

Deep Learning-Based Object Detection for Vehicular Safety: A Comparative Study on COCO, KITTI, and Merged Datasets

Bengisu Yıldırım¹, Elif Aydın²

¹ Department of Electrical and Electronics Engineering, Atılım University, Golbaşı/Ankara, Turkey
yildirim.bengisu@student.atilim.edu.tr

² Department of Electrical and Electronics Engineering, Cankaya University, Etimesgut/Ankara, Turkey
elif.aydin@cankaya.edu.tr

Abstract

This study investigates deep learning-based object detection for vehicular safety using the YOLOv8 architecture. The model was tested on three dataset configurations: COCO-only, KITTI-only, and a merged COCO-KITTI dataset. COCO-only training provided broad generalization across many object categories but lacked traffic specialization, while KITTI-only training achieved strong performance in road-centric classes such as cars, trucks, and pedestrians, yet underperformed on general objects. The merged dataset offered a balanced compromise, integrating COCO's diversity with KITTI's domain specificity. Fine-tuning produced limited benefits: on KITTI, it yielded marginal improvements in precision, recall, and mean Average Precision (mAP), while on COCO and the merged dataset the results were neutral or slightly degraded. This suggests fine-tuning is not universally effective and provides value mainly in domain-focused contexts. The model achieved real-time inference at approximately 263 FPS, corresponding to 3.8 milliseconds per image, confirming its potential for embedded and automotive applications such as autonomous driving and advanced driver assistance systems (ADAS).

Topic — Artificial Intelligence and its Applications

1. Introduction

Traffic accidents are still a continuous global public safety problem since they cause over 1.35 million deaths each year and millions of injuries [1]. Major factors are human mistakes like driving distracted, alcohol abuse, and breaking traffic laws [2][1]. Therefore, the development of advanced systems that reduce risks and improve road safety is essential.

Object detection-aided warning systems have emerged as one of the most promising solutions. These systems integrate both cameras and sensors plus real-time algorithms so as to identify surrounding objects that are such as vehicles, pedestrians, and cyclists. They can also issue timely alerts or initiate automated safety responses [3]. Object detection has achieved accuracy and robustness improvements along with the advent of deep learning, particularly with convolutional neural networks (CNNs), under dynamic traffic conditions [4][5].

Among state-of-the-art architectures, YOLO stands out for its real-time efficiency and high detection accuracy, while Faster Region-Based Convolutional Neural Network (Faster R-CNN) localizes precisely critical objects [6][7]. They are influential for use in automotive safety applications. Both have become regarded highly. Economic benefits also arise from these

advancements because safer plus more efficient transportation systems result from lower healthcare also insurance costs by reducing accidents [8][9]. Transfer learning also is helpful for the adapting of pretrained models to traffic-focused detection tasks quite valuably [10]. The remaining sections of this paper are structured as follows. In Section 2, the methodology is introduced, including dataset construction, model configuration, and training procedure. Section 3 reports the experimental outcomes and provides a discussion of the results. Section 4 concludes the study, presenting final remarks, identified limitations, and recommendations for future work.

1.1 Scope of The Study

This study investigates the development and evaluation of a deep learning-based object detection model trained on a merged dataset combining COCO 2017 and KITTI. While COCO offers diversity with 80 object categories [1], KITTI focuses on domain-specific driving scenarios [2]. The integration of these datasets aims to enhance generalization while preserving accuracy in traffic-related classes. The experiments employed YOLOv8, selected for its balance between detection accuracy and real-time inference speed [3], and compared training results across COCO-only, KITTI-only, and merged datasets, followed by a fine-tuning phase. Fine-tuning produced dataset-dependent outcomes: on KITTI, results remained strong with only marginal improvements, while on COCO and the merged dataset, performance gains were limited or slightly negative. This indicates that fine-tuning does not guarantee universal benefits and is more effective when the training data distribution closely matches the deployment domain. Although this study focuses exclusively on YOLOv8 to ensure a controlled and in-depth architectural evaluation, future research will incorporate comparative analyses with other state-of-the-art object detectors such as Faster R-CNN, DETR, and EfficientDet to strengthen the generalizability and comparative understanding of the findings. Retaining all COCO categories, including non-traffic-related classes, allows analysis of both benefits (improved generalization, richer feature learning) and drawbacks (class imbalance, false positives, reduced domain focus). Through systematic experimentation, the study evaluates whether integrating diverse object categories strengthens or weakens performance in autonomous driving contexts, ultimately contributing insights into dataset design strategies for deep learning-based vehicular safety systems.

2. Methodology

2.1 Dataset Preparation

Two benchmark datasets, COCO 2017 and KITTI, were combined to establish a comprehensive training set. COCO provides 80 diverse object categories, while KITTI focuses specifically on autonomous driving scenarios. The integration of these datasets aimed to leverage both diversity and domain specificity, thereby improving generalization while maintaining strong performance on traffic-related classes.

The dataset preparation followed several steps:

1. Conversion of KITTI into COCO format.
2. Merging COCO and converted KITTI, to avoid annotation ID conflicts.
3. Image conversion from PNG to JPEG.
4. Updating annotation files to match the new image formats.
5. Splitting into training (80%) and validation (20%) subsets.
6. Conversion into YOLO format for training.

All COCO categories, including non-vehicular classes, were retained to analyze the impact of broader diversity on performance. Data preprocessing methods such as normalization and augmentation play a crucial role in ensuring robustness [11].

2.2 Model and Training Settings

The YOLOv8 model was selected for its balance of accuracy, speed, and suitability for real-time deployment. Shared training parameters included an input size of 640×640, AdamW optimizer [12], initial learning rate of 0.001, 25 epochs per phase, batch size of 6, and AMP-enabled training. Experiments were performed on Google Colab with an NVIDIA L4 GPU.

2.3 Training Process

The training involved multiple stages: initial training (25 epochs), extended refinement, and final fine-tuning. Optimization strategies included data augmentation and mixed precision training. Performance was evaluated with precision, recall, mAP, and inference speed metrics to assess suitability for real-world vehicular safety applications. Image enhancement techniques are also a potential direction to improve detection under challenging conditions such as low-light or adverse weather [13].

2.4 Fine-Tuning Strategies

2.4.1 Single-Stage Fine-Tuning (Baseline)

In the baseline fine-tuning, the model is adapted to the target data in a single pass using a small learning rate and a short schedule to avoid catastrophic forgetting. Fine-tuning **was conducted** using the AdamW optimizer with an initial learning rate of 0.0005 for 10 epochs, a batch size of 6, AMP enabled, and an input resolution of 640×640. Data augmentation remains mild—limited geometric and photometric changes with mosaic kept around 0.5 and mixup disabled—to preserve the pretrained representations; mosaic is closed in the final epochs to stabilize average precision. The validation split and random seed are fixed across runs, and early stopping is applied with a short patience to prevent over-adaptation.

2.4.2 Two-Stage Fine-Tuning (Freeze → Unfreeze)

To mitigate over-adaptation on heterogeneous data and to improve stability and calibration, a two-stage protocol is adopted, consistent with gradual-unfreezing practices in transfer learning [14]. In the Freeze stage, the backbone is partially frozen so that only the detection head adapts to the target distribution, as supported by documented freezing options in Ultralytics workflows [15], [16]. This approach reduces overfitting risk and preserves pretrained representations while aligning class-specific logits and bounding-box scales, ensuring that general visual features remain stable before full adaptation [14], [15], [16].

Training is conducted briefly (approximately six epochs) with AdamW at a slightly higher initial learning rate of about 0.0015, batch size 6, AMP enabled, and 640×640 inputs; mixed-precision usage lowers compute and memory costs without degrading accuracy [17]. Augmentations are intentionally light to align class logits, objectness, and box scales without disturbing generic features; mosaic is kept modest and turned off in the final iterations to consolidate gains, in line with augmentation guidance and the mosaic strategy popularized within the YOLO family [18], [19]. In the subsequent Unfreeze stage, all layers are unfrozen and optimized with a smaller initial learning rate (≈ 0.001) under a cosine-style decay schedule (final LR ratio ≈ 0.05) for a short window of roughly twelve epochs, following evidence that cosine/warm-restart schedules improve late-stage convergence, as they help prevent premature convergence and stabilize fine-tuning by allowing periodic resets that refresh the optimization trajectory and improve late-stage adaptation [20]. The input resolution is increased to 704–768 to improve localization of small and medium objects, batch size is maintained between 6 and 8 depending on memory, and augmentation remains moderate (mosaic around 0.4–0.6, mixup at 0.0–0.05); mosaic is again closed in the last epochs to stabilize AP, and mixup usage is kept minimal in accordance with its known regularization behavior [18], [21]. Throughout both stages, the validation split and seed are kept fixed for reproducibility, and early stopping employs a shorter patience during the Freeze stage and a slightly longer patience during Unfreeze to allow controlled adaptation while reducing the risk of catastrophic forgetting [14].

3. Results and Discussion

The experiments evaluated YOLOv8 across COCO-only, KITTI-only, and merged COCO–KITTI datasets, with an additional fine-tuning stage. Results demonstrated that COCO-only training generalized broadly across diverse object categories but did not specialize in traffic-related classes. KITTI-only training achieved the strongest results in road-centric categories such as cars, trucks, and pedestrians, while showing weaker performance on general household categories. The merged dataset provided a compromise, combining KITTI’s traffic-specific knowledge with COCO’s broader object diversity, though its overall performance remained lower than KITTI alone.

In COCO precision and recall metrics, for baseline COCO2017 training, single-stage fine-tuning, and two-stage Freeze→Unfreeze the best early performance is in the freeze stage (epoch 6) and is $\text{mAP}@0.5 \approx 0.624$ (versus $\text{mAP}@0.5 \approx 0.593$ for baseline and 0.587 for single-stage fine-tuning) and recall ≈ 0.570 (versus 0.535 for baseline and 0.532 for single-stage fine-tuning). For the more difficult $\text{mAP}@0.5:0.95$, the scores are similar and span [0.427, 0.432], indicating that there is no major difference between the more difficult localization tasks. Effects were also dataset dependent. Compared to the baseline after fine-tuning, the domain features were already learned on KITTI, while precision and recall were reduced on COCO (as expected, due to overfitting to a very diverse dataset) and only marginally changed on the merged dataset. Some of the gains from the freeze phase were reversed when training was extended into the unfreeze phase. Therefore, fine-tuning was generally a regularizer, not an accelerator. Two-stage training achieved similar performance to fine-tuning at convergence, but rapidly reached high precision and recall with fewer training epochs. Other possible future work includes using other learning-rate schedules, freezing layers selectively, unfreezing and freezing them during training, and other methods for transfer between heterogeneous datasets.

The model was fine-tuned for KITTI with same as COCO parameters using a learning rate of 0.0005, 10 epochs, mixed precision, and AdamW on an NVIDIA L4 to preserve COCO weights. The best single-stage KITTI checkpoints were the 25-epoch baselines (precision ≈ 0.836 , recall ≈ 0.818 , $\text{mAP}@0.5 \approx 0.852$, $\text{mAP}@0.5:0.95 \approx 0.630$) and fine-tuning improved

precision to 0.012 (≈ 0.848), mAP@0.5 to 0.001 (≈ 0.853), mAP@0.5:0.95 to 0.007 (≈ 0.637) but recall dropped by -0.007 (≈ 0.811) because domain knowledge likely saturated during COCO pre-training. In the freeze/unfreeze regime, while freeze allowed for rapid early gain, mAP@0.5:0.95 remained low overall. Unfreeze restored mAP@0.5 to just above pre-adaptation levels, but could not achieve the single-stage performance for harder IoUs (other than 0.75/0.95). In general, performance on KITTI was not highly sensitive to regimes. Fine-tuning led to less variance and better detection quality, while Freeze/unfreeze led to faster convergence without impacting absolute ceiling performance.

In order for the transferred parameters to be updated less than otherwise during fine-tuning on the merged COCO+KITTI dataset, same hyperparameters were used as in fine-tuning on COCO dataset (10 epochs, 0.0005 learning rate, AdamW optimizer, mixed precision on NVIDIA L4). Compared to the second-stage baseline (precision of 0.672, recall of 0.560, mAP@0.5 of 0.608, mAP@0.5:0.95 of 0.446), fine-tuning on the single stage merged COCO+KITTI datasets gave near-zero performance gain (precision of 0.667, recall of 0.560 to 0.561, mAP@0.5 of 0.607, mAP@0.5:0.95 of 0.446), indicating that the performance of the model is saturated after second-stage training. In the two-stage setting, fine-tuning during the freeze phase provides slight improvement for recall/mAP@0.5 and slightly worse mAP@0.5:0.95 localization, which only partially recover after unfreezing and end up with a final score in the same narrow scoring band as baseline. It seems the main effect of fine-tuning is to do a better job at finding stable solutions with lower variance and more consistent detections. Likewise, the Freeze $\rightarrow$ Unfreeze phase acts as a convergence increase but does not change the accuracy ceiling.

A deeper comparative analysis indicates that fine-tuning outcomes are highly sensitive to dataset composition and tuning choices. On COCO, degradation was driven by over-adaptation of parameters, suggesting that a fixed learning rate is sub-optimal for large, heterogeneous data. On KITTI, modest gains were observed owing to its focused domain and consistent labels. On the merged COCO+KITTI set, performance saturation was reached after the second training phase; additional fine-tuning primarily stabilized predictions rather than improving accuracy. These results collectively indicate that success depends on similarity between pretraining and target distributions, and that adaptive learning-rate schedules, layer-wise (partial) freezing, or earlier stopping are likely to yield more balanced generalization in future experiments.

For merged model, it reached ≈ 263 FPS (≈ 3.8 ms/image) inference speed (640 $\times$ 640 input, batch=6, NVIDIA L4, mixed precision). Strong predictions remained for most of the traffic-dominant classes (e.g., Car, Truck, Pedestrian). However, there are still low-shot or noisy classes such as Handbag, Umbrella, Person_sitting which might stay undetected due to dataset imbalance or difficulties in annotating them. Although we remove label duplicates and use standard augmentations (e.g. random flips, brightness) on the data, the COCO and KITTI residual domain noise (imperfect matching of annotations) and differences in annotation styles might create performance imbalance across domains. Additional strategies to improve results should include class-weighted or focal loss, adaptive sampling, copy-paste/mosaic augmentation, and threshold calibration.

Limitations were also noted: disparities persisted between frequent and rare classes, and the study was confined to a single detector (YOLOv8). Future work should include architectural baselines and broader hyperparameter sweeps to validate the generality of these findings.

3.1 Class-wise Performance on Traffic-Related Categories (Best Results)

The class-wise results for traffic-related categories across COCO Best, KITTI Best, and Merged Best are summarized in Table 1. These results are mainly evaluated using the mean

Average Precision (mAP) metric, which provides a comprehensive measure of detection accuracy across all of the classes that is thorough.

mAP (mean Average Precision): The mean Average Precision averages the Average Precision (AP) values after calculation across all classes [22][23]. The model's overall object detection performance is summarized through it providing of a single aggregated metric.

mAP@0.5: A 0.5 Intersection over Union (IoU) threshold is used to compute this metric for mAP@0.5. A predicted bounding box is correct given overlap exceeding 50% with the ground truth box [23]. It is actually a more lenient criterion which generally results in higher scores for it reflects the model's own ability for the capture of objects with some moderate localization accuracy

mAP@0.5:0.95: This metric averages the AP values across IoU thresholds ranging from 0.5 to 0.95 at 0.05 increments. It is a stricter also more reliable measure of detection performance [22]. This is true especially in scenarios where objects are small, occluded, or of low contrast. This criterion assesses just how well that the model performs under varying localization requirements. This assessment is more thorough than most.

Table 1. Class-wise performance results for traffic-related categories (mAP@0.5/mAP@0.5:0.95).

Class	COCO (Best – First Training) mAP@0.5/ mAP@0.5:0. 95	KITTI (Best – Second Training) mAP@0.5/ mAP@0.5:0. 95	Merged (Best – Second Training) mAP@0.5/ mAP@0.5:0. 95
Car	0.694 / 0.484	0.97 / 0.817	0.60 / 0.388
Truck	0.584 / 0.420	0.988 / 0.864	0.639 / 0.483
Van	-	0.966 / 0.783	0.536 / 0.355
Person	0.814 / 0.598	0.809 / 0.489	0.818 / 0.598
Cyclist	-	0.853 / 0.591	0.601 / 0.372
Tram	-	0.969 / 0.778	0.860 / 0.676
Bus	0.842 / 0.719	0.801 / 0.661	0.801 / 0.661
Traffic Light	0.546 / 0.309	-	0.577 / 0.337

3.2 Discussion

According to Table 1, COCO (First Training) performed solidly regarding common categories because it detected Car (mAP@0.5 equals 0.69, about 69% detection accuracy) and Person (0.81, about 81% detection accuracy). According to these values, the model reliably localized and classified large, frequently occurring objects under a lenient IoU threshold of 0.5. However, COCO had lacked proper coverage for traffic-specific classes. Van, Cyclist, as well as Tram happen to be examples of these classes plus, they are critical regarding autonomous driving.

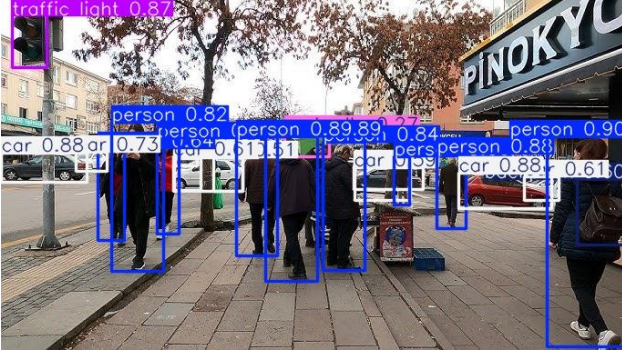


Figure 1. Example detection results of the YOLOv8 model trained on the COCO 2017 dataset before fine-tuning (shown as an example visualization; values correspond to Table 1).

The model demonstrates broad object recognition capability but lacks specialization for traffic-related objects such as Cyclist and Tram.

KITTI (Second Training) delivered the highest accuracy for road-related categories. For example, Car had (0.97, ~97% detection accuracy), Truck had (0.99, ~99%), and Van had (0.97, ~97%) (see Table 1) all achieving near-perfect performance, and it shows that the model almost always identified and localized these vehicles correctly at $\text{IoU} \geq 0.5$. Furthermore, KITTI provided strong results for vulnerable road users: Pedestrian (0.81, around 81%) along with Cyclist (0.85, around 85%). The model was verified to detect humans and bicycles. The strength of KITTI is stressed by each of these outcomes if strictly focusing on only traffic environments because even though its scope still narrows outside road-centric objects.

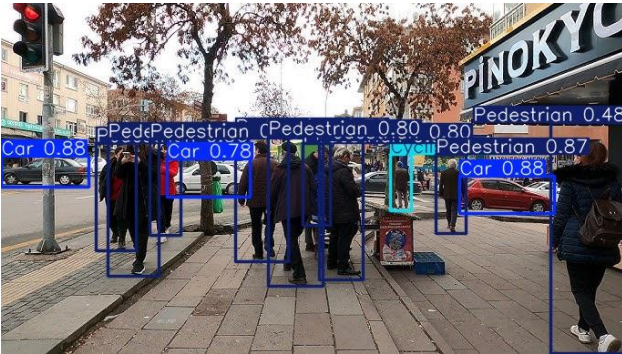


Figure 2. Example detection results of the YOLOv8 model trained on the KITTI dataset before fine-tuning (shown as an example visualization; values correspond to Table 1).

The model achieves high precision and accurate localization for traffic-domain objects such as Car, Truck, Pedestrian, and Cyclist, confirming strong specialization for driving scenarios.

Against KITTI, the Merged Dataset (Second Training) offered slightly weaker but competitive detection performance on traffic categories (e.g., Car: 0.60, ~60% vs. KITTI's 97%; Truck: 0.64, ~64% vs. 99%) as shown in Table 1. While the absolute accuracy is lower, the merged dataset preserved a larger pool of classes inherited from COCO, including non-traffic categories. This broader class coverage allows the model to operate effectively in complex urban scenarios, where both traffic-related and unrelated objects coexist.

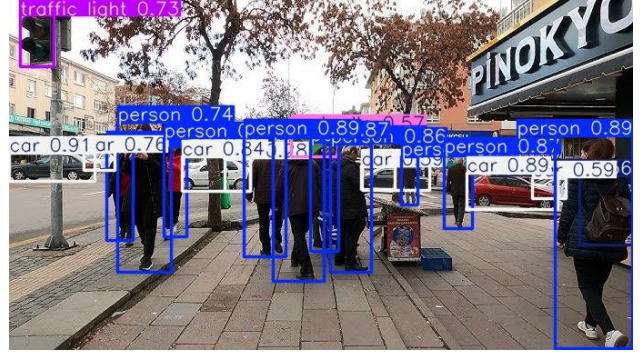


Figure 3. Example detection results of the YOLOv8 model trained on the COCO - KITTI merged dataset before fine-tuning (shown as an example visualization; values correspond to Table 1).

The model balances domain specialization and general recognition, maintaining stable detection across diverse traffic and pedestrian objects despite slight trade-offs in precision.

In summary, KITTI offers the best specialization within traffic-critical categories, evidenced by its exceedingly high $\text{mAP}@0.5$ scores (>95%, almost perfect detection) regarding vehicles plus solid pedestrian/cyclist detection. The Merged Dataset adaptability as well as broad coverage of heterogeneous classes do improve, by contrast. Because it must recognize various objects, this dataset trades Car accuracy such as ~60% for ADAS and autonomous driving application versatility.

4. Conclusion

This study evaluated the performance of the YOLOv8 object detection architecture across three dataset configurations — COCO-only, KITTI-only, and a merged COCO-KITTI dataset — to understand how dataset diversity and domain specialization influence detection accuracy and generalization. The integration of COCO's large-scale diversity with KITTI's road-centric focus aimed to enhance generalization while maintaining precise recognition of traffic-related objects.

Results confirmed that COCO generalized broadly across numerous categories yet lacked specialization for traffic scenes, whereas KITTI achieved exceptional accuracy within road-specific classes but exhibited weaker cross-domain generalization. The merged dataset provided a good base for training urban-scene generalization and domain adaptation. When fine-tuning on KITTI, the dataset dependence was shown with small improvements on precision, recall and mAP . However, on COCO and merged datasets, there was no or only slight gain in performance, which is possibly due to the small number of images. These results show that fine-tuning can still stabilize domain adaptation, but it cannot guarantee improved performance when training and deployment domains are greatly different.

At a real-time inference speed of 263 FPS (approximately 3.8 ms per image) under optimal conditions, embedded and in-vehicle performance feasibility was validated, and the results demonstrate (i) the effectiveness of the merged dataset for perception in autonomous driving, (ii) a systematic performance analysis of YOLOv8 across different dataset compositions, and (iii) a characterization showing that fine-tuning only marginally improves network performance while stabilizing convergence.

Limitations involve class imbalance, annotation inconsistencies, and reliance on a single detector architecture. Future work will incorporate comparative analyses with additional state-of-the-art models (e.g., Faster R-CNN, DETR, EfficientDet, and YOLOv9), adopt domain-specific augmentation and transfer-learning strategies, and further explore dataset-design methodologies to strengthen generalization for deep-learning-based vehicular-safety systems [10][13].

References

- [1] World Health Organization, Global Status Report on Road Safety, Geneva: WHO, 2018.
- [2] R. Treat, et al., Tri-level study of the causes of traffic accidents, Executive Summary, U.S. Department of Transportation, 1979.
- [3] J. Janai, F. Güney, A. Behl, A. Geiger, “Computer vision for autonomous vehicles: Problems, datasets and state-of-the-art,” *Foundations and Trends in Computer Graphics and Vision*, vol. 12, no. 1–3, pp. 1–308, 2020.
- [4] Y. LeCun, Y. Bengio, G. Hinton, “Deep learning,” *Nature*, vol. 521, pp. 436–444, 2015.
- [5] A. Krizhevsky, I. Sutskever, G. Hinton, “ImageNet classification with deep convolutional neural networks,” *NeurIPS*, 2012.
- [6] J. Redmon, S. Divvala, R. Girshick, A. Farhadi, “You Only Look Once: Unified, real-time object detection,” *CVPR*, 2016, pp. 779–788.
- [7] S. Ren, K. He, R. Girshick, J. Sun, “Faster R-CNN: Towards real-time object detection with region proposal networks,” *NeurIPS*, 2015.
- [8] T. Litman, *Autonomous Vehicle Implementation Predictions*, Victoria Transport Policy Institute, 2020.
- [9] McKinsey & Company, *Autonomous Driving: 10 Ways Automakers Can Drive Success*, McKinsey Report, 2015.
- [10] S. J. Pan, Q. Yang, “A survey on transfer learning,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, 2010.
- [11] J. Helin, et al., “Deep learning for data preprocessing in computer vision,” *Pattern Recognition Letters*, vol. 146, pp. 1–7, 2021.
- [12] I. Loshchilov, F. Hutter, “Decoupled Weight Decay Regularization,” *International Conference on Learning Representations (ICLR)*, 2019.
- [13] X. Wan, Y. Zhang, L. Wang, “Image enhancement techniques for deep learning-based detection,” *IEEE Access*, vol. 10, pp. 325–335, 2022.
- [14] J. Howard and S. Ruder, “Universal Language Model Fine-tuning for Text Classification,” in *Proc. 56th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2018.
- [15] Ultralytics, “Configuration—training arguments (freeze, lr0, etc.),” *Ultralytics YOLO Documentation*, 2025.
- [16] Ultralytics, “Transfer learning with frozen layers,” *Ultralytics YOLO Documentation*, 2025.
- [17] P. Micikevicius, S. Narang, J. Alben, et al., “Mixed Precision Training,” in *Proc. International Conference on Learning Representations (ICLR)*, 2018.
- [18] Ultralytics, “Data augmentation for Ultralytics YOLO (mosaic, mixup, copy-paste),” *Ultralytics YOLO Documentation*, 2025.
- [19] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, “YOLOv4: Optimal Speed and Accuracy of Object Detection,” *arXiv preprint arXiv:2004.10934*, 2020.
- [20] I. Loshchilov and F. Hutter, “SGDR: Stochastic Gradient Descent with Warm Restarts,” in *Proc. International Conference on Learning Representations (ICLR)*, 2017.
- [21] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, “mixup: Beyond Empirical Risk Minimization,” in *Proc. International Conference on Learning Representations (ICLR)*, 2018.
- [22] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, P. Dollár, “Microsoft COCO: Common Objects in Context,” *ECCV*, 2014.
- [23] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, A. Zisserman, “The Pascal Visual Object Classes (VOC) Challenge,” *International Journal of Computer Vision (IJCV)*, vol. 88, no. 2, pp. 303–338, 2010.