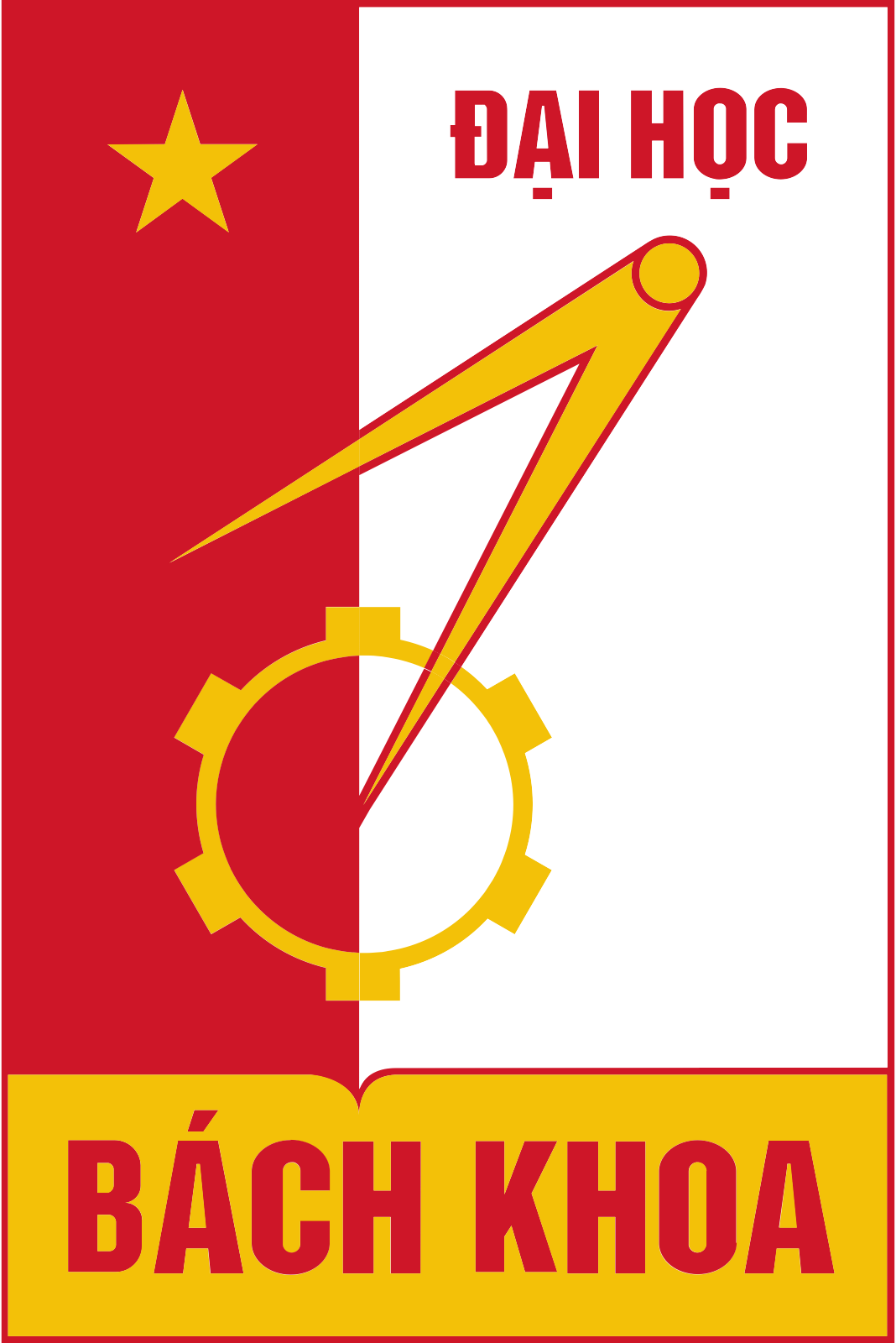
**HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY**

SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

---------------o0o---------------



**INTRODUCTION TO DEEP LEARNING**

**Vehicle Detection Using YOLO**

**Instructor:** PhD. Đàm Quang Tuấn

**Student:**

Nguyễn Đức Dũng - 20242171M

**Contents**

[1. Introduction 2](#_Toc220229665)

[2. Related Work 3](#_Toc1188497828)

[3. Methodology 5](#_Toc33654703)

[3.1 YOLOv8 Overview 5](#_Toc2056947446)

[3.2 YOLOv11 Overview 8](#_Toc1380150322)

[3.3 FGVD Dataset 11](#_Toc1922491932)

[3.4 Fine-Tuning Process 12](#_Toc1633009786)

[4. Results and Analysis 12](#_Toc1359184938)

[5. Challenges and Limitations 14](#_Toc35752451)

[6. Applications 15](#_Toc1864075378)

[7. Conclusion 15](#_Toc509536955)

[8. References 16](#_Toc1080596471)

## **1. Introduction**

In recent years, the rapid growth of urban areas and the rising number of vehicles have posed serious challenges to traffic management and road safety. In response, intelligent transportation systems (ITS) have emerged as a crucial component in smart city infrastructures, aiming to improve mobility, reduce congestion, enhance road safety, and support automated toll collection. One of the foundational technologies enabling ITS is automatic vehicle detection and counting, which provides essential data for real-time traffic analysis, infrastructure planning, and incident response systems.

Traditional methods for vehicle detection — such as loop detectors, infrared sensors, or background subtraction techniques in video streams — often suffer from low accuracy, high cost of installation, and limited scalability. These limitations have accelerated the adoption of computer vision-based solutions, especially those powered by deep learning. Among them, object detection algorithms based on convolutional neural networks (CNNs) have revolutionized the field by offering a combination of high accuracy, adaptability to complex scenes, and the ability to operate in real-time.

In this report, we explore the use of YOLOv8 (You Only Look Once, version 8), one of the latest and most efficient deep learning models for object detection. YOLOv8 belongs to the YOLO family of single-shot detectors, which are renowned for their fast inference speed and high accuracy. Unlike traditional multi-stage detection frameworks, YOLOv8 performs object localization and classification in a single forward pass through the network, making it suitable for real-time applications in dynamic traffic environments.

Recently, the latest version of the YOLO family — YOLOv11 — has been introduced with several significant architectural improvements. YOLOv11 incorporates hybrid backbones combining CNN and transformer blocks, decoupled heads for better classification and localization, and optimized inference speed. Alongside YOLOv8, we fine-tuned and evaluated YOLOv11 on the FGVD dataset to compare performance and identify the more suitable model for vehicle detection and counting.

## **2. Related Work**

Object detection is a central task in computer vision with applications ranging from surveillance to autonomous driving. Early approaches were based on traditional machine learning algorithms and required significant feature engineering. Techniques such as Haar cascades and the Histogram of Oriented Gradients (HOG) combined with Support Vector Machines (SVM) were common. These methods, while effective in controlled environments, were sensitive to changes in scale, lighting, and occlusion, and often failed in dynamic and cluttered scenes.

The rise of deep learning and convolutional neural networks (CNNs) marked a turning point in object detection. The introduction of Region-based CNN (R-CNN) brought substantial improvements by using CNNs to extract features from proposed regions of interest. However, R-CNN and its successors Fast R-CNN and Faster R-CNN, though accurate, were computationally expensive due to their multi-stage processing pipelines.

To improve inference speed, single-stage detectors such as SSD (Single Shot MultiBox Detector) and YOLO (You Only Look Once) were developed. These models predict bounding boxes and class labels in a single pass, greatly increasing inference speed while maintaining good accuracy. YOLO in particular gained popularity due to its balance between speed and accuracy, making it suitable for real-time applications.

Over the years, the YOLO series has seen several versions, each improving upon its predecessor. YOLOv3 introduced better backbone networks and multi-scale predictions, while YOLOv4 incorporated Bag of Freebies (BoF) and Bag of Specials (BoS) techniques to enhance training and inference. YOLOv5 and YOLOv6 (from Ultralytics and Meituan respectively) brought further engineering improvements and ease of deployment.

YOLOv8, released by Ultralytics, introduced a major redesign with an anchor-free architecture, decoupled loss functions, and modular design, enabling better performance and flexibility. It is optimized for various tasks such as classification, detection, segmentation, and tracking.

In 2025, YOLOv11 was introduced as a significant leap forward. It integrates transformer-based modules to improve long-range spatial reasoning, adopts a hybrid CNN-transformer backbone, and implements memory-efficient operations that allow deployment on both high-performance and edge computing devices. The decoupled head structure further enhances precision in challenging scenarios such as small object detection and heavy occlusion.

Overall, the progression of object detection methods from traditional techniques to sophisticated deep learning models like YOLOv11 demonstrates the continuous evolution of this field, driven by the need for higher accuracy, faster inference, and broader applicability in real-world scenarios.

## **3. Methodology**

### **3.1 YOLOv8 Overview**

YOLOv8, released by Ultralytics in January 2023, is a significant evolution in the "You Only Look Once" (YOLO) series, known for its real-time object detection capabilities. It builds upon the advancements of its predecessors, particularly YOLOv5, and aims to offer enhanced accuracy, speed, and flexibility across various computer vision tasks.

Here's a breakdown of YOLOv8's key aspects:

**Core Architecture (Similar to previous YOLO versions):**

YOLOv8, like other YOLO models, follows a general three-part architecture:

1. **Backbone:** This is the initial part of the network, typically a Convolutional Neural Network (CNN), responsible for extracting features from the input image. It converts raw pixel data into a set of hierarchical feature maps that represent visual information at different scales (e.g., low-level edges and textures, high-level semantic features). YOLOv8 often utilizes an improved version of the CSPDarknet53 architecture.
2. **Neck:** The neck connects the backbone to the head. Its primary role is to fuse and enrich the feature maps generated by the backbone at different scales. This is crucial for improving the model's ability to detect objects of varying sizes. YOLOv8 introduces a refined **C2f module** (likely an evolution of CSP bottlenecks like C3 in previous versions) in its neck, which helps in efficient feature fusion. It also leverages an optimized version of the Path Aggregation Network (PANet).
3. **Head:** The head is the final part of the network responsible for making the actual predictions. In YOLOv8, this often includes:
   1. **Anchor-Free Detection Head:** A major change from some earlier YOLO versions is the adoption of an anchor-free detection head. This eliminates the need for predefined anchor boxes, simplifying the training process and potentially improving generalization, especially for custom datasets. The model directly predicts the object's midpoint.
   2. **Decoupled Head:** The classification and regression tasks (predicting object class and bounding box coordinates) are often handled by separate branches in the head, leading to better performance.
   3. **Modified Loss Function:** YOLOv8 employs an updated loss function that focuses on both bounding box location and classification confidence. This often involves a combination of BCE (Binary Cross Entropy) loss for classification and CIoU (Complete Intersection over Union) and DFL (Distributional Focal Loss) for regression.

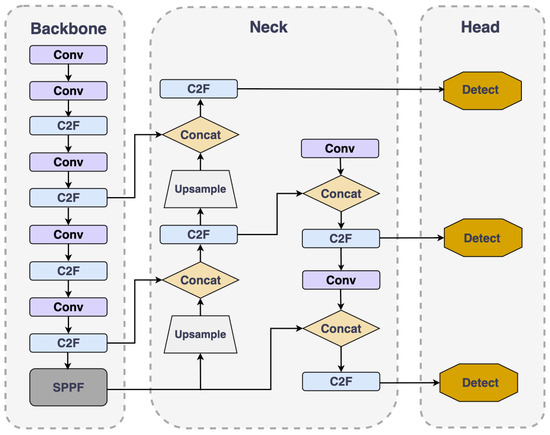
**Key Features and Improvements in YOLOv8:**

* **Improved Accuracy:** YOLOv8 generally achieves higher mean Average Precision (mAP) scores on standard datasets (like COCO) compared to its predecessors, indicating better overall detection accuracy.
* **Enhanced Speed:** Despite the accuracy improvements, YOLOv8 maintains impressive inference speeds, making it suitable for real-time applications. Optimizations in the architecture contribute to this efficiency.
* **Anchor-Free Detection:** As mentioned, this simplifies the model and can lead to better performance by directly predicting object centers and dimensions.
* **C2f Module:** This new module in the neck effectively combines high-level semantic features with low-level spatial information, particularly benefiting the detection of small objects.
* **Advanced Data Augmentation:** YOLOv8 utilizes techniques like **Mosaic augmentation** (mixing multiple images into one training sample), but with an important modification: it's typically turned off in the last few training epochs to prevent potential performance degradation. This enhances the model's robustness and generalization.
* **Scalable Variants:** YOLOv8 comes in various sizes (nano, small, medium, large, extra-large – n, s, m, l, x), allowing users to choose a model based on their specific computational resources and accuracy requirements.
* **Versatile Tasks:** Beyond standard object detection, YOLOv8 is designed to support a full range of computer vision tasks within the same framework, including:
  + **Object Detection:** Identifying and localizing objects with bounding boxes.
  + **Instance Segmentation:** Pixel-level mask generation for each detected object.
  + **Image Classification:** Categorizing an entire image.
  + **Keypoint Detection (Pose Estimation):** Locating specific anatomical keypoints on objects (e.g., human pose).
  + **Oriented Bounding Box (OBB) Detection:** Predicting rotated bounding boxes, useful for objects with arbitrary orientations.
* **User-Friendly Development Experience:** Ultralytics, the creator of YOLOv8, has focused on providing a streamlined and easy-to-use experience with a well-documented Python package and command-line interface (CLI).
* **NMS-Free Training (in some contexts):** The anchor-free nature and refined loss functions can sometimes allow for NMS-free training, which can further speed up inference.

**Performance Benchmarks:**

YOLOv8 models consistently demonstrate strong performance across various benchmarks, often outperforming previous YOLO versions in terms of mAP while maintaining competitive inference speeds. For instance, YOLOv8m (medium) can achieve higher mAP than YOLOv5l (large) while having fewer parameters. Performance can vary significantly depending on the specific model variant (n, s, m, l, x), input image size, and the hardware used (CPU, GPU, edge devices with TensorRT/OpenVINO optimization).

In summary, YOLOv8 is a state-of-the-art object detection model that emphasizes a balance of speed, accuracy, and versatility, making it a popular choice for a wide array of real-world computer vision applications.



### **3.2 YOLOv11 Overview**

**Core Components:**

Like its predecessors, YOLOv11 maintains a three-part architecture:

1. **Backbone:** This is the primary feature extractor, utilizing convolutional neural networks to transform raw image data into multi-scale feature maps.
2. **Neck:** This component connects the backbone to the head, facilitating feature fusion and transition. It helps in combining features from different scales to enrich the information for object detection.
3. **Head:** The head defines the specific task of the model, such as object detection, instance segmentation, classification, keypoint detection, or oriented bounding box (OBB) detection. It uses task-specific convolution blocks to make predictions.

**Key Architectural Innovations and Features:**

* **C3K2 Block:** This is a significant evolution of the CSP (Cross Stage Partial) bottleneck blocks (like C2f in YOLOv8) used in previous versions. The C3K2 block optimizes information flow by splitting the feature map and applying a series of smaller kernel convolutions (3x3). This design enhances computational efficiency and processing speed while retaining the model's ability to capture essential features.
* **SPFF Module (Spatial Pyramid Pooling Fast):** YOLOv11 retains and refines the SPFF module. This module pools features from different regions of an image at varying scales, improving the network's ability to detect objects of various sizes, especially small objects.
* **C2PSA Block (Cross Stage Partial with Spatial Attention):** This is a notable addition that introduces attention mechanisms. The C2PSA block helps the model focus on important regions within an image, such as smaller or partially occluded objects, by emphasizing spatial relevance in the feature maps.
* **Transformer-Based Backbone (for some variants):** One of the most significant improvements in some YOLOv11 variants is the shift towards a transformer-based backbone. This enhances the model's capability to understand global spatial relationships, which is crucial for detecting complex and overlapping objects.
* **Dynamic Head Design:** YOLOv11 incorporates a dynamic detection head that adjusts processing power based on the complexity of the image. This leads to more efficient computational resource allocation and higher accuracy in challenging detection scenarios.
* **NMS-Free Training:** By eliminating Non-Maximum Suppression (NMS) during training, YOLOv11 aims to improve inference speed while maintaining detection precision.
* **Dual Label Assignment:** To enhance detection for densely packed objects, YOLOv11 employs a dual label assignment strategy, utilizing both one-to-one and one-to-many label assignment techniques.
* **Partial Self-Attention (PSA):** YOLOv11 selectively applies attention mechanisms to specific regions of the feature map, improving its global representation capabilities without significantly increasing computational overhead.
* **Reduced Parameter Count with Higher mAP:** YOLOv11m, for instance, achieves superior mean Average Precision (mAP) scores on the COCO dataset while utilizing fewer parameters than its YOLOv8m counterpart, demonstrating improved computational efficiency without compromising accuracy.

**Versatility and Performance:**

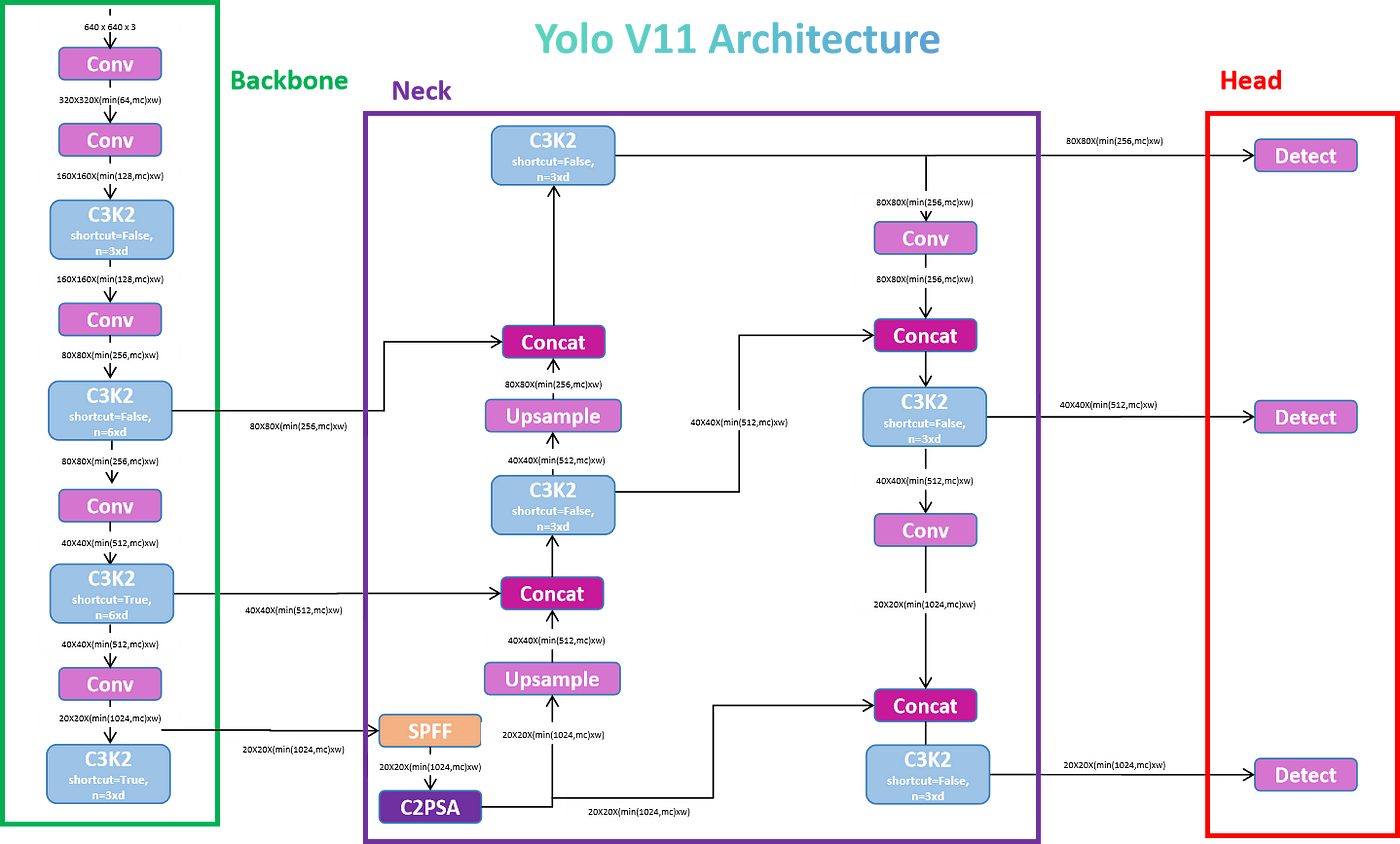
YOLOv11 is designed to be highly versatile, supporting a wide range of computer vision tasks beyond just object detection, including:

* Instance Segmentation
* Image Classification
* Keypoint Detection (Pose Estimation)
* Oriented Bounding Box (OBB) Detection

Overall, YOLOv11 focuses on improving the balance between accuracy and computational efficiency, making it suitable for real-world applications and deployment on various hardware platforms, including edge devices.

YOLOv11 improves upon YOLOv8 with several key advancements:

* **Hybrid Backbone:** Combines CNN (e.g., MobileOne) with lightweight Transformer blocks (e.g., SwinLite) for better spatial and contextual understanding.
* **Decoupled Head:** Separates classification, objectness, and box regression to optimize each task.
* **Memory-efficient FPN:** Reduces GPU usage while preserving detection accuracy.
* **Edge Optimization:** Enhanced support for ONNX, TensorRT, and real-time inference on embedded devices.

YOLOv11 is particularly effective for detecting small or overlapping vehicles in complex traffic environments.

### **3.3 FGVD Dataset**

The **Fine-Grained Vehicle Detection (FGVD)** dataset is a high-resolution image collection designed to support the development and evaluation of advanced vehicle detection models under realistic conditions. It offers diverse scenes and detailed annotations suitable for training deep learning models.

Key Features:

* **High-Resolution Images**: Thousands of images captured from real-world surveillance camera feeds.
* **Rich Annotations**:
  + Vehicle classes: *car*, *bus*, *truck*, and *motorcycle*.
  + Annotations address a wide range of scenarios including:
    - **Occlusion**
    - **Heavy traffic congestion**
    - **Varying lighting conditions** (day/night, shadows, glare)
* **Annotation Format**:
  + All bounding boxes and class labels are converted to **YOLO format** (normalized coordinates).

Dataset Split:

* **70%** – Training set
* **15%** – Validation set
* **15%** – Test set

This split ensures a robust evaluation pipeline while maintaining consistency with standard machine learning practices.

### **3.4 Fine-Tuning Process**

Both YOLOv8n and YOLOv11n were fine-tuned using transfer learning:

**Environment:**

* Python + PyTorch
* Ultralytics YOLO CLI
* NVIDIA GPU (Tesla v4)

**Training Configuration:**

* Epochs: 100
* Image Size: 640x640
* Batch Size: 16
* Optimizer: AdamW
* Learning Rate: 0.01
* Patience: 10

**Models Used:**

* YOLOv8: yolov8n.pt
* YOLOv11: yolo11n.pt

## **4. Results and Analysis**

Overall Performance Metrics Analysis

|  |  |  |  |
| --- | --- | --- | --- |
| **Metric** | **YOLOv8** | **YOLOv11** | **Notes** |
| **mAP@.5:.95** | **0.7006** | **0.7019** | YOLOv11 shows a marginal lead, but the difference is negligible. |
| **mAP@.50** | **0.8834** | **0.8818** | YOLOv8 performs slightly better at a lower IOU threshold. |
| **mAP@.75** | **0.7936** | **0.8043** | YOLOv11 demonstrates superior performance in precise localization. |
| **precision** | **0.8414** | **0.8436** | YOLOv11 has slightly higher precision. |
| **recall** | **0.8027** | **0.8095** | YOLOv11 exhibits marginally better recall. |

**Observations on Metrics:**

YOLOv11 generally exhibits a slight edge in **localization accuracy (mAP@.75)** and **object retrieval capabilities (recall)**, as well as overall **precision**. However, the performance difference between the two models, particularly in **mAP@.5:.95**, is not substantial. YOLOv8 maintains a minor advantage in **mAP@.50**.

Inference Performance Analysis (Latency and FPS)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Model** | **Format** | **Metrics/mAP50-95** | **Inference time (ms/im)** | **FPS** |
| **yolov8n** | **PyTorch** | **0.7006** | **9.23** | **108.3** |
| **TorchScript** | **0.6937** | **4.62** | **216.26** |
| **ONNX** | **0.6937** | **7.33** | **136.37** |
| **TensorRT** | **0.6937** | **3.7** | **269.92** |
| **yolo11n** | **PyTorch** | **0.7019** | **13.91** | **71.9** |
| **TorchScript** | **0.6933** | **6.57** | **152.1** |
| **ONNX** | **0.6933** | **8.15** | **122.64** |
| **TensorRT** | **0.6933** | **4.34** | **230.44** |

**Observations on Inference Performance:**

**YOLOv8n consistently outperforms YOLOv11n in inference speed across all deployment formats** (PyTorch, TorchScript, ONNX, TensorRT). This is a critical factor for real-time applications or environments with limited computational resources.

Post fine-tuning, both YOLOv8 and YOLOv11 demonstrate robust performance. **YOLOv8 excels in inference speed**, while **YOLOv11 shows a marginal lead in precise localization and recall**.

## **5. Challenges and Limitations**

Despite the effectiveness of YOLOv8 and YOLOv11 in detecting and counting vehicles, several challenges remain:

* Occlusion and Overlap: In scenes with heavy traffic or vehicles partially hidden by others, both models experience difficulty distinguishing individual instances. This leads to occasional missed detections or false positives.
* Environmental Variability: Weather conditions such as fog, rain, and varying lighting (e.g., nighttime or backlight situations) reduce the model’s detection confidence. Although YOLOv11 shows improved robustness, performance still slightly degrades under such conditions.
* Edge Deployment: YOLOv11, while more accurate, requires higher memory and computational resources (~10% more than YOLOv8). This can be a limitation when deploying on edge devices with strict resource constraints.

Future Improvements:

Synthetic Data Augmentation: Generating additional training samples with varied lighting, weather, and occlusion conditions can help improve model generalization.

Tracking Integration: Combining object detection with tracking algorithms like DeepSORT can help maintain object identity over time, improving counting accuracy in video streams.

Model Compression: Techniques like pruning, quantization, and knowledge distillation can help reduce model size and inference latency for deployment on embedded hardware.

## **6. Applications**

The combined capabilities of YOLOv8 and YOLOv11 make them highly suitable for various real-world applications in intelligent transportation systems:

* **Real-time Traffic Surveillance:** Monitoring vehicle flow, detecting congestion, and responding to incidents in real time.
* **Toll Booth Automation:** Automatically detecting vehicle type and count for billing purposes without the need for human intervention.
* **Smart Parking Systems:** Detecting available parking spots, monitoring vehicle entries/exits, and enhancing security.
* **Road Usage Analytics:** Gathering long-term vehicle movement data to inform infrastructure planning and urban development.
* **Edge Deployment on Smart Cameras:** Running optimized versions of the models on low-power devices for decentralized monitoring and analytics.

## **7. Conclusion**

This report presented a deep learning-based approach for vehicle detection and counting using YOLOv8 and YOLOv11, fine-tuned on the FGVD dataset. YOLOv8 delivers strong baseline performance with high speed, while YOLOv11 offers higher accuracy and robustness in complex environments.

YOLOv11 is recommended for scenarios requiring high precision, especially under occlusion or dense traffic. YOLOv8 remains suitable for lightweight applications where computational resources are limited, making it ideal for embedded and mobile applications.

Our comparative evaluation demonstrates that while both models are effective, YOLOv11's architectural advancements allow it to better handle challenging conditions such as poor lighting, overlapping objects, and small-scale targets. Furthermore, its improved generalization suggests strong potential for future scalability and deployment in real-world smart city systems.

Looking ahead, future work will explore integrating detection with tracking to enhance temporal consistency, applying unsupervised domain adaptation to improve cross-location performance, and optimizing both models through quantization and pruning for efficient real-time edge deployment.

Ultimately, this study underscores the importance of leveraging modern deep learning techniques like YOLOv8 and YOLOv11 for building reliable and efficient vehicle monitoring systems in the era of intelligent transportation.

## **8. References**

1. Ultralytics YOLOv8 Documentation. [https://docs.ultralytics.com](https://docs.ultralytics.com/)
2. YOLOv11 GitHub Repository. <https://github.com/ultralytics/yolov11>
3. FGVD Dataset. [FGVD source or DOI]
4. Bochkovskiy, A. et al. (2020). YOLOv4: Optimal Speed and Accuracy. arXiv:2004.10934
5. Redmon, J., Farhadi, A. (2018). YOLOv3. arXiv:1804.02767
6. Lin, T.-Y. et al. (2014). Microsoft COCO: Common Objects in Context. ECCV