

**DEEP LEARNING-BASED DETECTION AND SEGMENTATION FOR 3D
PERCEPTION OF THE URBAN WORLD**

(KENTSEL DÜNYANIN 3D ALGISI İÇİN DERİN ÖĞRENME TABANLI TESPİT
VE SEGMENTASYON)

by

Bahadır Akın Akgül, B.S.

Thesis

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

MASTER OF SCIENCE

in

COMPUTER ENGINEERING

in the

GRADUATE SCHOOL OF SCIENCE AND ENGINEERING

of

GALATASARAY UNIVERSITY

Supervisor: Asst. Prof. İsmail Burak PARLAK
July 2025

ACKNOWLEDGEMENTS

The numerical calculations reported in this thesis were fully performed at TUBITAK ULAKBİM, High Performance and Grid Computing Center (TRUBA resources).

I would like to thank my dear wife Yasemin for all the support she has given me during this thesis.

July 2025

Bahadır Akın AKGÜL

TABLE OF CONTENTS

LIST OF SYMBOLS	vi
LIST OF FIGURES	vii
LIST OF TABLES	x
ABSTRACT.....	xi
ÖZET	xii
1. INTRODUCTION	1
2. LITERATURE REVIEW	6
3. METHODS & MATERIALS	10
3.1 Dataset.....	10
3.2 Detection Techniques	17
3.3 Segmentation Techniques	18
3.4 Evaluation Techniques	20
3.5 Material	22
4. RESULTS	25
4.1 Training of Neural Networks	26
4.2 Detection	36
4.3 Segmentation.....	69
4.4 Statistical Evaluation.....	89
5. DISCUSSION	102
6. CONCLUSION	106
6.1 Thesis Contribution.....	107
6.2 Limitations and Future Work	107

REFERENCES.....	109
BIOGRAPHICAL SKETCH.....	113



LIST OF SYMBOLS

API	: Application Programming Interface
AP	: Average Precision
ASPP	: Atrous Spatial Pyramid Pooling
BEV	: Bird's Eye View
CIoU	: Complete Intersection over Union
CNN	: Convolutional Neural Network
FCN-8	: Fully Convolutional Network with 8 layers
GPU	: Graphics Processing Unit
GSV	: Google Street View
IoU	: Intersection over Union
ITS	: Intelligent Transport Systems
JSON	: JavaScript Object Notation
KITTI	: Karlsruhe Institute of Technology and Toyota Technological Institute
LiDAR	: Light Detection and Ranging
mAP	: mean Average Precision
mIoU	: mean Intersection over Union
NMS	: Non-Maximum Suppression
PANet	: Prototype Alignment Network
RGB	: Red-Green-Blue
PR	: Precision-Recall
ResNet	: Residual Network
SGD	: Stochastic Gradient Descent
TRUBA	: Turkish National e-Science e-Infrastructure
VOC	: Visual Object Classes
YOLO	: You Only Look Once

LIST OF FIGURES

Figure 1.1: An image that contains distortions because of 360-degree camera.	3
Figure 3.1: An example GSV image from our dataset	13
Figure 3.2: Flowchart for YOLOv8 and YOLOv10	17
Figure 3.3: Flowchart for DeepLabV3	19
Figure 4.1: Results for YOLOv8 65-35	26
Figure 4.2: Results for YOLOv10 65-35	27
Figure 4.3: Train loss for DeepLabV3 65-35	28
Figure 4.4: Results for YOLOv8 75-25	28
Figure 4.5: Results for YOLOv10 75-25	29
Figure 4.6: Train loss graph for DeepLabV3 75-25	30
Figure 4.7: Results for YOLOv8 70-30	31
Figure 4.8: Results for YOLOv10 70-30	32
Figure 4.9: Train loss for DeepLabV3 70-30	33
Figure 4.10: Results for YOLOv8 95-5	33
Figure 4.11: Results for YOLOv10 95-5	34
Figure 4.12: Train loss graph for DeepLabV3 95-5	35
Figure 4.13: Confusion Matrix Normalized for YOLOv8 65-35	36
Figure 4.14: F1-Curve and PR-Curve for YOLOv8 65-35	37
Figure 4.15: P-Curve and R-Curve for YOLOv8 65-35	38
Figure 4.16: Confusion Matrix Normalized for YOLOv10 65-35	39
Figure 4.17: F1-Curve and PR-Curve for YOLOv10 65-35	39
Figure 4.18: P-Curve and R-Curve for YOLOv10 65-35	40
Figure 4.19: Confusion Matrix Normalized for YOLOv8 75-25	41
Figure 4.20: F1-Curve and PR-Curve for YOLOv8 75-25 split	42
Figure 4.21: P-Curve and R-Curve for YOLOv8 75-25	43
Figure 4.22: Confusion Matrix Normalized for YOLOv10 75-25 split	44
Figure 4.23: F1-Curve and PR-Curve for YOLOv10 75-25	44
Figure 4.24: P-Curve and R-Curve for YOLOv10 75-25	45
Figure 4.25: Confusion matrix normalized for YOLOv8 70-30	46
Figure 4.26: F1-Curve and PR-Curve for YOLOv8 70-30	47
Figure 4.27: P-Curve and R-Curve for YOLOv8 70-30	48
Figure 4.28: Confusion matrix normalized for YOLOv10 70-30	49
Figure 4.29: F1-Curve and PR-Curve for YOLOv10 70-30	49
Figure 4.30: P-Curve and R-Curve for YOLOv10 70-30	50
Figure 4.31: Confusion matrix for YOLOv8 95-5	51

Figure 4.32: F1-Curve and PR-Curve for YOLOv8 95-5.....	52
Figure 4.33: P-Curve and R-Curve for YOLOv8 95-5.....	53
Figure 4.34: Confusion Matrix Normalized for YOLOv10 95-5	54
Figure 4.35: F1-Curve and PR-Curve for YOLOv10 95-5.....	54
Figure 4.36: P-Curve and R-Curve for YOLOv10 95-5.....	55
Figure 4.37: Ground truth and prediction for road class from Istanbul streets.....	56
Figure 4.38: Ground truth and prediction for road class from Marseille streets	57
Figure 4.39: Ground truth and detection image from Munich streets for road class.....	57
Figure 4.40 - Ground truth and detection image from Paris streets for road class	58
Figure 4.41: Ground truth image and detection from Istanbul streets for vehicle class	58
Figure 4.42: Ground truth and detection from Marseille streets for vehicle class	59
Figure 4.43: Ground truth image and detection for vehicle class from Munich streets	59
Figure 4.44: Ground truth and detection for pedestrian class from Istanbul streets.....	60
Figure 4.45: Ground truth image and detection for pedestrian class from Marseille	61
Figure 4.46: Ground truth and detection for pedestrian class from Munich streets	61
Figure 4.47: Ground truth and detection for pedestrian class from Paris streets.....	62
Figure 4.48: Ground truth and unsuccessful detection from Istanbul for pedestrian	63
Figure 4.49: Ground truth and unsuccessful detection for pedestrian from Marseille ..	63
Figure 4.50: Ground truth & unsuccessful detection for pedestrian from Munich	64
Figure 4.51: Ground truth & unsuccessful detection for pedestrian from Paris streets.	64
Figure 4.52: Ground truth & unsuccessful detection for road class from Istanbul.....	65
Figure 4.53: Ground truth & unsuccessful detection image for road from Marseille ...	65
Figure 4.54: Ground truth and unsuccessful detection result for road from Munich	66
Figure 4.55: Ground truth & unsuccessful detection for road class from Paris	66
Figure 4.56: Ground truth & unsuccessful detection for vehicle from Istanbul.....	67
Figure 4.57: Ground truth & unsuccessful detection for vehicle from Marseille.....	68
Figure 4.58: Ground truth & unsuccessful detection for vehicle from Munich	68
Figure 4.59: Ground truth and unsuccessful detection for vehicle from Paris	69
Figure 4.60: Precision and Recall graphs for DeepLabV3 65-35.....	69
Figure 4.61: F1 Score and IoU graphs for DeepLabV3 65-35	70
Figure 4.62: Precision and Recall graphs for DeepLabV3 75-25.....	71
Figure 4.63: F1-Score and IoU graphs for DeepLabV3 75-25.....	71
Figure 4.64: Precision and Recall graphs for DeepLabV3 70-30.....	72
Figure 4.65: F1-Score and IoU graphs for DeepLabV3 70-30.....	73
Figure 4.66: Precision and Recall graphs for DeepLabV3 95-5.....	73
Figure 4.67: F1-Score and IoU graphs for DeepLabV3 95-5.....	74
Figure 4.68: Ground truth and segmentation result for road in Istanbul streets	75
Figure 4.69: Ground truth and segmentation for road class from Marseille streets	75
Figure 4.70: Ground truth & segmentation for road class from Munich streets.....	76
Figure 4.71: Ground truth & segmentation for a road class from Paris streets	76
Figure 4.72: Ground truth & segmentation for pedestrian class from Istanbul	77

Figure 4.73: Ground truth & segmentation for pedestrian class from Marseille	78
Figure 4.74: Ground truth & segmentation for pedestrian class from Munich	78
Figure 4.75: Ground truth and segmentation for pedestrian from Paris streets	79
Figure 4.76: Ground truth and segmentation for vehicle class from Istanbul streets	80
Figure 4.77: Ground truth and segmentation for vehicle class from Marseille streets ..	80
Figure 4.78: Ground truth and segmentation for vehicle class from Munich streets	81
Figure 4.79: Ground truth and segmentation for vehicle class from Paris streets	81
Figure 4.80: Ground truth & unsuccessful segmentation for pedestrian from Istanbul	82
Figure 4.81: Ground truth & failed segmentation for pedestrian from Marseille	83
Figure 4.82: Ground truth and failed segmentation for pedestrian from Munich	83
Figure 4.83: Ground truth and failed segmentation for pedestrian from Paris	84
Figure 4.84: Ground truth and failed segmentation for road from Istanbul	84
Figure 4.85: Ground truth and unsuccessful segmentation for road from Marseille	85
Figure 4.86: Ground truth and unsuccessful result for road class from Munich streets	85
Figure 4.87: Ground truth and unsuccessful segmentation for road from Paris streets .	86
Figure 4.88: Ground truth and unsuccessful segmentation for vehicle from Istanbul...	87
Figure 4.89: Ground truth & failed segmentation for vehicle from Marseille	87
Figure 4.90: Ground truth & failed segmentation for vehicle from Munich	88
Figure 4.91: Ground truth & failed segmentation for vehicle class from Paris	88

LIST OF TABLES

Table 4.1: City based metrics table for YOLO 65-35 models	89
Table 4.2: City based metrics for DeepLabV3 65-35	89
Table 4.3: Class and city-based results for YOLOv8 65-35	90
Table 4.4: Class and city-based results for YOLOv10 65-35	91
Table 4.5: Class and city-based results for DeepLabV3 65-35	91
Table 4.6: City and total metrics for YOLO 75-25	92
Table 4.7: City and total metrics for DeepLabV3 75-25	92
Table 4.8: Class based metrics for YOLOv8 75-25	93
Table 4.9: Class based metrics for YOLOv10 75-25	94
Table 4.10: Class based metrics for DeepLabV3 75-25	94
Table 4.11: City based results for YOLO models with 70-30	95
Table 4.12: City based results for DeepLabV3 70-30	95
Table 4.13: Class based results for YOLOv8 70-30	96
Table 4.14: Class based results for YOLOv10 70-30	97
Table 4.15: Class based results for DeepLabV3 70-30	97
Table 4.16: City based results for YOLO models with 95-5	98
Table 4.17: City based results for DeepLabV3 95-5	98
Table 4.18: Class based results for YOLOv8 95-5	99
Table 4.19: Class based results for YOLOv10 95-5	100
Table 4.20: Class based results for DeepLabV3 95-5	100

ABSTRACT

The rapid development of autonomous driving technologies has highlighted the growing need for perception systems that can understand and interpret complex urban traffic scenes. This thesis presents a deep learning-based, vision-only framework that focuses on the detection and segmentation of three key elements in traffic environments: roads, vehicles, and pedestrians. The system operates using 2D image projections generated from 360-degree cameras, without relying on LiDAR or any other external sensors. This approach aims to offer a scalable and cost-effective solution for intelligent transportation applications.

To train and evaluate the proposed models, a custom dataset was created using Google Street View imagery collected from four major European cities: Istanbul, Paris, Munich, and Marseille. The dataset includes 8,932 labeled images and more than 149,000 object annotations, providing a diverse range of traffic scenes under varying urban conditions.

YOLOv8 and YOLOv10 models were used for object detection, while DeepLabV3 was applied for semantic segmentation. The models were evaluated across different train-validation splits using standard metrics. YOLOv10 achieved the best detection performance with a mAP@0.5 score of 0.685 and an overall precision of 0.76. DeepLabV3 produced strong segmentation results, including IoU scores above 0.85 for road detection and F1 scores exceeding 0.80 in clearly defined object regions.

These results indicate that image-only systems can effectively perform traffic scene analysis in real time. The framework developed in this thesis demonstrates the potential of deep learning for urban scene understanding and contributes a new dataset that supports future research in vision-based autonomous navigation.

ÖZET

Otonom sürüş teknolojilerindeki hızlı ilerleme, karmaşık kentsel trafik sahnelerini anlayabilen algılama sistemlerine olan ihtiyacı daha da belirgin hale getirmiştir. Bu tez çalışması, yalnızca görüntü verilerine dayalı olarak çalışan bir derin öğrenme sistemi sunmaktadır. Sistem, trafik ortamlarında yer alan üç temel öğeyi, yani yolu, aracı ve yayayı tanıma ve ayırt etme üzerine odaklanmaktadır. Yöntem, 360 derece kameralarla elde edilen görüntülerin 2D projeksiyonlarını kullanmakta ve LiDAR gibi ek sensörlere ihtiyaç duymamaktadır. Bu yaklaşım, özellikle düşük maliyetli ve ölçeklenebilir çözümler sunmayı hedefleyen akıllı ulaşım sistemleri için önemlidir.

Geliştirilen modellerin eğitimi ve test edilmesi için, İstanbul, Paris, Münih ve Marsilya şehirlerinden alınan Google Street View görüntüleriyle oluşturulmuş özel bir veri seti kullanılmıştır. Veri seti, 8.932 etiketli görüntü ve 149.000'in üzerinde nesne anotasyonu içermektedir. Farklı şehir yapılarına ve trafik koşullarına sahip bu veri seti, modelin genel performansını değerlendirmek için zengin bir içerik sunmaktadır.

Nesne tespiti için YOLOv8 ve YOLOv10 modelleri, semantik segmentasyon için ise DeepLabV3 modeli tercih edilmiştir. Modeller farklı eğitim-doğrulama oranlarında test edilmiş ve standart ölçütlerle değerlendirilmiştir. En iyi tespit performansı YOLOv10 ile elde edilmiş, model mAP@0.5 değerinde 0.685 ve genel kesinlikte 0.76 sonuçlarına ulaşmıştır. Segmentasyon tarafında ise DeepLabV3 modeli, yol sınıfında 0.85'in üzerinde IoU ve net bölgelerde 0.80'i aşan F1 skorları ile yüksek başarı göstermiştir. Bu sonuçlar, yalnızca görüntü kullanan sistemlerin, kentsel trafik sahnelerinin analizinde gerçek zamanlı ve etkili biçimde çalışabileceğini göstermektedir. Bu tez kapsamında geliştirilen sistem, görüntü tabanlı otonom araç teknolojileri için hem uygulanabilir hem de geliştirilebilir bir temel sunmakta ve literatüre yeni bir veri seti kazandırmaktadır.

1. INTRODUCTION

An automobile, also known as a car, is a vehicle with its own engine, traditionally operated by a human driver to carry passengers or goods. The traditional car is a complex system of interrelated parts that includes an internal combustion engine, chassis and transmission. Over time, cars have developed into complex machines used around the world. Today, cars are widely accessible and offer convenient, flexible means of personal mobility.

The automotive industry, composed of many manufacturers, produces vehicles with diverse features and designs. Key design priorities in the automotive industry include safety, fuel efficiency, and environmental sustainability. This leads to using advanced technologies like hybrid and electric systems.

Autonomous cars, or self-driving cars, use advanced technologies to drive without humans controlling them. They have sensors, cameras, radar, and artificial intelligence to sense the surroundings and make driving decisions. The main goal is to drive and handle traffic by themselves, making roads safer and more efficient.

Companies in the automotive and technology sectors are investing heavily in research to improve these systems. The benefits of autonomous cars include improved road safety, reduced congestion, and enhanced accessibility for individuals with disabilities. However, widespread adoption faces challenges such as regulatory, ethical, and technological issues. Ethical considerations involve decision algorithms in emergency

situations and liability in the event of a crash, prompting collaboration among politicians and industry stakeholders to establish regulations and standards.

As technology advances, careful testing is essential to ensure the safety and reliability of self-driving cars. The vision of a future where self-driving cars coexist with traditional vehicles promises a more efficient, safer, and accessible transport landscape.

As these technologies evolve, one of the key challenges shifts toward enabling safe navigation in complex urban environments. For autonomous vehicles to navigate urban traffic scenes effectively, it is crucial to ensure the continuous and real-time recognition of road users, traffic signs, road markings, and other dynamic variables. Recognizing these factors and making decisions based on the resulting data pose a complex problem with numerous variables.

These challenges become even more pronounced in large-scale urban datasets, such as those derived from platforms like Google Street View. One of the most significant challenges in 3D object detection and segmentation, especially in real-world environments like Google Street View, is the inherent geometric complexity of urban scenes (Long et al., 2015). Objects vary widely in shape, scale, and orientation, and are often partially occluded or overlapping. Capturing reliable 3D structure from images alone—particularly from monocular or stereo sources—is susceptible to depth ambiguity, especially at large distances or in textureless regions (Mazzini et al., 2018; Ahn and Kwak, 2018). Even with depth sensors or structure-from-motion techniques, the resulting 3D reconstructions can be sparse, noisy, or incomplete, which affects the accuracy of downstream object localization and segmentation.

An alternative idea for reducing computational load is to consider static scene components for reference, allowing vehicles to anticipate conditions based on past observations. However, this approach may be risky and inefficient in urban traffic scenes with a high number of variables.

In street-level imagery, objects are typically captured across multiple overlapping views from different positions and angles. This multi-angle coverage introduces the need for accurate multi-view data fusion and instance-level 3D tracking. Establishing correspondences between object detections or segments across views requires reliable

pose estimation, depth alignment, and appearance matching—tasks that are all error-prone in real-world, dynamic settings. Misalignment between images, due to GPS drift, moving objects, or inconsistent lighting, can lead to redundant or fragmented detections in the 3D space. Efficiently combining 2D detections into consistent 3D object instances remains an open problem, especially when scalability and real-time performance are required.



Figure 1.1: An image that contains distortions because of 360-degree camera.

Instantaneous recognition of elements in urban traffic scenes and making decisions based on collected data in real-time appears to be an effective method for solving this problem. Various data sources, including audio and video, along with different sensors, are needed for instantaneous identification. A 360-degree camera can capture an entire urban traffic scene, providing a comprehensive view for evaluation. Object detection in Google Street View images benefits from the rich visual context and comprehensive scene coverage provided by 360-degree panoramic imagery. Unlike standard camera views, Street View captures the full environment around the camera, allowing models to observe objects from all directions (Huang et al., 2019). This helps in reducing the blind spots common in narrow field-of-view datasets and supports more robust recognition of large-scale scene

structures such as road networks, buildings, and signage. Additionally, the geospatial and temporal continuity of Street View data allows for longitudinal analysis—tracking changes in object presence, infrastructure, and urban growth over time (Ning et al., 2025). Google Street View panoramic images are typically encoded using a spherical equirectangular projection, which introduces significant geometric distortions, particularly near the poles. Conventional object detection models, trained on perspective images, often struggle to generalize to these projections without adaptation. Features such as lines, shapes, and object edges become distorted, which can hinder the performance of both handcrafted and learned feature extractors. While some solutions involve slicing panoramas into perspective views or retraining models on warped images, these workarounds add preprocessing complexity and may not fully capture the spatial continuity of the original scene (Kim et al., 2025).

To address these challenges, recent research has increasingly turned to deep learning-based approaches. This thesis explores using deep learning models, such as neural networks, to analyze various traffic elements, including vehicles, pedestrians and roads. The goal is to enhance deep learning systems' ability to rapidly detect and respond to dynamic urban traffic conditions. This study addresses the challenges presented by the dynamic and unpredictable nature of urban traffic. The goal is to enhance the overall performance of Intelligent Transport Systems (ITS) and help create urban mobility solutions that are safer and more efficient by using deep learning algorithms.

A key contribution of this thesis is the introduction of a novel dataset composed of real-world street images captured with 3D cameras. All annotations were manually created, ensuring high-quality labels with consistent ground truth across diverse urban scenes. The dataset covers multiple cities, allowing a comparative evaluation of how deep learning models generalize across environments with different visual and structural characteristics. The goal of this study is to test how well selected object detection and semantic segmentation models work in different conditions. The models included in the experiments are YOLOv8, YOLOv10, and DeepLabV3, chosen for their high relevance in the literature and complementary strengths. The original You Only Look Once (YOLO) framework proposed by Redmon et al. (2016) introduced a unified, real-time

object detection system that significantly accelerated inference while maintaining competitive accuracy. This pioneering architecture laid the foundation for subsequent YOLO versions used in this study. YOLOv8 and YOLOv10 represent two successive advancements in the YOLO family of object detectors, designed for real-time object recognition with high accuracy and computational efficiency (Ultralytics, 2023; Wang et al., 2024). YOLOv8 employs a decoupled head architecture, separating classification and localization branches to improve task specialization. It introduces anchor-free detection and adopts the Complete IoU (CIoU) loss for bounding box regression, enhancing convergence and robustness (Ultralytics, 2023). YOLOv10 builds upon this foundation by incorporating a Unified Detection Paradigm (UDP), which bridges object detection and instance segmentation under a single architecture. It uses RepOptimizer for training stability and architectural re-parameterization to improve inference speed without sacrificing accuracy (Wang et al., 2024). Additionally, DeepLabV3 is a state-of-the-art semantic segmentation architecture that leverages atrous convolution to effectively capture multi-scale contextual information (Chen et al., 2017). By employing Atrous Spatial Pyramid Pooling (ASPP), it enables robust dense prediction without sacrificing resolution. This makes DeepLabV3 particularly effective in urban scene understanding, where fine object boundaries and large-scale contextual cues are crucial. The model also integrates batch normalization and deeper backbone networks, such as ResNet and Xception, to improve accuracy in complex scenes (Chen et al., 2017; Zhao et al., 2018). DeepLabV3 achieves superior mean Intersection over Union (mIoU) scores on benchmarks such as Cityscapes and PASCAL VOC, particularly excelling in fine-grained categories like pedestrians and traffic signage (Chen et al., 2018).

This thesis is structured into five chapters. Chapter 2 explains the methods, model choice, data collection, and training steps. Chapter 3 shows the results with visuals and performance scores from different versions of our dataset. Chapter 4 discusses what the results mean and compares key points. Chapter 5 gives the conclusion and suggests ideas for future work. To sum up, this thesis adds to current research by providing detailed performance results per class and per city, using a fully labeled real-world dataset.

2. LITERATURE REVIEW

Understanding urban traffic scenes has significantly advanced in recent years, largely due to the emergence of deep learning techniques. Recent advances in image segmentation—particularly in the recognition of urban traffic scene elements—have been largely driven by deep learning models.

Perez and Fusco (2025) demonstrated how the temporal and spatial continuity of Google Street View imagery can be used for longitudinal urban analysis, enabling the tracking of infrastructure changes and city growth through multimodal deep learning frameworks. Long, Shelhamer, and Darrell (2015) were pioneers in using deep learning for pixel-level classification in urban environments. Building on this work, Mazzini et al. (2018) presented an efficient CNN architecture tailored for street scenes, offering further evidence of segmentation effectiveness.

AffinityNet, introduced by Ahn and Kwak (2018), predicts pixel-level affinities between adjacent coordinates using only image-level labels, enabling weakly supervised semantic segmentation. It learns to construct an affinity graph and applies random walk propagation to expand sparse class activation maps into full object masks, effectively capturing object boundaries. Evaluated on PASCAL VOC 2012, the method significantly outperforms prior weakly supervised techniques and approaches performance levels comparable to fully supervised segmentation models.

ICNet, developed by Zhao et al. (2018), introduces a multi-resolution cascade network designed for real-time semantic segmentation of high-resolution urban imagery. The model combines low-, medium-, and high-resolution inputs through a cascade feature fusion strategy to balance speed and accuracy. Tested on datasets like Cityscapes and CamVid, ICNet achieves competitive segmentation accuracy while maintaining real-time inference performance, making it suitable for applications in autonomous driving.

Pohlen et al. (2019) introduced Full-Resolution Residual Networks (FRRN) to address the challenge of balancing semantic abstraction and spatial precision in urban scene segmentation. Their approach processes feature maps along two paths—one preserving full resolution and another capturing contextual semantics—and combines them using specialized residual units. This dual-stream design allows the network to segment complex street scenes with enhanced boundary accuracy and overall coherence, without depending on post-processing techniques.

Huang et al. (2019) introduced Criss-Cross Network (CCNet), which enhances context modeling by using repeated operations to capture long-range dependencies within feature maps. This mechanism allows each pixel to aggregate information from the entire image in a memory-efficient manner, improving segmentation performance in complex urban scenes. CCNet delivers richer contextual understanding without significantly increasing computational cost, making it well-suited for high-resolution street scene tasks.

Wang et al. (2019) introduced Prototype Alignment Network (PANet), a novel approach for few-shot semantic segmentation that aligns feature prototypes across different classes. The method uses class-specific feature centroids, or prototypes, to guide pixel-wise segmentation, enabling effective generalization to unseen categories with minimal examples. PANet demonstrates robust performance on diverse datasets by enhancing inter-class discrimination while maintaining model flexibility.

Wahid and Lee (2019) emphasized the importance of deconvolution layers in their study on semantic segmentation. Their model, based on Fully Convolutional Network with 8 layers (FCN-8), showed good results in IoU scores and fast processing times. Tan et al. (2020) explored the challenges of nighttime semantic segmentation and demonstrated that their exposure-guided network adapts effectively to varying lighting conditions.

Among the commonly used datasets, KITTI is widely regarded as a benchmark for training and evaluating deep neural networks related to autonomous driving and urban scene analysis (Geiger, Lenz, & Urtasun, 2012). With annotations for traffic elements like vehicles, pedestrians, cyclists, and traffic signs, it is essential for training algorithms to recognize and understand the intricacies of urban traffic.

Also, the Cityscapes dataset is an important resource. It was created together by universities and companies to focus on detailed understanding of city scenes by labeling every pixel. With its high-quality images and detailed annotations, Cityscapes remains the benchmark dataset for researchers working on urban semantic segmentation. (Cordts et al., 2016). Besides these two traditional datasets, other datasets have also been examined on Dataset subsection.

Despite these comprehensive resources, traditional datasets lack support for 2D projections derived from 360-degree panoramic images. In recent comprehensive reviews, the geometric and structural complexities of 3D object detection, especially in street-level scenes, have been highlighted as major bottlenecks for real-time systems in autonomous driving (Pravallika et al., 2024).

Ning et al. (2025) emphasized that annotating 360-degree panoramic images is labor-intensive and error-prone, particularly when extended to 3D domains where spatial distortions and occlusions are more complex.

Kim et al. (2025) investigated how panoramic image distortion, particularly in equirectangular projections, poses challenges for standard convolutional networks, and highlighted that retraining on warped or perspective-sliced views often compromises spatial coherence.

Zhang et al. (2025) introduce SMM-POD, a panoramic 3D object detector designed to handle the challenges of wide field-of-view 360° imagery. By fusing spherical image data with synchronized LiDAR scans, their model adaptively mitigates projection distortions and improves detection robustness in urban street environments. SMM-POD demonstrates substantial gains in accurately recognizing small and occluded objects like pedestrians and cyclists compared to conventional monocular approaches.

Li et al. (2025) propose RADLER, a radar-based object detection framework that integrates contrastive self-supervised learning with semantic 3D city model priors. By combining radar reflections and high-level semantic information from city structures, their network better differentiates between vehicles, cyclists, and pedestrians—even in noisy or occluded urban settings. Evaluated on the new RadarCity dataset, RADLER

improves mAP compared to baseline radar detectors, showcasing the value of multi-modal and map-informed detection.

Wang et al. (2025) adapted the YOLOv10 model to improve the detection of small objects in challenging environments such as aerial or street scenes. By adding modules that help the model focus better on important areas, they achieved more accurate results without losing speed. This study shows that YOLOv10 can be successfully adjusted for more specific detection tasks in complex real-world conditions.

Despite the availability of global datasets, no dataset specifically representing street scenes in Türkiye was found in the existing literature. However, since our aim in this study is to develop an algorithm that can help autonomous swarms using only image processing algorithms without additional decision support systems such as LiDAR, we decided to create our own dataset. Recent surveys emphasize that current 3D object detection systems face challenges in sensor fusion, scene reconstruction, and maintaining performance under occlusion, supporting the rationale for vision-only alternatives (Arnold et al., 2019). According to Qian et al. (2022), data derived from 360-degree panoramic imagery still lacks standardized benchmarks for urban scene analysis, underscoring the need for localized datasets like the one proposed in this study. The dataset is a collection of 2D images from four different cities—Istanbul, Paris, Marseille, and Munich—taken with a 360-degree camera.

3. METHODS & MATERIALS

This section presents the methodology followed in this study, focusing on urban scene understanding using both object detection and semantic segmentation approaches. The general process includes collecting data, labeling it by hand, training the model, and checking how well it works. Each step is explained clearly and separately to make the study easy to understand and replicate.

The experiments are conducted on a custom dataset created from street-view images captured with 3D camera systems in multiple cities. These images reflect a wide range of urban characteristics and visual complexity. All samples were labeled by hand to make sure the data was correct and reliable. This is important for testing how well deep learning models work in real-life situations.

Three models are selected for comparison: YOLOv8, YOLOv10, and DeepLabV3. These models are chosen based on their popularity in recent literature and their distinct design goals. The former two are real-time object detectors, while the latter focuses on dense pixel-wise segmentation. Further details on the dataset structure, model architectures, training setup, and evaluation procedures are provided in the following subsections.

3.1 Dataset

For our study, we needed a dataset shot with a 360-degree camera but projected in 2D. In addition, since the main goal of our work is to recognize the three most basic classes in a traffic scene, namely road, vehicle and pedestrian, we searched for a database with the relevant classes. Afterwards, we did not want to work with data enriched with LiDAR sensors, because in this work we only wanted to recognize the relevant classes from the cameras. Because we believe that instead of expensive LiDAR sensors for autonomous driving, the future of autonomous driving will be to recognize the relevant classes using

only the 360-degree camera in the vehicle or the 2D 360-degree projection of images from different cameras. When we examined the datasets used in different academic studies, we found that the datasets with the largest number of samples/images were those consisting of images from a single camera. Examples include datasets such as Cityscapes (Cordts et al., 2016), KITTI (Geiger, Lenz, & Urtasun, 2012), and others. In addition to standard ones, the Equirectangular Panorama Object Detection Dataset, introduced by Yang et al. (2018), provides a format that is very close to what we need. However, it is not suitable for our study because the images, although partially containing traffic objects, do not include a sufficient variety of classes to represent real traffic scenes. Despite recent advances in autonomous driving datasets, there remains a notable gap in publicly available resources that offer true 360-degree equirectangular imagery without LiDAR, specifically designed for traffic scene understanding and object detection tasks. The widely used Waymo Open Dataset, for example, utilizes a multi-camera system that provides surround view coverage, but not a single panoramic 360-degree image. Moreover, it includes extensive LiDAR data, which disqualifies it under LiDAR-free constraints, as noted by Sun et al. (2020). Similarly, the Honda Driving Dataset (HDD) only provides conventional rectilinear front-facing camera images and lacks panoramic coverage, as described by Ramanishka et al. (2018), while its extended version, H3D, focuses exclusively on LiDAR-based 3D object detection (Yao et al., 2019). Another line of work explores fisheye and surround-view systems, such as those analyzed by Dosovitskiy et al. (2024), but these optical systems do not produce spherical equirectangular projections and often lack object-level annotations for detection. A recent dataset, Dur360BEV, combines fisheye imagery with LiDAR in a vehicle-centered 360-degree setup, yet still does not meet the strict requirement of LiDAR-free, fully spherical Red-Green-Blue (RGB) imagery, as shown by Zhu et al. (2025).

In summary, while several datasets attempt to address parts of this problem space, a publicly accessible, LiDAR-free dataset with true 360-degree equirectangular traffic scenes and object detection annotations remains unavailable.

Therefore, to best support our work, we decided to create our own dataset, which would include different streets, cities, and countries with different traffic habits. Traffic habits,

traffic participants, streets, cities and countries vary considerably. Therefore, to be able to train against as many different scenes as possible, we had to have a suitable data set. However, creating a dataset with these characteristics would be a time-consuming and costly process. This is because we would need to create a dataset based on a vehicle participating in traffic in different countries, and images taken at regular intervals by a 360-degree camera on the vehicle. Unfortunately, we don't have the means to do this, so we must rely on data that meets the specified qualifications. In this regard, we realized that the images on Google Street View (GSV) were very suitable for our work. In this study, for the reasons stated, Google Street View images were used for academic research purposes, and no redistribution of the images or trained models was made. Google's usage policy does not allow using Street View images for machine learning. However, some researchers argue that EU law especially Article 3 of Directive 2019/790 may allow data mining for research if certain rules are followed. Several peer-reviewed studies, such as the one by Helbich et al. (2024), have used Street View data in similar contexts, and this study follows this precedent in a limited, non-commercial, and illustrative way.

In this study, we used street-level panoramic images from the Google Street View Static API, which is part of the Google Maps Platform (Google, 2025). This API allows users to get static, 360-degree images of streets by entering location coordinates and camera settings such as direction and field of view. The images we collected were downloaded through the official API using our developer key.

Sample images of various streets from the cities of Istanbul, Munich, Paris, and Marseille were used to generate the dataset. The specific locations included were:

Istanbul:

- Libadiye Street
- Muallim Naci Street
- Bağdat Street
- The Dolmabahçe area and its surrounding coastal roads
- The Taksim district and nearby urban streets
- Levent district and its surrounding main avenues

Munich:

- Einstein Street
- Leopold Street
- Other major arterial roads in central Munich

Paris:

- Champs-Élysées
- Boulevard Richard-Lenoir
- Surrounding streets in the city center

Marseille:

- Avenue Pierre Mendès-France (coastal boulevard)
- A7 motorway section near Aix-Marseille University (urban expressway)

The images were then labeled by hand using an online tool called Roboflow (Dwyer et al., 2024). After uploading the images to Roboflow, the labeling process was started using the Annotate menu. During the annotation process, 8,932 images were manually labeled. The dataset was built using data from four major cities: Istanbul, Munich, Paris, and Marseille. The resulting dataset is composed of a total of 8,932 images captured in 360-degree views, which have undergone conversion into a 2D format. The images have been labeled manually, with a view to ensuring both high accuracy and detail in the annotations.



Figure 3.1: An example GSV image from our dataset

The dataset includes 42,424 instances of the class "pedestrian", 9,482 instances of the class "road" and 97,744 instances of the class "vehicle". On the Roboflow platform, in addition to the labeling process, it is possible to do various preprocessing operations on the dataset, along with other operations to create a version of the dataset and train it using YOLO or the platform's own model.

It is also possible to use the models you have already trained as an aid in the labeling process in the Roboflow interface.

All images in the dataset were manually labeled using bounding boxes via the Roboflow platform. This process, carried out one by one for each instance, was highly time-consuming but ensured precise and consistent ground truth across the dataset. Although the platform allows polygon annotations, only rectangular bounding boxes were used because YOLO supports only bounding boxes, and the same dataset was needed for DeepLabV3.

In addition, as we have already mentioned, it is possible to use models previously trained on the Roboflow platform with different datasets to assist in the labeling process. If other researchers have shared their models on the platform as open source, these models can also be used as an aid during manual labeling.

In the manual labeling process, there was no external user assistance, the labeling process was done entirely by us. At the beginning of the labeling process, the road class was labeled as a polygon. In the rest of the process, the polygonal labeling was abandoned, and bounding box labeling was done for the road class, especially since bounding box labeling is mandatory for the training of the object detection algorithm and both polygonal and bounding box labeling are possible for the semantic segmentation algorithm.

In addition, different classes were started to be labeled according to the type of vehicles in the traffic scene (e.g. trucks, cars, buses, bicycles, etc.), but then it was thought that this would create an imbalance in the sample size and that the type of traffic participants in the vehicle type would not provide a significant benefit in the first place, so all such classes were grouped under the vehicle class.

In addition, in the traffic scenes taken from the GSV, parked vehicles are often seen alongside moving vehicles. However, vehicles such as rentable electric scooters and

bicycles, which are included in the vehicle class and which we also see in traffic scenes, were initially included in the labeling process even without a driver, but parked scooters and bicycles, which are required by regulations to be parked in a restricted area, especially in Western European cities such as Paris, and which have no driver and are unlikely to move because they have no driver, have been excluded from the labeling process. This principle has also been applied to data from all cities to ensure consistency across cities. However, vehicles such as cars, trucks, buses, etc., where it is very difficult to determine from the image whether there is a driver or not, are included in the tagging process even if they are parked. In addition to these, GSV images also include pedestrians as traffic participants. In the images, people's faces are completely blurred, preventing violations of personal rights. A pedestrian can walk on the sidewalk or on the road. However, in order not to negatively affect the educational process, participants in the GSV images who are clearly not active traffic participants (e.g. a person sitting on a chair on the side of a street) are also labeled. However, especially after this decision, it has been observed that there is a confusion between labels with vehicle class whose drivers are clearly visible in the traffic scene, such as bicycles, scooters, open-top cars, etc., and labels with pedestrian class. To avoid this confusion, the rider on the bicycle is also labeled as pedestrian, even though this person is clearly a rider, and the bicycle below him is labeled as a vehicle. However, as this would have added to the confusion, human and bicycle, human and scooter have been included in a single class, vehicle, even though they are clearly human. In addition, particularly large construction machinery in scenes is often not included in the labeling. Because, in a hypothetical traffic scene, these vehicles would not be present, and even if they were, according to the regulations applied in most countries, traffic signs and markers and the places where these vehicles operate, or pass would be closed to other participants.

However, these construction vehicles are included in the labeling process, even though they may be in a conventional traffic scene, even if they are stopping work in a traffic-free area. While labeling the roads, several difficulties were encountered since the images used are 2D projections of a 360-degree image. This is because often, even though a path is enclosed in a bounding box, it contains image fragments that are not actually part of

the path. However, despite this situation due to the limitations of object detection models, the labelling and training process of the road class was continued, considering it as a strong reference.

In addition, the shape and location of the road is often very similar in the images from GSV.

All images were sourced from Google Street View using the Street View Download 360 tool. While the metadata indicates that some images were captured using panoramic cameras such as the RICOH THETA S, the dataset may include images from a variety of camera models, as GSV aggregates data from multiple sources. All files were downloaded in .jpg format at a resolution of 8192×4096 and subsequently resized to 2048×1024 for training purposes.

Furthermore, while the original images are quite high resolution, the images included in the dataset are 2048×1024 . Since the GSV images project a very large area at the same time, there are difficulties in labeling the instances, especially pedestrian and sometimes vehicle class instances, which in our opinion are not of high importance for the instantaneous decisions of the autonomous vehicle that our study hopes to help, as they are located too far away and occupy a very small area in the image. However, since most instances belonging to the vehicle class are over a certain size, even if they are relatively far away, we did not encounter as many problems as we did with the pedestrian class. Since most people are of a certain size, instances belonging to the pedestrian class, especially those that are far away from the camera, were tried to be labeled, even though they may seem insignificant, but during the labeling process, especially for the pedestrian class, some points were missed or not noticed. When labeling, especially for the vehicle and pedestrian classes, even if the object is not completely visible, if a person thinks that it belongs to the relevant class, it is labeled even though it is not completely in the image. However, for the vehicle class, for example, when there are dozens of cars parked side by side in a traffic scene, it is not always possible to label the smallest parts of each car. As the number of labeling increased, the model was trained repeatedly to improve the performance of this decision support system. Even though a decision support system was

used, all images in the dataset were also manually re-examined and only the output of the model was used.

3.2 Detection Techniques

Following the original YOLO algorithm introduced by Redmon et al. (2016), this study focuses on two recent versions, YOLOv8 (Ultralytics, 2023) and YOLOv10 (Wang et al., 2024), which offer significant advancements in speed, modularity, and real-time performance.

In this study, two advanced versions of object detection algorithms were utilized: YOLOv8-large (Ultralytics, 2023) and YOLOv10-large (Wang et al., 2024). These models were chosen because they are accurate and can make fast predictions, which is useful for understanding city scenes.

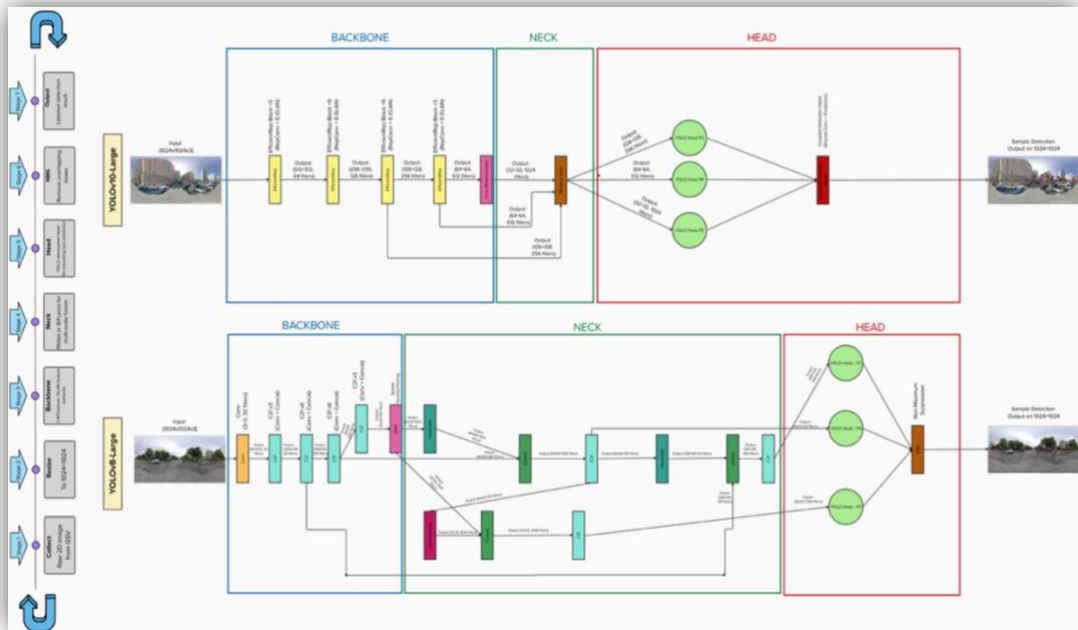


Figure 3.2: Flowchart for YOLOv8 and YOLOv10

The training of both YOLO models was conducted using the Ultralytics library, which offers a streamlined interface for model training, validation, and testing. For both models, the input image size (imgsz) was set to 1024, and training was performed for 100 epochs. A batch size of 12 was used, and the computation was distributed across two Graphics Processing Units (GPU) using the device='0,1' parameter. The training process was conducted using a batch size of 12, with computations distributed across two GPUs via the device= '0,1' setting. On average, each epoch took approximately 12 minutes to complete, resulting in a total training time of nearly 20 hours per full run. However, due to the usage of the TRUBA national computing infrastructure, where GPU sessions are limited to 4-hour intervals, the entire process was executed in multiple consecutive batches. This queue-based scheduling occasionally led to extended waiting times between sessions, making the actual wall-clock training duration longer than the compute time alone.

The Stochastic Gradient Descent (SGD) optimizer was used during training, with default momentum and weight decay parameters provided by the Ultralytics framework. No manual modifications were made to the default preprocessing settings such as blur or mosaic augmentations. The number of workers was set to 16 to optimize data loading on the high-performance computing infrastructure.

To remove redundant detections, Non-Maximum Suppression (NMS) was applied during post-processing. These settings ensured that both models were trained under consistent conditions, enabling a fair performance comparison in later sections.

3.3 Segmentation Techniques

Semantic segmentation helps interpret the content of an image by assigning a class label to each pixel. Unlike object detection, which only draws bounding boxes around objects, segmentation provides precise shapes and regions for elements such as roads, people, and vehicles. In this project, we chose DeepLabv3 because it performs well in many real-time systems and effectively captures both coarse and fine details in an image, as shown by Chen et al. (2017).

The code used for training DeepLabV3 is implemented in Python using the PyTorch library. The training code has been shared as open source on GitHub. For DeepLabV3 training, 100 epochs were used, as recommended by Chen et al. (2017). Input images were resized to 1024x768 pixels, and standard augmentations such as horizontal flipping and normalization were applied using the Albumentations library.

A custom dataset class was implemented to load “.jpg” images and their corresponding “_mask.png” segmentation masks.

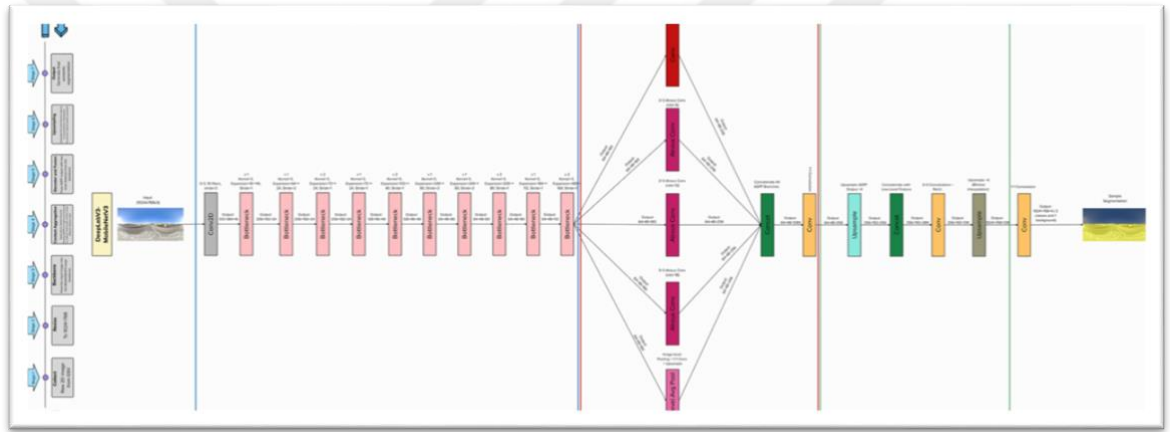


Figure 3.3 – Flowchart for DeepLabV3

The DeepLabv3 model with a MobileNetV3 Large backbone was used and adapted for four output classes. If multiple GPUs are available, the model is parallelized with DataParallel to use them effectively. The model was trained using the Adam optimizer with a learning rate of 0.0001. To handle class imbalance, a weighted CrossEntropy loss was used.

The training and validation data were loaded with a batch size of 24 and 8 workers using pin memory True for better performance. A checkpoint system allows training to continue from the last saved state. After each epoch loss values, and model weights are saved. At the end of training, performance is evaluated on the validation set using per-class precision, recall, F1 score, and IoU metrics. Results are saved both as plots and JSON files and the final trained model is stored for later use.

3.4 Evaluation Techniques

To check how well the object detection and segmentation models worked, several common evaluation methods were used. These metrics helped measure the correctness of predictions, the difference between classes, and the overall model quality, both numerically and visually.

Precision shows how many of the predicted positive results are correct. It is calculated using this formula:

$$Precision = \frac{TP}{(TP+FP)} \quad (1)$$

Recall shows how many of the real positive cases the model was able to find correctly. It is calculated with this formula:

$$Recall = \frac{TP}{(TP+FN)} \quad (2)$$

The F1 Score combines precision and recall into one number. It is the harmonic mean of the two and is helpful when the classes are not balanced. It is calculated using the following formula:

$$F1 = 2 * \frac{(Precision * Recall)}{(Precision+Recall)} \quad (3)$$

In this study, F1-confidence, precision-confidence, and recall-confidence curves were used to show how model performance changes when the confidence threshold gets higher. In general, a balanced F1 curve should show a peak in the middle. Irregular or flat curves may show that the model does not behave well at different confidence levels.

Precision-Recall (PR) curves were also generated. A curve close to the upper right corner is expected for good models. If the curve drops too fast, it may mean that the model struggles with some classes.

In object detection, mean Average Precision (mAP) is a standard way to evaluate how well the predicted bounding boxes match the real ones. It is calculated using different Intersection over Union (IoU) thresholds. The most common are mAP@50 and mAP@50-95:

$$\text{mAP}_T = \frac{1}{|T|} \sum_{t \in T} \left(\frac{1}{C} \sum_{c=1}^C AP_c^{\text{IoU} \geq t} \right) \quad (4)$$

$$\text{mAP}_{50} = \frac{1}{C} \sum_{c=1}^C AP_c^{\text{IoU} \geq 0.5} \quad (5)$$

$$\text{mAP}_{50:95} = \frac{1}{10} \sum_{k=1}^{10} \left(\frac{1}{C} \sum_{c=1}^C AP_c^{\text{IoU} \geq t_k} \right) \quad (6)$$

In this formula, T denotes the set of IoU thresholds used for evaluation, and $|T|$ represents the number of thresholds in that set. C is the total number of object classes. The term $AP_c^{\text{IoU} \geq t}$ refers to the average precision for class c at a specific IoU threshold t . This general formulation allows mAP to summarize model performance over multiple object categories and levels of spatial overlap, as described by Lin et al. (2014). This evaluation follows the COCO protocol and was computed using the Ultralytics library (Ultralytics, 2023; Wang et al., 2024).

IoU (Intersection over Union) is not directly reported in YOLO-based object detection but is used to decide whether a prediction is correct. For example, if the IoU between a predicted box and the real box is at least 0.5, the prediction is counted as a True Positive. In segmentation tasks, IoU is used as a direct performance metric:

$$\text{IoU} = \frac{|\text{Prediction} \cap \text{GroundTruth}|}{|\text{Prediction} \cup \text{GroundTruth}|} \quad (7)$$

Dice Coefficient is another common segmentation metric. It is more sensitive to small objects than IoU and is calculated as:

$$\text{Dice Coefficient} = \frac{2 \cdot |\text{Prediction} \cap \text{GroundTruth}|}{|\text{Prediction}| + |\text{GroundTruth}|} \quad (8)$$

During training, model optimization was monitored through different loss functions. These included box loss (location error), class loss (classification error), and Distribution Focal Loss (for better bounding box prediction). The training and evaluation process tracked several key metrics using Ultralytics YOLOv8's default logging system. These are train/box_loss, train/cls_loss, train/dfl_loss, val/box_loss, val/cls_loss, val/dfl_loss, metrics/precision(B), metrics/recall(B), metrics/mAP50(B), and metrics/mAP50-95(B) (Ultralytics, 2023). The loss curves were expected to decrease over time. If validation loss increases while training loss continues to drop, this may show overfitting.

Normalized confusion matrices were also used to analyze per-class accuracy. These helped understand which classes were often misclassified.

By combining all these metrics and visualizations, the performance of the models was compared in a reliable and interpretable way.

In addition to evaluating performance using common metrics, this study also applies K-fold cross validation to improve the reliability of the results.

K-fold cross validation is a method used to evaluate the performance of machine learning models in a more reliable way (Kohavi, 1995). This evaluation technique was first discussed and analyzed in detail by Kohavi (1995) and it is now widely used in many areas of machine learning and deep learning to ensure more robust performance comparisons. In this method, the dataset is split into K equal parts called folds. In each round, one fold is used for validation, while the other K−1 folds are used for training. This is repeated K times so that every part is used once as the validation set. After all rounds, the results are averaged. This helps reduce the impact of random data splits and gives more reliable performance estimates (Kohavi, 1995).

3.5 Material

The deep learning models used in this study were trained and evaluated under the following hardware and software configurations.

During the training, the barbun-cuda machine on Turkish National e-Science e-Infrastructure (TRUBA) was used. Added to TRUBA in 2018, this machine has 28 cores x 2 CPUs and 2 x Nvidia P100 GPUs. SPECfp_rate_base2006 is 1400 and the theoretical gflops are 2048Gflops & 9400Gflops. Total memory space is defined as 384 GB & 2x16 GB HBM2. (TRUBA, 2023)

To access this supercomputer on TRUBA, the Open OnDemand platform on the already installed interface server was used. Through this platform, it is possible to request access for different types of devices. During the trainings, it is possible to get 4 hours of access to the Jupyter Notebook - GPU option on barbun-cuda machines.

In addition, since the models used in our training are not models that can be completed within the 4-hour access time, the training code has been written in such a way that the training can be resumed from where it left off when the training is interrupted.

The code for training the YOLOv8 and YOLOv10 models was written in Python using the Ultralytics library (Ultralytics, 2023; Wang et al., 2024). The library provides a built-in interface, which allows easy training, validation, and testing for both YOLO versions (Ultralytics, 2023). The code used for training DeepLabV3 is also written in Python, utilizing the PyTorch library. The training code has been shared as open source on GitHub, as based on the implementation by Chen et al. (2017).

For YOLO trainings, 100 epochs were used. The parameter that sets the image size is given as `imgsz 1024`. The batch size was set to 12 and the device parameter was set to `"[0,1]"` to allow 2 GPUs to be used simultaneously in parallel. SGD was used for the optimizer. The default settings in the ultralytics library for blur and other similar settings in the preprocessing process have not been changed. The number of workers is set to 16, so that the training process can be completed optimally on the supercomputer.

For DeepLabV3 training, 100 epochs were used. Input images were resized to 1024x768, and standard augmentations such as horizontal flipping and normalization were applied using the Albumentations library. A custom dataset class was implemented to load ".jpg" images along with their corresponding "_mask.png" segmentation masks.

The DeepLabv3 model with a MobileNetV3-Large backbone was used and adapted for 4 output classes. If multiple GPUs are available, the model is parallelized with DataParallel

to utilize them effectively. The optimizer used is Adam with a learning rate of 0.0001, and the loss function is weighted CrossEntropy to balance class distribution.

The training and validation data were loaded with a batch size of 24 and 8 workers, using `pin_memory=True` for better performance. A checkpoint mechanism allows training to resume from the last saved state. After each epoch, loss values and model weights are saved. At the end of training, performance is evaluated on the validation set using per-class precision, recall, F1 score, and IoU metrics. Results are saved both as plots and JSON files, and the final trained model is stored for later use.

4. RESULTS

The dataset was used to train the YOLOv8 (Ultralytics, 2023), YOLOv10 (Wang et al., 2024), and DeepLabV3 (Chen et al., 2017) algorithms. Although the original image resolution was 2048x1024, all models were trained on 1024-pixel width images due to hardware performance limitations. The dataset was first split into three parts: 70% for training the model, 15% for validating it during training, and 15% for testing its final performance. Additionally, the K-fold cross-validation method was applied to evaluate model performance across different data segmentations.

In addition, the images in the test datasets are also analyzed per city, and the results are presented. The aim here is to understand whether images from different cities produce significant differences.

At the end of training, performance is evaluated on the validation set using per-class precision, recall, F1 score, and IoU metrics. Results are saved both as plots and JSON files, and the final trained model is stored for later use.

4.1 Training of Neural Networks

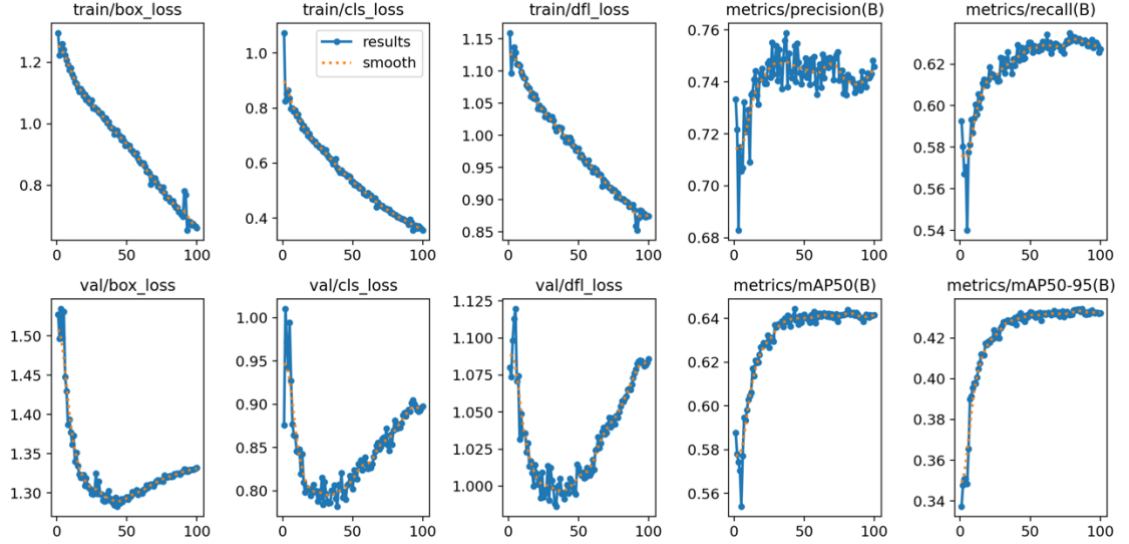


Figure 4.1: Results for YOLOv8 65-35

As shown in Figure 4.1, the 65-35 configuration yielded steadily decreasing training losses, indicating a well-converging optimization process. However, validation losses began to rise slightly after epoch 50, suggesting a mild overfitting trend despite the model's stable mAP50 plateauing at 0.64. The precision and recall metrics remained consistent, supporting the model's general ability to retain class-level balance during inference.

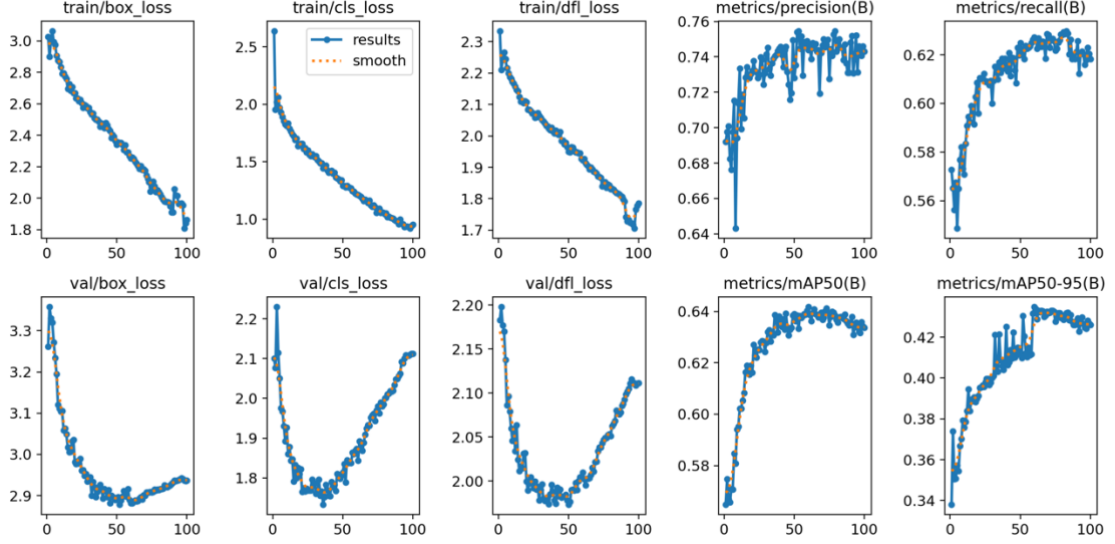


Figure 4.2: Results for YOLOv10 65-35

In the YOLOv10 (65-35 split) training process (Figure 4.2), the box, classification, and distribution focal losses consistently decreased across epochs, indicating a stable learning trend. However, validation losses reached their minimum around epoch 40-50 and gradually increased afterward, suggesting mild overfitting. Precision fluctuated around 0.74, while recall steadily improved from 0.55 to approximately 0.63, reflecting moderate detection sensitivity. The model achieved a final mAP50 of about 0.64 and mAP50-95 of 0.43, showing solid but slightly limited performance at stricter IoU thresholds. These results are largely comparable to YOLOv8 under the same split, though YOLOv10 exhibited slightly higher training and validation losses.

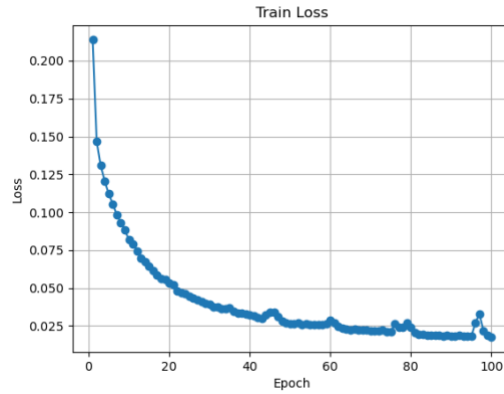


Figure 4.3: Train loss for DeepLabV3 65-35

The training loss of the DeepLabv3 model consistently decreased throughout the training process, indicating effective learning. Initially high, the loss dropped below 0.025 after approximately 20 epochs, suggesting rapid convergence during early training. As illustrated in Figure 4.3, in the later stages, the loss remained low and stable, reflecting a well-optimized model with minimal fluctuation.

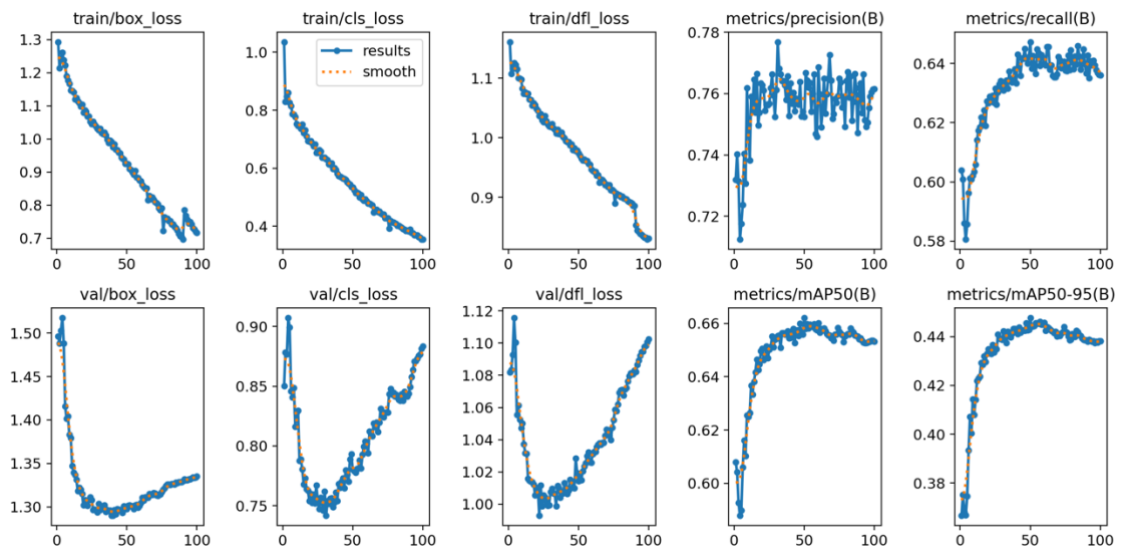


Figure 4.4: Results for YOLOv8 75-25

As seen in Figure 4.4, the training graphs show that the train/box_loss, train/cls_loss, and train/dfl_loss values decrease steadily throughout the training, which means the model is learning object locations, class labels, and distribution-related features effectively. On the other hand, the box loss, class loss, and distribution focal loss on the validation set decrease until about epoch 40, but then they start to increase again. This suggests that the model begins to overfit the training data and struggles to generalize well on the validation set. The precision metric stays around 0.74 but waves, while the recall steadily increases and levels off near 0.645, showing that the model becomes better at finding true objects but its confidence in predictions may vary between classes. The overall performance is reflected in the mAP@0.5 value reaching about 0.66, and mAP@0.5:0.95 reaching 0.44, which indicates moderate to good detection accuracy. In summary, the model learns effectively on the training set, but signs of overfitting on the validation set suggest that class imbalance or limited data diversity may be affecting generalization.

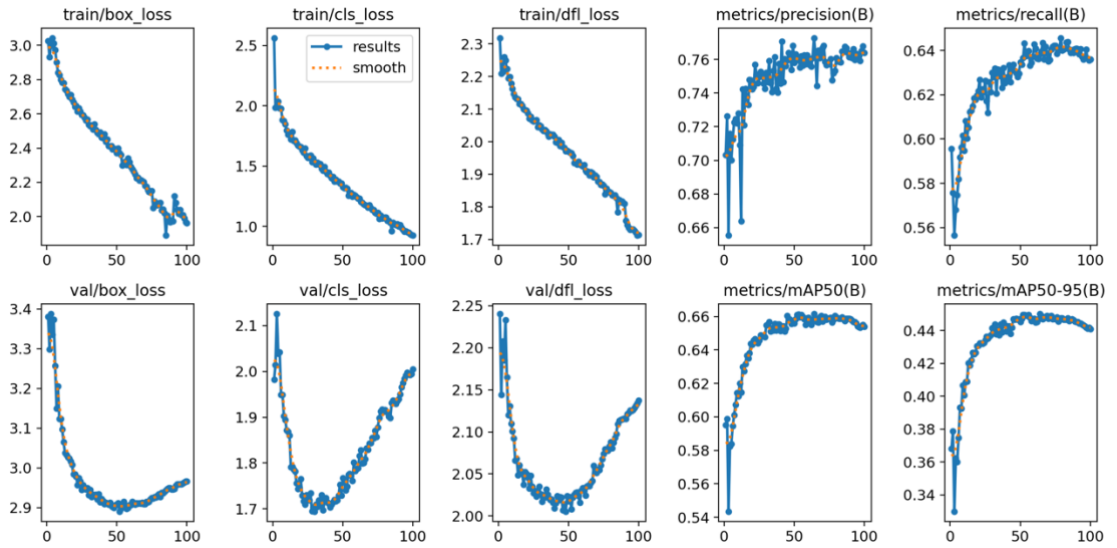


Figure 4.5: Results for YOLOv10 75-25

As illustrated by Figure 4.5, during training, YOLOv10 showed a consistent decrease in all training losses, while validation losses reached their minimum around epoch 40 before increasing, indicating overfitting. Compared to YOLOv8, YOLOv10 achieved similar

recall and mAP values (0.645 and 0.662 respectively), but delivered slightly higher and more stable precision, suggesting improved prediction consistency. Despite comparable final accuracy, YOLOv10 exhibited smoother metric curves, reflecting a more balanced learning behavior.

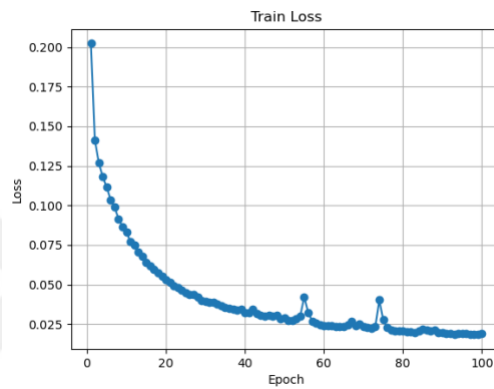


Figure 4.6: Train loss graph for DeepLabV3 75-25

Figure 4.6 clearly demonstrates that the training loss curve of DeepLabv3 shows a steady and consistent decline from 0.20 to 0.017 across 100 epochs, indicating effective convergence without signs of overfitting. Minor spikes observed around epochs 60 and 75 are likely due to batch variability but do not affect the overall learning trend. Compared to YOLOv10, DeepLabv3 reaches convergence with significantly lower loss values, reflecting the difference in task complexity between segmentation and detection models.

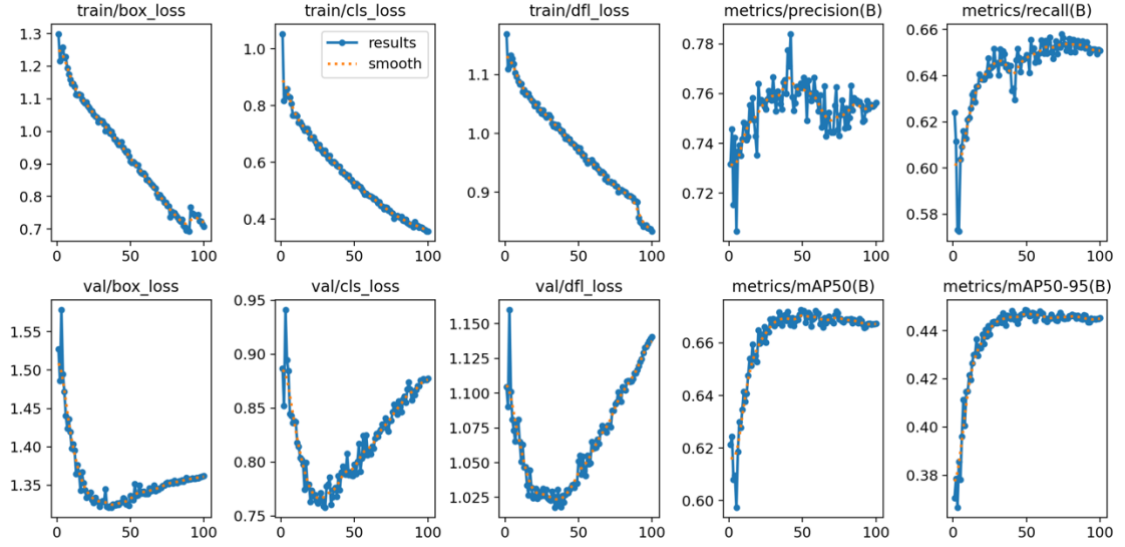


Figure 4.7: Results for YOLOv8 70-30

Figure 4.7 presents the training results of the YOLOv8 model using a 70-30 train-validation split. The training losses consistently decreased over the epochs, while the validation losses reached their lowest point around epoch 40 before gradually increasing, indicating a mild overfitting trend. Nevertheless, the model achieved stable precision around 0.76, recall approaching 0.66, and final mAP@0.5 and mAP@0.5:0.95 scores of approximately 0.67 and 0.45, respectively, demonstrating effective detection performance.

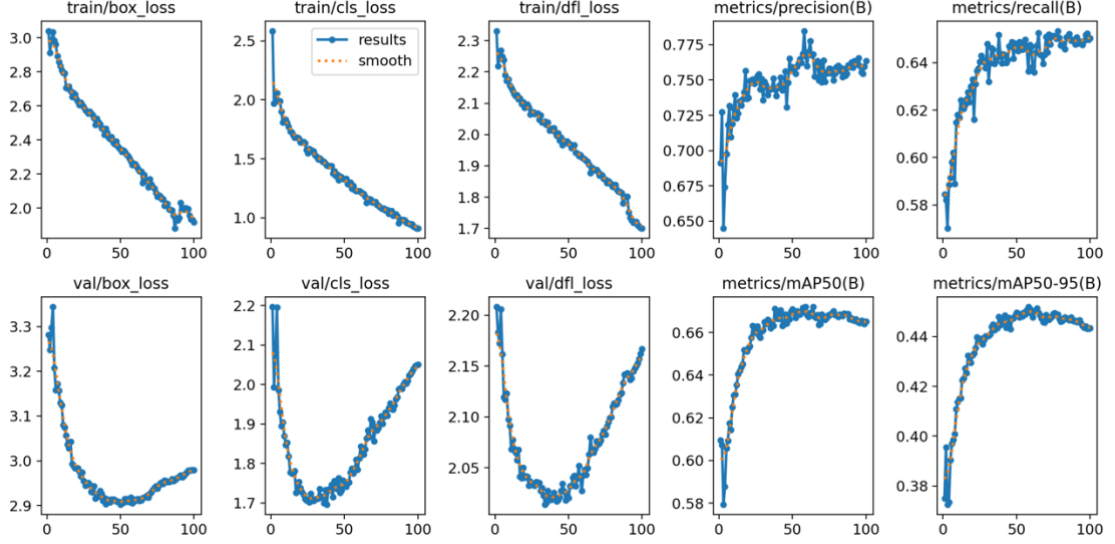


Figure 4.8: Results for YOLOv10 70-30

Figure 4.8 presents the training and validation performance of the YOLOv10 model using a 70-30 data split. All training losses steadily decreased throughout the training process, reflecting consistent learning and convergence. Validation losses reached their minimum around epoch 40 and then gradually increased, indicating a mild overfitting pattern like what was observed in YOLOv8 under the same split (Figure 11). Despite slightly higher training and validation loss values than YOLOv8, YOLOv10 achieved comparable or slightly better detection metrics, with precision stabilizing near 0.76, recall improving up to 0.65, and final $\text{mAP}@0.5$ and $\text{mAP}@0.5:0.95$ scores of approximately 0.667 and 0.447, respectively.

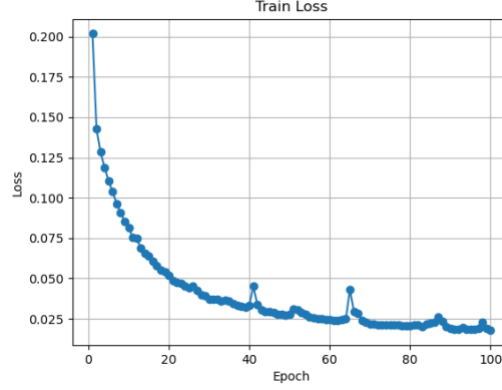


Figure 4.9: Train loss for DeepLabV3 70-30

Figure 4.9 illustrates the training loss curve of the DeepLabV3 model under a 70-30 train-validation split. The loss steadily declined from approximately 0.20 to below 0.02 over 100 epochs, indicating effective convergence and a smooth learning process. Minor spikes observed around epochs 55 and 70 are likely due to temporary fluctuations in batch composition, but they do not significantly affect the overall downward trend.

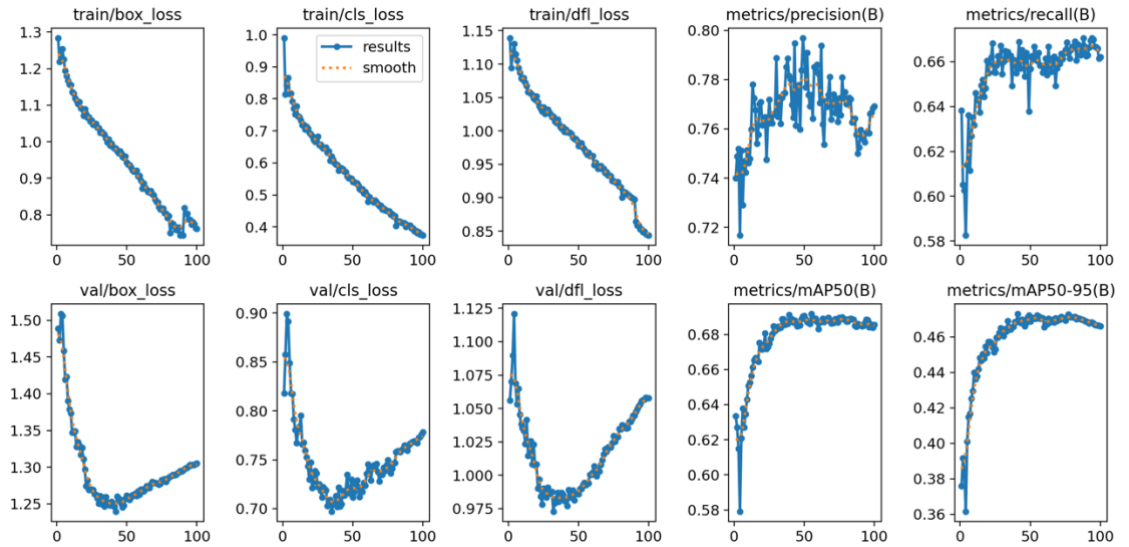


Figure 4.10: Results for YOLOv8 95-5

As depicted in Figure 4.10, the box, class, and distribution focal losses in both the training and validation sets decrease steadily, showing that the model is learning in a stable way. However, validation losses begin to rise slightly after around epoch 50, suggesting potential overfitting. Precision improves over time, although it shows some fluctuations, stabilizing around 0.76. Recall increases more consistently, leveling off near 0.66. Both $\text{mAP}@0.5$ and $\text{mAP}@0.5:0.95$ demonstrate continuous growth, reaching approximately 0.685 and 0.475 respectively, which reflects the model's improving detection performance.

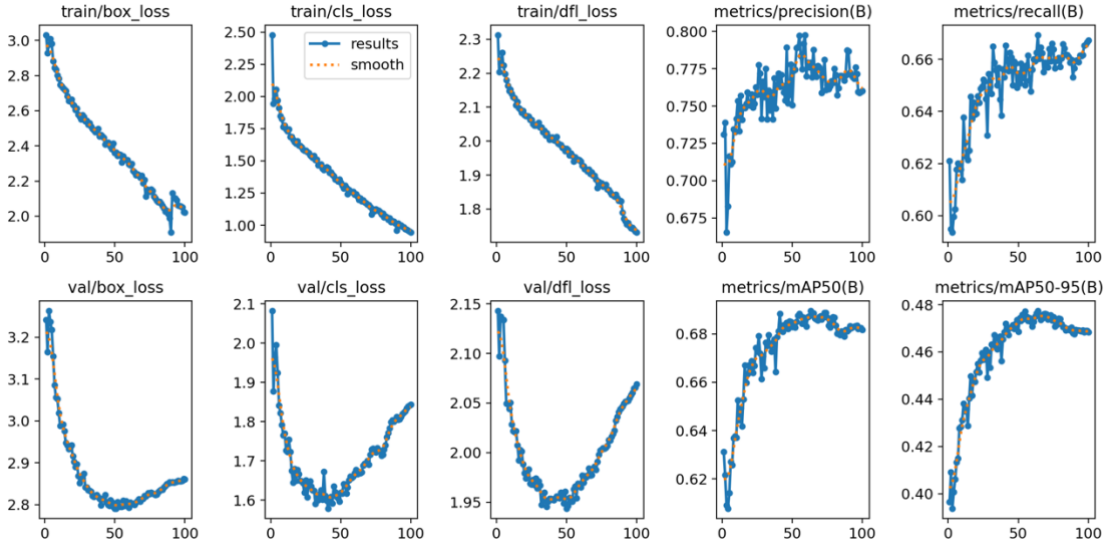


Figure 4.11: Results for YOLOv10 95-5

As illustrated in Figure 4.11, the YOLOv10 model trained with 95-5 split exhibits a consistent decrease in training losses and achieves a final $\text{mAP}@0.5$ of approximately 0.68. Validation losses show a slight upward trend after epoch 50, suggesting potential early overfitting. Compared to YOLOv8 under the same data distribution, YOLOv10 achieves very similar performance in both $\text{mAP}@0.5$ and $\text{mAP}@0.5-0.95$. However, YOLOv10 demonstrates a more stable learning pattern with less fluctuation in precision curves. These results highlight YOLOv10's robustness and reliability, even under limited validation data.

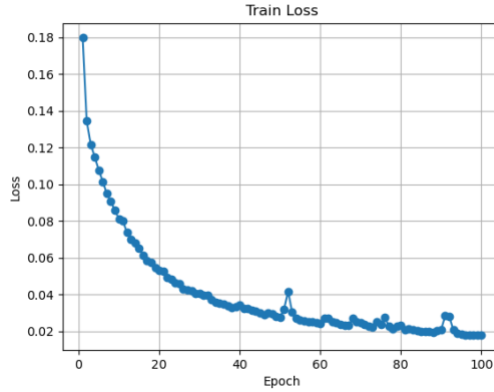


Figure 4.12: Train loss graph for DeepLabV3 95-5

Figure 4.12 shows the training loss curve of the DeepLabV3 model trained with a 95-5 train-validation split. The loss decreases consistently from approximately 0.18 to below 0.02 across 100 epochs, indicating efficient and stable convergence. Minor spikes around epochs 50 and 85 likely result from random mini-batch fluctuations and do not significantly affect the overall trend.

When compared to earlier DeepLabV3 configurations, the 95-5 split demonstrates a slightly smoother and faster convergence. The final training loss in Figure 16 is marginally lower than in the other cases, which may reflect the benefits of a larger training dataset. However, this comes at the cost of reduced validation feedback, which can make it more difficult to assess the model's generalization capability.

4.2 Detection

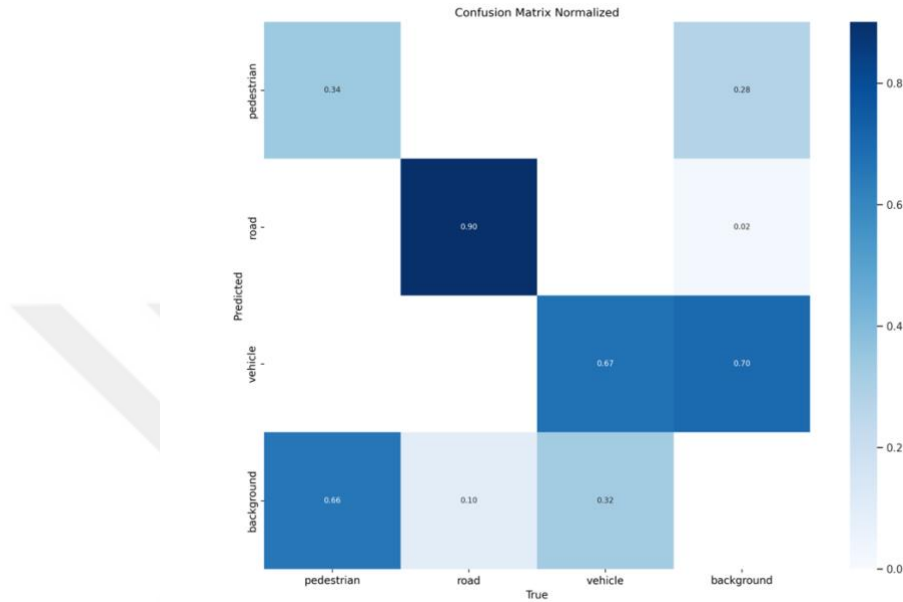


Figure 4.13: Confusion Matrix Normalized for YOLOv8 65-35

Figure 4.13 reveals that the normalized confusion matrix for YOLOv8 under the 65-35 split shows high accuracy in the road class, with 90% of the road pixels correctly classified. The vehicle class also maintains relatively high performance (70%), although some confusion with the background remains. The pedestrian class, however, is significantly misclassified, with 66% of its pixels predicted as background. This pattern suggests that YOLOv8 has limited discriminative ability for small or low-contrast objects like pedestrians, especially under challenging segmentation conditions.

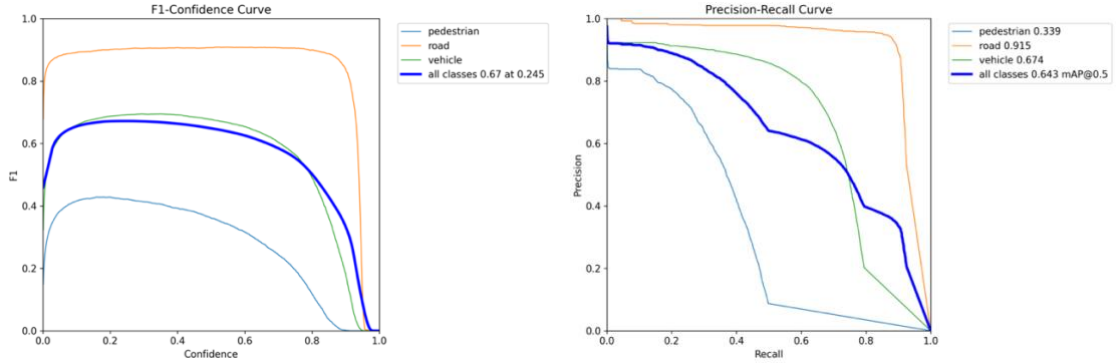


Figure 4.14: F1-Curve and PR-Curve for YOLOv8 65-35

As presented in Figure 4.14, in the 65-35 data split, YOLOv8 reached a peak average F1 score of 0.67 at a confidence threshold of 0.245. The model performed particularly well in the road (~ 0.91) and vehicle (~ 0.70) categories, whereas pedestrian detection remained the weakest, with a maximum F1 of approximately 0.45 followed by a rapid decline at higher confidence thresholds. Compared to configurations with larger training sets, this setup resulted in only a slight reduction in overall F1, while maintaining similar class-level performance characteristics.

YOLOv8 achieved an overall $\text{mAP}@0.5$ of 0.643 in the 65-35 split, with strong class-wise separation in the PR domain. The road class yielded excellent performance (0.915), while the vehicle class maintained a moderate balance (0.674) before dropping in precision at higher recall levels. Pedestrian detection remained the weakest link with a mAP of 0.339, indicating frequent false positives and missed instances across recall thresholds.

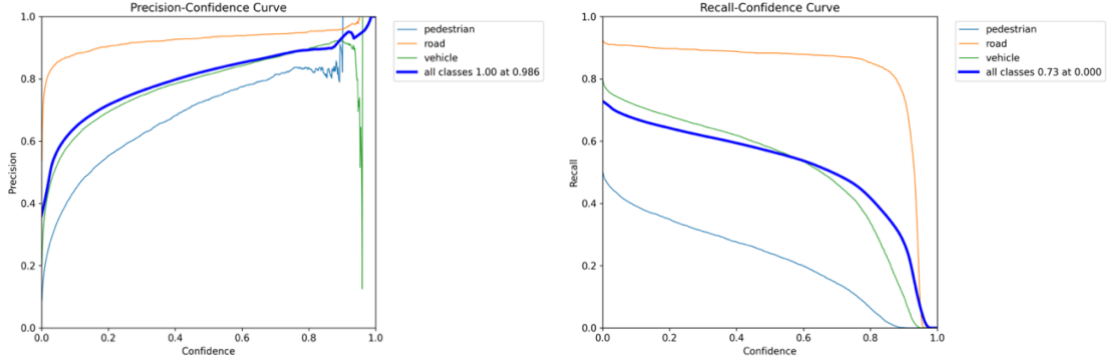


Figure 4.15: P-Curve and R-Curve for YOLOv8 65-35

Figure 4.15 demonstrates that in the 65-35 configuration, YOLOv8 achieved class-wise precision values that steadily increased with confidence, reaching 1.00 at a threshold of 0.986 for all classes. While the road class showed highly stable precision, the vehicle curve experienced a sharp drop near the end, likely due to a small number of high-confidence predictions containing errors. This highlights that although the model is generally accurate when confident, it may become unstable at extreme thresholds in non-dominant classes.

In the 65-35 configuration, YOLOv8 achieved a maximum average recall of 0.73 across all classes at a confidence level of 0.0. The road class maintained high recall values (>0.90) even at moderate confidence thresholds, whereas the vehicle class showed a steady decline beyond 0.85. Recall of the Pedestrian class rapidly deteriorated above 0.6 confidence, reflecting the model's difficulty in consistently detecting this class under stricter thresholds.

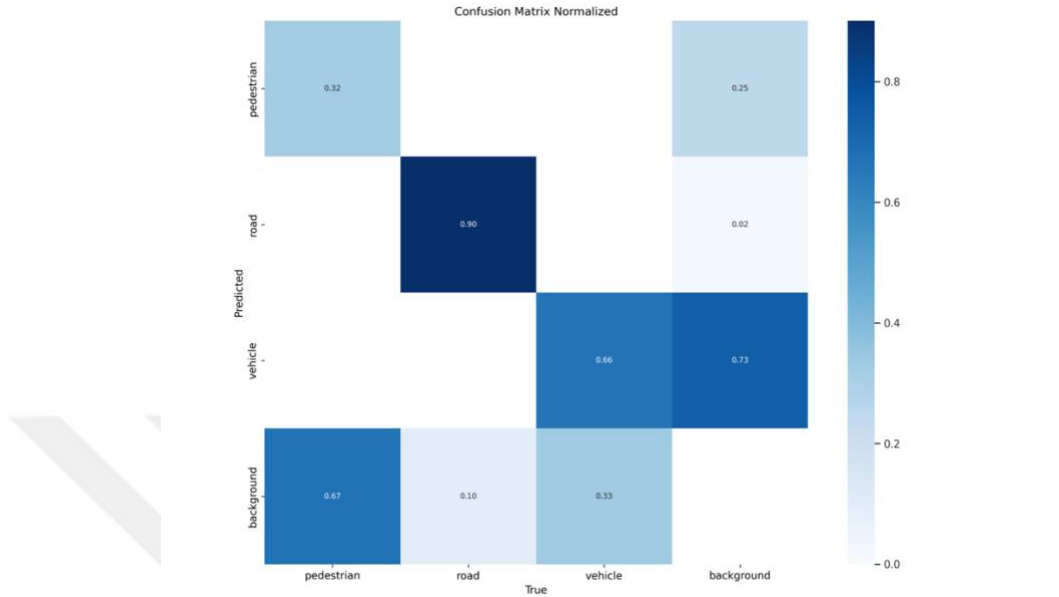


Figure 4.16: Confusion Matrix Normalized for YOLOv10 65-35

As shown in Figure 4.16, in the 65-35 configuration, YOLOv10 maintained a strong performance on the road class with a 0.90 true positive rate, consistent with YOLOv8. While vehicle classification remained stable at 0.66, the pedestrian class showed a slight decline to 0.32, along with increased confusion with the background. These results suggest that YOLOv10's ability to distinguish smaller or less distinct objects like pedestrians may require further refinement.

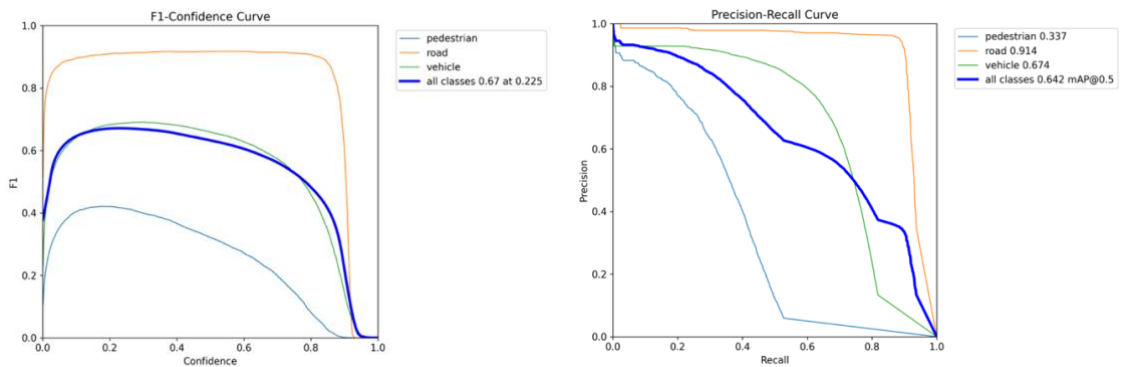


Figure 4.17: F1-Curve and PR-Curve for YOLOv10 65-35

Figure 4.17 confirms that YOLOv10 achieved a maximum F1 score of 0.67 at a confidence threshold of 0.225, nearly identical to YOLOv8's performance under the same 65-35 setting. The road class continued to dominate with an F1 score above 0.90, while the pedestrian class remained the most challenging, peaking at around 0.43. Despite these class-wise differences, both models demonstrated similar overall balance between precision and recall.

In the PR curve of YOLOv10 (65-35 split), the overall mAP@0.5 is 0.642, with the road class maintaining a strong precision level above 0.90 across nearly all recall levels. The vehicle class achieved 0.674 mAP, but its precision declined as recall increased, indicating instability in detection certainty. Compared to YOLOv8 with the same data split, both models performed similarly overall, but YOLOv10 showed a more consistent performance for the road class.

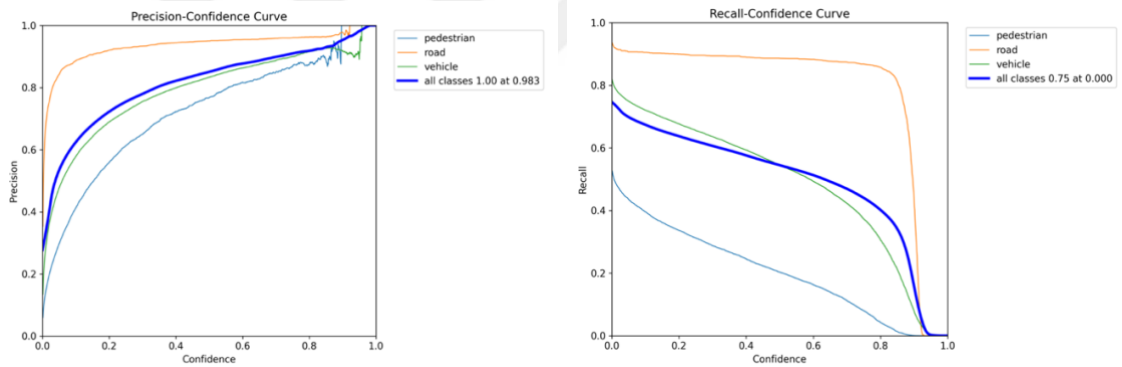


Figure 4.18: P-Curve and R-Curve for YOLOv10 65-35

As depicted in Figure 4.18, in the precision-confidence curve of YOLOv10 trained with a 65-35 split, the model achieved a maximum average precision of 1.00 at a confidence threshold of 0.983. The road class exhibited consistently high precision (~ 0.98), while the vehicle and pedestrian classes showed gradual improvements in precision as confidence increased. Compared to YOLOv8, YOLOv10 demonstrated slightly more stable behavior in the vehicle class, with reduced fluctuations at higher confidence thresholds.

In the recall-confidence curve of YOLOv10 (65-35), the overall recall reached 0.75 at a confidence threshold of 0.0, showing high sensitivity at low thresholds. The road class

preserved recall above 0.85 across a wide confidence range, while the vehicle class showed a steady decline and the pedestrian class dropped significantly, falling below 0.3 even at moderate confidence levels. Compared to YOLOv8, YOLOv10 achieved slightly better overall recall and more stable performance in the road class.

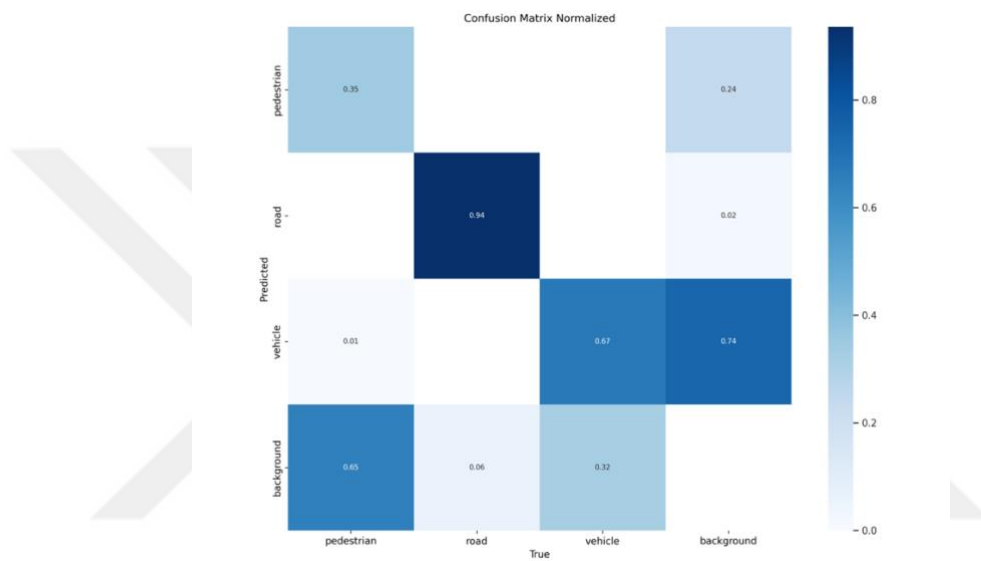


Figure 4.19: Confusion Matrix Normalized for YOLOv8 75-25

As highlighted in Figure 4.19, the normalized confusion matrix shows that the model is quite successful in this class with 94% correct prediction in the road class. Although 67% accuracy is achieved in the vehicle class, the confusion with background is particularly noticeable. In general, the model performs strongly in the clear classes, while there is room for improvement in the instances that make class distinction difficult.

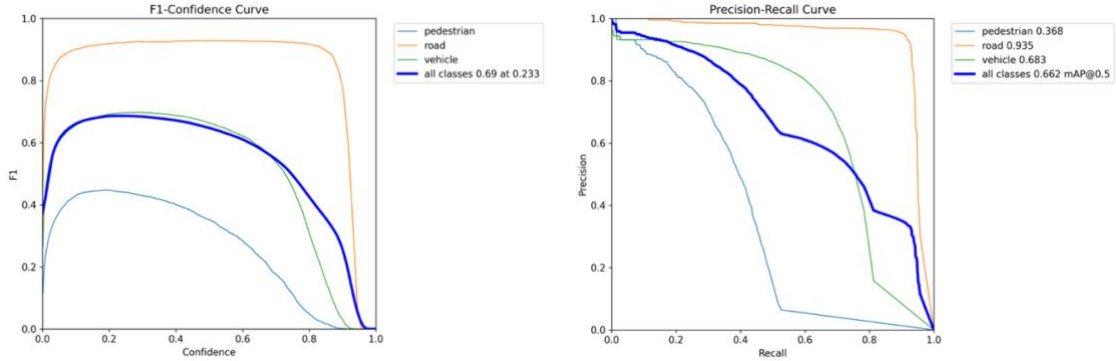


Figure 4.20: F1-Curve and PR-Curve for YOLOv8 75-25 split

As shown by Figure 4.20, the F1-confidence curve shows that YOLOv8 achieves its highest average F1 score of 0.69 at a confidence threshold of 0.233. The road class consistently outperforms the others with an F1 score exceeding 0.90 across most confidence levels. In contrast, the pedestrian class demonstrates a steep decline in performance at higher confidence values, highlighting difficulty in producing reliable predictions for less prominent objects. Overall, the model maintains more balanced and stable F1 performance at lower confidence thresholds, indicating better generalization under less strict detection criteria. The PR curve presents the model's class-specific detection capabilities. The road class achieves a high and stable performance, with an area under the curve of 0.935, which indicates strong confidence and low error rates. In contrast, the precision in the vehicle and pedestrian classes decreases as recall increases, especially in the pedestrian class, where limited sample representation leads to more false positives. This trend becomes more evident at lower confidence thresholds. The model achieves a mAP at IoU 0.5 (mAP@0.5) of 0.662, which reflects a generally reliable detection performance across all classes.

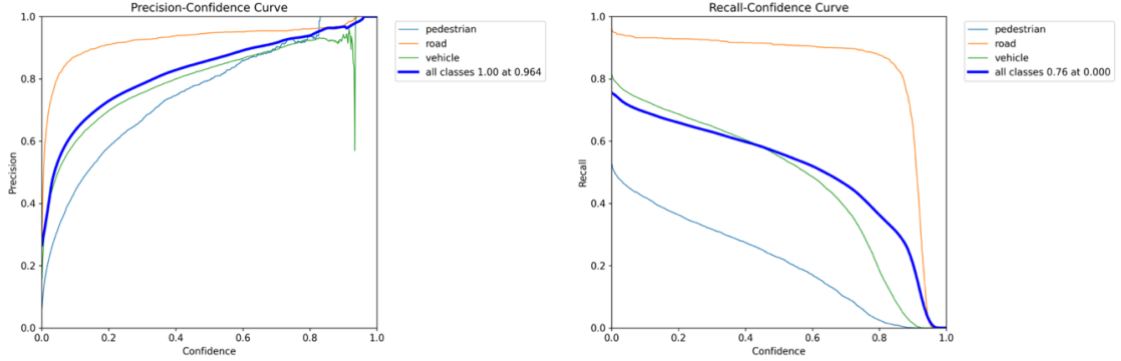


Figure 4.21: P-Curve and R-Curve for YOLOv8 75-25

As seen in Figure 4.21, the precision-confidence curve for YOLOv8 trained with 75-25 split reveals that the model achieves perfect precision (1.00) at a confidence threshold of 0.964. While the road class maintains consistently high precision across all confidence levels, reflecting the model's strong ability to identify well-defined surfaces, the vehicle class exhibits noticeable fluctuations in the upper confidence range. These variations likely stem from statistical noise due to a limited number of high-confidence samples in that class. Nevertheless, the overall trend of the model's precision increases steadily, and reliable performance is observed even at lower confidence levels, particularly for dominant classes.

The recall-confidence curve illustrates that the model retrieves fewer true positives as the confidence threshold increases, which leads to a gradual decrease in overall recall. The road class demonstrates robust performance, maintaining a recall rate close to 90% even at high confidence levels. In contrast, the vehicle and pedestrian classes experience a rapid decline in recall, particularly in the pedestrian class where recall begins to drop sharply at low confidence levels. This behavior suggests that the model has limited confidence in detecting pedestrian instances, likely due to class imbalance or subtle visual features. Across all classes, the maximum recall is achieved at a confidence threshold of 0.0, corresponding to an average recall of 76%, which reflects the model's upper bound in terms of sensitivity.

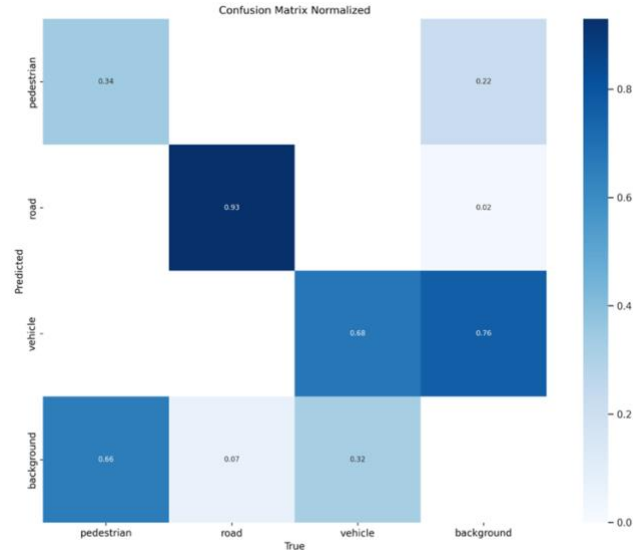


Figure 4.22: Confusion Matrix Normalized for YOLOv10 75-25 split

Figure 4.22 highlights that the YOLOv10 model achieves high accuracy in the road class and reasonable success in the vehicle class, but experiences serious confusion in the pedestrian class. The dominant use of the background class indicates that the model is cautious in low confidence predictions and tends to miss some classes. Although it is like YOLOv8 in terms of its general structure, there is room for improvement in terms of class-based discrimination power, especially in the pedestrian class.

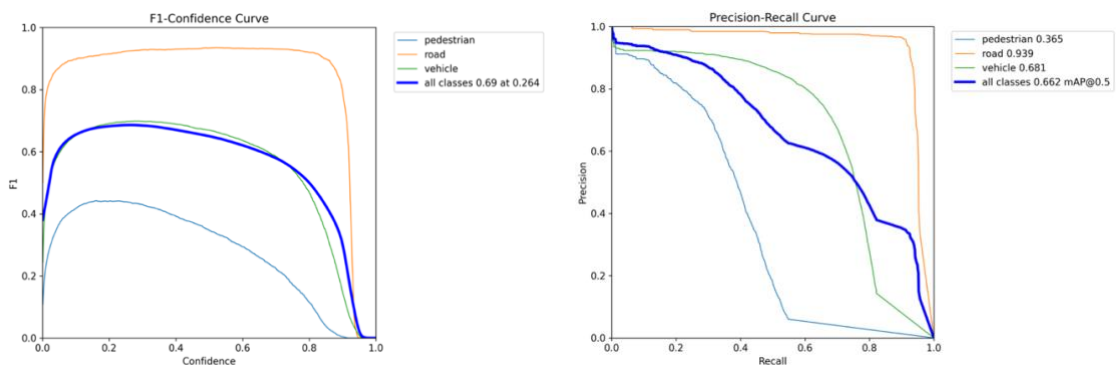


Figure 4.23: F1-Curve and PR-Curve for YOLOv10 75-25

As depicted in Figure 4.23, the F1-confidence curve of YOLOv10 under the 75-25 split shows that the model achieves its maximum average F1 score of 0.69 at a confidence threshold of 0.264. The road class stands out with a high and stable F1 score above 0.90, indicating reliable detection across various confidence levels. While the vehicle class demonstrates moderate F1 performance, the pedestrian class remains considerably lower, highlighting the model's limited discriminative capacity for small or less distinct objects. The PR curve of YOLOv10 under the 75-25 split shows that the model achieves its best class-wise performance in the road category with a $mAP@0.5$ of 0.939, slightly outperforming YOLOv8's 0.935. The vehicle class shows nearly identical performance between models, with 0.681 for YOLOv10 and 0.683 for YOLOv8. Pedestrian detection remains the weakest aspect, with a class-wise mAP of 0.365, indicating that YOLOv10 does not significantly improve over its predecessor in handling smaller or less distinct objects. Although the average $mAP@0.5$ remains unchanged at 0.662, YOLOv10 displays a slightly more stable PR distribution across recall values.

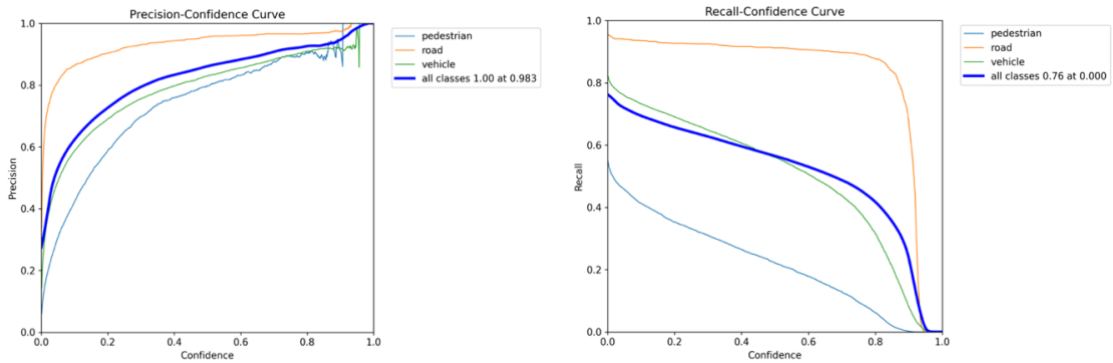


Figure 4.24: P-Curve and R-Curve for YOLOv10 75-25

As observed in Figure 4.24, the precision-confidence curve of YOLOv10 trained with a 75-25 split reveals a steady increase in precision across all classes, reaching a maximum of 1.00 at a confidence threshold of 0.983. The road class shows consistently high precision values above 0.95 across the full confidence spectrum, reflecting the model's strong ability to correctly classify clearly defined structures. While the vehicle and pedestrian classes demonstrate lower precision overall, they follow an upward trend as

confidence increases, with the vehicle class approaching 0.90 and the pedestrian class reaching just over 0.80 at higher thresholds.

The recall-confidence curve shows that the model reaches a maximum average recall of 0.76 at a confidence threshold of 0.00, like YOLOv8. While recall for the road class remains consistently high (above 0.90) in both models, pedestrian recall is limited to 0.55 and drops below 0.10 after 0.6 confidence, with no notable improvement over YOLOv8. Although the general trend is preserved, YOLOv10 displays a slightly smoother recall decline in the vehicle class, suggesting more stable detection behavior.

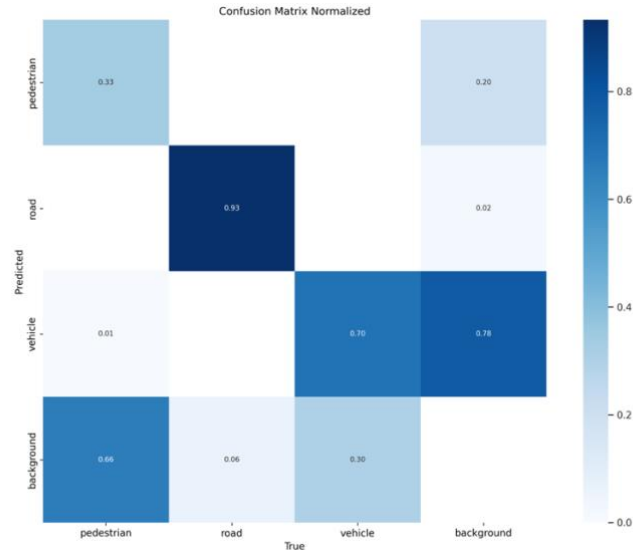


Figure 4.25: Confusion matrix normalized for YOLOv8 70-30

As shown in Figure 4.25, the normalized confusion matrix for YOLOv8 trained with a 70-30 split shows strong class separation for the road category, with a 93% correct prediction rate. While vehicle classification remains reliable (70%), the pedestrian class still suffers from significant confusion, with only 33% correctly identified and a large portion misclassified as background. Compared to previous splits such as 75-25 and 65-35, this configuration offers marginal gains in vehicle and background distinction, but pedestrian detection remains consistently weak.

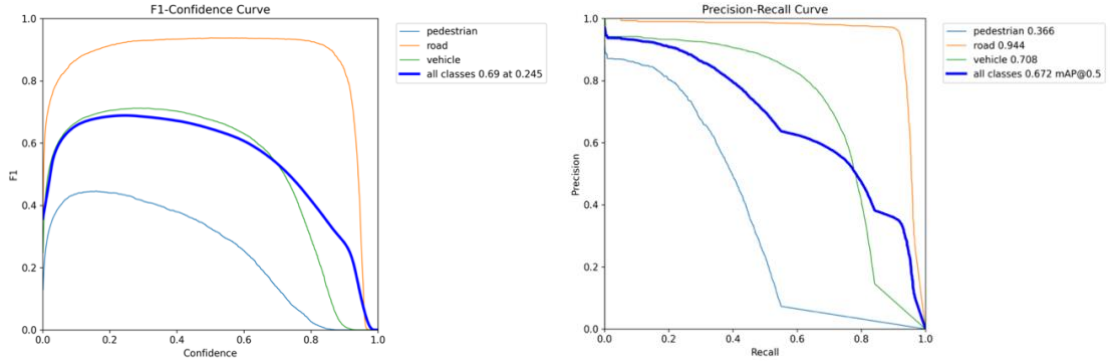


Figure 4.26: F1-Curve and PR-Curve for YOLOv8 70-30

As illustrated in Figure 4.26, the F1-confidence curve of YOLOv8 with 70-30 split shows that the model achieves its peak F1 score of 0.69 at a confidence threshold of 0.245. While the road class consistently reaches above 0.90, the vehicle class maintains a moderate level, and pedestrian performance remains significantly lower. Compared to previous configurations, the overall F1 performance improves slightly, reflecting a more stable trade-off between precision and recall.

The PR curve for YOLOv8 with a 70-30 split shows strong class-wise performance, particularly in the road class with a high $\text{mAP}@0.5$ of 0.944. Vehicle detection yields a moderately high mAP of 0.708, while pedestrian detection remains limited at 0.366, consistent with earlier splits. Compared to 75-25 and 65-35 configurations, the 70-30 distribution leads to slightly improved mAP across all classes, reflecting the benefit of a more balanced validation set.

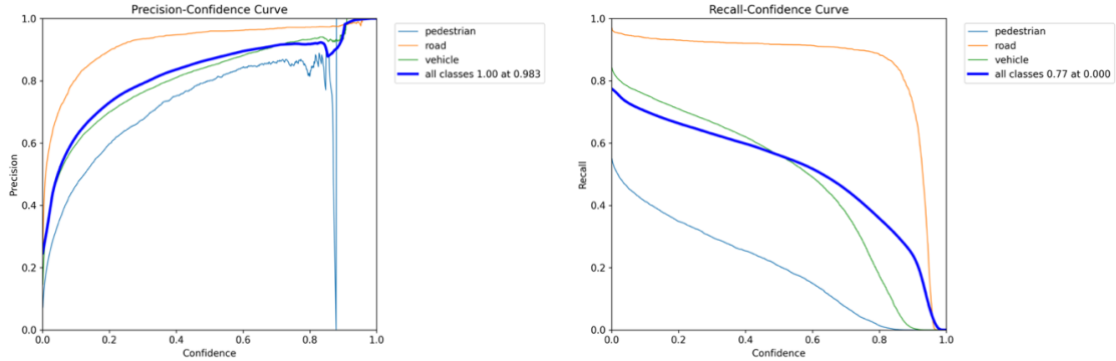


Figure 4.27: P-Curve and R-Curve for YOLOv8 70-30

As presented in Figure 4.27, the precision-confidence curve of YOLOv8 trained on the 70-30 split shows a smooth increase in precision across all classes, reaching a perfect average precision of 1.00 at a threshold of 0.983. The road class maintains high precision throughout, while the vehicle class shows slightly more stable precision than in the 75-25 split. Although pedestrian precision improves slightly compared to 65-35, sharp fluctuations at higher confidence values indicate that the model still lacks reliability in this class.

The road class retains excellent recall above 0.90 across a wide confidence range, while the vehicle class exhibits a slightly smoother decline than in detection models. Pedestrian recall, although still the lowest among classes, demonstrates a more gradual drop compared to previous YOLO-based results, indicating relatively better sensitivity under uncertain conditions.

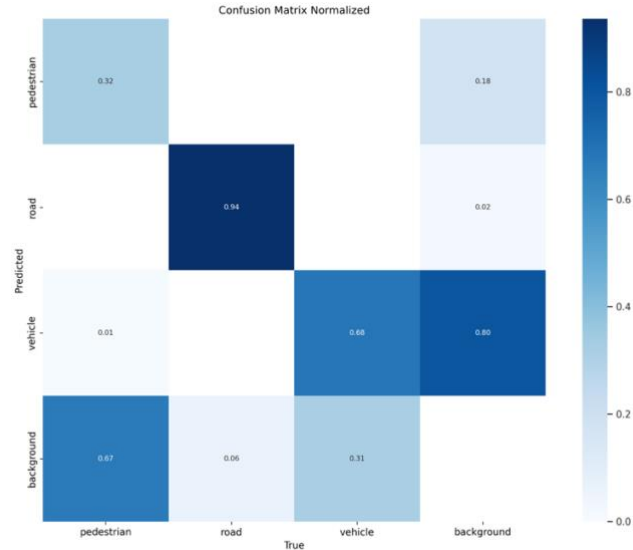


Figure 4.28: Confusion matrix normalized for YOLOv10 70-30

As shown in Figure 4.28, the normalized confusion matrix for the YOLOv10 model with the 70-30 data split indicates a strong performance in the road class, achieving a 94% correct classification rate. While the vehicle class shows moderate accuracy at 68%, the pedestrian class remains challenging, with only 32% correctly identified and considerable confusion with the background. Compared to YOLOv8, the results are largely consistent, though YOLOv10 appears slightly more effective at distinguishing background instances.

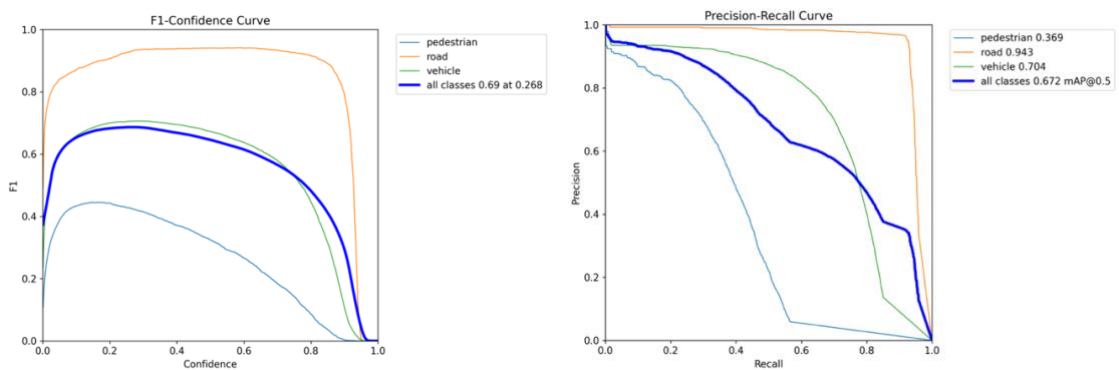


Figure 4.29: F1-Curve and PR-Curve for YOLOv10 70-30

According to the results shown in Figure 4.29, the F1-Confidence curve for YOLOv10 in the 70-30 split shows a peak average F1 score of 0.69 at a confidence threshold of 0.268. The road class consistently outperforms the others with F1 scores above 0.90, while the vehicle class maintains moderate performance, peaking near 0.70. Despite slight gains over previous splits, pedestrian detection remains challenging, with its F1 curve peaking low and dropping sharply, indicating persistent difficulties in reliable identification of this class.

The PR curve for YOLOv10 in the 70-30 data split demonstrates strong performance in the road class with a $\text{mAP}@0.5$ of 0.943, which is among the highest across all configurations. Vehicle detection shows a balanced trade-off, achieving a mAP of 0.704, while pedestrian detection remains the weakest at 0.369, indicating persistent challenges in distinguishing this class. Compared to YOLOv8, YOLOv10 offers marginal gains in road and vehicle classes, but pedestrian-level improvements remain minimal.

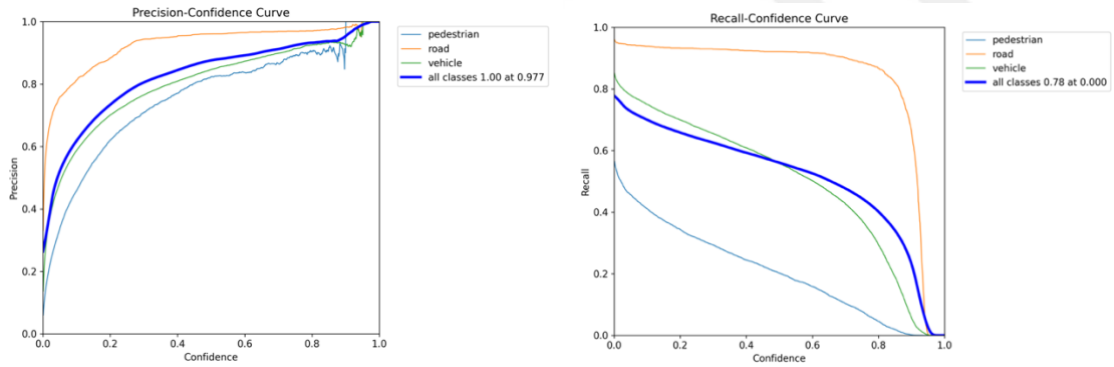


Figure 4.30: P-Curve and R-Curve for YOLOv10 70-30

As depicted in Figure 4.30, the precision-confidence curve for YOLOv10 under the 70-30 split shows a steady increase in precision as confidence rises, reaching a perfect score of 1.00 at a threshold of 0.977. The road class demonstrates highly consistent and high precision across nearly the entire range, confirming the model's strength in detecting clearly defined features. While the vehicle and pedestrian classes improve with increasing confidence, the pedestrian curve remains lower, indicating persistent difficulty in making confident and accurate predictions for this category compared to YOLOv8.

The recall-confidence curve for YOLOv10 in the 70-30 split shows that the model achieves its highest average recall (0.78) at a confidence threshold of 0.0, indicating high sensitivity when no filtering is applied. The road class maintains excellent recall across the confidence range, demonstrating strong and consistent detection capability. In contrast, the pedestrian class recall declines rapidly as confidence increases, like previous splits, reflecting the model's ongoing difficulty in reliably detecting smaller or less distinct objects.

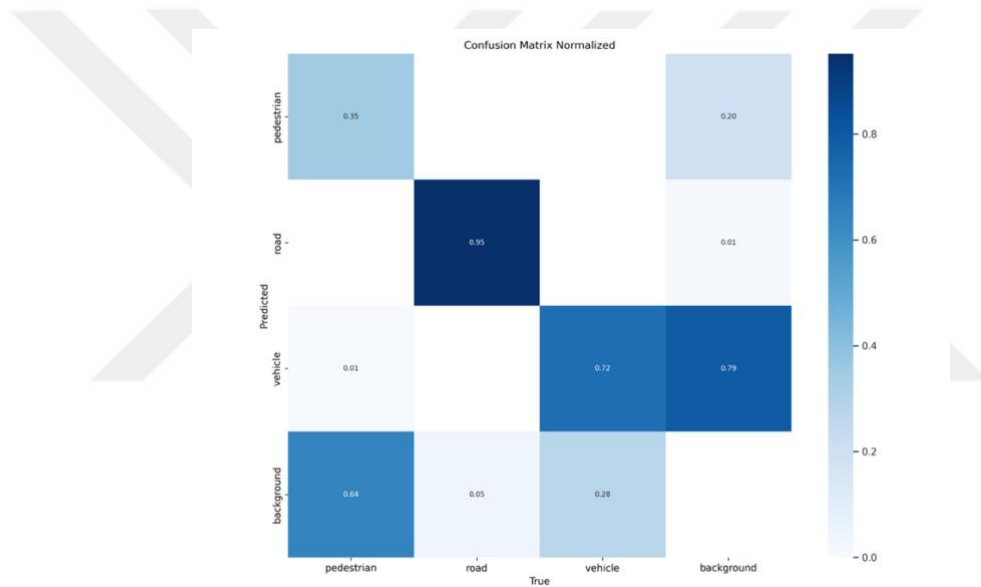


Figure 4.31: Confusion matrix for YOLOv8 95-5

According to the results illustrated in Figure 4.31, the confusion matrix of the YOLOv8 model trained with a 95-5 data split shows very strong performance on the road class, with a high prediction accuracy of 95%. Vehicle detection is good at 72%, though a notable portion is misclassified as background. However, pedestrian recognition remains weak, with only 35% accuracy and a high confusion rate with the background class.

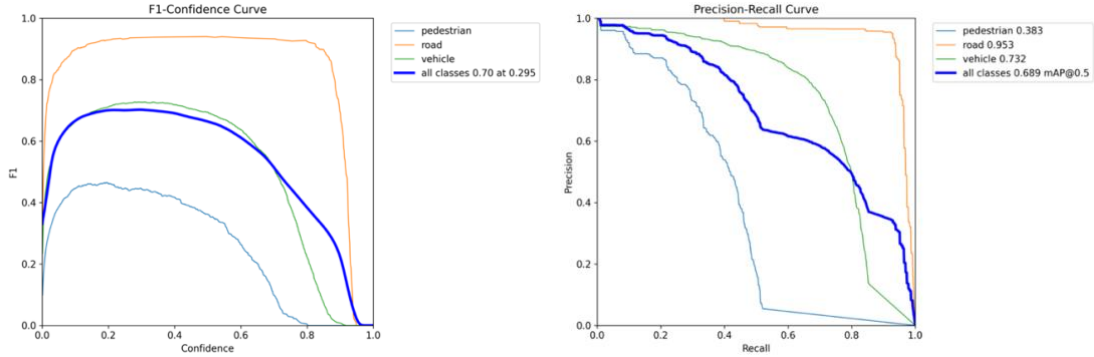


Figure 4.32: F1-Curve and PR-Curve for YOLOv8 95-5

As shown in Figure 4.32, in the F1-Confidence Curve of the YOLOv8 model trained with a 95-5 data split, the overall F1 score reaches its peak at 0.70 around a confidence threshold of 0.295. The road class maintains consistently high performance across different confidence levels, indicating strong detection reliability. In contrast, the vehicle class achieves a moderate F1 score, while the pedestrian class shows the weakest performance among the three. The pedestrian curve declines sharply at higher confidence levels, which suggests that the model struggles to identify this class confidently likely due to fewer samples or higher variability in its visual characteristics within the dataset. In the PR curve for YOLOv8 (95-5 split), the road class shows excellent performance with a precision of 0.953, followed by the vehicle class at 0.732. The pedestrian class remains the weakest, with precision dropping steeply and averaging around 0.383. The overall mAP@0.5 reaches 0.689, one of the highest values across all splits. This suggests that the model, despite having a very small validation set, produces highly confident predictions.

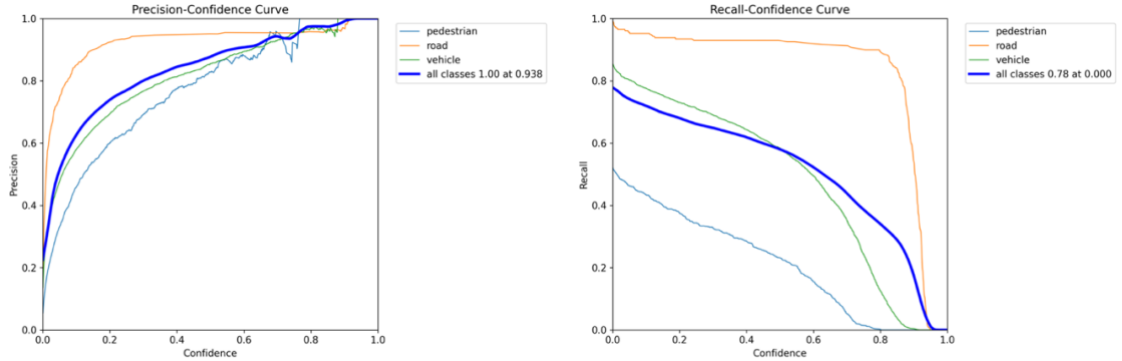


Figure 4.33: P-Curve and R-Curve for YOLOv8 95-5

As depicted in Figure 4.33, in the Precision-Confidence Curve for the YOLOv8 model (95-5 split), the overall precision for all classes peaks at 1.00 when the confidence threshold reaches approximately 0.938. The road class maintains high and stable precision throughout the range, reflecting strong prediction confidence. In contrast, the vehicle and pedestrian classes show more gradual increases, with the pedestrian curve exhibiting a sharp jump after 0.85 confidence, suggesting the model avoids classifying pedestrians unless it is very certain. This indicates high precision for pedestrians, but at the cost of recall, since many pedestrian instances are likely missed at lower confidence levels.

The Recall-Confidence curve shows that the road class maintains a consistently high recall above 0.9 across nearly all confidence levels. The vehicle class also performs well, starting around 0.8 and gradually declining. In contrast, the pedestrian class shows significantly lower recall, with a more noticeable decline as confidence increases. Overall, the maximum recall across all classes reaches 0.78, indicating a generally strong retrieval performance for the model.

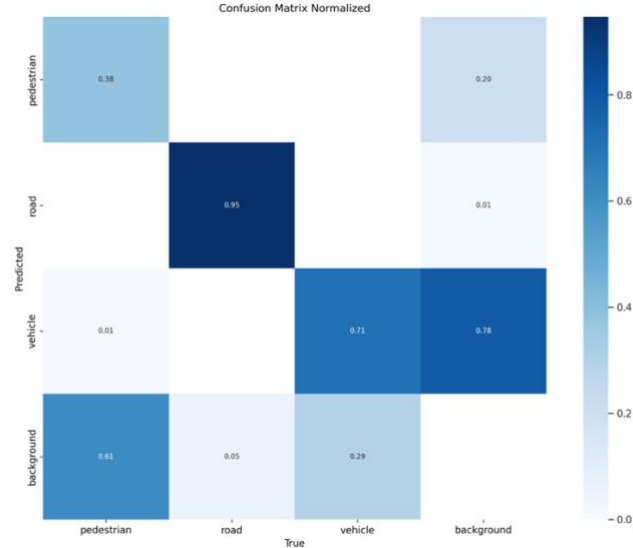


Figure 4.34: Confusion Matrix Normalized for YOLOv10 95-5

As illustrated in Figure 4.34, the confusion matrix reveals that the YOLOv10 model performs exceptionally well on the "road" class, with a 95% correct classification rate. The "vehicle" class is also handled well, though 29% of the instances are misclassified as background. The "pedestrian" class remains the most challenging, with only 38% accuracy and a significant 61% being misclassified as background.

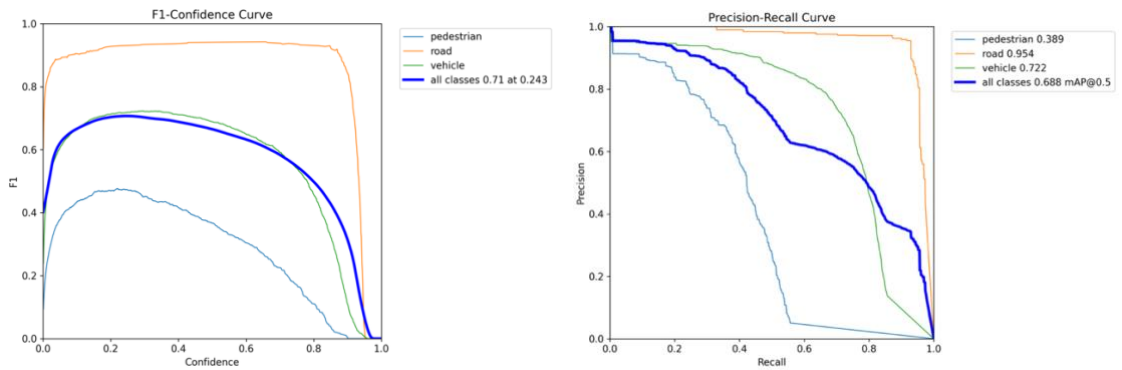


Figure 4.35: F1-Curve and PR-Curve for YOLOv10 95-5

As shown in Figure 4.35, the F1-Confidence Curve for YOLOv10 trained with a 95-5 data split shows an overall optimal F1 score of 0.71 at a confidence threshold of 0.243.

The road class consistently achieves the highest F1 values across the confidence range, maintaining nearly perfect performance and indicating strong and stable detection. The vehicle class shows moderate F1 values, peaking below the road class but still contributing positively to the overall metric. In contrast, the pedestrian class demonstrates significantly weaker performance, with a steep drop in F1 as confidence increases, suggesting challenges in distinguishing this class reliably, especially under higher certainty thresholds. Overall, while the model is highly confident and effective for large, structured objects like roads and vehicles, it remains limited in fine-grained pedestrian identification.

The PR curve shows stable and near-perfect performance for the "road" class across all recall levels. In contrast, "vehicle" performance gradually drops as recall increases, indicating some trade-off with precision at higher recall. The "pedestrian" class suffers a sharp decline in precision after 0.5 recall, revealing a weakness in detecting this category under more permissive conditions.

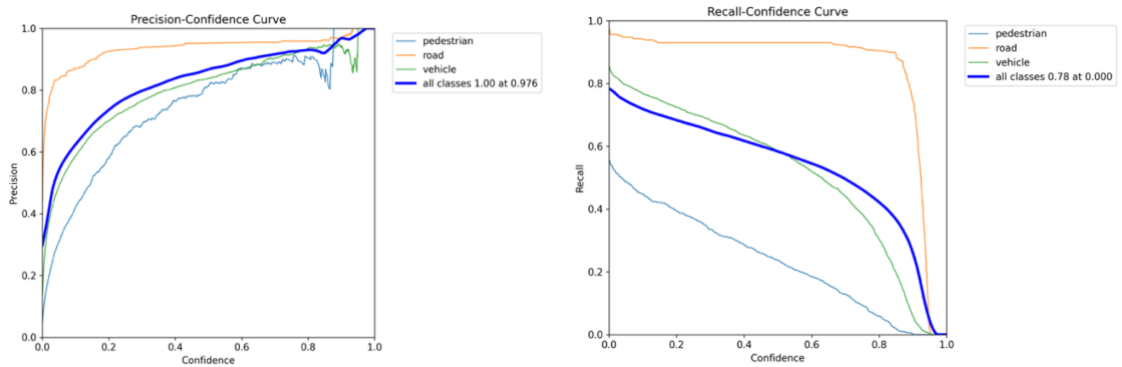


Figure 4.36: P-Curve and R-Curve for YOLOv10 95-5

As seen in Figure 4.36, the precision-confidence curve indicates consistently high precision for the "road" class, maintaining over 95% even at lower confidence levels. In contrast, "vehicle" and especially "pedestrian" classes show a more gradual increase, with noticeable fluctuations at high confidence, particularly beyond 0.95. These drops suggest unstable predictions at high thresholds, especially for smaller or more ambiguous objects.

The recall-confidence curve highlights a consistently high recall for the "road" class, which drops sharply beyond the 0.95 confidence level. Classes "vehicle" and "pedestrian" experience a steady decline in recall as confidence increases, with the pedestrian class exhibiting low recall even at lower confidence values. This suggests the model struggles more with pedestrian detection across all confidence levels.



Figure 4.37: Ground truth and prediction for road class from Istanbul streets

In Figure 4.37, the image on the left shows the ground truth from the Istanbul dataset, where the road class is marked with a green bounding box. On the right, the YOLO model's detection is shown with a black bounding box. When the results are compared, it is seen that the road class is detected in accordance with the ground truth.

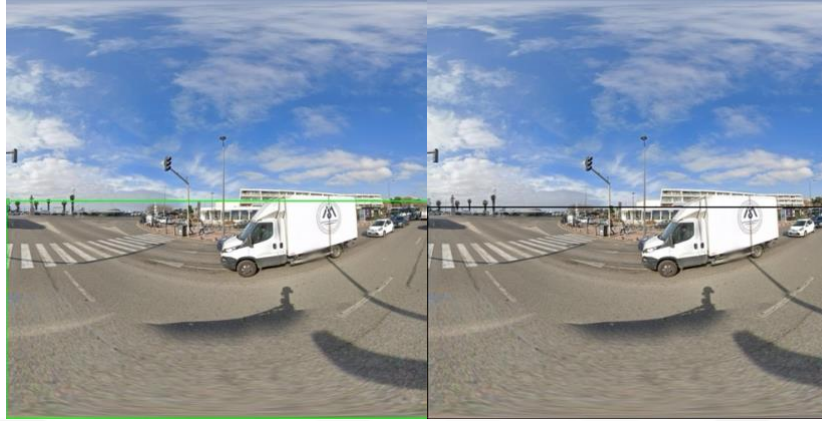


Figure 4.38: Ground truth and prediction for road class from Marseille streets

Figure 4.38 shows an image of a traffic scene in the city of Marseille, while on the right, the bounding box detected by our object recognition algorithm is colored in black. Compared to the green colored ground truth, it is a successful detection. Shadows on the road did not adversely impact detection performance.



Figure 4.39: Ground truth and detection image from Munich streets for road class

Figure 4.39 shows a traffic scene from the city of Munich, including a road blocked due to road construction works. When the results are examined, it is observed that the green colored bounding box and the black colored object detection match the output of our algorithm and are considered successful.



Figure 4.40: Ground truth and detection image from Paris streets for road class

Figure 4.40 shows a cobblestone road from the city of Paris. In this traffic scene, the open road, which used to be a narrow street, is seen marked as ground truth with green color. The detection result on the right side matches the ground truth.



Figure 4.41: Ground truth image and detection from Istanbul streets for vehicle class

Figure 4.41 shows a scene from Bagdat Street, a busy street in the city of Istanbul. In the scene, many vehicle classes are present at the same time. On the left side, the ground truths are indicated in green and on the right side, the vehicle classes detected are marked in red. Examples of the vehicle class have been identified, albeit quite small. Small

vehicle class instances are indicated with arrows pointing to zoomed-in views. When the results are analyzed, our object recognition algorithm is successful in this scene.



Figure 4.42: Ground truth and detection from Marseille streets for vehicle class

Figure 4.42 shows a traffic scene from the city of Marseille. There are many vehicles parked on the stage. Many vehicles near and far from the camera angle can be seen as ground truth in green on the left side. On the right side, the result of our object recognition algorithm is shown, where examples belonging to the vehicle class were successfully found. Distant vehicles are also shown with arrows and zoom.

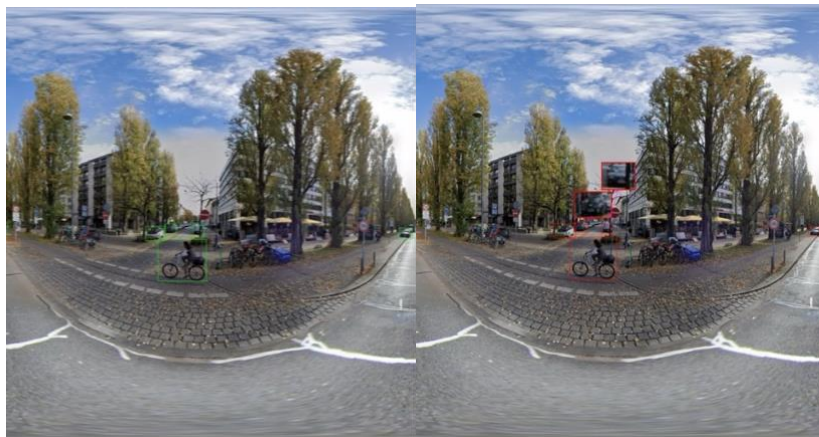


Figure 4.43: Ground truth image and detection for vehicle class from Munich streets

Figure 4.43 shows a scene from Leopold Street, one of the most important streets in Munich, with a pedestrian walkway consisting of a traditional carriageway, bicycle path, trees and cobblestone pavement. In the scene we see a bicycle, which is part of the vehicle class, and cars of different brands. Looking at our object detection result on the right side, bicycles belonging to the vehicle class and cars located relatively far away were successfully detected. Our model is also successful in this scene.



Figure 4.44: Ground truth and detection for pedestrian class from Istanbul streets

In Figure 4.44, we see a scene from Bağdat Street in Istanbul, where pedestrian traffic is heavy as well as vehicular traffic. As can be seen on the left side of the scene, many examples of the pedestrian class have been labeled, while on the right side, most of the relevant examples have been successfully detected by our algorithm.



Figure 4.45: Ground truth image and detection for pedestrian class from Marseille

In the traffic scene in Figure 4.45, where a cafe, a pedestrian crossing and a conventional driveway can be seen, examples of many pedestrian classes can be seen. Individuals seated in front of the café are also labeled as pedestrians. The results show that the yellow-colored detections contain a significant number of pedestrian classes found in the ground truth, so our object detection model is successful here.



Figure 4.46: Ground truth and detection for pedestrian class from Munich streets

A scene near Odeonsplatz, one of the most important squares in Munich, is shown in Figure 4.46. A bicycle and its rider are clearly visible in the scene, as well as pedestrians.

However, when examining the result on the right, the person on the bicycle is successfully disregarded as a pedestrian and the other pedestrians are successfully identified.



Figure 4.47: Ground truth and detection for pedestrian class from Paris streets

A traffic scene from the Champs Élysées, one of the world-famous avenues of Paris, is shown in Figure 4.47. As it is a wide street, the pedestrian path is far away and the instances of the pedestrian class in the scene are labeled as very small. When the results were checked, only a few instances of the pedestrian class were successfully detected, and the distant ones were zoomed in so that they could be seen.



Figure 4.48: Ground truth and unsuccessful detection from Istanbul for pedestrian

A traffic scene from Libadiye Street, one of the well-known streets of Istanbul, is shown in Figure 4.48. Due to the width of the road, the instances of the pedestrian class in the roadside park and walkway can be seen in a very small way. Our object detection algorithm on the right did not find any of these pedestrians in this scene. Therefore, it can be considered an unsuccessful example.



Figure 4.49: Ground truth and unsuccessful detection for pedestrian from Marseille

A view of an alleyway in Marseilles is shown in Figure 4.49. The scene shows vehicles, cyclists and some pedestrians. However, as can be seen on the right, our object detection model failed to detect any pedestrian class in this scene.



Figure 4.50: Ground truth & unsuccessful detection for pedestrian from Munich

Figure 4.50 shows a traffic scene in Munich. In this scene where the driveway and the sidewalk are intertwined, a pedestrian can be seen on the left side labeled in green in the ground truth image. The detection result on the right side shows that this pedestrian was not recognized by our model and our model was unsuccessful for this image.

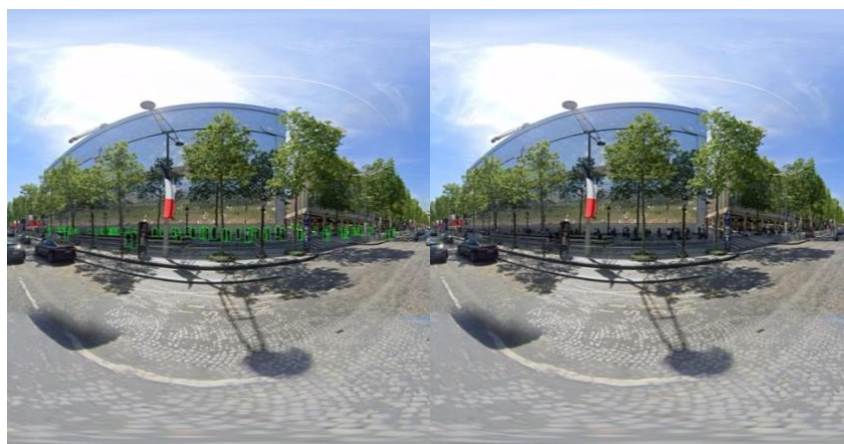


Figure 4.51: Ground truth & unsuccessful detection for pedestrian from Paris streets

Figure 4.51 shows a traffic scene, again taken from the Champs-Élysées, the famous avenue of Paris. Although there are about 50 pedestrian classes in the ground truth on the left, none of them are detected in the detection result on the right.



Figure 4.52: Ground truth & unsuccessful detection for road class from Istanbul

Figure 4.52 shows a traffic scene inside Istanbul Technical University in Istanbul. Unlike most other road classes in the dataset, the road in this ground truth image appears to be marked at a different point which appears differently due to the camera angle. Our detection result was not successful in this rare scene and failed to detect the road.



Figure 4.53: Ground truth & unsuccessful detection image for road from Marseille

Another traffic scene from Marseille is shown in Figure 4.53. In this image, the model predicts the road class, but it misses the area on the right side of the camera view. So, the model's performance in this case can be seen as partly successful and partly unsuccessful.

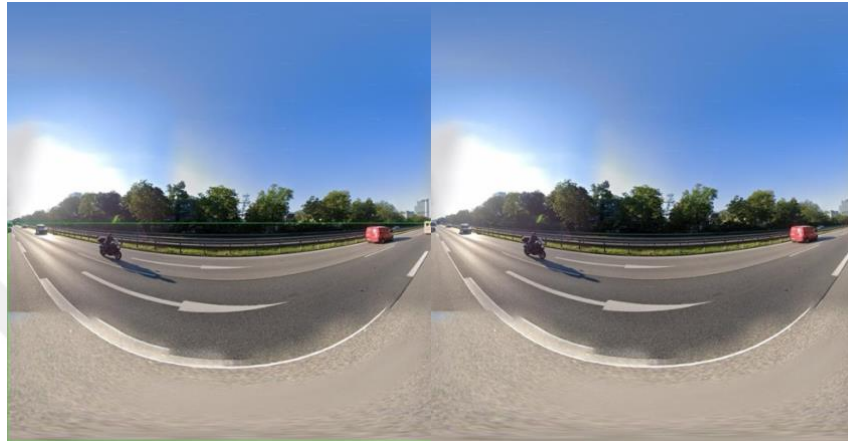


Figure 4.54: Ground truth and unsuccessful detection result for road from Munich

The traffic scene in Figure 4.54 was taken from a highway in one of the inner-city areas of Munich. There are two wide roads side by side in this scene. In the image here, only a certain part of the road was captured, and as can be seen in this part, our model was unsuccessful in this scene as well, as the road class could not be determined.



Figure 4.55: Ground truth & unsuccessful detection for road class from Paris

The traffic scene in Figure 4.55, with many parked rental bicycles and a wide pedestrian walkway, is typical of the streets of Paris. In this scene, the wide pedestrian walkway probably created a confusion and the road class in the ground track was not detected, so our model failed to detect it.



Figure 4.56: Ground truth & unsuccessful detection for vehicle from Istanbul

The traffic scene in Figure 4.56, with vehicles on two driveways separated by a railing, was taken from Libadiye Street in Istanbul. In this scene, the vehicles identified in ground truth were not detected by our model. The reason for the non-detection here is that there is an obstruction because the vehicles were behind the guardrail. Therefore, our model failed in this scene.



Figure 4.57: Ground truth & unsuccessful detection for vehicle from Marseille

Figure 4.57 shows a traffic scene on a highway in the city of Marseille. In this scene, vehicles are labeled on the left on the ramp at the road exit. In the detection result on the right, these vehicles were not detected. Since the vehicles are on a ramp, a rotated bounding box is detected instead of the standard horizontal bounding box. In addition, not all vehicles can be seen from the camera angle.



Figure 4.58: Ground truth & unsuccessful detection for vehicle from Munich

Figure 4.58 depicts a traffic scene on a railroad street in the city center of Munich. This scene shows two commercial class cars and a motorcycle. The reasons for the failure of our model to detect these vehicles are that the image of one of the vehicles is blocked due

to the fences affecting the bounding lengths and that it cannot be trained with a sufficient sample of these vehicle types. It can also be said that our model has some complexities in the detection of vehicles such as motorcycles where a human driver is clearly visible.



Figure 4.59: Ground truth and unsuccessful detection for vehicle from Paris

Figure 4.59 shows a traffic scene from the streets of Paris. The shapes and sizes of the vehicles seen in this scene differ from those of the traditional vehicle class. This is likely due to insufficient training samples of such vehicle types.

4.3 Segmentation

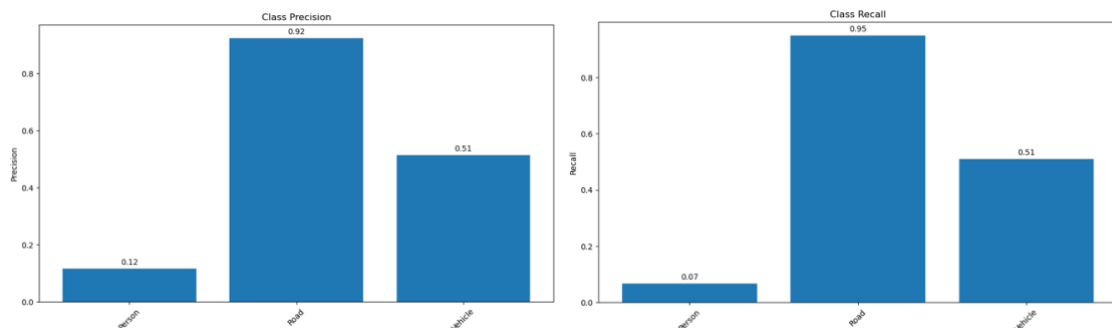


Figure 4.60: Precision and Recall graphs for DeepLabV3 65-35

As presented in Figure 4.60, the class-wise precision for DeepLabV3 (65-35 split) shows exceptionally high performance in detecting the road class, achieving a precision of 0.92. The vehicle class follows with moderate precision at 0.51, indicating some misclassifications or overlaps. In contrast, the pedestrian class performs poorly with a precision of just 0.12, suggesting a high rate of false positives in this category.

The class recall values for DeepLabV3 (65-35 split) reveal excellent performance in detecting the road class, with a recall of 0.95. The vehicle class shows moderate recall at 0.51, indicating that the model misses nearly half of the vehicle instances. The pedestrian class performs very poorly with a recall of just 0.07, meaning the model rarely detects pedestrians correctly.

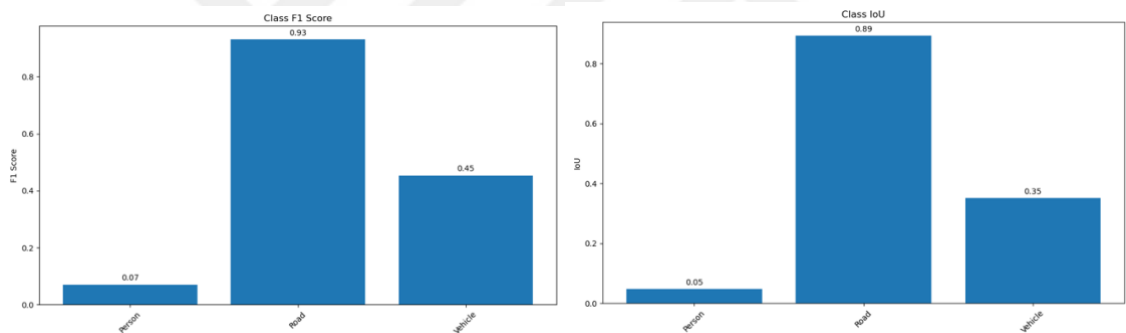


Figure 4.61: F1 Score and IoU graphs for DeepLabV3 65-35

As shown in Figure 4.61, the F1 Score results for DeepLabV3 (65-35 split) show that the model performs best on the road class with a strong score of 0.93. The vehicle class achieves a moderate F1 score of 0.45, suggesting room for improvement in both precision and recall. The pedestrian class performs very poorly with an F1 score of only 0.07, indicating a severe weakness in detecting this category.

The IoU values for DeepLabV3 (65-35 split) show that the model performs very well on the road class with a score of 0.79. The vehicle class achieves a moderate IoU of 0.35, indicating partial overlap between predictions and ground truth. The pedestrian class performs poorly with an IoU of just 0.03, revealing that the model rarely predicts this class accurately.

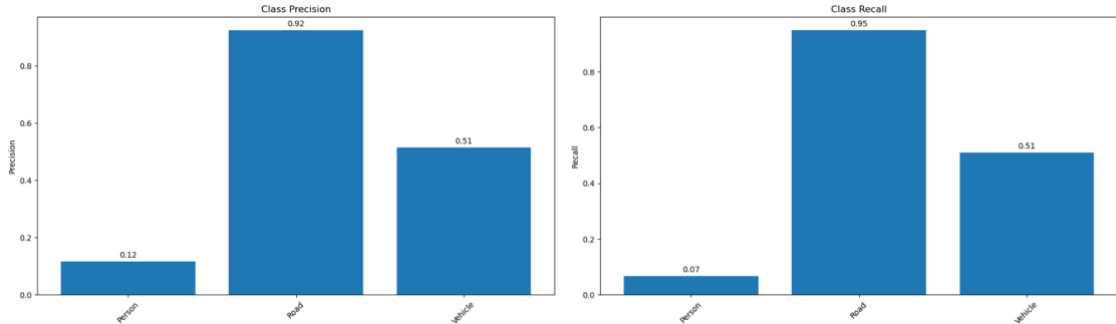


Figure 4.62: Precision and Recall graphs for DeepLabV3 75-25

As shown in Figure 4.62, compared to the 65-35 split, DeepLabV3's precision performance remains virtually unchanged for the road class, maintaining a high value of 0.92. The vehicle class also shows consistent behavior, with a slight improvement from 0.50 to 0.51, indicating marginally more confident predictions. However, the pedestrian class still exhibits very low precision (0.12 compared to 0.13), confirming that class imbalance or visual ambiguity continues to limit detection accuracy.

In the 75-25 split, recall values for the road and vehicle classes remain stable, both maintaining a score of 0.93 and 0.51 respectively identical to the 65-35 results. However, the pedestrian class staying at a very low recall value of 0.07. This continued deficiency suggests that the increased training data did not help the model better capture rare or complex instances of pedestrians.

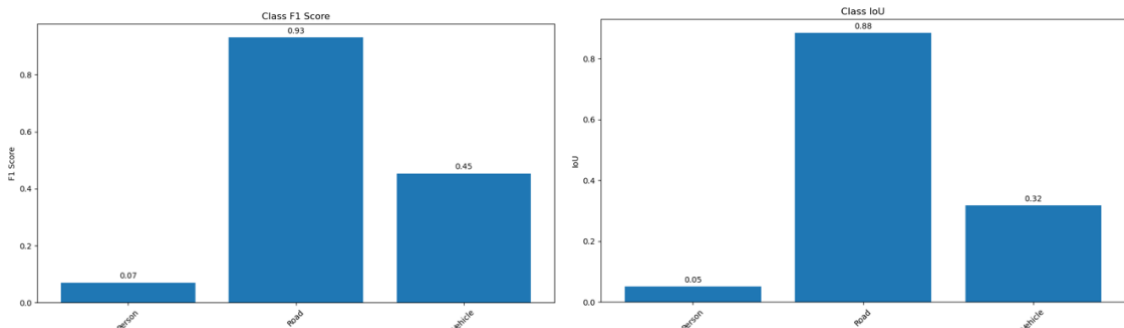


Figure 4.63: F1-Score and IoU graphs for DeepLabV3 75-25

Figure 4.63 demonstrates that the F1 scores in the 75-25 split mirror those of the 65-35 configuration, with 0.93 for road, 0.45 for vehicle, and a notably low 0.07 for the pedestrian class. This consistency indicates that increasing the training data portion did not lead to any tangible improvement in class-level balance. Particularly, the unchanged F1 score for the pedestrian class emphasizes the model's continued struggle with this category.

The IoU values in the 75-25 split are nearly identical to those in the 65-35 configuration, with the road class at 0.88, vehicle at 0.32, and pedestrian remaining extremely low at 0.05. This suggests that increasing the training set does not necessarily enhance spatial overlap performance for underrepresented classes. The persistent gap between the road and other classes reflects a strong class imbalance and limited generalization for pedestrians and vehicles.

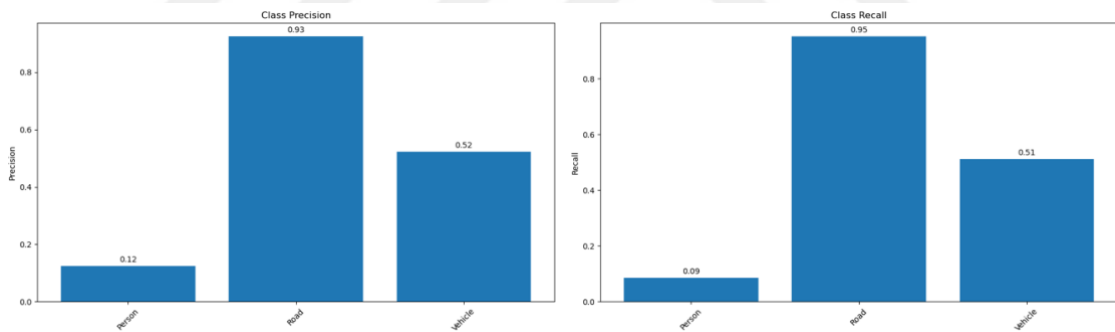


Figure 4.64: Precision and Recall graphs for DeepLabV3 70-30

Figure 4.64 reveals that in the 70-30 split, the road class achieves a precision of 0.93, remaining consistently high across all configurations. The vehicle class shows a slight improvement to 0.52 compared to 0.51 in the 65-35 and 75-25 splits. However, the pedestrian class still exhibits poor precision at 0.12, indicating persistent challenges in achieving accurate pedestrian detection, regardless of training size.

In the 70-30 split, the road class sustains a high recall of 0.90, like earlier splits. The vehicle class recall increases slightly to 0.53, showing minor improvement from the 0.51 seen in the 65-35 and 75-25 splits. Meanwhile, the pedestrian class reaches 0.09 recall,

an improvement over the 0.07 in previous splits, yet still indicating the model's weak ability to retrieve this class reliably.

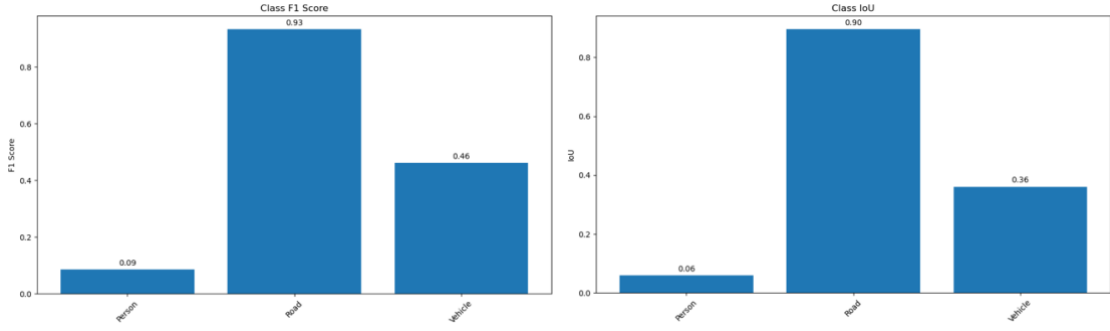


Figure 4.65: F1-Score and IoU graphs for DeepLabV3 70-30

Figure 4.65 highlights that in the 70-30 configuration, the road class maintains a strong F1 score of 0.93, in line with previous splits. The vehicle class remains moderate with an F1 score of 0.46, showing consistent performance. The pedestrian class continues to underperform with an F1 score of 0.09, reflecting persistent detection challenges.

In the 70-30 configuration, the IoU for the road class is remarkably high at 0.90, indicating highly accurate segmentation. The vehicle class follows with a moderate IoU of 0.36, reflecting fair localization but with room for improvement. The pedestrian class lags with an IoU of only 0.06, confirming its continued detection challenges across all splits.

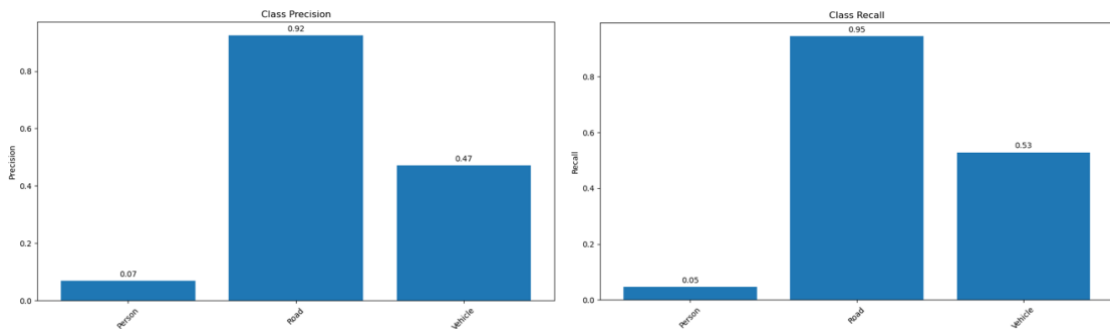


Figure 4.66: Precision and Recall graphs for DeepLabV3 95-5

Figure 4.66 indicates that in the 95-5 configuration, the road class maintains a high precision of 0.92, consistent with previous data splits. The vehicle class shows a slight decrease to 0.47 compared to the 70-30 configuration, suggesting a mild drop in prediction accuracy with less validation data. The pedestrian class remains critically low at 0.07, reaffirming its persistent difficulty across all evaluation settings.

In the 95-5 configuration, the road class again achieves a recall of 0.95, maintaining its consistently high detection rate. The vehicle class records a recall of 0.53, showing stable performance compared to earlier splits. The pedestrian class, however, declines to 0.05, reflecting a further deterioration in sensitivity under this data ratio.

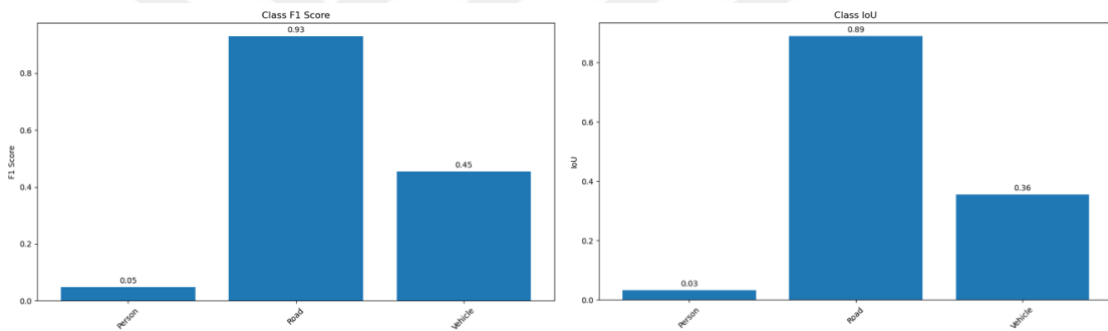


Figure 4.67: F1-Score and IoU graphs for DeepLabV3 95-5

Figure 4.67 confirms that in the 95-5 setting, the road class achieves a strong F1 score of 0.93, preserving its lead in both precision and recall balance. The vehicle class remains stable at 0.45, showing no notable improvement over previous splits. Conversely, the pedestrian class sees its F1 score fall to 0.05, indicating significant difficulty in correctly identifying this category with such a small validation subset.

The IoU graph for DeepLabV3 under the 95-5 split indicates excellent performance for the road class with a value of 0.89, consistent with earlier splits. The vehicle class maintains a moderate IoU of 0.36, showing only marginal improvement. The pedestrian class, however, remains critically low at 0.03, underlining persistent segmentation challenges despite training on a larger dataset portion.

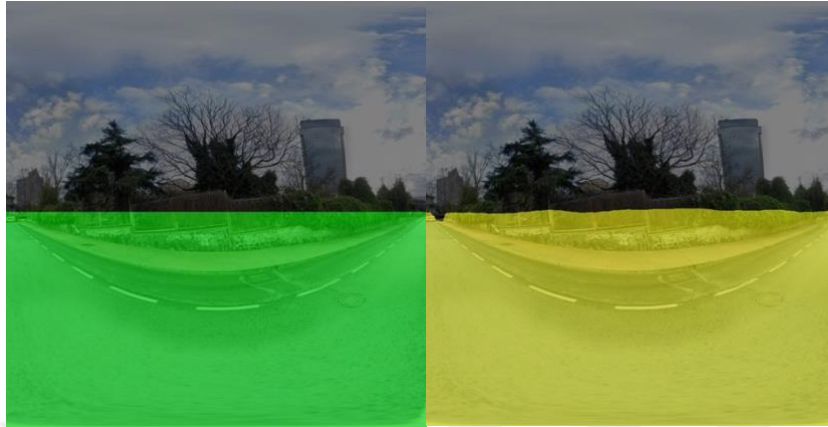


Figure 4.68: Ground truth and segmentation result for road in Istanbul streets

In the traffic scene shown in Figure 4.68, the shows the annotated road segment in the Dolmabahçe area of Istanbul. This road was successfully detected by our segmentation model and is shown in yellow in the image on the right.

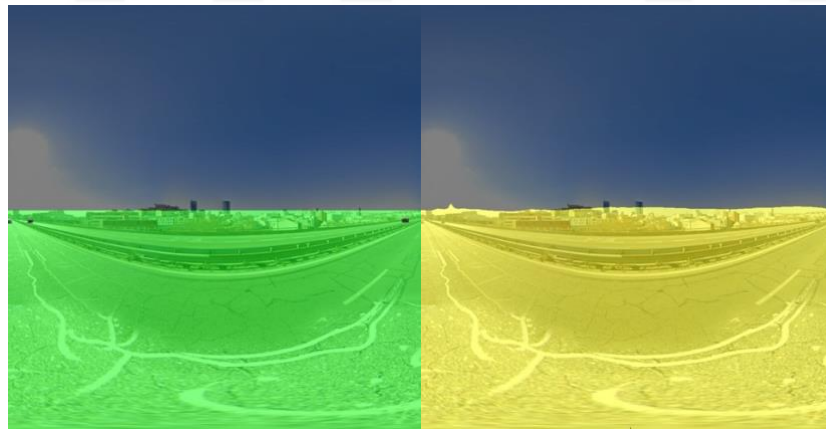


Figure 4.69: Ground truth and segmentation for road class from Marseille streets

The traffic scene with different lines and stripes, including a road instance separated by a railing, which can be seen in Figure 4.69, belongs to the streets of Marseille. The ground truth, shown in green on the left, and our segmentation result, shown in yellow on the right, clearly match, indicating that the segmentation model was successful in this scene.

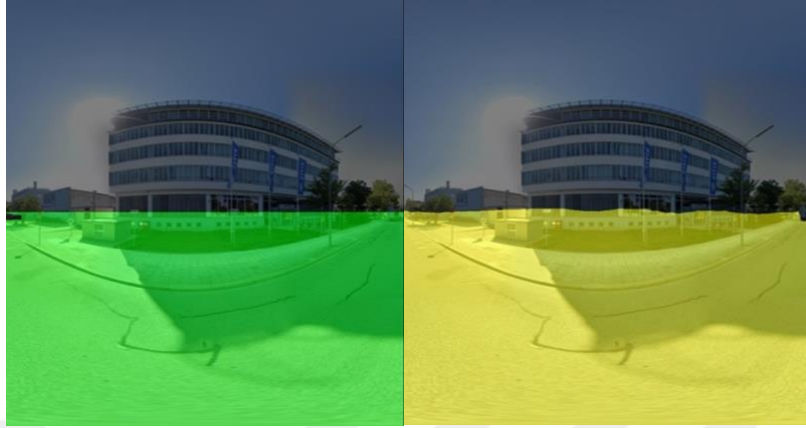


Figure 4.70: Ground truth & segmentation for road class from Munich streets

The traffic scene in Figure 4.70 is of a street in Munich. A pedestrian path and a road instance are visible. Although there are black lines on the road, the ground truth image on the left and the segmentation model result on the right successfully match. Therefore, we can consider our model as successful in segmenting the road instance in this traffic scene.

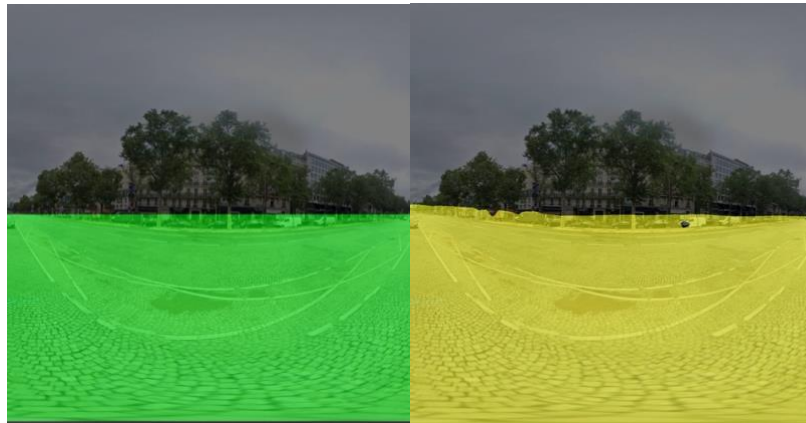


Figure 4.71: Ground truth & segmentation for a road class from Paris streets

The traffic scene in Figure 4.71 is one of the rare scenes of the streets of Paris. This is because the road is made of a cobblestone-like material instead of asphalt and yet there are lane lines on the road. Despite this, our model successfully segmented the scene compared to the ground truth on the left.



Figure 4.72: Ground truth & segmentation for pedestrian class from Istanbul

Figure 4.72 is a traffic scene of Bağdat Street, one of the popular streets of Istanbul. In this scene, on the left, pedestrians are seen walking on the wide sidewalk on the street, while on the right, the result of the segmentation of these pedestrians can be seen. Since our dataset is labeled with bounding boxes, the masks created are also based on these bounding boxes. It would be possible to have more accurate ground truth and segmentation with polygon labeling, but at this point only bounding box labels are available in our dataset. When the segmentation result is displayed, although some pedestrians' entire bodies are not recognized, most of the instances can be segmented, so our segmentation model is considered successful in this scene.



Figure 4.73: Ground truth & segmentation for pedestrian class from Marseille

The traffic scene in Figure 4.73 is from the city of Marseille. In this scene a pedestrian can be seen walking on the pedestrian walkway on the ground level on the left. Our segmentation model on the right can also segment this pedestrian in pink color. Despite two minor false positives, our model was partially successful in this scene.

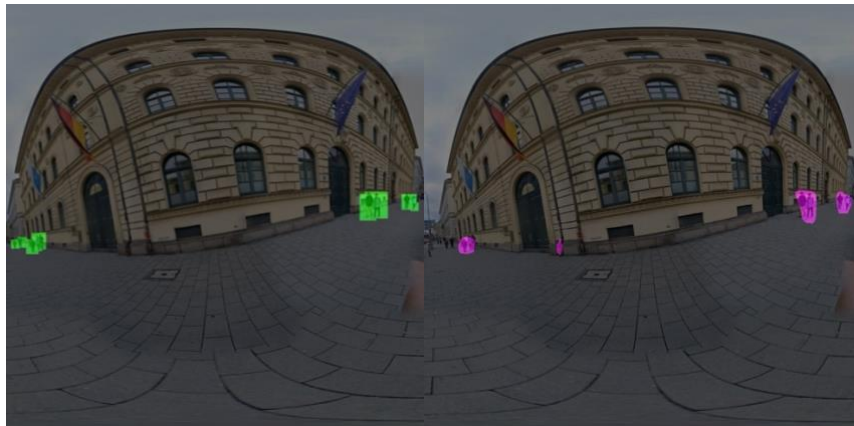


Figure 4.74: Ground truth & segmentation for pedestrian class from Munich

The traffic scene in Figure 4.74 is taken from Ludwig Street in Munich. In this scene, pedestrians on the pedestrian walkway can be seen in the ground truth image on the left. The segmentation model on the right shows that most pedestrians can be detected, and

one false positive is detected. Therefore, our segmentation model is considered partially successful for this scene.



Figure 4.75: Ground truth and segmentation for pedestrian from Paris streets

The image in Figure 4.75 depicts a traffic scene in Paris. The distortion observed at the bottom of the image is due to seen at the bottom of the image is that when shooting with a 360-degree camera, another vehicle in front of the vehicle is distorted while transferring to the 2D plane. The scene also shows that the ground truth annotation does not fully cover some pedestrians with full body integrity. As a result of the segmentation on the right, although the pedestrians were mostly detected, there were some pedestrians that could not be detected and there were false positives, so it was considered partially successful.

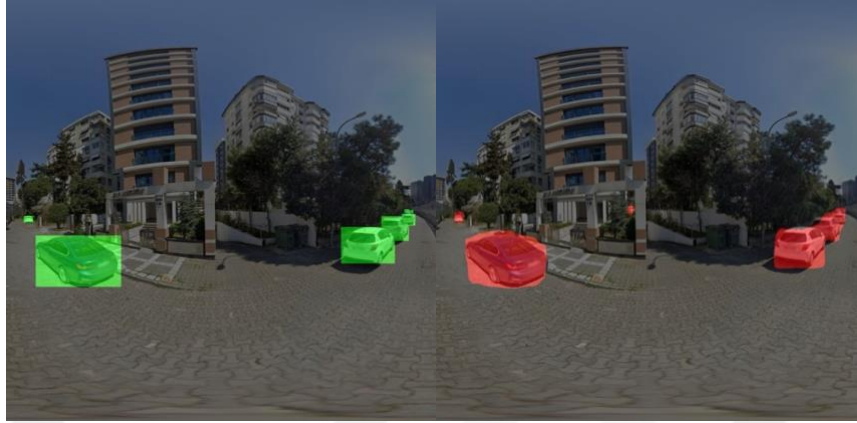


Figure 4.76: Ground truth and segmentation for vehicle class from Istanbul streets

Another traffic scene from the streets of Istanbul can be seen in Figure 4.76. In this example taken on an inclined road segment, the instances of the vehicle class of the ground truth image can be seen on the left side. On the right, our segmentation model has identified these vehicles. Although there is only a false positive in a very small area, our model was successful in this scene.

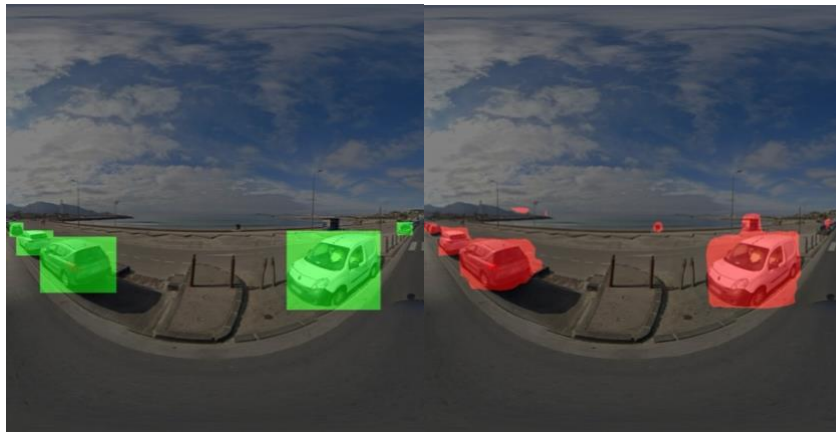


Figure 4.77: Ground truth and segmentation for vehicle class from Marseille streets

A traffic scene on a coastal road in the city of Marseille, with a bicycle path and a pedestrian sidewalk alongside the road, can be seen in Figure 4.77. On the left, the vehicles marked in green are ground truth and, on the right, the results of our segmentation

model can be seen in the regions marked in red. Our model is considered successful in this scene as it detected most of the vehicles despite false positives.



Figure 4.78: Ground truth and segmentation for vehicle class from Munich streets

A traffic scene on a street on the banks of the Isar river running through the city of Munich can be seen in Figure 4.78. In this scene, on the left you can see the instances of a class of vehicles on the road, mostly parked, while on the right you can see the output of our segmentation model. In this scene, our model largely detected the vehicles in the ground truth, therefore, the segmentation is deemed successful in this scene.



Figure 4.79: Ground truth and segmentation for vehicle class from Paris streets

The traffic scene in Figure 4.79 is a street in the city of Paris. As ground truth, the instance of the vehicle class marked as the bounding box is segmented almost exclusively as the region belonging to the vehicle, which is visible in red in our segmentation result on the right. Although false positives were found, they were deemed insignificant, and our segmentation model performed successfully in this scene.



Figure 4.80: Ground truth & unsuccessful segmentation for pedestrian from Istanbul

The traffic scene in Figure 4.80 shows a road, various shops and a pedestrian walkway. Pedestrians in this traffic scene in the city of Istanbul are visible to the camera even though they are mostly inside shops. However, our segmentation model was not able to detect these pedestrians. This may be due to the model's limited ability to detect small regions at detecting relatively small areas. In addition, the fact that the pedestrians in this scene were inside the shop instead of on the sidewalk may have also contributed to the lack of success.



Figure 4.81: Ground truth & failed segmentation for pedestrian from Marseille

Figure 4.81 shows a traffic scene on a street in the city of Marseille, with a road, a pedestrian sidewalk and a bus lane. In this scene, our segmentation model is clearly unsuccessful, as none of the pedestrians labeled green in the ground truth are detected.

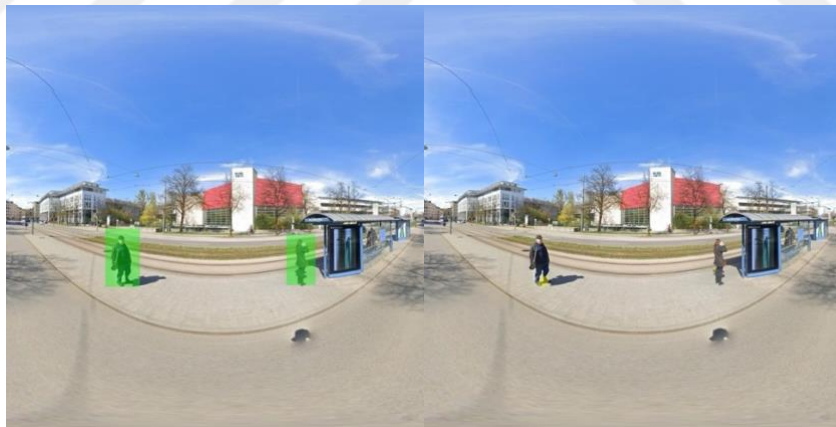


Figure 4.82: Ground truth and failed segmentation for pedestrian from Munich

A traffic scene of the streets of Munich with a road and a tram stop in the middle of the road and pedestrians waiting at the tram stop can be seen in Figure 4.82. In this scene our model was able to predict an object of the feet of one of the pedestrians, but in this scene also the model is considered unsuccessful.



Figure 4.83: Ground truth and failed segmentation for pedestrian from Paris

A street in the city of Paris can be seen in Figure 4.83 with a traffic scene with a wide pedestrian walkway and road. In this scene, pedestrians walking on the pedestrian sidewalk on the left are marked in green, but as can be seen from the output of our segmentation model on the right, these pedestrians were not detected by our model. Therefore, our model can be classified as unsuccessful here.



Figure 4.84: Ground truth and failed segmentation for road from Istanbul

Figure 4.84 shows a traffic scene on the streets of Istanbul. In the traffic scene there is a wide sidewalk and road. Although the ground truth on the left side is labeled as a road

without sidewalk in green, our segmentation model on the right includes the sidewalk in the road class. Therefore, our model fails in this scene.

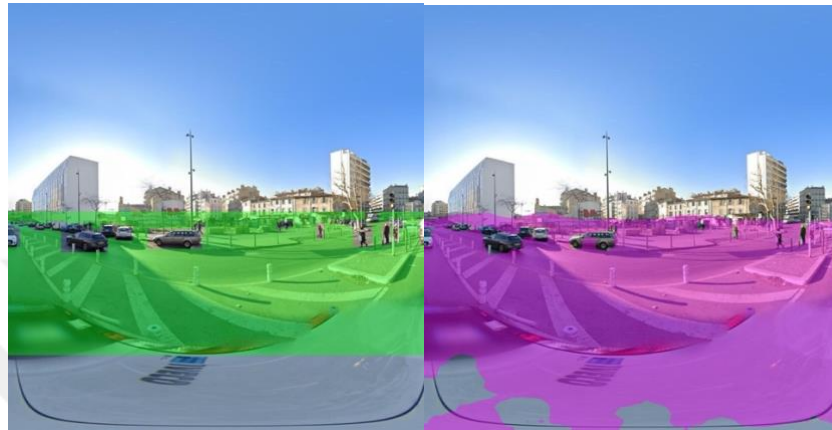


Figure 4.85: Ground truth and unsuccessful segmentation for road from Marseille

Figure 4.85 shows a traffic scene in the city of Marseille. Since there is another vehicle directly in front of the vehicle with the camera traveling in traffic, a distortion occurs during the transfer of the 360-degree image to the 2D plane. In the ground truth image on the left, the distorted part of this vehicle at the bottom of the scene is not labeled, but in the output of our segmentation model on the right, this part is segmented as a road, so our model fails in this scene.

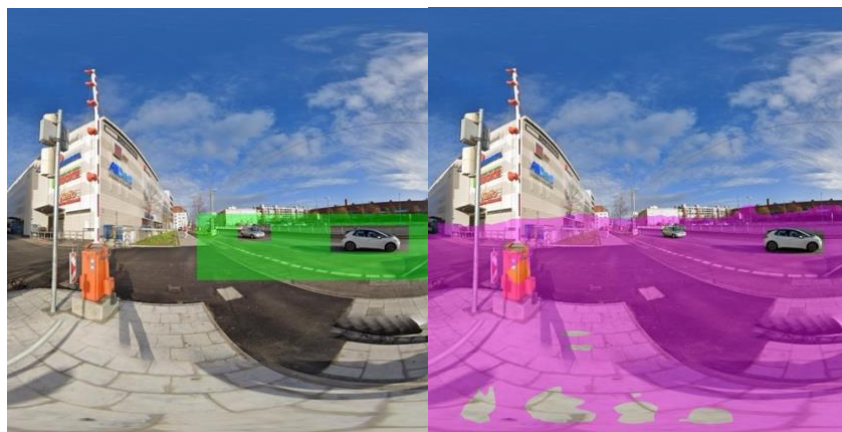


Figure 4.86: Ground truth and unsuccessful result for road class from Munich streets

The traffic scene in Figure 4.86 is located near Einstein Street in Munich. Unlike most examples in our dataset, in this scene the road class appears on the right side of the camera view. There is a pedestrian walkway directly opposite the viewpoint. Although this is labeled accordingly in the ground truth on the left, the result of our segmentation model on the right indicates a segmentation error, as the pedestrian sidewalk was incorrectly included in the road class.

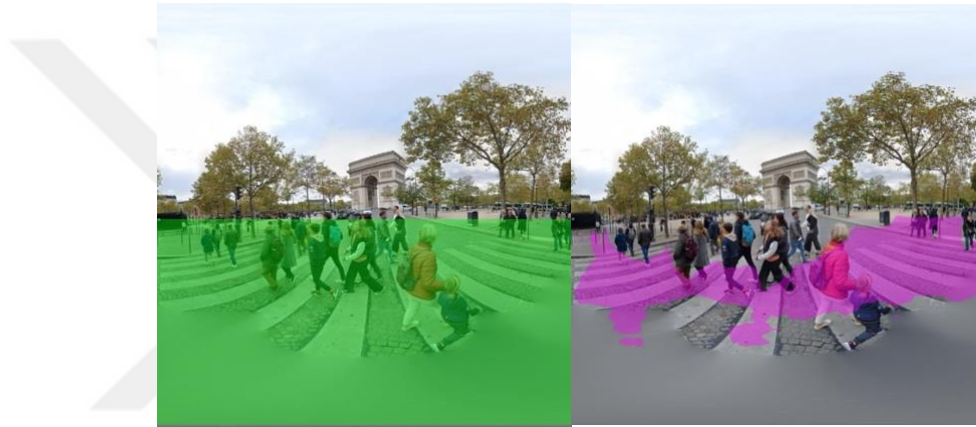


Figure 4.87: Ground truth and unsuccessful segmentation for road from Paris streets

A traffic scene with a pedestrian walkway and pedestrians crossing a road in the city of Paris can be seen in Figure 4.87. In this scene, the image from the GSV has been blurred. Although the blurred area is also included in the ground truth of the road class, our segmentation model could not identify the blurred area as part of the road, and the rest of the road was segmented in a fragmented manner due to the large number of pedestrians in the scene. Therefore, our segmentation model was unsuccessful in this scene.



Figure 4.88: Ground truth and unsuccessful segmentation for vehicle from Istanbul

Figure 4.88 shows a traffic scene in the city of Istanbul. In this scene, the instance of the vehicle class on the main street is labeled as ground truth, but our segmentation model is unable to detect this vehicle. This may be because it is far away from the camera angle and relatively small. Therefore, our model was unsuccessful in this scene.



Figure 4.89: Ground truth & failed segmentation for vehicle from Marseille

A traffic scene in the city of Marseille, with a bridge crossing and two roads and vehicles separated by railings, can be seen in Figure 4.89. In this scene, two vehicles are ground truth under and above the bridge at a distant point, but these were not detected by our segmentation model, as shown on the right, therefore this scene is also classified as unsuccessful.

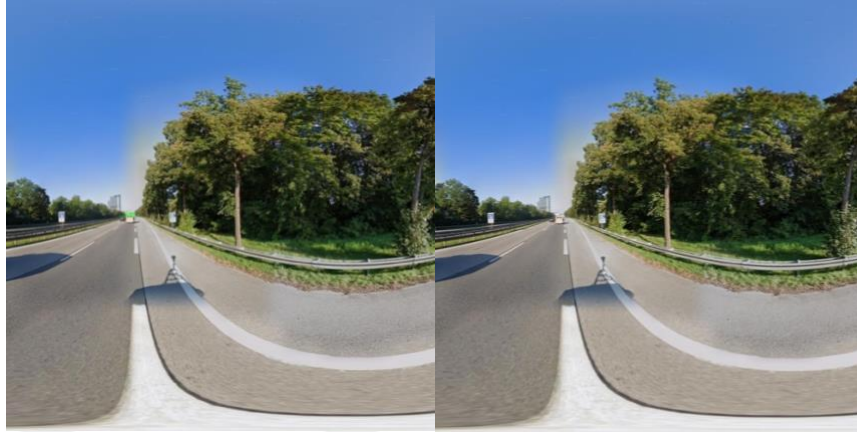


Figure 4.90: Ground truth & failed segmentation for vehicle from Munich

Figure 4.90 shows a traffic scene from the city of Munich, featuring a highway through the city. In the scene, the distant vehicle is marked as ground truth, but our segmentation model on the right fails to detect it. Therefore, the result of our model in this scene is classified as an unsuccessful segmentation result.



Figure 4.91: Ground truth & failed segmentation for vehicle class from Paris

Figure 4.91 shows a pedestrian crossing and a road along a road in the city of Paris. In the segmentation result, pedestrians were incorrectly classified as vehicles. Therefore, this segmentation result is far from accurate.

4.4 Statistical Evaluation

Table 4.1: City based metrics table for YOLO 65-35 models

	Istanbul			Munich			Paris			Marseille			Total		
Model	Precision	Recall	mAP50	Precision	Recall	mAP50	Precision	Recall	mAP50	Precision	Recall	mAP50	Precision	Recall	mAP50
YOLOv8	0.712	0.579	0.582	0.732	0.659	0.69	0.751	0.667	0.687	0.805	0.685	0.719	0.75	0.648	0.67
YOLOv10	0.721	0.574	0.582	0.716	0.674	0.704	0.765	0.668	0.698	0.797	0.683	0.714	0.749	0.65	0.674

Table 4.1 presents a comparison of the performance of YOLOv8 and YOLOv10 across four cities using precision, recall, and mAP@0.50 metrics. Both models demonstrate similar overall results, but YOLOv10 achieves a slightly higher total mAP (0.674 vs. 0.67), indicating a modest improvement in average detection accuracy. In Munich and Paris, YOLOv10 outperforms YOLOv8 in all three metrics, suggesting more robust generalization in these cities. However, performance in Istanbul and Marseille remains nearly equal, with minimal gains or slight drops. Overall, YOLOv10 shows more consistent performance across cities, reflecting incremental yet meaningful architectural enhancements.

Table 4.2: City based metrics for DeepLabV3 65-35

	Istanbul					Munich					Paris					Marseille					Total				
Model	Precision	Recall	F1-Score	IoU	Dice	Precision	Recall	F1-Score	IoU	Dice	Precision	Recall	F1-Score	IoU	Dice	Precision	Recall	F1-Score	IoU	Dice	Precision	Recall	F1-Score	IoU	Dice
DeepLabv3	0.701	0.703	0.679	0.629	0.702	0.619	0.647	0.599	0.556	0.632	0.731	0.708	0.692	0.639	0.72	0.762	0.73	0.725	0.67	0.746	0.703	0.697	0.674	0.624	0.7

Table 4.2 shows that the DeepLabV3 model demonstrates consistent and balanced performance across different cities under the 65-35 split. Among the cities, the highest F1-score is achieved in Paris (0.692), while Munich yields the lowest (0.599), indicating some variation in segmentation quality depending on regional data characteristics. The model's overall precision and recall values are close (0.703 and 0.697, respectively), which suggests a stable balance between false positives and false negatives. IoU scores are highest in Marseille (0.67) and Paris (0.639), pointing to strong localization accuracy in those cities. Despite slightly lower values in Istanbul and Munich, their Dice coefficients still exceed 0.63, confirming that the model maintains reasonable overlap between predicted and ground-truth regions across all locations. On average, DeepLabV3

achieves a total F1-score of 0.674 and IoU of 0.624, reflecting reliable generalization capabilities across different urban contexts.

Table 4.3: Class and city-based results for YOLOv8 65-35

	Istanbul			Munich			Paris			Marseilles			Total		
	Precision	Recall	mAP50	Precision	Recall	mAP50	Precision	Recall	mAP50	Precision	Recall	mAP50	Precision	Recall	mAP50
Pedestrian	0.582	0.295	0.307	0.560	0.446	0.445	0.698	0.429	0.467	0.681	0.337	0.386	0.630	0.377	0.401
Vehicle	0.627	0.533	0.510	0.730	0.729	0.754	0.741	0.635	0.681	0.793	0.766	0.809	0.723	0.666	0.689
Road	0.910	0.919	0.922	0.877	0.880	0.906	0.902	0.859	0.901	0.960	0.981	0.969	0.912	0.910	0.924

Table 4.3 illustrates that the YOLOv8 model exhibits strong performance in the "road" class across all cities, with a consistently high mAP@50 exceeding 0.90 and reaching up to 0.969 in Marseille. In contrast, the "pedestrian" class remains the most challenging, with a notably low total recall of 0.377 and mAP@50 of only 0.401, suggesting that the model struggles with detecting this class reliably, especially in Istanbul and Paris. The "vehicle" class achieves moderate performance, with relatively balanced precision and recall values across cities and the highest mAP@50 of 0.809 in Marseille. Overall, the total mAP@50 scores reflect the model's superior ability to segment and classify road elements, while human detection remains less robust. This disparity highlights the need for targeted improvements in handling underrepresented or visually complex categories like pedestrians.

Table 4.4: Class and city-based results for YOLOv10 65-35

	Istanbul			Munich			Paris			Marseilles			Total		
	Precision	Recall	mAP50	Precision	Recall	mAP50	Precision	Recall	mAP50	Precision	Recall	mAP50	Precision	Recall	mAP50
Pedestrian	0.594	0.301	0.306	0.622	0.456	0.488	0.606	0.379	0.387	0.673	0.330	0.380	0.624	0.366	0.390
Vehicle	0.632	0.534	0.519	0.727	0.697	0.709	0.726	0.728	0.752	0.778	0.741	0.792	0.715	0.675	0.693
Road	0.910	0.903	0.922	0.846	0.824	0.873	0.922	0.895	0.922	0.964	0.983	0.986	0.911	0.901	0.926

Table 4.4 highlights that the YOLOv10 model shows consistent and superior performance in the "road" class across all cities, with mAP@50 values exceeding 0.92 and peaking at 0.986 in Marseille. Despite slight fluctuations in recall, both precision and recall values for the road class remain high, confirming its reliability in segmenting well-defined road structures. While the "vehicle" class performs relatively well overall (total mAP@50 of 0.693), slight variations across cities indicate some sensitivity to regional differences. The "pedestrian" class, however, continues to be the weakest, with a low overall mAP@50 of 0.390, and particularly low recall values in Istanbul and Marseille. Compared to YOLOv8, YOLOv10 provides marginal improvements in vehicle detection but struggles similarly with pedestrian recognition, underscoring the persistent challenge posed by this class.

Table 4.5: Class and city-based results for DeepLabV3 65-35

	Istanbul				Munich				Paris				Marseilles				Total			
	Precision	Recall	F1	IoU	Dice	Precision	Recall	F1	IoU	Dice	Precision	Recall	F1	IoU	Dice	Precision	Recall	F1	IoU	Dice
Pedestrian	0.131	0.085	0.089	0.061	0.103	0.112	0.058	0.063	0.043	0.077	0.083	0.061	0.045	0.025	0.070	0.107	0.052	0.059	0.041	0.070
Vehicle	0.452	0.484	0.410	0.313	0.467	0.504	0.460	0.420	0.323	0.481	0.338	0.452	0.303	0.205	0.387	0.637	0.600	0.570	0.450	0.618
Road	0.936	0.968	0.948	0.911	0.952	0.912	0.956	0.924	0.880	0.933	0.963	0.972	0.965	0.936	0.967	0.913	0.914	0.911	0.876	0.914
Background	0.986	0.964	0.973	0.950	0.974	0.982	0.954	0.963	0.937	0.968	0.975	0.977	0.969	0.952	0.976	0.982	0.961	0.968	0.946	0.971

According to Table 4.5, the DeepLabV3 model demonstrates excellent performance in the "road" and "background" classes across all cities, with total IoU values of 0.904 and 0.946 respectively, and Dice scores exceeding 0.94. Particularly, the road class maintains high precision and recall above 0.91 in every city, highlighting its robustness in

identifying structured regions. The "vehicle" class yields moderate results, achieving a total F1 score of 0.432, and shows its highest city-wise performance in Marseille, suggesting variability in detection depending on urban context. In contrast, the "pedestrian" class continues to underperform significantly, with a total IoU of only 0.052 and an F1 score of 0.077, indicating severe limitations in segmenting small and less defined objects. These results underline that while DeepLabV3 is highly effective in large-area semantic segmentation, it faces substantial challenges in detecting fine-grained categories like pedestrians.

Table 4.6: City and total metrics for YOLO 75-25

	Istanbul			Munich			Paris			Marseille			Total		
Model	Precision	Recall	mAP50	Precision	Recall	mAP50	Precision	Recall	mAP50	Precision	Recall	mAP50	Precision	Recall	mAP50
YOLOv8	0.706	0.582	0.58	0.722	0.685	0.702	0.78	0.641	0.683	0.811	0.695	0.721	0.755	0.651	0.671
YOLOv10	0.717	0.587	0.579	0.724	0.676	0.692	0.762	0.647	0.684	0.817	0.687	0.722	0.755	0.649	0.669

Table 4.6 reveals that the city-wise evaluation of YOLOv8 and YOLOv10 models trained with a 75-25 split shows overall comparable performance across all metrics. YOLOv10 slightly improves upon YOLOv8 in precision across all cities except for Munich, although the gain is marginal. Recall values are relatively stable between models, with YOLOv8 slightly outperforming YOLOv10 in total recall (0.651 vs. 0.649), indicating better coverage of positive detections. The mAP@50 metric shows near-identical performance, with only minimal fluctuations across cities and a total difference of 0.002, favoring YOLOv8. Overall, both models exhibit robust results in this setting, with YOLOv10 offering slightly better precision, while YOLOv8 maintains a marginal recall advantage.

Table 4.7: City and total metrics for DeepLabV3 75-25

	Istanbul					Munich					Paris					Marseille					Total				
Model	Precision	Recall	F1-Score	IoU	Dice	Precision	Recall	F1-Score	IoU	Dice	Precision	Recall	F1-Score	IoU	Dice	Precision	Recall	F1-Score	IoU	Dice	Precision	Recall	F1-Score	IoU	Dice
DeepLabv3	0.689	0.682	0.663	0.613	0.686	0.626	0.658	0.61	0.567	0.642	0.733	0.699	0.687	0.631	0.716	0.753	0.709	0.707	0.649	0.73	0.7	0.687	0.667	0.615	0.693

Table 4.7 shows that the performance table for DeepLabV3 under the 75-25 split demonstrates consistent results across all evaluated cities. Precision and recall values are

balanced, with overall scores of 0.700 and 0.687 respectively, leading to a total F1-Score of 0.667. Among the cities, Paris exhibits the highest individual F1-Score (0.687), while Munich shows relatively lower values across all metrics. The IoU and Dice scores follow a similar trend, with total values of 0.615 and 0.693 respectively, indicating reliable overlap and segmentation performance. These metrics suggest that DeepLabV3 maintains steady segmentation capability across diverse urban environments, with only slight variations depending on location.

Table 4.8: Class based metrics for YOLOv8 75-25

	Istanbul			Munich			Paris			Marseilles			Total		
	Precision	Recall	mAP50	Precision	Recall	mAP50	Precision	Recall	mAP50	Precision	Recall	mAP50	Precision	Recall	mAP50
Pedestrian	0.605	0.306	0.310	0.574	0.405	0.429	0.653	0.420	0.445	0.705	0.318	0.393	0.634	0.356	0.394
Vehicle	0.634	0.537	0.503	0.714	0.731	0.745	0.740	0.647	0.699	0.779	0.770	0.808	0.717	0.671	0.689
Road	0.911	0.919	0.924	0.885	0.891	0.901	0.893	0.875	0.909	0.966	0.974	0.966	0.914	0.914	0.925

Table 4.8 presents the class-based performance metrics of YOLOv8 under the 75-25 data split. The road class consistently outperforms others with a total precision and recall of 0.914 and a remarkably high mAP50 of 0.925, indicating exceptional detection capability. The vehicle class shows solid performance across cities, achieving a total mAP50 of 0.689 with well-balanced precision (0.717) and recall (0.671). However, pedestrian detection remains the most challenging, with the lowest total mAP50 (0.394) and notably weaker recall (0.356), despite moderate precision (0.634). These results suggest that while YOLOv8 performs reliably for structured and larger objects like roads and vehicles, its effectiveness in detecting smaller or more variable objects like pedestrians is still limited.

Table 4.9: Class based metrics for YOLOv10 75-25

	Istanbul			Munich			Paris			Marseilles			Total		
	Precision	Recall	mAP50	Precision	Recall	mAP50	Precision	Recall	mAP50	Precision	Recall	mAP50	Precision	Recall	mAP50
Pedestrian	0.582	0.295	0.307	0.560	0.446	0.445	0.698	0.429	0.467	0.681	0.337	0.386	0.630	0.377	0.401
Vehicle	0.627	0.533	0.510	0.730	0.729	0.754	0.741	0.635	0.681	0.793	0.766	0.809	0.723	0.666	0.689
Road	0.910	0.919	0.922	0.877	0.880	0.906	0.902	0.859	0.901	0.960	0.981	0.969	0.912	0.910	0.924

Table 4.9 displays the class-based performance metrics of the YOLOv10 model trained with a 75-25 data split. The road class continues to dominate in performance, achieving a total mAP50 of 0.924 with very high precision (0.912) and recall (0.910), indicating reliable and consistent detection. The vehicle class shows strong results with a total mAP50 of 0.689, supported by a good balance between precision (0.723) and recall (0.666), confirming the model’s effectiveness in identifying vehicles across cities. The pedestrian class again exhibits the weakest performance, with a low recall of 0.377 and the lowest total mAP50 at 0.401, despite slightly improved precision compared to previous splits. These results emphasize YOLOv10’s robustness in structured object detection, especially for roads and vehicles, while underlining the continued difficulty in pedestrian detection due to complexity and variability in the class.

Table 4.10: Class based metrics for DeepLabV3 75-25

	Istanbul					Munich					Paris					Marseilles					Total				
	Precision	Recall	F1	IoU	Dice	Precision	Recall	F1	IoU	Dice	Precision	Recall	F1	IoU	Dice	Precision	Recall	F1	IoU	Dice	Precision	Recall	F1	IoU	Dice
Pedestrian	0.131	0.085	0.089	0.061	0.103	0.112	0.058	0.063	0.043	0.077	0.083	0.061	0.045	0.025	0.070	0.107	0.052	0.059	0.041	0.070	0.050	0.031	0.033	0.023	0.038
Vehicle	0.452	0.484	0.410	0.313	0.467	0.504	0.460	0.420	0.323	0.481	0.338	0.452	0.303	0.205	0.387	0.637	0.600	0.570	0.450	0.618	0.448	0.557	0.439	0.340	0.496
Road	0.936	0.968	0.948	0.911	0.952	0.912	0.956	0.924	0.880	0.933	0.963	0.972	0.965	0.936	0.967	0.913	0.914	0.911	0.876	0.914	0.936	0.917	0.918	0.871	0.926
Background	0.986	0.964	0.973	0.950	0.974	0.982	0.954	0.963	0.937	0.968	0.975	0.977	0.969	0.952	0.976	0.982	0.961	0.968	0.946	0.971	0.940	0.964	0.926	0.904	0.952

Table 4.10 presents class-based performance metrics for the DeepLabV3 model trained on a 75-25 split. The road class maintains superior detection results across all cities, with total F1-score, IoU, and Dice values of 0.918, 0.871, and 0.926 respectively, confirming DeepLabV3’s high reliability in segmenting structured surfaces. The vehicle class

performs moderately well, achieving an overall F1-score of 0.439 and an IoU of 0.340, though performance remains significantly lower than for the road class. The pedestrian class exhibits extremely poor results, with a total F1-score of only 0.033 and IoU of 0.023, indicating that the model struggles to segment small, variable objects in complex environments. Compared to previous splits, results here show little to no improvement for pedestrians, reinforcing the need for better class balance and data augmentation.

Table 4.11: City based results for YOLO models with 70-30

Model	Istanbul			Munich			Paris			Marseille			Total		
	Precision	Recall	mAP50	Precision	Recall	mAP50	Precision	Recall	mAP50	Precision	Recall	mAP50	Precision	Recall	mAP50
YOLOv8	0.672	0.591	0.567	0.742	0.677	0.705	0.778	0.649	0.674	0.815	0.688	0.718	0.752	0.651	0.666
YOLOv10	0.671	0.59	0.567	0.761	0.663	0.709	0.786	0.642	0.674	0.815	0.686	0.716	0.758	0.645	0.667

Table 4.11 displays the city-based evaluation results for the YOLOv8 and YOLOv10 models trained with a 70-30 split. Both models show very similar overall performance, with YOLOv10 slightly outperforming YOLOv8 in total precision (0.758 vs. 0.752) and mAP50 (0.667 vs. 0.666), while YOLOv8 achieves marginally higher recall (0.651 vs. 0.645). City-specific performance is generally stable, with Marseille consistently delivering high scores across all metrics for both models. Notably, YOLOv10 shows a marked improvement in Munich, achieving a higher mAP50 (0.709) compared to YOLOv8 (0.705), despite a slight drop in recall. The small performance gap between the models suggests that YOLOv10 offers incremental improvements in precision and detection accuracy under this split.

Table 4.12: City based results for DeepLabV3 70-30

Model	Istanbul			Munich			Paris			Marseille			Total		
	Precision	Recall	F1-Score	Precision	Recall	F1-Score	Precision	Recall	F1-Score	Precision	Recall	F1-Score	Precision	Recall	F1-Score
DeepLabv3	0.703	0.707	0.683	0.633	0.705	0.619	0.663	0.612	0.569	0.64	0.742	0.719	0.704	0.65	0.73

Table 4.12 summarizes the city-based performance of the DeepLabV3 model trained with a 70-30 split. The model demonstrates strong and consistent results across all cities, with total precision, recall, and F1-Score values of 0.707, 0.704, and 0.681, respectively. Among the cities, Paris shows the highest F1-Score (0.704) and IoU (0.65), indicating

particularly reliable performance in this location. While precision is slightly lower in Munich (0.619), the recall and overall F1-Score remain competitive, suggesting balanced predictions. Overall, DeepLabV3 performs robustly across all cities, with minimal performance variation and high generalization under the 70-30 data split.

Table 4.13: Class based results for YOLOv8 70-30

	Istanbul			Munich			Paris			Marseilles			Total		
	Precision	Recall	mAP50	Precision	Recall	mAP50	Precision	Recall	mAP50	Precision	Recall	mAP50	Precision	Recall	mAP50
Pedestrian	0.740	0.349	0.417	0.661	0.639	0.638	0.578	0.363	0.384	0.675	0.331	0.385	0.664	0.421	0.456
Vehicle	0.806	0.680	0.735	0.777	0.807	0.836	0.774	0.744	0.771	0.812	0.764	0.817	0.792	0.749	0.790
Road	0.935	0.930	0.948	0.944	0.943	0.976	0.880	0.882	0.912	0.971	0.989	0.981	0.933	0.936	0.954

Table 4.13 presents class-based results for the YOLOv8 model trained with a 70-30 data split. The "road" class continues to show outstanding performance with nearly perfect precision (0.933), recall (0.936), and mAP50 (0.954), confirming the model's ability to detect road elements reliably across all cities. The "vehicle" class also performs well, achieving a high precision of 0.792 and a recall of 0.749, indicating strong localization and classification capabilities. However, the "pedestrian" class demonstrates weaker results with a relatively low recall of 0.421, despite a moderate precision of 0.664, suggesting the model misses many pedestrian instances. Among the cities, Munich achieves the best overall balance for pedestrian detection, with both precision and recall above 0.6. Overall, while YOLOv8 performs robustly for road and vehicle classes, improvements are still needed for pedestrian detection under this data configuration.

Table 4.14: Class based results for YOLOv10 70-30

	Istanbul			Munich			Paris			Marseilles			Total		
	Precision	Recall	mAP50	Precision	Recall	mAP50	Precision	Recall	mAP50	Precision	Recall	mAP50	Precision	Recall	mAP50
Pedestrian	0.621	0.306	0.320	0.599	0.427	0.414	0.511	0.251	0.262	0.688	0.277	0.340	0.605	0.315	0.330
Vehicle	0.698	0.620	0.616	0.720	0.709	0.704	0.697	0.650	0.650	0.761	0.743	0.788	0.719	0.681	0.690
Road	0.916	0.876	0.876	0.891	0.866	0.885	0.866	0.828	0.831	0.968	0.985	0.973	0.910	0.889	0.891

Table 4.14 displays the class-based performance metrics for the YOLOv10 model trained with a 70-30 data split. The "road" class continues to exhibit excellent results, with total precision at 0.910 and recall at 0.889, indicating the model's strong and consistent ability to detect road features across all cities. The "vehicle" class also performs well, especially in Marseille where both recall (0.743) and mAP50 (0.788) are particularly high. In contrast, the "pedestrian" class shows the weakest performance, with a total recall of only 0.315 and precision of 0.605, suggesting that the model struggles to detect this class, especially in Paris where recall drops to 0.251. These results imply that while YOLOv10 is effective for well-defined and prominent classes like roads and vehicles, its detection capabilities for smaller or less distinct objects like pedestrians remain limited.

Table 4.15: Class based results for DeepLabV3 70-30

	Istanbul					Munich					Paris					Marseilles					Total				
	Precision	Recall	F1	IoU	Dice	Precision	Recall	F1	IoU	Dice	Precision	Recall	F1	IoU	Dice	Precision	Recall	F1	IoU	Dice	Precision	Recall	F1	IoU	Dice
Pedestrian	0.131	0.085	0.089	0.061	0.103	0.112	0.058	0.063	0.043	0.077	0.083	0.061	0.045	0.025	0.070	0.107	0.052	0.059	0.041	0.070	0.111	0.065	0.068	0.045	0.082
Vehicle	0.452	0.484	0.410	0.313	0.467	0.504	0.460	0.420	0.323	0.481	0.338	0.452	0.303	0.205	0.387	0.637	0.600	0.570	0.450	0.618	0.472	0.463	0.400	0.297	0.467
Road	0.936	0.968	0.948	0.911	0.952	0.912	0.956	0.924	0.880	0.933	0.963	0.972	0.965	0.936	0.967	0.913	0.914	0.911	0.876	0.914	0.930	0.949	0.934	0.895	0.939
Background	0.986	0.964	0.973	0.950	0.974	0.982	0.954	0.963	0.937	0.968	0.975	0.977	0.969	0.952	0.976	0.982	0.961	0.968	0.946	0.971	0.974	0.962	0.963	0.938	0.968

Table 4.15 presents the class-based performance metrics of the DeepLabV3 model trained with a 70-30 data split. The model demonstrates excellent results in the "road" class, achieving a total F1 score of 0.949 and an IoU of 0.934, indicating highly accurate and consistent segmentation across all cities. The "background" class also performs strongly, with a Dice score of 0.968, reflecting robust classification in non-target regions. However,

the performance on the "pedestrian" class remains critically low, with an overall IoU of just 0.045 and F1 score of 0.068, suggesting the model struggles considerably with detecting smaller, less frequent objects. The "vehicle" class shows moderate success (F1: 0.463, IoU: 0.297), but its performance is still far below that of the road and background categories.

Table 4.16: City based results for YOLO models with 95-5

	Istanbul			Munich			Paris			Marseille			Total		
Model	Precision	Recall	mAP50	Precision	Recall	mAP50	Precision	Recall	mAP50	Precision	Recall	mAP50	Precision	Recall	mAP50
YOLOv8	0.739	0.593	0.611	0.741	0.643	0.655	0.875	0.733	0.787	0.763	0.663	0.673	0.78	0.658	0.682
YOLOv10	0.712	0.575	0.585	0.725	0.666	0.661	0.861	0.78	0.795	0.766	0.681	0.678	0.766	0.675	0.68

Table 4.16 provides a comparative overview of YOLOv8 and YOLOv10 models trained with a 95-5 data split, evaluated across four cities and in total. YOLOv8 achieves slightly higher overall precision (0.780) compared to YOLOv10 (0.766), though YOLOv10 yields better recall (0.675 vs. 0.658), indicating it retrieves more true positives. In terms of mAP50, both models perform similarly, with YOLOv8 marginally ahead (0.682 vs. 0.680). Paris stands out as the best-performing city for both models, particularly in recall and precision metrics, suggesting that the dataset from this city may be more consistent or easier to segment. Overall, both models deliver strong performances, with minor trade-offs between precision and recall depending on the city and architecture.

Table 4.17: City based results for DeepLabV3 95-5

	Istanbul			Munich			Paris			Marseille			Total		
Model	Precision	Recall	F1-Score	Precision	Recall	F1-Score	Precision	Recall	F1-Score	Precision	Recall	F1-Score	Precision	Recall	F1-Score
DeepLabv3	0.679	0.7	0.674	0.627	0.69	0.604	0.695	0.587	0.538	0.647	0.69	0.7	0.674	0.628	0.695

Table 4.17 presents the city-wise and overall performance metrics of the DeepLabv3 model trained on a 95-5 data split. The model demonstrates balanced precisions and recalls across all cities, with the highest total recall (0.705) and an F1-score of 0.664, indicating a well-rounded detection capability. Marseille stands out with the highest individual performance, especially in recall (0.741) and IoU (0.662), reflecting better segmentation quality in that region. Conversely, Munich exhibits the lowest IoU (0.538),

suggesting that the model struggled more with segmentation accuracy in this city. Overall, DeepLabv3 maintains consistent generalization across cities, though with observable room for improvement in spatial overlap measures like IoU.

Table 4.18: Class based results for YOLOv8 95-5

	Istanbul			Munich			Paris			Marseilles			Total		
	Precision	Recall	mAP50	Precision	Recall	mAP50	Precision	Recall	mAP50	Precision	Recall	mAP50	Precision	Recall	mAP50
Pedestrian	0.618	0.305	0.341	0.592	0.496	0.442	0.830	0.463	0.520	0.605	0.291	0.297	0.661	0.389	0.400
Vehicle	0.629	0.521	0.488	0.657	0.670	0.682	0.752	0.878	0.869	0.747	0.751	0.760	0.703	0.707	0.725
Road	0.888	0.898	0.926	0.926	0.831	0.858	1.000	1.000	0.995	0.946	1.000	0.977	0.952	0.932	0.939

Table 4.18 presents the class-based evaluation metrics of the YOLOv8 model trained with a 95-5 data split. The "Road" class consistently achieves superior results across all cities, culminating in an outstanding total mAP50 of 0.939 and recall of 0.932, indicating highly accurate and complete detections. The "vehicle" class also performs well especially in Paris and Marseille where recall exceeds 0.75 and leading to a strong total mAP50 of 0.725. In contrast, the "Pedestrian" class continues to underperform, with a total mAP50 of only 0.400 and the lowest recall (0.389), suggesting significant difficulty in identifying this class. These results highlight the model's strength in detecting structured and visually distinct classes like roads, while underscoring the challenges posed by less frequent or more variable objects like pedestrians.

Table 4.19: Class based results for YOLOv10 95-5

	Istanbul			Munich			Paris			Marseilles			Total		
	Precision	Recall	mAP50	Precision	Recall	mAP50	Precision	Recall	mAP50	Precision	Recall	mAP50	Precision	Recall	mAP50
Pedestrian	0.692	0.336	0.385	0.599	0.436	0.437	0.928	0.347	0.492	0.590	0.256	0.288	0.702	0.344	0.401
Vehicle	0.622	0.546	0.528	0.695	0.662	0.678	0.777	0.851	0.874	0.753	0.734	0.754	0.719	0.698	0.733
Road	0.902	0.898	0.920	0.929	0.831	0.849	0.920	1.000	0.995	0.947	1.000	0.978	0.934	0.933	0.935

Table 4.19 shows the class-based evaluation results of the YOLOv10 model trained with a 95-5 data split. The "road" class once again demonstrates exceptional performance, achieving a total precision of 0.934, recall of 0.933, and an impressive mAP50 of 0.935 indicating consistent and highly accurate predictions. The "vehicle" class also performs strongly, especially in Paris (mAP50 = 0.874) and Marseille (mAP50 = 0.754), culminating in a total mAP50 of 0.733. However, the "Pedestrian" class continues to be the most challenging, with a low total recall of 0.344 and a modest mAP50 of 0.401. Despite high precision values in cities like Paris (0.928), the corresponding low recall suggests that the model identifies only a limited portion of pedestrian instances, likely due to data imbalance or visual complexity. Overall, YOLOv10 maintains strong performance on well-defined classes but struggles to generalize for more variable and underrepresented categories.

Table 4.20: Class based results for DeepLabV3 95-5

	Istanbul					Munich					Paris					Marseilles					Total				
	Precision	Recall	F1	IoU	Dice	Precision	Recall	F1	IoU	Dice	Precision	Recall	F1	IoU	Dice	Precision	Recall	F1	IoU	Dice	Precision	Recall	F1	IoU	Dice
Pedestrian	0.131	0.085	0.089	0.061	0.103	0.112	0.058	0.063	0.043	0.077	0.083	0.061	0.045	0.025	0.070	0.107	0.052	0.059	0.041	0.070	0.108	0.064	0.064	0.043	0.081
Vehicle	0.452	0.484	0.410	0.313	0.467	0.504	0.460	0.420	0.323	0.481	0.338	0.452	0.303	0.205	0.387	0.637	0.600	0.570	0.450	0.618	0.483	0.499	0.425	0.323	0.491
Road	0.936	0.968	0.948	0.911	0.952	0.912	0.956	0.924	0.880	0.933	0.963	0.972	0.965	0.936	0.967	0.913	0.914	0.911	0.876	0.914	0.931	0.953	0.937	0.901	0.942
Background	0.986	0.964	0.973	0.950	0.974	0.982	0.954	0.963	0.937	0.968	0.975	0.977	0.969	0.952	0.976	0.982	0.961	0.968	0.946	0.971	0.981	0.964	0.968	0.946	0.972

Table 4.20 presents the class-based evaluation results of the DeepLabV3 model trained using a 95-5 split. The "road" and "background" classes exhibit exceptional segmentation performance, with total F1-scores of 0.937 and 0.968, and IoU scores of 0.901 and 0.946

respectively, indicating precise and consistent predictions. The "vehicle" class shows moderate performance with a total F1-score of 0.425 and IoU of 0.323, revealing room for improvement in handling vehicle boundaries. The "pedestrian" class performs poorly, with very low total recall (0.064) and IoU (0.043), suggesting that the model struggles to detect or segment pedestrians effectively, likely due to their relatively small size or visual variability in the dataset. Overall, DeepLabV3 delivers highly reliable results for dominant classes, but its performance significantly deteriorates for smaller or underrepresented classes like pedestrians.

5. DISCUSSION

This chapter seeks to critically evaluate the methodologies employed and the findings reported in this study. While the previous sections focused on the design, implementation, and quantitative evaluation of the models, this section explores their broader implications, limitations, and practical relevance in real-world scenarios. In particular, the discussion revisits the proposed deep learning pipeline and evaluates its performance in the context of urban traffic scene recognition, using only image-based data without the aid of LiDAR or other external sensors.

The deep learning models developed for this research include YOLOv8, YOLOv10, and DeepLabV3. These models were selected based on their suitability for real-time tasks and their strong performance in object detection and semantic segmentation. Each model was tested using a custom dataset collected from Google Street View, covering urban scenes from various cities with different traffic structures and environmental conditions. This diversity allowed for a realistic assessment of model generalization and adaptability.

The discussion has two main parts. The first part explains the features of the proposed methods. It explains the architectural design of the models, the training configurations, and the approach used to construct and annotate the dataset. The reasons behind the selection of each model are presented clearly, along with their advantages in terms of detection accuracy and ease of implementation.

The second section focuses on comparing the performance of the models. Standard metrics like precision, recall, mAP, and IoU are analyzed in detail. The results are discussed both in terms of overall performance and class-specific outcomes, including roads, vehicles, and pedestrians, as summarized in the city-based and class-based metrics in Tables 4.1–4.2, 4.6–4.7, 4.11–4.12, 4.16–4.17 (overall), and Tables 4.3–4.5, 4.8–4.10,

4.13–4.15, 4.18–4.20 (class-specific). This class-based analysis helps to identify the strengths and weaknesses of the models under different conditions.

In conclusion, this chapter highlights the practical opportunities and technical challenges of using a complete vision-based system for urban traffic scene understanding. The results suggest that such systems have the potential to support future intelligent transport applications. On the other hand, further research is needed to improve the recognition of complex and less visible elements like crowded or visually noisy urban environments.

In this study, a deep learning-based framework was proposed to detect and segment key elements in urban traffic scenes using 2D projections of 360-degree camera images. The system was made to work without LiDAR or radar sensors, using only camera images to recognize objects. This choice was based on the idea that using only cameras would be cheaper and easier for self-driving cars, especially in cities where visual information is usually enough for driving.

The detection tasks were handled using two advanced object detection models: YOLOv8 and YOLOv10. These models belong to the YOLO family, which is known for real-time detection and fast inference (Redmon et al., 2016; Ultralytics, 2023; Wang et al., 2024). Both YOLOv8 and YOLOv10 were trained using the Ultralytics library with consistent hyperparameters to allow for a fair comparison. The training was completed for 100 epochs using images downsized to 1024x768 pixels. SGD was selected as the optimization method, and performance was monitored through metrics such as loss values, precision, recall, and mAP.

For semantic segmentation, DeepLabV3 was used. It has worked well in earlier studies because it can find features at different scales using a method called atrous spatial pyramid pooling (Chen et al., 2017). In this study, DeepLabV3 was implemented with a MobileNetV3 Large backbone to balance performance and computational efficiency. The segmentation model was trained for 100 epochs using the PyTorch library and evaluated using standard metrics like precision, recall, F1 score, and IoU for each class.

The dataset used for training and validation was created manually using Google Street View imagery from four cities: Istanbul, Paris, Marseille, and Munich. The images were labeled through the Roboflow platform, resulting in a dataset containing over 8,900

annotated images and more than 149,000 labeled instances. The classes of interest were "road", "vehicle", and "pedestrian" which represent the fundamental components required for understanding traffic scenes. The labeling process focused on consistency and accuracy, and bounding boxes were used for detection tasks, while pixel-wise masks were created for segmentation.

The proposed method uses strong detection models and an efficient segmentation system with a special dataset to develop lightweight vision-based systems for smart transportation. Using different model types together in one system helps it adapt to many city situations and work in real time on devices with limited hardware.

The system's performance was evaluated through a variety of experiments involving both object detection and semantic segmentation models. The evaluation checked not only how accurate the models are but also how well they work on different parts of the dataset and in various city environments.

For object detection, YOLOv8 and YOLOv10 were tested using several performance measures like mAP, precision, recall, and F1 score. These measures were calculated separately for the road, vehicle, and pedestrian classes using different train-validation-test splits such as 65-35, 70-30, 75-25, and 95-5. Across all splits, the road class consistently showed the highest detection accuracy, as seen in class-based results in Tables 4.3, 4.4, 4.8, 4.9, 4.13, 4.14, 4.18, and 4.19. Both YOLO models achieved over 90% correct identification for roads, indicating that the models could easily distinguish road surfaces due to their uniform structure and high visual contrast.

Vehicle detection also demonstrated solid results, with YOLOv8 and YOLOv10 reaching moderate to high mAP scores, typically above 0.65 (Tables 4.3, 4.4, 4.8, 4.9, 4.13, 4.14, 4.18, 4.19). However, the detection of pedestrians remained the most challenging task, as seen in Tables 4.3, 4.4, 4.8, 4.9, 4.13, 4.14, 4.18, and 4.19, and Figures 4.44 to 4.51. In all configurations, the pedestrian class had the lowest precision and recall scores, often due to factors such as small object size, partial occlusions, and visual similarity with background elements. However, YOLOv10 showed more consistent results than YOLOv8 in high-confidence predictions, with smoother evaluation metrics, which can

be observed in Figures 4.17, 4.23, 4.29, and 4.35. It had smoother learning curves and fewer changes in evaluation results as illustrated in Figures 4.3, 4.6, 4.9, and 4.12.

In the semantic segmentation task, DeepLabV3 showed strong performance, especially in the road class, where the IoU exceeded 0.85 in most cases. The training loss decreased steadily across all data splits, indicating a stable optimization process (Figures 4.60, 4.63, 4.64 and 4.66). The model worked well in finding clear edges and segmenting large, visible objects, as seen in Figures 4.68 to 4.71 and Figures 4.76 to 4.79. As expected, it made more mistakes with pedestrians because they appear smaller and less clear in the 360-degree images, making consistent labeling harder (Figures 4.72 to 4.75). When comparing the models, DeepLabV3 gave better accuracy for each pixel in the road and vehicle classes (Figures 4.68–4.71 and 4.76–4.79). YOLO models, on the other hand, were better at detecting objects with bounding boxes and were faster in making predictions (Figures 4.37–4.40 and 4.41–4.43). The performance patterns were similar in different cities, but some differences appeared due to urban layout and lighting conditions (see Tables 4.1, 4.6, 4.11 and 4.16).

In conclusion, each model has its own strengths, but using detection and segmentation together provides a balanced and complete solution. YOLOv10 stands out for detection tasks requiring real-time performance and stability, while DeepLabV3 is more suitable for applications that prioritize spatial accuracy (Figures 4.1–4.12 and 4.60–4.67; Tables 4.1–4.20). The analysis shows that vision-only systems can perform well in understanding city traffic scenes, but more work is needed, especially to better detect and segment pedestrians in difficult environments (Figures 4.72–4.75, 4.80–4.83; Tables 4.3, 4.4, 4.8, 4.9, 4.13, 4.14, 4.18, 4.19).

6. CONCLUSION

This thesis explored a deep learning-based framework for the recognition of urban traffic scene elements using image-only data, specifically 2D projections derived from 360-degree camera views. The study used a mix of object detection and semantic segmentation methods, with YOLOv8, YOLOv10, and DeepLabV3 models. A custom dataset was created from Google Street View images collected across four European cities, focusing on three key traffic scene classes: roads, vehicles, and pedestrians.

The models were trained and tested on different parts of the data. Their performance was measured using common metrics like precision, recall, F1 score, mAP, and IoU, as detailed in Tables 4.1–4.20 and visualized in Figures 4.1–4.36. The results showed that vision-only systems can effectively detect and segment large, well-defined objects like roads and vehicles. However, challenges remain in detecting smaller or visually complex elements such as pedestrians (see Tables 4.5, 4.10, 4.15, 4.20 and Figures 4.44–4.51, 4.80–4.83).

The main findings show that a well-designed vision-based system can help develop smart transportation, even without LiDAR or radar. The study's results reveal both the strengths and the current weaknesses of deep learning models in understanding complex city scenes, highlighting their effectiveness in detecting large and well-defined objects while also underscoring the limitations these models face in handling smaller, occluded, or visually ambiguous elements commonly found in real-world urban environments. These limitations emphasize the importance of improving model sensitivity to subtle visual cues and incorporating more diverse urban scenarios in training data. Addressing these gaps could significantly enhance the reliability and applicability of vision-based systems in real-world autonomous driving contexts.

6.1 Thesis Contribution

The main contribution of this thesis is creating a deep learning system that recognizes important parts of city traffic scenes using only 2D images from 360-degree cameras. Unlike many existing approaches that depend on expensive and complex sensor systems such as LiDAR or radar, this study proposed a vision-only solution, demonstrating that meaningful object detection and semantic segmentation can be achieved with significantly lower hardware requirements. Another important contribution is the creation of a custom dataset composed of thousands of manually labeled panoramic images sourced from Google Street View. This dataset includes different street scenes from Istanbul, Paris, Munich, and Marseille. It offers a new resource for future research on visual traffic analysis without using LiDAR depth data. The thesis also compares advanced object detection models, YOLOv8 and YOLOv10, with the DeepLabV3 segmentation model, based on the performance results illustrated in Tables 4.3–4.5, 4.8–4.10, 4.13–4.15, 4.18–4.20 and corresponding Figures 4.1–4.12, 4.60–4.67. By applying consistent evaluation metrics across all models and dataset splits, the study presents a reliable performance benchmark for real-world applications. Combining detection and segmentation in one system shows how the proposed method can be useful for real-time understanding of city scenes in smart transport. This unified approach enables comprehensive scene interpretation, capturing both object boundaries and contextual relationships. Such integration is particularly valuable for advanced driver assistance systems and future autonomous navigation technologies.

6.2 Limitations and Future Work

Although the results of this research are promising, several limitations must be acknowledged. One of the most notable challenges encountered was the relatively low performance in detecting and segmenting pedestrians, particularly in complex scenes or when the individuals were small or partially obscured. This issue may be attributed to

class imbalance in the dataset as well as the inherent difficulty of capturing fine-grained detail using 2D projections of wide-angle imagery. Future research could address this by employing targeted data augmentation techniques, increasing the diversity of labeled pedestrian instances, or implementing attention-based mechanisms that improve small object sensitivity. Another limitation concerns the geographic and environmental scope of the dataset. Although four cities were included, the dataset still lacks variability in terms of weather conditions, lighting scenarios, and street types, which may affect the generalizability of the models. Adding more diverse urban scenes to the dataset would make the system stronger and more reliable. Additionally, the use of equirectangular projections, while effective for compatibility, introduces distortions that could be mitigated by exploring alternative representations such as cubemap or spherical CNN approaches. Lastly, the models were not tested in real-time embedded environments, which leaves open the question of system performance under hardware constraints. Future studies could focus on deploying and optimizing the models on edge devices or in-vehicle platforms to better assess their readiness for practical deployment in autonomous systems.

REFERENCES

- Ahn, D. and Kwak, B. (2018). Learning pixel-level semantic affinity with image-level supervision. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
- Arnold, E., Al-Jarrah, O. Y., Dianati, M., Fallah, S., Oxtoby, D., & Mouzakitis, A. (2019). A Survey on 3D Object Detection Methods for Autonomous Driving Applications. *IEEE Transactions on Intelligent Transportation Systems*, 20(10), 3782–3795.
- Chen, L.-C., Papandreou, G., Schroff, F., and Adam, H. (2017). Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*.
- Chen, L.-C., Zhu, Y., Papandreou, G., Schroff, F., & Adam, H. (2018). Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European conference on computer vision (ECCV)*.
- Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., ... & Schiele, B. (2016). *The Cityscapes Dataset for Semantic Urban Scene Understanding*. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 3213–3223.
- Dwyer, B., Nelson, J., Hansen, T., et al. (2024). Roboflow (Version 1.0) [Software]. URL: <https://roboflow.com>
- Geiger, A., Lenz, P., & Urtasun, R. (2012). *Are we ready for autonomous driving? The KITTI vision benchmark suite*. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 3354–3361.
- Google (2025). Street View Static API overview. *Google Maps Platform Documentation*. URL: <https://developers.google.com/maps/documentation/streetview/overview>

Helbich, M., Danish, M., Labib, S. M., and Ricker, B. (2024). To use or not to use proprietary street view images in (health and place) research? That is the question. *Health & Place*, 87, 103244.

Huang, L.-C., et al. (2019). CCNet: Criss-Cross Attention for semantic segmentation. *CVPR*.

Kim, J., Jang, L., Chiang, Y. Y., Wang, G., & Pasco, M. (2025). StreetLens: Enabling Human-Centered AI Agents for Neighborhood Assessment from Street View Imagery. *arXiv preprint arXiv:2506.14670*.

Kohavi, R. (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection. *IJCAI*, 1137-1143.

Li, Z., Wang, H., Sun, Y., Chen, M., & Liu, J. (2025). RADLER: Radar object detection leveraging semantic 3D city models. *arXiv preprint arXiv:2504.12167*.

Lin, T.-Y., Maire, M., Belongie, S., et al. (2014). Microsoft COCO: Common Objects in Context. *ECCV*, 740-755.

Mazzini, M., et al. (2018). A CNN architecture for efficient semantic segmentation of street scenes. *ECCV Workshops*.

Ning, H., Li, Z., Yu, M., & Yin, W. (2025). SIM: A Mapping Framework for Built Environment Auditing Based on Street View Imagery. *arXiv preprint arXiv:2505.24076*.

Perez, J., & Fusco, G. (2025). Streetscape Analysis with Generative AI (SAGAI): Vision-Language Assessment and Mapping of Urban Scenes. Preprint, 5226191.

Pohlen, T., et al. (2019). Full-resolution residual networks for semantic segmentation. *CVPR*.

Pravallika, A., Hashmi, M. F., & Gupta, A. (2024). Deep Learning Frontiers in 3D Object Detection: A Comprehensive Review for Autonomous Driving. *IEEE Access*, 12, 173936–173980. doi:10.1109/ACCESS.2024.3456893.

Qian, R., Lai, X., & Li, X. (2022). 3D Object Detection for Autonomous Driving: A Survey. *Pattern Recognition*, 130, 108796.

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You Only Look Once: Unified, Real-Time Object Detection. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 779–788.

Sun, P., Kretschmar, H., Dotiwalla, X., et al. (2020). Scalability in perception for autonomous driving: Waymo Open Dataset. *CVPR*, 2446-2454.

Tan, Y., et al. (2020). Night-time semantic segmentation with a large real dataset. *ICCV*.

TRUBA. (2023). Turkish National e-Science e-Infrastructure. Retrieved from <https://truba.gov.tr>

Ultralytics (2023). Ultralytics YOLOv8 [software].
URL: <https://github.com/ultralytics/ultralytics>

Wahid, S. and Lee, Y. (2019). Deconvolution pixel layer based semantic segmentation for street view images. *IEEE ICIP*.

Wang, A., Chen, H., Liu, L., Chen, K., Lin, Z., & Han, J. (2024). YOLOv10: Real-Time End-to-End Object Detection. *Advances in Neural Information Processing Systems*, 37, 107984–108011.

Wang, J., Su, J., Wen, Z., & Sun, Y. (2025). Enhanced YOLOv10 for small object detection with context-aware and adaptive modules. *International Journal of Multimedia Information Retrieval*. [Online].
<https://www.researchgate.net/publication/392916992>

Wang, S., et al. (2019). PANet: Few-shot image semantic segmentation with prototype alignment. *ICCV*.

Yang, W., Qian, Y., Kämäräinen, J.-K., Cricri, F., and Fan, L. (2018). Object detection in equirectangular panorama. *arXiv preprint arXiv:1805.08009*.

Yao, B., Chen, Y.-T., Liang, F., et al. (2019). The H3D dataset for full-surround 3D multi-object detection and tracking in crowded urban scenes. *IEEE ICRA*, 9236-9243.

Zhang, J., Yang, Y., Gao, Z., Shi, H., & Xie, Y. (2025). SMM-POD: Panoramic 3D object detection via spherical multi-stage multi-modal fusion. *Remote Sensing*, 17(12), 2089.

Zhao, H., et al. (2018). ICNet for real-time semantic segmentation on high-resolution images. *ECCV*.

Zhu, Y., et al. (2025). Dur360BEV: A real-world 360-degree single camera dataset and benchmark for autonomous driving. *arXiv preprint arXiv:2503.00675*.



BIOGRAPHICAL SKETCH

Bahadır Akın Akgül

Galatasaray University M.S. Computer Engineering (2022 -)

Eskişehir Osmangazi University B.S. Electrical and Electronics Engineering (2010-2016)

PUBLICATIONS

B. Akgül and K. Fidanboyu, "A Review for Artificial Intelligence Solutions at E-commerce field", 4th International Conference on Data Science and Applications (ICONDATA'21), 2021.