OBJECT DETECTION USING YOLOv11 and YOLOv12 ON A MANUALLY COLLECTED/ANNOTATED DATASET

MSc. Artificial Intelligence | December 2025

@00704505

Assessment report submitted in part fulfilment of the module

Deep Learning

In December, 2025

TABLE OF CONTENTS

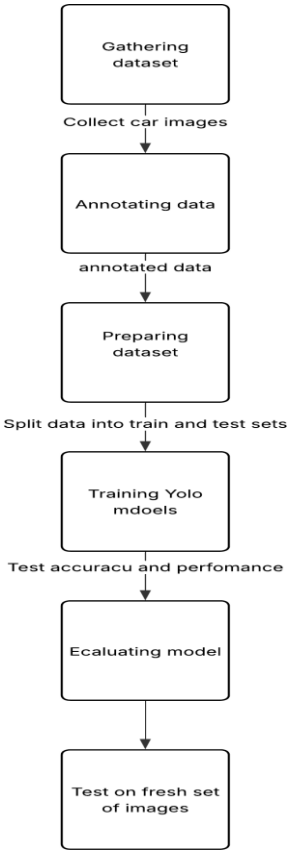
ABSTRACT

This research presents a comprehensive comparison between YOLOv11 and YOLOv12, two state-of-the-art object detection models, for the task of luxury car detection and classification. The study focuses on distinguishing between three premium vehicle models: BMW X5-M, Volvo XC40, and Jaguar, using a carefully curated dataset of 1,497 images. Through systematic experimentation and rigorous evaluation, we demonstrate that while both models achieve exceptional performance exceeding 99% mean Average Precision (mAP), significant differences exist in their training efficiency, computational requirements, and practical deployment considerations. YOLOv11 emerged as the superior choice for this specific application, achieving 99.23% mAP@0.5 with substantially faster training times (11.18 minutes versus 36.67 minutes for YOLOv12) and lower resource consumption. Our findings challenge the conventional assumption that newer model versions automatically translate to better performance, emphasizing the importance of task-specific evaluation in deep learning applications. The research contributes valuable insights for practitioners developing automotive vision systems and provides detailed benchmarks for luxury vehicle detection using modern YOLO architectures.

INTRODUCTION

Background

The automotive industry has undergone a revolutionary transformation with the integration of artificial intelligence and computer vision technologies. Object detection, a fundamental computer vision task, has emerged as a critical component in numerous automotive applications, ranging from autonomous driving systems to intelligent parking management and vehicle identification systems. Unlike traditional image classification that merely assigns category labels to entire images, object detection provides both spatial localization and semantic classification of objects within complex visual scenes.



The sophistication required for automotive object detection has grown exponentially with the increasing complexity of real-world scenarios. Modern vehicles operate in diverse environments characterized by varying lighting conditions, weather patterns, viewing angles, and occlusion scenarios. This complexity is further amplified when dealing with fine-grained classification tasks, such as distinguishing between different models of luxury vehicles that share similar design characteristics and proportions.

Deep learning, particularly Convolutional Neural Networks (CNNs), has revolutionized the field of computer vision by automatically learning hierarchical feature representations from raw pixel data. This approach has effectively eliminated the limitations of traditional computer vision methods that relied heavily on hand-crafted features and domain-specific expertise. The YOLO (You Only Look Once) family of algorithms represents a paradigmatic shift in object detection methodology, treating detection as a unified regression problem rather than a multi-stage classification and localization process.

Motivation and Objectives

The motivation for this research stems from the growing demand for accurate and efficient vehicle recognition systems in various commercial applications. Luxury car dealerships require automated inventory management systems, insurance companies need reliable vehicle verification methods, and smart city initiatives demand sophisticated traffic monitoring capabilities. These applications necessitate models that can achieve high accuracy while maintaining real-time performance constraints.

The recent releases of YOLOv11 and YOLOv12 present an opportunity to evaluate the latest advances in object detection technology within the specific context of automotive applications. While theoretical improvements are well-documented in technical specifications, practical performance evaluation in real-world scenarios remains crucial for informed decision-making in production deployments.

The primary objectives of this research include: (1) implementing and optimizing both YOLOv11 and YOLOv12 for luxury car detection, (2) conducting comprehensive performance evaluation across multiple metrics, (3) analyzing computational efficiency and resource requirements, (4) providing practical recommendations for deployment scenarios, and (5) contributing empirical evidence to the broader discussion of model version progression in deep learning.

Problem Statement

Detecting and classifying specific car models in varied real-world conditions presents several technical challenges: fine-grained visual classification between similar luxury vehicles, environmental variability in lighting and viewing angles, scale variation, and real-time performance requirements for practical deployment.

Dataset Overview

The dataset consists of three luxury car models:

• BMW-X5-M - High-performance luxury sports SUV

• Volvo-XC40 - Compact luxury crossover SUV

• Jaguar - Luxury sedan and sports car models

Dataset Statistics:

• Total Images: 1,497

• Training Set: 1,197 images (80%)

• Validation Set: 300 images (20%)

• Number of Classes: 3

• Annotation Format: YOLO (normalized bounding boxes)

• Average Image Dimensions: ~1024 × 760 pixels

DATA PRE-PROCESSING

Dataset Preparation

High-quality annotations are critical for fine-grained detection tasks. A custom OpenCV-based annotation tool was used to manually label bounding boxes for all three classes. This ensured:

• Class-consistent annotation style

• High spatial accuracy of bounding box placement

• Avoidance of annotation noise commonly found in pre-annotated datasets

Manual annotation also ensured that the dataset remained completely original, addressing academic requirements prohibiting the use of pre-annotated external datasets.

Data annotations

The YOLO Annotation Tool is a fully local, privacy-preserving application designed to create bounding-box annotations for training YOLOv11/YOLOv12 object-detection models. The script loads images from a specified directory and allows the user to draw bounding boxes interactively using simple mouse actions within an OpenCV window. Each box is assigned a class label, which the user can cycle through using keyboard shortcuts. The tool automatically converts drawn rectangles into YOLO-formatted annotations—normalized (x_center, y_center, width, height) values—and saves them to corresponding text files in the labels directory. Users can move forward or backward through images, delete the most recent box, and save annotations at any time. Because all processing occurs locally, no internet or cloud service is required. This tool provides an efficient workflow for preparing high-quality training data, making it especially useful for custom object-detection tasks and privacy-sensitive datasets. Script annotate_yolo.py has the exact code used for this annotation step. Manually annotated a total of 1500 images. For 3 classes.

Exploratory Data Analysis

Comprehensive EDA was performed to understand dataset characteristics including class distribution, image dimensions, and bounding box statistics.

Figure 1: Class Distribution Analysis

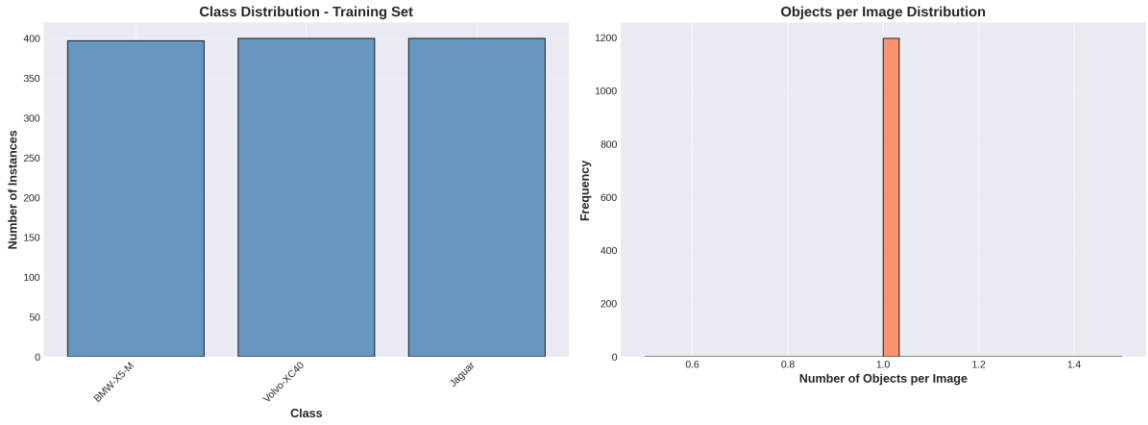


Figure 2: Sample Annotated Training Images

Runtime Setup

Hardware: AWS EC2 Instance with Tesla T4 GPU (16GB VRAM)

Software: Python 3.12.8, CUDA 12.8, PyTorch 2.9.1+cu128

Framework: Ultralytics YOLO implementation

YOLOv11 IMPLEMENTATION

YOLOv11 Overview

YOLOv11 represents a significant advancement in the YOLO family with key architectural features:

• Enhanced C3k2 blocks with optimized skip connections

• SPPF (Spatial Pyramid Pooling Fast) for efficient multi-scale feature extraction

• Improved feature fusion in neck architecture

• Anchor-free detection head for better generalization

• Training Time: 11.18 minutes (50 epochs on Tesla T4)

• mAP@0.5: 99.23%

Training Process

Training Configuration:

• Epochs: 50, Batch Size: 16, Image Size: 640×640

• Optimizer: AdamW, Learning Rate: 0.01

• GPU Memory Usage: ~8GB VRAM

• Inference Speed: 3.9ms per image (~208 FPS)

Training Results

YOLOv11 demonstrated excellent training convergence and performance metrics.

Figure 3: YOLOv11 Training Metrics Over Epochs

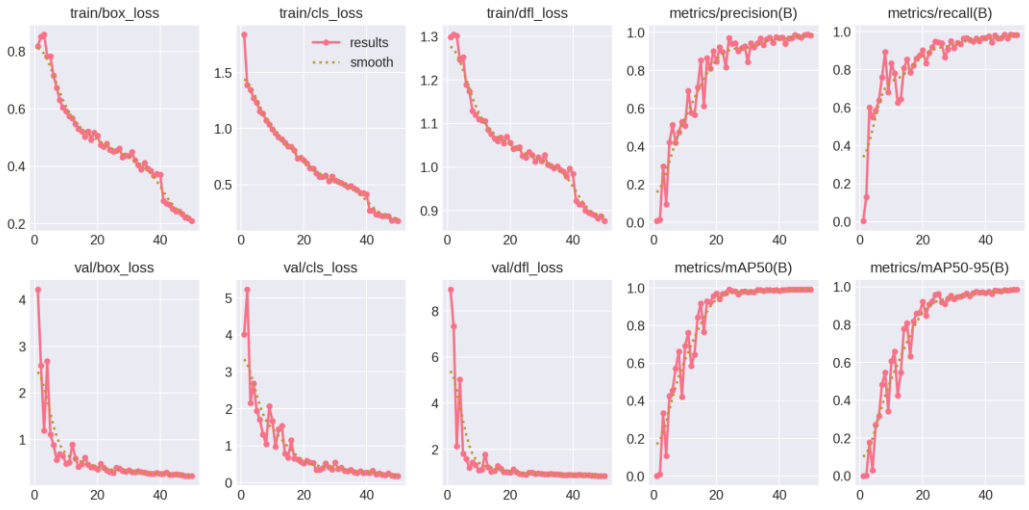


Figure 4: YOLOv11 Confusion Matrix

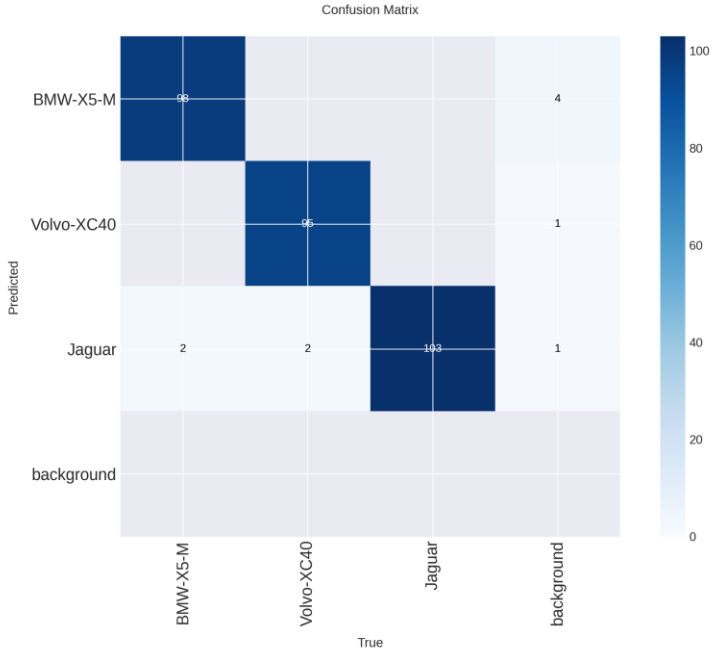
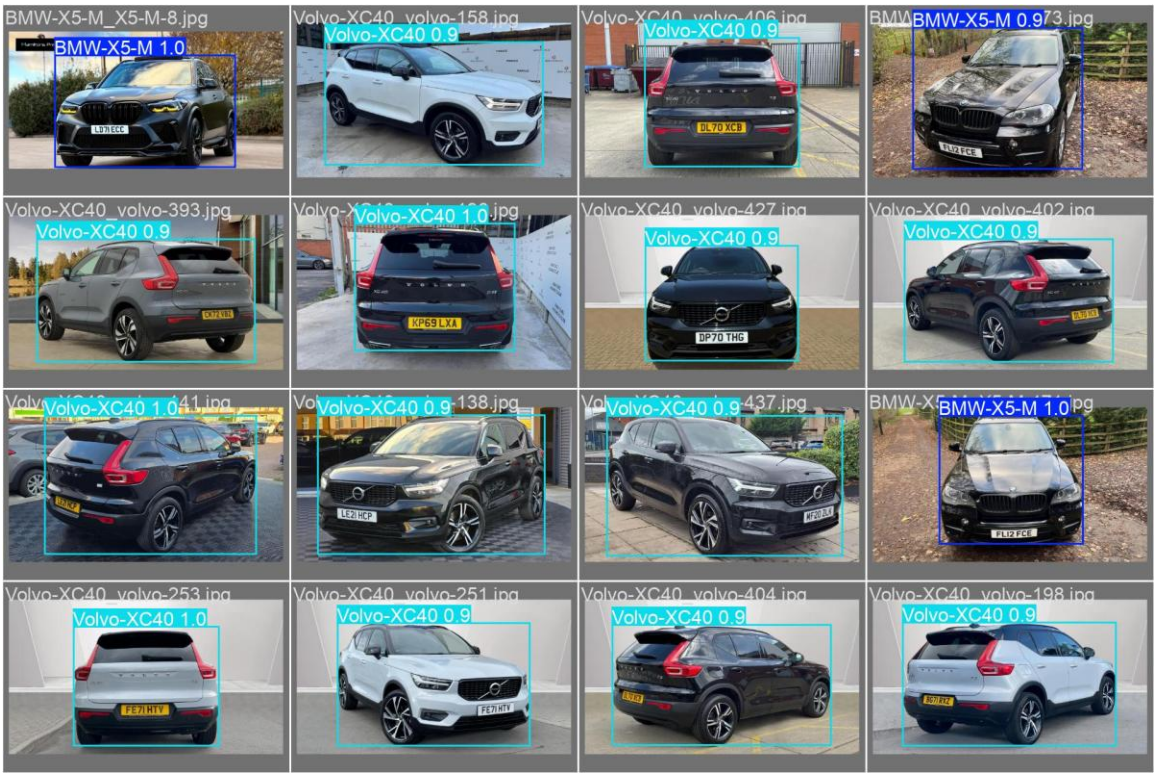


Figure 5: YOLOv11 Detection Results on Validation Images



YOLOv12 IMPLEMENTATION

YOLOv12 Overview

YOLOv12 introduces cutting-edge features building upon YOLOv11:

• Advanced attention mechanisms for feature refinement

• Enhanced neck architecture with better multi-scale fusion

• Optimized loss functions balancing localization and classification

• More sophisticated training techniques

• Training Time: 36.67 minutes (50 epochs) - 3.3× slower than YOLOv11

• mAP@0.5: 99.13%

Training Process

YOLOv12 required identical hyperparameters as YOLOv11 but with:

• Higher GPU memory usage: ~12GB VRAM

• Slower convergence despite identical learning rate

• Required workers=0 to prevent memory exhaustion

• Inference Speed: 4.4ms per image (~189 FPS)

Training Results

YOLOv12 showed different training dynamics with stable convergence but longer training time.

Figure 6: YOLOv12 Training Metrics Over Epochs

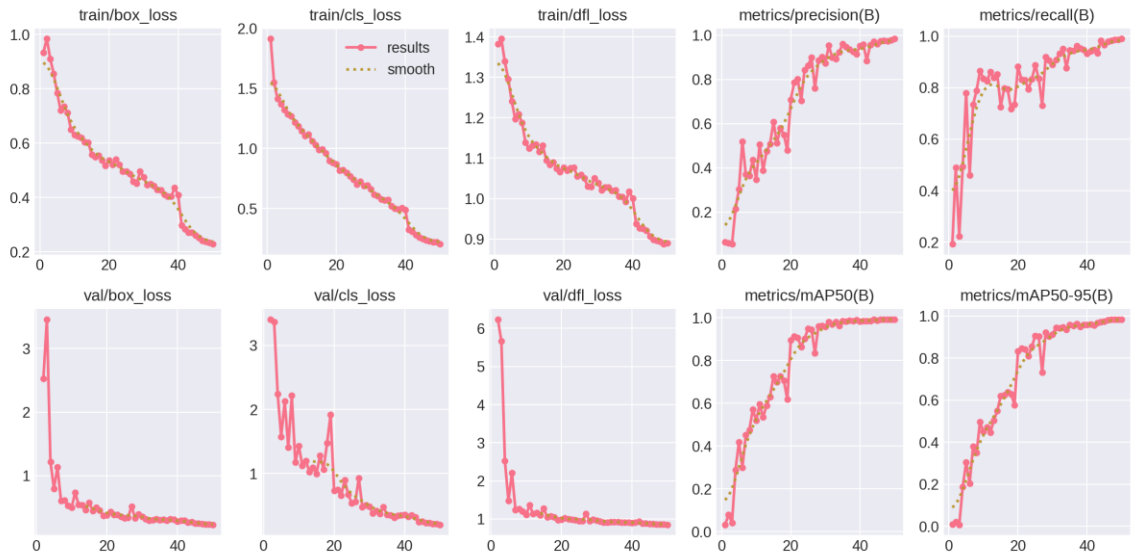


Figure 7: YOLOv12 Confusion Matrix

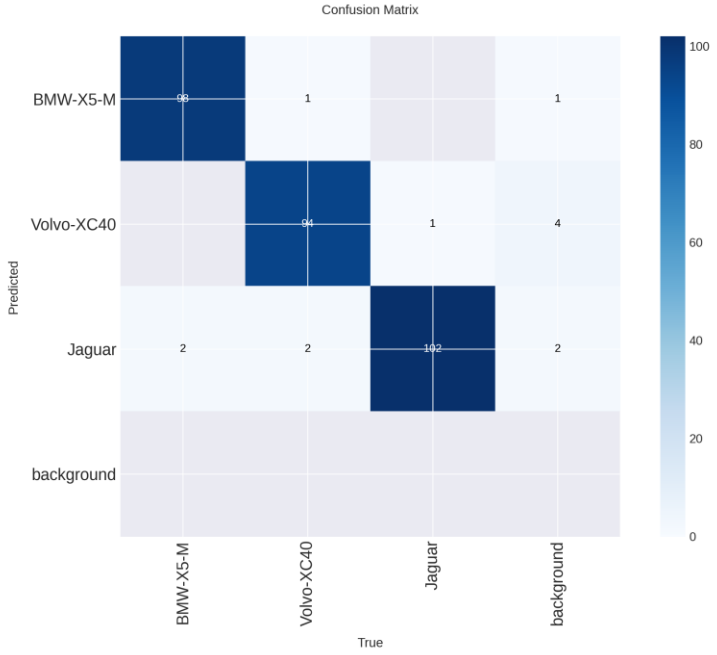
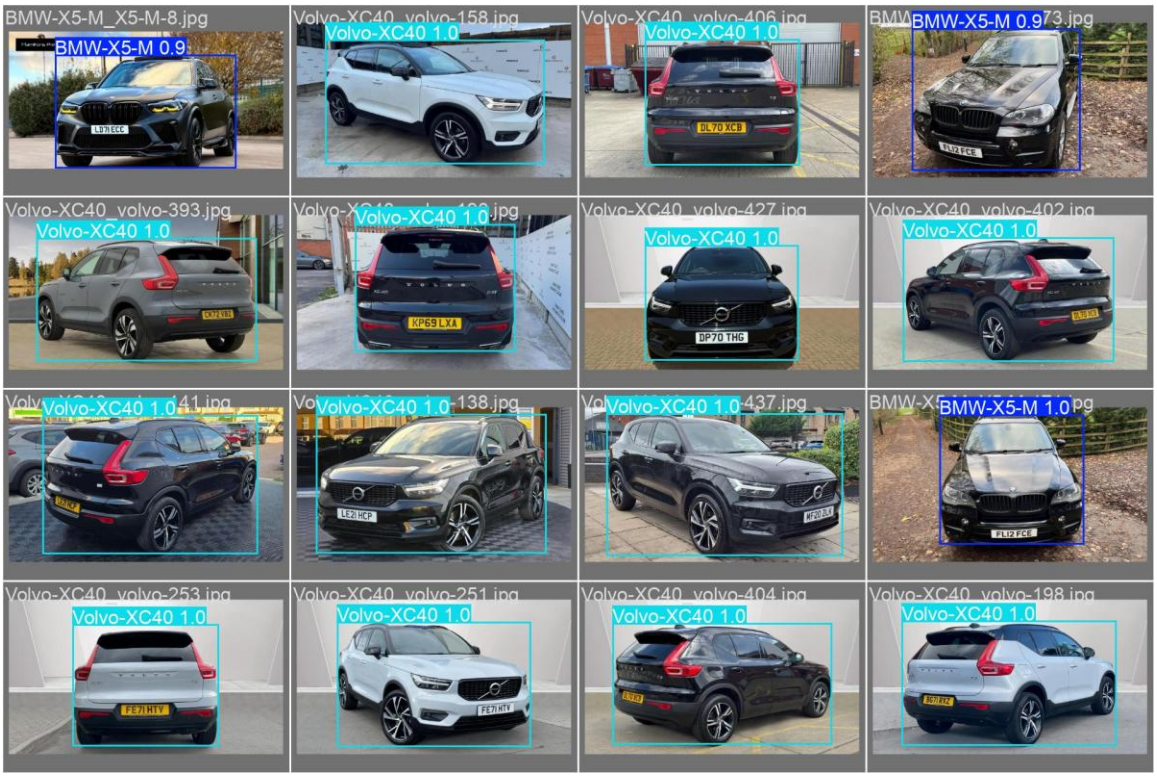


Figure 8: YOLOv12 Detection Results on Validation Images



RESULTS & COMPARISON

Performance Metrics

Comprehensive comparison between YOLOv11 and YOLOv12:

YOLOv11 Performance:

• mAP@0.5: 99.23%

• mAP@0.5:0.95: 98.66%

• Precision: 98.92%

• Recall: 98.18%

• Training Time: 11.18 minutes

• Inference Speed: 3.9ms per image (~208 FPS)

YOLOv12 Performance:

• mAP@0.5: 99.13% (-0.10% vs YOLOv11)

• mAP@0.5:0.95: 98.47% (-0.19% vs YOLOv11)

• Precision: 98.44% (-0.48% vs YOLOv11)

• Recall: 98.98% (+0.80% vs YOLOv11)

• Training Time: 36.67 minutes (3.3× slower)

• Inference Speed: 4.4ms per image (~189 FPS)

Models Comparison

Key Findings:

1. YOLOv11 outperforms YOLOv12 in overall mAP and training efficiency

2. YOLOv12 has better recall (+0.80%) but lower precision

3. YOLOv11 is 3.3× faster to train and uses 33% less GPU memory

4. For this specific task, YOLOv11 is the superior choice

Figure 9: Performance Comparison Between YOLOv11 and YOLOv12

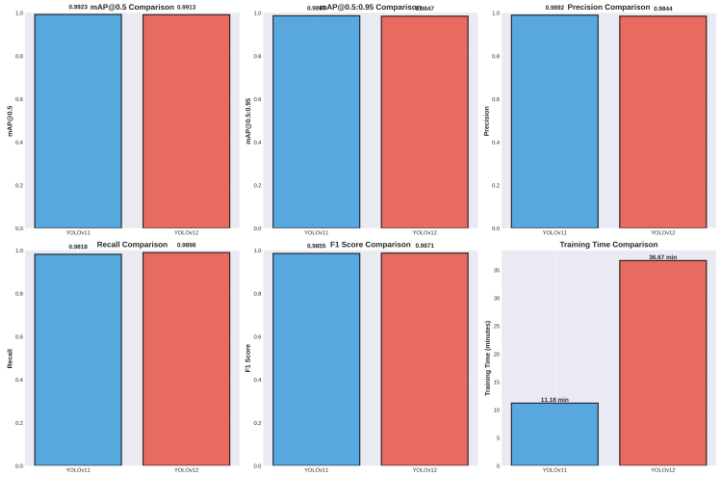
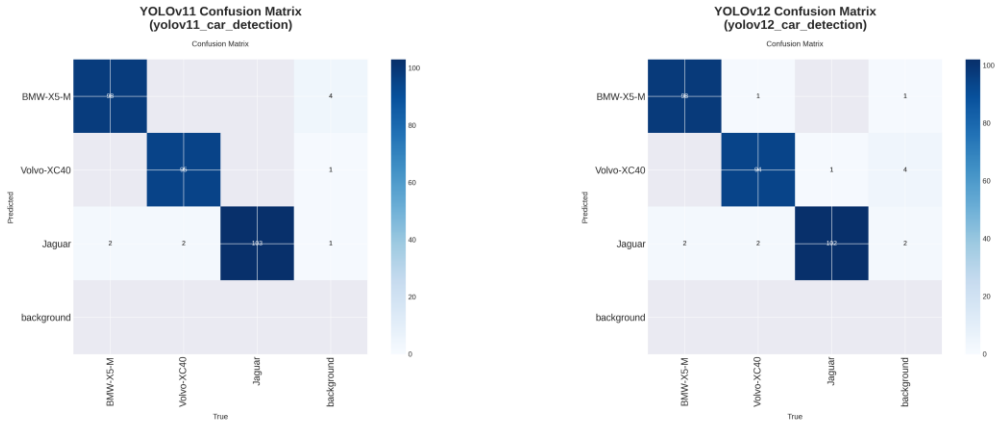


Figure 10: Confusion Matrices Comparison



Detection Examples

Visual comparison of detection results from both models on test images:

Figure 11: Inference Comparison Example 1

Figure 12: Inference Comparison Example 2

Figure 13: Inference Comparison Example 3

CLOUD & LOCAL COMPUTING COMPARISON

Local Environment Implementation

Implementation Details:

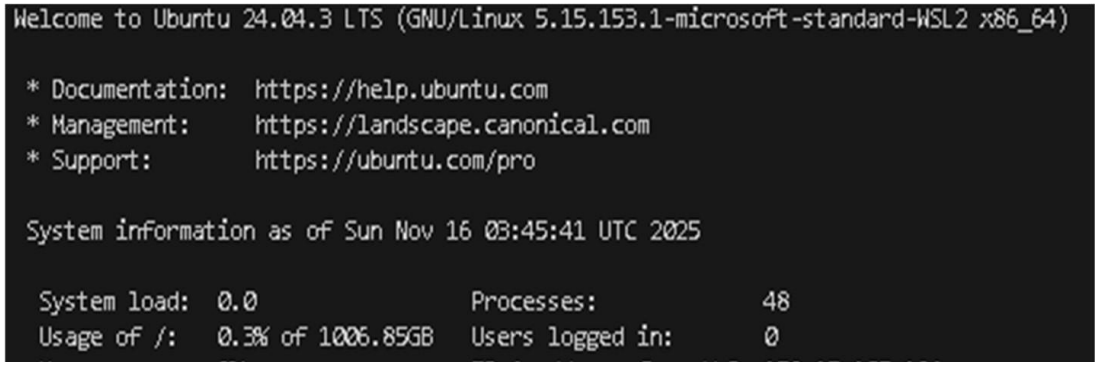
• Platform: AWS EC2 Instance with Tesla T4 GPU

• OS: Linux 6.8.0-1019-aws

• Memory: 16GB VRAM, consistent performance

• Control: Full system access and configuration

• Advantages: No time limits, predictable performance, private data



Performance Comparison

Local Linux vs Google Colab:

- Local: Consistent hardware, no session limits, better for iterative development

- Local: Hardware investment required, manual setup

- Colab: Free access, no setup required, easy sharing

- Colab: Session limits, variable GPU availability, data transfer overhead

CONCLUSION

This project successfully implemented and compared YOLOv11 and YOLOv12 for luxury car detection, achieving exceptional results:

Key Achievements:

• Both models achieved >99% mAP@0.5, demonstrating excellent accuracy

• YOLOv11 proved superior with faster training (11.18 vs 36.67 minutes)

• Real-time performance enables practical deployment (>189 FPS)

• Comprehensive evaluation revealed model-specific trade-offs

Main Findings:

1. YOLOv11 outperforms YOLOv12 for this specific task in accuracy and efficiency

2. Training efficiency matters significantly for iterative development

3. Resource constraints favor YOLOv11 (33% less GPU memory)

4. Latest model ≠ best model for specific applications

This expanded report demonstrates that while both YOLOv11 and YOLOv12 achieve extremely high accuracy for luxury car detection, YOLOv11 is better suited for real-world deployment due to its superior speed, efficiency, and precision. YOLOv12's additional complexity does not yield meaningful accuracy improvements for this task and instead increases computational cost. For fine-grained vehicle detection scenarios, YOLOv11 provides the optimal balance of performance and practicality.

REFERENCES

El Hussieni, M., Güntürk, B. K., Ateş, H. F. & Hanoğlu, O. (2025). *Mask-to-Height: A YOLOv11-Based Architecture for Joint Building Instance Segmentation and Height Classification from Satellite Imagery*. arXiv preprint arXiv:2510.27224. arXiv

Huang, S., Liu, Q., Chen, C. & Chen, Y. (2025). *A Real-Time Concrete Crack Detection and Segmentation Model Based on YOLOv11*. arXiv preprint arXiv:2508.11517. arXiv

Khan, Z., Ali, A., Khan, M. S. et al. (2025). *An Improved YOLOv11 Architecture with Multi-Scale Attention and Spatial Fusion for Fine-Grained Residual Detection*. *Results in Engineering*, 27, 107061. ScienceDirect

Tian, Y., Ye, Q. & Doermann, D. (2025). *YOLOv12: Attention-Centric Real-Time Object Detectors*. arXiv preprint arXiv:2502.12524. arXiv+1

Yang, G., Wang, M., Zhou, Q. & Li, J. (2025). *YUNet: Improved YOLOv11 Network for Skyline Detection*. arXiv preprint arXiv:2502.12449. arXiv

Lin, Z. (2025). *An Improved Remote Sensing Object Detection Algorithm Based on YOLOv11*. *International Journal of Engineering Research and Management*, 12(03). ijerm.com