reliably within the available GPU memory. A model-specific batch size is necessary because the five architectures have substantially different memory requirements. The batch size is therefore not treated as a controlled variable. However, its value is reported for all detectors, as it may influence optimization and constitutes a limitation of the comparison.

Settings that cannot be made equivalent across frameworks are not artificially forced to be identical. The Ultralytics training implementation is used for YOLOv8s, YOLOv12s, and RT-DETR-L, whereas the TorchVision training implementation is used for Faster R-CNN and RetinaNet. As a result, optimizer behavior, learning rate schedules, loss functions, preprocessing, and augmentations specific to the architecture follow their corresponding implementation. The batch size, optimizer, initial learning rate, schedule, and selected checkpoint epoch are reported in Table 17. Each detector is trained once. The experiment therefore controls the training budget and data partitions, but does not estimate variation across multiple random training seeds.

4.6 Evaluation preparation

Predictions from the saved checkpoints are exported for the validation and test partitions for common evaluation. Export uses an input size of 640 pixels, a low confidence threshold of 0.001, and a maximum of 300 detections per image. The low threshold preserves low-confidence detections that may contribute to the precision-recall curve and WBF. Each exported image record contains its filename, image dimensions, absolute bounding box coordinates, class labels, and confidence scores. The exported predictions are then evaluated by one framework-independent evaluator.

4.7 Evaluation protocol

4.7.1 Individual detector evaluation

The common evaluator measures mAP@50:95 and other secondary metrics, such as mAP@50, which are used to analyze the models in more detail. To identify hidden complementary strengths, the evaluator measures class AP and AP by object size. Lastly, performance for weather, time of day,

scene type, occlusion, and truncation conditions is also measured.

Detection efficiency is measured separately from accuracy on the same NVIDIA L40S GPU using an input size of 640 and a batch size of 1. Each detector received 100 warm-up inferences followed by 2,000 timed images. The measurement is repeated five times, resulting in 10,000 timed observations per detector. In addition, CUDA synchronization is performed immediately before and after each timed model call to prevent asynchronous GPU execution, producing artificially low latency. Image file reading and decoding are excluded from the timer.

4.7.2 WBF pair evaluation

All ten unique pairs of the five detectors are evaluated, allowing us to examine both combinations of similarly performing detectors and combinations of detectors from different architectural families. Before fusion, each detector's boxes are clipped to the image boundaries and normalized. Both component models receive equal weight, giving a fixed model weight ratio of 1:1. The minimum box confidence threshold supplied to WBF is 0.001, and a maximum of 300 detections are kept from each detector and from the fused output.

The WBF box matching IoU threshold is selected independently for each pair using the validation partition. An initial grid from 0.45 to 0.70 in increments of 0.05 was evaluated. Because many pairs achieved their highest validation AP at the upper limit, the search was extended to a grid from 0.75 to 0.90. For each detector pair, the IoU threshold producing the highest validation mAP@50:95 is selected, and the resulting configuration is frozen. The frozen configuration is then used to evaluate their test performance. The test WBF accuracy is measured using the same common evaluator as the individual detectors.

The efficiency of every WBF pair is measured using the same image size, batch size, warm-up count, timed subset, and repetition count used for the individual detectors. Both component detectors are loaded simultaneously, and peak GPU memory is measured while both models are run sequentially.

4.7.3 Average-scoring baseline

To determine whether WBF provides an advantage over a simpler ensemble, an average-scoring baseline is evaluated on the same ten detector pairs. Both methods combine the confidence from two detectors, but the average-scoring baseline does not average their bounding-box coordinates. As a result, a higher result for WBF would indicate that its complete fusion procedure, including the use of localization agreement between overlapping predictions, is more effective than confidence averaging alone. Overall, this baseline provides a controlled comparison with WBF.

An average-scoring baseline works by matching predictions from the two detectors only when they belong to the same class, and their bounding-box IoU exceeds a selected threshold. For every matched pair, the coordinates of the higher-confidence box are retained, while the two confidence scores are replaced by their arithmetic mean. Unmatched predictions are kept unchanged. The resulting detections are then sorted by confidence, and a maximum of 300 detections per image is retained.

The matching IoU threshold is selected independently for each detector pair using the validation partition and the same search range used for WBF. The threshold producing the highest validation mAP@50:95 is frozen and then applied once to the test partition. Test performance is measured using the same common evaluator as the individual detectors and WBF pairs.

5 Experiments

Table 17 summarizes the concrete training configuration used for each detector.

5.1 Individual detector performance

Table 3 presents the overall performance of the five individual detectors on the test partition. mAP@50:95 is treated as the primary accuracy measure, while mAP@50 and mAP@75 provide additional information about detection and localization performance. Precision, recall, and F1 are reported at the fixed confidence threshold of 0.25 and matching IoU threshold of 0.50.

Table 3: Overall object detection performance on the test set.

Model	**mAP@50:95**	**mAP@50**	**mAP@75**	**Precision**	**Recall**	**F1**
YOLOv8s	0.285	0.510	0.269	0.747	0.675	0.709
YOLOv12s	0.287	0.508	0.271	**0.759**	0.669	**0.711**
RT-DETR-L	**0.299**	**0.551**	**0.273**	0.509	**0.809**	0.625
Faster R-CNN	0.195	0.385	0.170	0.371	0.664	0.476
RetinaNet	0.105	0.195	0.098	0.183	0.545	0.275

RT-DETR-L achieved the highest mAP@50:95, mAP@50, and mAP@75, with values of 0.299, 0.551, and 0.273, respectively. It also achieved the highest recall, at 0.809. YOLOv12s achieved the highest precision and F1 score, at 0.759 and 0.711, respectively. YOLOv8s and YOLOv12s performed very similarly in overall AP, whereas Faster R-CNN and RetinaNet achieved substantially lower mAP@50:95.

Class AP was examined to determine whether the overall ranking was consistent across the ten object categories. The complete per-class results are provided in Table 20. RT-DETR-L obtained the highest AP for most classes, while YOLOv12s achieved the highest AP for the car and truck classes.

Table 4: Detection performance by object size on the test set.

Model	$\mathbf{AP_{small}}$	$\mathbf{AP_{medium}}$	$\mathbf{AP_{large}}$
YOLOv8s	0.103	0.337	**0.546**
YOLOv12s	0.104	0.337	0.544
RT-DETR-L	**0.128**	**0.341**	0.524
Faster R-CNN	0.053	0.219	0.445
RetinaNet	0.016	0.122	0.296

All detectors performed substantially better on large objects than on small objects. Most interestingly, RT-DETR-L achieved the highest AP for small objects by a significant margin. It also achieved the highest AP for medium objects, whereas YOLOv8s achieved the highest AP for large objects.

Detector performance was further evaluated across weather, time of day, and scene categories. Since several categories contain comparatively few images, the subgroup results are interpreted together with their sample counts. Complete absolute AP tables and subgroup counts are provided in Appendix E.

To make changes across conditions easier to compare between detectors, the absolute difference between the model's overall result and condition is defined as

$$\Delta AP_c = AP_c - AP_{\text{overall}},$$

where AP_c denotes mAP@50:95 for condition c. Therefore, a negative value indicates that performance in the condition was below the model's overall result, whereas a positive value indicates performance was above the overall result.

Table 5: Change in mAP@50:95 under different weather conditions relative to each model's overall test performance.

Model	Clear	Foggy	Overcast	Partly Cloudy	Rainy	Snowy
YOLOv8s	-0.009	-0.002	+0.003	+0.022	+0.003	-0.025
YOLOv12s	-0.008	-0.045	+0.003	+0.015	+0.009	-0.020
RT-DETR-L	-0.011	-0.057	+0.012	+0.024	-0.018	-0.009
Faster R-CNN	-0.007	-0.033	-0.002	+0.012	+0.005	-0.016
RetinaNet	-0.006	+0.018	-0.001	+0.010	+0.009	-0.002

Table 6: Change in mAP@50:95 across times of day relative to each model's overall test performance.

Model	Dawn/Dusk	Daytime	Night
YOLOv8s	-0.002	+0.008	-0.016
YOLOv12s	+0.005	+0.008	-0.016
RT-DETR-L	+0.002	+0.015	-0.031
Faster R-CNN	-0.005	+0.004	-0.009
RetinaNet	-0.002	+0.006	-0.007

Table 7: Change in mAP@50:95 across driving scene types relative to each model's overall test performance.

Model	City Street	Gas Station	Highway	Parking Lot	Residential	Tunnel
YOLOv8s	+0.004	+0.050	-0.046	+0.025	+0.036	+0.123
YOLOv12s	+0.003	+0.041	-0.043	+0.024	+0.039	+0.066
RT-DETR-L	+0.001	+0.057	-0.026	+0.038	+0.042	+0.122
Faster R-CNN	+0.002	+0.044	-0.021	+0.023	+0.027	+0.048
RetinaNet	+0.005	+0.051	-0.024	+0.056	+0.024	+0.092

The time-of-day results show that nighttime performance was below the overall result for all detector. The largest nighttime decrease was observed for RT-DETR-L, although it remained one of the strongest models in absolute terms. Daytime performance was above the overall result for all five models. Highway scenes also produced lower AP than each model's overall result. Conversely, tunnel scenes produced higher AP.

Table 8: AP@50:95 for occluded and truncated objects.

Model	$\mathbf{AP}_{\mathbf{occluded}}$	$\mathbf{AP}_{\mathbf{truncated}}$
YOLOv8s	0.182	0.289
YOLOv12s	0.183	**0.301**
RT-DETR-L	**0.194**	0.297
Faster R-CNN	0.111	0.201
RetinaNet	0.042	0.119

Occlusion and truncation results are summarized in Table 8. RT-DETR-L achieved the highest conditional AP for occluded objects, while YOLOv12s achieved the highest AP for truncated objects.

Table 9 compares the training cost with the inference efficiency. YOLOv8s required the shortest training time and achieved the lowest measured latency, with 3.45 ms per image and approximately 289.68 frames per second. YOLOv12s was slower than YOLOv8s but had the smallest checkpoint file. RT-DETR-L achieved the highest detection accuracy, but required the longest training time, and had the highest inference latency among the detectors. Faster R-CNN and RetinaNet had substantially larger checkpoint files than the other configurations. Faster R-CNN also had the highest measured peak GPU memory usage, being almost twice as much as the second highest.

Table 9: Training cost and inference efficiency.

Model	Training time (h)	Time/epoch (min)	Best epoch	Latency (ms/image)	FPS	Model size (MB)	Peak VRAM (GB)
YOLOv8s	**12.62**	**7.57**	85	**3.45**	**289.68**	21.47	**0.096**
YOLOv12s	16.84	10.10	93	5.75	173.83	**18.04**	0.150
RT-DETR-L	109.97	65.98	100	9.60	104.20	63.19	0.346
Faster R-CNN	29.83	17.90	14	7.56	132.23	315.10	0.777
RetinaNet	23.08	13.85	83	6.71	149.05	246.35	0.385

5.2 WBF performance

Table 10 presents the final test accuracy of all ten WBF pairs. The table reports WBF mAP@50:95, the stronger constituent's mAP@50:95, and the improvement relative to the stronger constituent model. Table 28 reports the selected threshold for each WBF pair.

Table 10: WBF mAP@50:95 performance relative to the stronger individual constituent.

Model pair	mAP@50:95	Stronger constituent mAP@50:95	ΔmAP@50:95
YOLOv12s + RT-DETR-L	**0.3077**	0.2992	+0.0085
YOLOv8s + RT-DETR-L	0.3065	0.2992	+0.0073
YOLOv8s + YOLOv12s	0.3000	0.2869	+0.0131
RT-DETR-L + Faster R-CNN	0.2702	0.2992	-0.0290
YOLOv8s + Faster R-CNN	0.2574	0.2853	-0.0278
YOLOv12s + Faster R-CNN	0.2584	0.2869	-0.0286
RT-DETR-L + RetinaNet	0.2567	0.2992	-0.0425
YOLOv8s + RetinaNet	0.2229	0.2853	-0.0624
YOLOv12s + RetinaNet	0.2222	0.2869	-0.0647
Faster R-CNN + RetinaNet	0.1862	0.1950	-0.0089

Only the three combinations, formed from YOLOv8s, YOLOv12s, and RT-DETR-L, improved over their stronger constituent. YOLOv12s combined with RT-DETR-L achieved the highest mAP@50:95 of 0.3077, improving upon the strongest individual detector by 0.0085. YOLOv8s combined with RT-DETR-L followed closely at 0.3065. The YOLOv8s and YOLOv12s pair produced the largest improvement over its stronger constituent, at 0.0131, although its gain over the globally strongest individual detector was only 0.0008. All pairs containing Faster R-CNN or RetinaNet reduced the accuracy of their stronger constituent.

Table 11 shows that four of the ten WBF pairs improved upon the F1 score of their stronger constituent. The YOLOv8s and YOLOv12s pair achieved the highest overall F1 score of 0.7170, representing an improvement of 0.0056. The two YOLO and RT-DETR-L pairs also produced small positive gains. Faster R-CNN combined with RetinaNet showed the largest F1 increase, at 0.0077, although its absolute F1 remained comparatively low at 0.4837. The remaining six pairs reduced F1, with the largest decreases observed for the YOLO and RetinaNet pairs.

Table 12 reports the end-to-end latency of each fusion pipeline. The measurement includes sequential execution of both detectors, conversion and transfer of their outputs, WBF, and sorting

Table 11: WBF F1 performance relative to the stronger individual constituent.

Model pair	WBF F1	Stronger constituent F1	ΔF1
YOLOv12s + RT-DETR-L	0.7159	0.7115	+0.0045
YOLOv8s + RT-DETR-L	0.7132	0.7092	+0.0039
YOLOv8s + YOLOv12s	**0.7170**	0.7115	+0.0056
RT-DETR-L + Faster R-CNN	0.5734	0.6251	-0.0518
YOLOv8s + Faster R-CNN	0.5798	0.7092	-0.1295
YOLOv12s + Faster R-CNN	0.5799	0.7115	-0.1315
RT-DETR-L + RetinaNet	0.5874	0.6251	-0.0377
YOLOv8s + RetinaNet	0.5615	0.7092	-0.1477
YOLOv12s + RetinaNet	0.5588	0.7115	-0.1527
Faster R-CNN + RetinaNet	0.4837	0.4760	+0.0077

of the fused detections.

Table 12: Latency includes both detector calls, output conversion and transfer, WBF, and fused output sorting. Image decoding is excluded.

Model pair	Mean latency ± SD (ms)	FPS	Pipeline overhead (ms)	Peak VRAM (GB)
YOLOv12s + RT-DETR-L	26.88 ± 0.09	37.20	11.53	0.272
YOLOv8s + RT-DETR-L	24.94 ± 0.03	40.09	11.90	0.280
YOLOv8s + YOLOv12s	18.13 ± 0.05	55.17	8.92	**0.133**
RT-DETR-L + Faster R-CNN	27.21 ± 0.45	36.75	10.05	0.436
YOLOv8s + Faster R-CNN	**17.77 ± 0.33**	**56.26**	**6.76**	0.359
YOLOv12s + Faster R-CNN	20.56 ± 0.04	48.64	7.25	0.350
RT-DETR-L + RetinaNet	28.95 ± 0.08	34.54	12.65	0.372
YOLOv8s + RetinaNet	19.81 ± 0.06	50.48	9.65	0.261
YOLOv12s + RetinaNet	22.77 ± 0.28	43.92	10.31	0.254
Faster R-CNN + RetinaNet	21.72 ± 0.04	46.04	7.45	0.439

All fusion configurations were substantially slower than their individual components. The YOLOv8s and YOLOv12s pair was among the fastest fusion configurations and had the smallest combined checkpoint size. Pairs involving RT-DETR-L generally had higher latency, while combinations involving Faster R-CNN or RetinaNet had substantially larger combined checkpoint sizes.

Figure 3 compares accuracy and latency, where points closer to the upper left represent more favorable trade-offs. The Pareto frontier contains configurations for which no other evaluated option is both faster and more accurate. YOLOv8s, YOLOv12s, and RT-DETR-L are at the frontier along with three WBF pairs. Among the WBF pairs, the YOLOv8s and YOLOv12s pair provided the lowest latency, whereas the YOLOv12s and RT-DETR-L pair achieved the highest accuracy. All remaining WBF pairs were dominated by another configuration by both higher accuracy and lower latency.

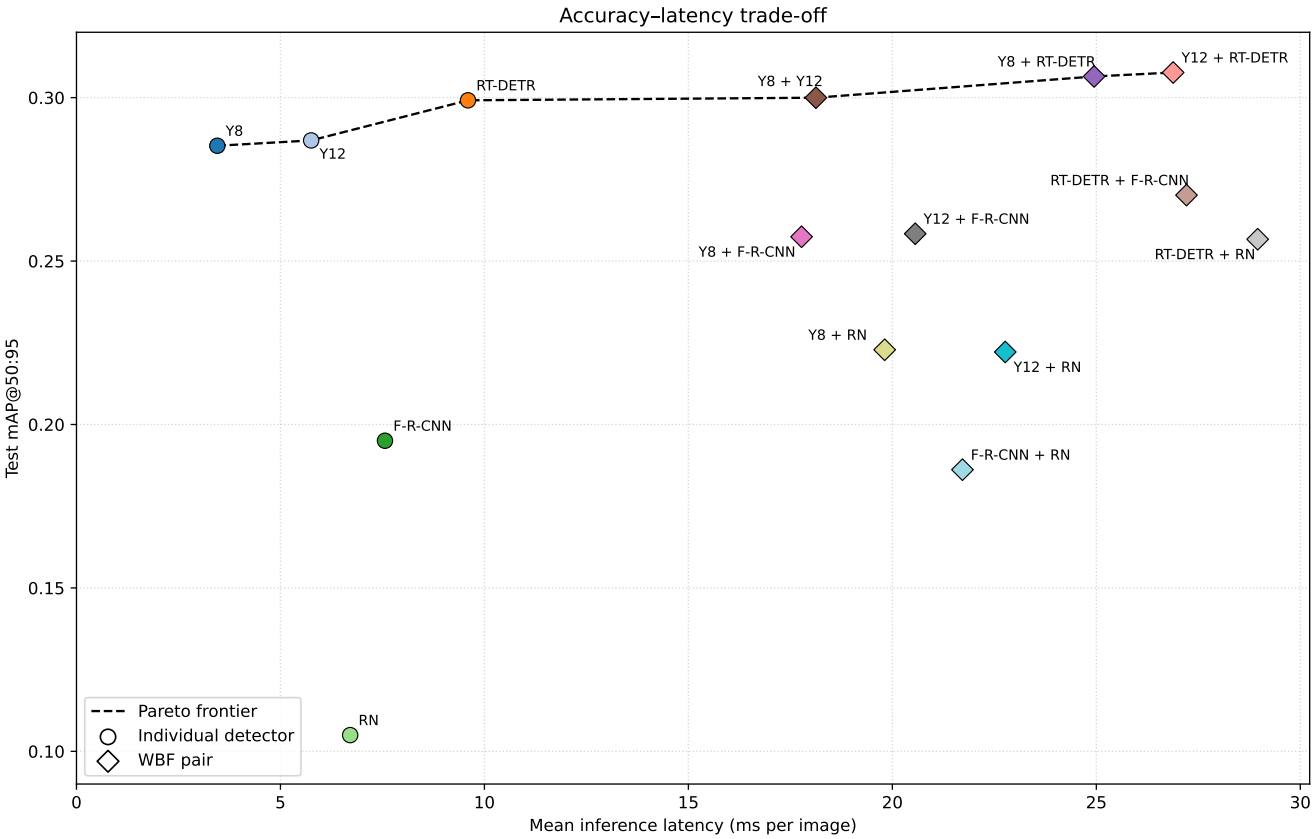

Figure 3: Accuracy-latency trade-off for the individual detectors and WBF configurations. Higher AP and lower latency are preferred.

Table 13: Test mAP@50:95 of the score-averaging baseline and WBF.

Model pair	**Score-averaging mAP**	**WBF mAP**
YOLOv12s + RT-DETR-L	0.2909	**0.3077**
YOLOv8s + RT-DETR-L	0.2916	0.3065
YOLOv8s + YOLOv12s	**0.2953**	0.3000
RT-DETR-L + Faster R-CNN	0.2596	0.2702
YOLOv12s + Faster R-CNN	0.2275	0.2584
YOLOv8s + Faster R-CNN	0.2276	0.2574
RT-DETR-L + RetinaNet	0.2405	0.2567
YOLOv8s + RetinaNet	0.1834	0.2229
YOLOv12s + RetinaNet	0.1825	0.2222
Faster R-CNN + RetinaNet	0.1610	0.1862

Table 13 shows that WBF outperformed the score-averaging baseline for all ten detector pairs. The smallest difference was observed for the YOLOv8s and YOLOv12s pair, where WBF improved mAP from 0.2953 to 0.3000. The largest differences occurred for the YOLO and RetinaNet pairs.

Overall, these results indicate that averaging both the localization and confidence information through WBF was more effective than averaging confidence scores while retaining one original bounding box. Table 29 reports the selected threshold for each score-averaging baseline pair.

6 Discussion

The results show a clear advantage for RT-DETR-L, YOLOv12s, and YOLOv8s configurations. RT-DETR-L achieved the highest overall individual mAP@50:95, while YOLOv8s and YOLOv12s followed closely. Faster R-CNN and RetinaNet achieved substantially lower accuracy under the same maximum epoch budget and dataset. As a result, newer configurations appear to have transferred more effectively to BDD100K within our experimental setup.

RT-DETR-L provided the strongest overall AP and led under most evaluated conditions. Its main distinguishing characteristics were high recall and stronger small-object detection. The image resolution was reduced to increase training and inference speed [19], resulting in an approximate 75% reduction in the number of content pixels in the image. Small objects already contain fewer discriminative visual features and are therefore particularly sensitive to resolution reduction [22]. RT-DETR-L's advantage under these conditions suggests a useful relative strength in recovering small objects among the evaluated configurations. This may be related to its use of multi-scale feature processing and query-based prediction, although this study cannot determine which component caused the observed advantage.

The YOLO configurations produced slightly lower aggregate AP but substantially higher precision and F1. This indicates that their predictions provided a stronger balance between false positives and false negatives. Their higher precision may partly reflect differences in confidence assignment and post-processing, which can make the models more conservative about which detections are retained. The YOLO models also performed slightly better on large objects, while YOLOv12s obtained the strongest truncated-object result. However, YOLOv8s and YOLOv12s were very similar in most categories, suggesting that YOLOv12s provided an incremental rather than transformative improvement in this experimental setting. As a result, the additional architectural changes in YOLOv12s may have improved its prediction quality. Still, they did not produce a large aggregate AP advantage under the selected resolution and training budget.

Faster R-CNN consistently outperformed RetinaNet, but both remained below the three newer configurations under most conditions. Nonetheless, the model ranking was broadly stable across the detailed evaluations, although the magnitude of the differences varied by condition.

When considering the trade-off between detection accuracy and computational efficiency, YOLOv8s showed the strongest overall balance in this experiment. It had substantially lower latency, training cost, memory use, and model size than RT-DETR-L. Although RT-DETR-L achieved the highest accuracy, this gain was accompanied by considerably greater inference cost. The accuracy-efficiency trade-off is affected by input resolution [25], and research on DETR-style detectors has shown that input resolution, among other factors, can change their efficiency-accuracy balance [24]. At the original image resolution, the detectors could obtain different accuracy improvements and substantially different latency and memory requirements. As a result, the relative balance between YOLOv8s, YOLOv12s, and RT-DETR-L could differ under full-resolution evaluation.

YOLOv12s occupied an intermediate position, offering a slight improvement in accuracy and precision over YOLOv8s at a noticeable latency cost. Consequently, the preferred detector depends on the deployment objective. YOLOv8s is the strongest option when real-time throughput and resource efficiency are prioritized, whereas RT-DETR-L is preferred when recovering additional objects, particularly small and occluded, justifies the higher computational cost. In the end, YOLOv12s did not substantially separate itself from YOLOv8s in aggregate AP and instead appears to be an intermediate accuracy-latency option.

WBF produced a modest improvement over the strongest individual detector. The strongest fused configuration, YOLOv12s combined with RT-DETR-L, achieved an absolute improvement of 0.0085 AP, or approximately 2.8% relative to RT-DETR-L. YOLOv8s combined with RT-DETR-L achieved a similar improvement of 0.0073, while the YOLOv8s and YOLOv12s pair exceeded RT-DETR-L by only 0.0008. The remaining seven pairs did not outperform the strongest individual detector. WBF therefore improved performance only to a limited extent and only when combining the three strongest individual configurations.

The success of these three pairs may be related to the quality and complementarity of their constituent predictions. RT-DETR-L produced high recall and stronger small-object performance, whereas the YOLO models produced higher precision and F1. Fusion may therefore have retained some additional detections from RT-DETR-L while benefiting from the more precise predictions of the YOLO configurations. However, this interpretation remains a hypothesis and future research should measure the exact false-positive and false-negative overlap between the detectors.

The YOLOv8s and YOLOv12s pair achieved the highest overall F1 score of 0.7170, improving upon its stronger constituent by 0.0056. As a result, the pair produced the most favorable precision-recall balance, though it did not achieve the highest AP. Four of the ten WBF pairs improved on the F1 score, whereas the remaining six pairs reduced the F1 score of their stronger constituent, with the largest decreases observed for the YOLO and RetinaNet pair. These results again show that fusion with a weaker detector can negatively change the confidence distribution and the precision-recall balance of the stronger model.

The score-averaging experiment provides additional evidence that the WBF itself was useful. WBF outperformed the score-averaging baseline for all ten detector pairs. The score-averaging baseline used information from both confidence scores but retained the coordinates of one original prediction. WBF additionally combined the localization estimates of overlapping detections. Its consistent advantage therefore suggests that using localization agreement, rather than merely averaging confidence values, improved the fused predictions.

Previous work has shown that modifying how overlapping detector predictions are combined or suppressed can improve detection AP without retraining the underlying models [8]. However, such improvements are not guaranteed when the constituent models do not provide sufficiently accurate or complementary predictions. One possible explanation for the weaker WBF pairs is that confidence scores from independently trained detectors may not be equally calibrated. Confidence scores can vary with localization quality, object scale, and image position, which affects post-processing methods that use these values to weight or rank detections [9]. Since the WBF experiment used equal model weights, a less accurate or overconfident detector may have received excessive influence over the fused coordinates or confidence score. Different model weights or confidence calibration could potentially reduce this effect.

Furthermore, the accuracy improvement introduced considerable inference cost. The YOLOv12s and RT-DETR-L pair required 26.88 ms per image, compared with 9.60 ms for RT-DETR-L alone. This represents an additional 17.28 ms per image and approximately 2.8 times the latency of the individual RT-DETR-L configuration. Throughput consequently decreased from 104.20 to 37.20 FPS. Overall, though WBF increased the best mAP@50:95 by 0.0085, this gain required a substantial reduction in inference efficiency. Its practical value therefore depends on whether the modest accuracy improvement justifies the additional end-to-end latency per image.

6.1 Limitations

Several limitations affect the interpretation of the results. For instance, each model was trained only once. This is undesirable since studies of neural network evaluation have shown that uncontrolled sources of randomness can produce variation and affect the reproducibility of reported comparisons [7, 14]. This means random initialization, data order, or non-deterministic GPU operations may have produced differences in our results that were not accounted for in the experimental setup.

In addition, the training batch size differed between models due to GPU memory constraints. Under a fixed epoch budget, different batch sizes result in different numbers of parameter updates and alter the optimization conditions. Large mini-batch object detection research has shown that changing batch size may require adjustments to the learning rate to maintain effective training [17]. Learning rates were selected specifically for the models and feasible batch sizes to compensate for these effects; however, this likely did not fully remove them.

Another limitation is that the detectors were initialized using public pretrained checkpoints. Previous research has shown that pretraining and the method of pretraining can affect detector convergence and final performance [5]. Consequently, the models may not have started from equally transferable representations. Therefore, the observed performance differences cannot be attributed solely to the detector architectures, as they may also reflect differences in checkpoint quality and pretraining procedure.

7 Conclusion

This thesis investigated whether newer object detection configurations consistently outperform established detector architectures when fine-tuned and evaluated under a shared road scene protocol. Five publicly available COCO-pretrained configurations were adapted to a BDD100K-derived dataset. In their evaluation, this study reported aggregate detection accuracy, performance by class and size, performance under road scene conditions, computational efficiency, and prediction-level fusion through Weighted Boxes Fusion.

Within the experimental protocol, the newer detector configurations achieved the strongest overall results. RT-DETR-L obtained the highest mAP@50:95 score of 0.299, as well as the highest mAP@50, mAP@75, and recall. It also performed particularly well for small and occluded objects. YOLOv8s and YOLOv12s followed closely in aggregate accuracy, with mAP@50:95 scores of 0.285 and 0.287, respectively. Faster R-CNN outperformed RetinaNet, but both configurations obtained

substantially lower aggregate accuracy than the three newer configurations.

The results of YOLOv8s and YOLOv12s were highly similar in most analyses. YOLOv12s slightly outperformed YOLOv8s in aggregate accuracy, precision, and truncated object performance. As a result, YOLOv12s showed an incremental rather than transformative improvement over YOLOv8s in accuracy. Furthermore, the efficiency analyses demonstrated a clear accuracy-latency trade-off. YOLOv8s was the fastest individual detector, requiring 3.45 ms per image and achieving approximately 289.68 FPS. RT-DETR-L achieved the highest detection accuracy, but required 9.60 ms per image and substantially more training time.

Consequently, YOLOv8s is the most suitable configuration when real-time throughput, low latency, and lower resource consumption are the main priorities. RT-DETR-L is preferable when higher detection accuracy, recall, and small object performance justify greater computational cost.

Weighted Boxes Fusion produced modest improvements only for combinations of the three strongest individual detectors. The best fusion configuration, the YOLOv12s and RT-DETR-L pair, achieved a final mAP@50:95 of 0.3077 and a latency of 26.88 ms per image. The remaining fusion pairs, particularly those involving Faster R-CNN or RetinaNet, did not improve upon their strongest component model. The results show that WBF can exploit complementary predictions. However, its benefit heavily depends on the strength of the selected models and therefore may be too small to justify the additional inference cost in strict real-time applications.

Overall, this thesis shows that recent detector configurations transferred more effectively to the evaluated BDD100K road-scene setting than the selected established Faster R-CNN and RetinaNet configurations. However, the results also show that the newer models are not necessarily decisively better, as the older YOLOv8s remained highly competitive with a more recent YOLOv12s. Lastly, WBF provided a small additional accuracy gain for selected strong model pairs, but a considerable latency penalty accompanied this improvement.

8 Further Research

Future research could further investigate how model initialization and training configuration affect the comparison. A controlled experiment could compare the public pretrained checkpoints used in this thesis with detectors trained from random initialization. This could be accompanied by hyperparameter optimization and longer training schedules.

The benchmark could also be extended to include more detector architectures and model sizes. Evaluating several variants from each detector family would help determine whether the observed ranking reflects general architectural characteristics or the particular configurations selected. Training and evaluating the models at multiple input resolutions, including the original image resolution, would also clarify how the accuracy-latency trade-off changes when more spatial information is retained. This would be especially relevant for the small objects that are common in road scenes.

Furthermore, the WBF experiments could be extended to optimize unequal model weights on the validation partitions. Confidence calibration could also be investigated before fusion. Previous work has found that differences in detector calibration can cause an overly confident model to

dominate an ensemble even when its predictions are not the most accurate [16]. Calibration-aware fusion [18], learned weighting mechanisms, and alternative ensemble methods could therefore be compared with standard WBF. These experiments could determine whether the weaker detector pairs in this thesis suffered from limited complementary information, lack of confidence calibration, or the use of fixed fusion parameters.

References

[1] L. Cai, Z. Zhang, Y. Zhu, L. Zhang, M. Li, and X. Xue, "BigDetection: A large-scale benchmark for improved object detector pre-training," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2022, pp. 4777–4787.

[2] M. Chaman, A. El Maliki, H. Dahou, and A. Hadjoudja, "Benchmarking YOLO-based deep learning models for real-time object detection in hybrid ADAS and intelligent transportation systems," *Results in Engineering*, vol. 29, p. 108942, 2026.

[3] K. Chen, J. Wang, J. Pang *et al.*, "MMDetection: Open MMLab detection toolbox and benchmark," *arXiv preprint arXiv:1906.07155*, 2019.

[4] Dataset Ninja, "Visualization tools for bdd100k: Images 100k dataset," https://datasetninja.com/bdd100k, 2026, accessed: 2026-08-06.

[5] K. He, R. Girshick, and P. Dollar, "Rethinking ImageNet pre-training," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 4918–4927.

[6] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[7] Y. Ji, D. Kaestner, O. Wirth, and C. Wressnegger, "Randomness is the root of all evil: More reliable evaluation of deep active learning," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2023, pp. 3943–3952.

[8] Y. Jiang and J. Ma, "Combination features and models for human detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 240–248.

[9] F. Kuppers, J. Kronenberger, A. Shantia, and A. Haselhoff, "Multivariate confidence calibration for object detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2020, pp. 326–327.

[10] T.-Y. Lin, P. Dollar, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 2117–2125.

[11] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar, "Focal loss for dense object detection," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 2980–2988.

[12] T.-Y. Lin, M. Maire, S. Belongie *et al.*, "Microsoft COCO: Common objects in context," in *European Conference on Computer Vision*, 2014, pp. 740–755.

[13] X. Mao, Y. Chen, Y. Zhu *et al.*, "COCO-O: A benchmark for object detectors under natural distribution shifts," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 6339–6350.

[14] P. Munjal, N. Hayat, M. Hayat, J. Sourati, and S. Khan, "Towards robust and reproducible active learning using neural networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 223–232.

[15] A. Neubeck and L. Van Gool, "Efficient non-maximum suppression," in *Proceedings of the 18th International Conference on Pattern Recognition*, vol. 3, 2006, pp. 850–855.

[16] K. Oksuz, S. Kuzucu, T. Joy, and P. K. Dokania, "MoCaE: Mixture of calibrated experts significantly improves object detection," *Transactions on Machine Learning Research*, 2024.

[17] C. Peng, T. Xiao, Z. Li, Y. Jiang, X. Zhang, K. Jia, G. Yu, and J. Sun, "MegDet: A large mini-batch object detector," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 6181–6189.

[18] T. Popordanoska, A. Tiulpin, and M. B. Blaschko, "Beyond classification: Definition and density-based estimation of calibration in object detection," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2024, pp. 585–594.

[19] L. Qi, J. Kuen, J. Gu, Z. Lin, Y. Wang, Y. Chen, Y. Li, and J. Jia, "Multi-scale aligned distillation for low-resolution detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 14 443–14 453.

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

[21] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Advances in Neural Information Processing Systems*, vol. 28, 2015.

[22] B. Singh and L. S. Davis, "An analysis of scale invariance in object detection—SNIP," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 3578–3587.

[23] R. Solovyev, W. Wang, and T. Gabruseva, "Weighted boxes fusion: Ensembling boxes from different object detection models," *Image and Vision Computing*, vol. 107, p. 104117, 2021.

[24] Y. Suh, D. Kim, A. Aich, S. Schulter, J.-C. Su, B. Han, and M. Chandraker, "Improving the efficiency-accuracy trade-off of DETR-style models in practice," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2024, pp. 8027–8031.

[25] M. Tan, R. Pang, and Q. V. Le, "EfficientDet: Scalable and efficient object detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 10 781–10 790.

[26] Y. Tian, Q. Ye, and D. Doermann, "YOLOv12: Attention-centric real-time object detectors," *arXiv preprint arXiv:2502.12524*, 2025.

[27] Ultralytics, "YOLOv8 documentation," https://docs.ultralytics.com/models/yolov8/, 2023, accessed: 2026-08-02.

[28] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?" in *Advances in Neural Information Processing Systems*, vol. 27, 2014.

[29] F. Yu, H. Chen, X. Wang *et al.*, "BDD100K: A diverse driving dataset for heterogeneous multitask learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 2636–2645.

[30] Y. Zhao, W. Lv, S. Xu *et al.*, "DETRs beat YOLOs on real-time object detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 16 965–16 974.

[31] Z. Zou, Z. Shi, Y. Guo, and J. Ye, "Object detection in 20 years: A survey," *Proceedings of the IEEE*, vol. 111, no. 3, pp. 257–276, 2023.

A Data split and preprocessing

Table 14: Dataset partitions used in the experiments.

Partition	**Images**	**Source**	**Purpose**
Training	60,000	Original BDD100K training set	Model training
Validation	10,000	Official BDD100K validation set	Model and fusion selection
Test	10,000	Original BDD100K training set	Final evaluation

B Experimental environment

Table 15: Software and hardware environment used for model training and evaluation.

Component	Configuration
Execution environment	Linux-based high-performance computing cluster managed with SLURM
GPU	NVIDIA L40S
Python	3.11.15
PyTorch	2.12.1+cu126
CUDA	12.6
Ultralytics	8.4.82
TorchVision	0.26.0+cu128

C Pretraining and training configurations

Table 16: Pretrained initialization of the evaluated detector configurations.

Detector	Implementation	Pretrained weights
YOLOv8s	Ultralytics	`yolov8s.pt`
YOLOv12s	Ultralytics	`yolo12s.pt`
RT-DETR-L	Ultralytics	`rtdetr-l.pt`
Faster R-CNN ResNet-50-FPN	TorchVision	`FasterRCNN_ResNet50_FPN_Weights.COCO_V1`
RetinaNet ResNet-50-FPN	TorchVision	`RetinaNet_ResNet50_FPN_Weights.COCO_V1`

Table 17: Training configurations used for the five detectors.

Model	Batch size	Optimizer	Initial learning rate	Learning-rate schedule	Selected epoch
YOLOv8s	32	SGD	1.0×10^{-2}	Linear decay to 1% with 3-epoch warm-up	85
YOLOv12s	16	SGD	1.0×10^{-2}	Linear decay to 1% with 3-epoch warm-up	93
RT-DETR-L	8	AdamW	1.0×10^{-4}	Linear decay to 1% with 3-epoch warm-up	100
Faster R-CNN	24	SGD	2.5×10^{-3}	StepLR: $\times 0.1$ every 30 epochs	14
RetinaNet	16	SGD	1.0×10^{-3}	StepLR: $\times 0.1$ every 30 epochs	83

Table 18: Training settings shared across the evaluated detector configurations.

Setting	**Configuration**
Training images	60,000
Validation images	10,000
Input resolution	640×640 pixels
Maximum training epochs	100
Early stopping	Disabled
Initialization	COCO-pretrained weights
Number of target classes	10
Checkpoint-selection metric	Validation mAP@50:95
Final checkpoint selection	Epoch with the highest validation mAP@50:95
Number of training runs	One run per detector configuration

D Evaluation configuration

Table 19: Common evaluation and prediction-export settings.

Setting	**Value**
Input resolution	640×640 pixels
Primary metric	mAP@50:95
Maximum evaluator detections	100 per image
Prediction-export confidence threshold	0.001
Maximum exported detections	300 per image
Precision/recall confidence threshold	0.25
Precision/recall matching IoU	0.50
YOLO NMS IoU threshold	0.70
WBF minimum confidence threshold	0.001
WBF model weights	1:1
Maximum fused detections	300 per image

E Individual model data

E.1 Results

Table 20: Per-class AP@50:95 on the BDD100K test set.

Model	Ped.	Rider	Car	Truck	Bus	Train	Motor.	Bicycle	T. Light	T. Sign
YOLOv8s	0.305	0.215	0.475	0.421	0.433	0.000	0.203	0.227	0.232	0.342
YOLOv12s	0.304	0.211	**0.477**	**0.422**	0.438	0.003	0.207	0.225	0.235	0.347
RT-DETR-L	**0.324**	**0.231**	0.474	0.421	**0.449**	**0.006**	**0.232**	**0.244**	**0.243**	**0.368**
Faster R-CNN	0.186	0.148	0.384	0.300	0.325	0.000	0.121	0.114	0.146	0.228
RetinaNet	0.124	0.001	0.302	0.233	0.188	0.000	0.037	0.012	0.045	0.106

Table 21: mAP@50:95 under different weather conditions.

Model	Clear	Foggy	Overcast	Partly Cloudy	Rainy	Snowy
YOLOv8s	0.276	**0.284**	0.288	0.307	0.288	0.260
YOLOv12s	0.279	0.242	0.290	0.302	**0.295**	0.267
RT-DETR-L	**0.288**	0.242	**0.311**	**0.324**	0.281	**0.291**
Faster R-CNN	0.188	0.162	0.193	0.207	0.200	0.179
RetinaNet	0.099	0.123	0.104	0.115	0.114	0.102

Table 22: mAP@50:95 under different times of day.

Model	Dawn/Dusk	Daytime	Night
YOLOv8s	0.284	0.293	0.269
YOLOv12s	0.291	0.295	**0.270**
RT-DETR-L	**0.301**	**0.314**	0.268
Faster R-CNN	0.190	0.199	0.186
RetinaNet	0.103	0.111	0.098

Table 23: mAP@50:95 across different driving scene types.

Model	City Street	Gas Station	Highway	Parking Lot	Residential	Tunnel
YOLOv8s	0.289	0.335	0.239	0.310	0.321	0.408
YOLOv12s	0.290	0.328	0.243	0.311	0.326	0.353
RT-DETR-L	**0.300**	**0.356**	**0.273**	**0.337**	**0.341**	**0.421**
Faster R-CNN	0.197	0.239	0.174	0.218	0.222	0.243
RetinaNet	0.110	0.156	0.081	0.161	0.129	0.197

E.2 Counts

Table 24: Class distribution in the BDD100K test partition.

Class	**Annotated instances**	**Images containing class**
Pedestrian	13,262	3,220
Rider	649	515
Car	102,506	9,879
Truck	4,245	2,689
Bus	1,597	1,242
Train	15	14
Motorcycle	452	334
Bicycle	1,007	578
Traffic light	26,885	5,653
Traffic sign	34,908	8,221
Total	**185,526**	**10,000 test images**

Table 25: Weather-condition distribution in the BDD100K test partition.

Weather condition	**Number of images**	**Share of test set**
Clear	5,346	53.46%
Foggy	13	0.13%
Overcast	1,239	12.39%
Partly cloudy	738	7.38%
Rainy	738	7.38%
Snowy	769	7.69%
Undefined	1,157	11.57%
Total	**10,000**	**100.00%**

Table 26: Time-of-day distribution in the BDD100K test partition.

Time of day	**Number of images**	**Share of test set**
Dawn/dusk	778	7.78%
Daytime	5,258	52.58%
Night	3,929	39.29%
Undefined	35	0.35%
Total	**10,000**	**100.00%**

Table 27: Driving-scene distribution in the BDD100K test partition.

Scene type	**Number of images**	**Share of test set**
City street	6,112	61.12%
Gas station	7	0.07%
Highway	2,499	24.99%
Parking lot	49	0.49%
Residential	1,253	12.53%
Tunnel	27	0.27%
Undefined	53	0.53%
Total	**10,000**	**100.00%**

F Fusion configurations

Table 28: Validation-selected WBF configurations. Pairs are ordered by validation mAP@50:95.

Model pair	**Selected WBF IoU**	**Validation mAP@50:95**
YOLOv12s + RT-DETR-L	0.80	0.3063
YOLOv8s + RT-DETR-L	0.80	0.3055
YOLOv8s + YOLOv12s	0.75	0.2995
RT-DETR-L + Faster R-CNN	0.75	0.2701
YOLOv8s + Faster R-CNN	0.70	0.2578
YOLOv12s + Faster R-CNN	0.70	0.2574
RT-DETR-L + RetinaNet	0.85	0.2554
YOLOv8s + RetinaNet	0.70	0.2214
YOLOv12s + RetinaNet	0.70	0.2207
Faster R-CNN + RetinaNet	0.60	0.1833

Table 29: Selected thresholds for the score-averaging baseline.

Model pair	Selected IoU	Validation mAP@50:95
YOLOv8s + YOLOv12s	0.45	0.2949
YOLOv8s + RT-DETR-L	0.55	0.2909
YOLOv12s + RT-DETR-L	0.45	0.2891
RT-DETR-L + Faster R-CNN	0.45	0.2542
RT-DETR-L + RetinaNet	0.90	0.2382
YOLOv8s + Faster R-CNN	0.45	0.2248
YOLOv12s + Faster R-CNN	0.45	0.2240
YOLOv8s + RetinaNet	0.85	0.1833
YOLOv12s + RetinaNet	0.85	0.1811
Faster R-CNN + RetinaNet	0.90	0.1598

G WBF pair data

Table 30: Final test set accuracy of the WBF pairs.

Model pair	mAP@50:95	AP@50	AP@75	Precision	Recall	F1
YOLOv12s + RT-DETR-L	**0.3077**	**0.5559**	0.2858	0.6916	0.7420	0.7159
YOLOv8s + RT-DETR-L	0.3065	0.5547	0.2844	0.6860	0.7426	0.7132
YOLOv8s + YOLOv12s	0.3000	0.5269	**0.2879**	**0.7984**	0.6507	**0.7170**
RT-DETR-L + Faster R-CNN	0.2702	0.5018	0.2464	0.4530	**0.7807**	0.5734
YOLOv8s + Faster R-CNN	0.2574	0.4643	0.2425	0.5152	0.6628	0.5798
YOLOv12s + Faster R-CNN	0.2584	0.4629	0.2442	0.5173	0.6599	0.5799
RT-DETR-L + RetinaNet	0.2567	0.4555	0.2425	0.4848	0.7451	0.5874
YOLOv8s + RetinaNet	0.2229	0.3774	0.2220	0.5528	0.5706	0.5615
YOLOv12s + RetinaNet	0.2222	0.3744	0.2211	0.5541	0.5635	0.5588
Faster R-CNN + RetinaNet	0.1862	0.3558	0.1684	0.3866	0.6459	0.4837

H Usage of generative AI

ChatGPT has been utilized in the following aspects of the work:

1. Troubleshooting code
2. Formatting and converting data into LaTeX tables and figures
3. Language refinement and grammar correction

Grammarly has been used for further language refinement and grammar correction.