الجمهورية الجزائرية الديمقراطية الشعبية

MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH

UNIVERSITY YAHIA FARES - MEDEA

FACULTY OF SCIENCES

Department of Mathematics and Computer Science



End of Studies Project

Master's degree in Computer Science

Option : System Engineering and Web Technologies

THEME

Motorcycle Helmet Detection using Deep Learning

Proposed and directed by:

Mr. BABAALI Baligh

Presented by :

Mr. BENILLES Hatem

Mr. DAHMANI Abdeljalil

Academic Year 2024-2025

Dedication

This thesis is dedicated to: My beloved grandfather, for your unwavering support, invaluable wisdom, and the countless hours you spent mentoring me. Your passion for learning has been an inspiration throughout my academic journey. My dear grandmother, for your unconditional love, endless encouragement, and the warmth you have brought to my life. Your resilience and kindness have been a constant source of strength. My wonderful father, for your guidance, tough love, and belief in my abilities. Your dedication to our family has shaped the person I am today. My caring mother, for your unconditional support, unwavering patience, and the sacrifices you've made to provide me with every opportunity. Your compassion and nurturing spirit have been instrumental in my success. My supportive brother, for your camaraderie, healthy competition, and the many memories we've shared. Your presence in my life has enriched it in ways I cannot fully express. And to my dear friends, who have become an extended family. Thank you for your friendship, laughter, and for being there for me through the highs and lows. Your presence in my life has made this journey all the more meaningful. I am truly grateful to have you all in my life, and I hope that this work reflects the positive impact you have had on me.

BENILLES Hatem

Dedication

This thesis is dedicated to: My beloved grandfather, for your unwavering support, invaluable wisdom, and the countless hours you spent mentoring me. Your passion for learning has been an inspiration throughout my academic journey. My dear grandmother, for your unconditional love, endless encouragement, and the warmth you have brought to my life. Your resilience and kindness have been a constant source of strength. My wonderful father, for your guidance, tough love, and belief in my abilities. Your dedication to our family has shaped the person I am today. My caring mother, for your unconditional support, unwavering patience, and the sacrifices you've made to provide me with every opportunity. Your compassion and nurturing spirit have been instrumental in my success. My supportive brother, for your camaraderie, healthy competition, and the many memories we've shared. Your presence in my life has enriched it in ways I cannot fully express. And to my dear friends, who have become an extended family. Thank you for your friendship, laughter, and for being there for me through the highs and lows. Your presence in my life has made this journey all the more meaningful. I am truly grateful to have you all in my life, and I hope that this work reflects the positive impact you have had on me.

.

DAHMANI Abdeljalil

Acknowledgment

First and foremost, we would like to thank Almighty Allah for granting us the courage and patience to successfully complete this work. We would like to express our sincere gratitude to our advisors, Dr. Baligh for his invaluable guidance and support throughout our master's program. his expertise, encouragement, and critical feedback have been extremely valuable in structuring the work and enhancing the quality of the various sections. We would like to extend our sincere appreciation to the members of the jury Dr. and DR. for the honor they bestow upon us by taking the time to read and evaluate this work. Our dear parents are our pride and the source of our success, as they have made sacrifices to provide us with a conducive working environment and the best possible conditions. Without them, nothing would have been easy. May God protect and bless them so that we can repay them for all that they have given us. We would also like to thank our friends and family for their love and support during this process. Without their encouragement and motivation, we would not have been able to complete this journey.

ملخص :

تمثل حوادث الدراجات النارية أكثر من 23٪ من الوفيات الناجمة عن حوادث الطرق عالمياً، حيث تكون إصابات الرأس السبب الرئيسي. تشير الأبحاث إلى أن الخوذات يمكن أن تقلل من خطر الإصابات القاتلة بنسبة 42-69٪، ومع ذلك يظل الالتزام غير متسق بسبب الاعتماد على المراقبة اليدوية. تقترح هذه الدراسة حلاً آلياً باستخدام رؤية الحاسوب ونظام YOLOv8، حيث تم تدريبه على 400 صورة موضحة (تشمل: راكبي الدراجات مع خوذات، راكبي الدراجات بدون خوذات، والركاب). حقق النموذج دقة بمقدار mAP 0.677 ودرجة F1 تبلغ 0.67، مما يثبت جدواه في المراقبة المرورية باستخدام كاميرات المراقبة. بينما يمكن لمزيد من التحسين أن يعزز المتانة، فإن هذا النهج يتيح المراقبة على مدار الساعة دون تكاليف بنية تحتية إضافية.

كلمات مفتاحية: رؤية الحاسوب , كاميرات المراقبة , حوادث الطرق , YOLOv8 .

Abstract:

Motorcycle accidents account for over 23% of global road fatalities, with head injuries as the leading cause. Research indicates helmets can reduce fatal injury risk by 42–69%, yet compliance remains inconsistent due to reliance on manual monitoring. This work proposes an automated computer vision solution using YOLOv8, trained on 400 annotated images (classes: helmeted riders, non-helmeted riders, occupants). The model achieves 0.677 mAP and 0.67 F1-score, demonstrating viability for CCTV-based traffic monitoring. While further optimization could improve robustness, this approach enables 24/7 enforcement without infrastructure costs.

Keywords: Road fatalities, YOLOv8, computer vision ,CCTV.

Résumé :

Les accidents de moto représentent plus de 23 % des décès routiers dans le monde, les traumatismes crâniens en étant la principale cause. Les recherches indiquent que les casques peuvent réduire le risque de blessures mortelles de 42 à 69 %, mais le respect reste inégal en raison de la surveillance manuelle. Ce travail propose une solution automatisée de vision par ordinateur utilisant YOLOv8, entraînée sur 400 images annotées (classes : conducteurs casqués, conducteurs non casqués, passagers). Le modèle atteint une précision moyenne (mAP) de 0,677 et un score F1 de 0,67, démontrant sa viabilité pour la surveillance routière par caméras. Bien qu'une optimisation supplémentaire puisse améliorer sa robustesse, cette approche permet une surveillance 24h/24 sans coûts d'infrastructure supplémentaires.

Mots clés : décès routiers , YOLOv8, la surveillance routière par caméras, vision par ordinateur.

Table of Contents

Table of Contents **8**
General Introduction **4**
Chapter 1. LITERATURE REVIEW **6**
1.1. Introduction 7
1.2. Road Safety and Regulations 7
1.2.1. Statistics on motorcycle accidents 8
1.2.2. Existing regulations on helmet usage 8
1.3. Artificial Intelligence and Computer Vision 8
1.3.1. Definition of AI and computer vision 8
1.3.2. Applications in road safety 9
1.4. Existing Approaches for Helmet Detection 9
1.4.1. YOLOv8-Powered Helmet Detection (NUST, Pakistan) 9
1.4.2. Few-Shot YOLOv8 Helmet Violation (AI City Challenge) 9
1.4.3 Improved YOLOv8 (“Helmet Net”) 10
1.4.4 Improvement of Helmet Detection Algorithm Based on YOLOv8s 10
1.5. Traditional detection techniques 11
1.6. Introduction to neural networks and deep learning 12
1.7. Deep Learning Models for Object Detection 13
1.7.1. Convolutional Neural Networks (CNNs) 13
1.7.2. Faster R-CNN 14
1.7.3. YOLO (You Only Look Once) – YOLOv5, YOLOv8 14
1.7.4. Advantages and limitations of existing approaches 19
1.8. Evaluation Metrics 21
1.8.1 Mean Average Precision (mAP@0.5): 21
1.8.2 Precision (P) 21
1.8.3 Recall (R) 21
1.8.4 F1-Score 21
1.9. Conclusion 22
Chapter 2. Proposed Model And Methodology **23**
2.1. Introduction 24
2.2. Overview of the Proposed Solution 24
2.3. Model Selection 24
2.4. Dataset and Preprocessing 25

2.4.1 Dataset Description 25
2.4.2 Annotation Process 26
2.4.3 Preprocessing Pipeline 26
2.5. Model Training 28
2.5.1. Training parameters 28
2.6. YOLOv8 Model Variants 35
2.7. Implementation and Tools Used 37
2.7.1 Framework 37
2.7.2 Training Interface 38
2.7.3 Bounding Box 38
2.8. Conclusion 38
Chapter 3. EXPERIMENTS AND RESULTS 39
3.1. Introduction 40
3.2. Results Analysis 40
3.2.1. Evaluation metrics (accuracy, recall,…) 41
3.2.2 Comparison of YOLOv8 Model Variants (Small, Nano, Medium) 48
3.2.3 Comparison with Yolov3 53
3.3. Discussion 55
3.3.1 Initial Experiment with Default Preprocessing 55
3.3.2 Dataset Expansion and Minor Improvement 55
3.3.3 Custom Preprocessing Attempts 55
3.3.4 Dataset Merging and Model Scaling 55
3.3.5 Image Size Optimization 56
3.3.6 Key Takeaways 56
3.4. Conclusion 56
GENERAL CONCLUSION 58
4.1 Contributions of the Study 59
4.2 Future Improvements 59
4.3 Summary of Results 60
4.4 Final Remarks 60
4.5 Conclusion 61
References 62
Appendices 68
4.6 Application Snapshots 68
4.6.1 Yolov8 User Interface 68

List of Figures

List Of Images

Figure 1.1 YOLOv8 architecture 14

Figure 2.1. Bounding box 26

Figure 3.1. Precision-confidence curve 41

Figure 3.2. Recall-confidence curve 42

Figure 3.3. F1-Score Confidence Curve 44

Figure 3.4. Precision-Recall Curve 45

Figure 3.5. Normalized Confusion Matrix 47

Figure 3.6. Class-wise mAP@0.5 48

Figure 3.7. Precision vs. Recall Trade-off 49

Figure 3.8. Yolov8 Models Size 50

Figure 3.9. YOLOv8 models map Scores 51

Figure 3.9. Yolov3 example detection 53

Figure 3.10. Yolov8 example detection 53

Figure 4.1. User Interface for Image Upload (Initial State) 68

Figure 4.2. Image Selection Interface 69

Figure 4.3. Detection Results Output 69

Figure 4.4. frontend code example 70

Figure 4.5. frontend code example 2 70

Figure 4.6. backend code example 71

Figure 4.7. backend code example 2 71

List of Tables

List Of Tables

- ***Table 1.1.*** *Difference Table: CSP vs. C2f vs. SPPF* *18*
- ***Table 2.1.*** *Mosaic Augmentation Parameters* *32*
- ***Table 2.2.*** *Flip Augmentation Parameters* *34*
- ***Table 2.3.*** *Training Parameter Summary* *35*
- ***Table 2.4.*** *Implementation and Tools Used* *37*
- ***Table 3.1.*** *Class-wise mAP@0.5 for multi classes* *49*
- ***Table 3.2.*** *Precision vs Recall for Yolov8 variants* *42*
- ***Table 3.3****. Map@0.5 for Yolov8 variants* *51*
- ***Table 3.4.*** *Evaluation metrics summary of Yolov8 variants* *52*

List of Acronyms

Adam: Adaptive Moment Estimation

AdamW: Adaptive Moment Estimation with Weight Decay

CCTV: Closed-Circuit Television

CIoU: Complete Intersection over Union

CNN: Convolutional Neural Network

DL: Deep Learning

Faster R-CNN: Faster Region-Based Convolutional Neural Network

HSV: Hue, Saturation, Value

IoU: Intersection over Union

ITS: Intelligent Transportation Systems

lr0: Initial Learning Rate

mAP@0.5: mean Average Precision at IoU=0.5

NHTSA: National Highway Traffic Safety Administration

NMS: Non-Maximum Suppression

SGD: Stochastic Gradient Descent

WHO: World Health Organization

YOLO: You Only Look Once

General Introduction

Existing automated helmet detection systems primarily rely on older deep learning models like CNNs, Faster R-CNN, and early YOLO versions (YOLOv3-YOLOv5), which suffer from critical limitations: slow inference speeds (notably Faster R-CNN's two-stage detection), reduced accuracy in complex scenarios (e.g., occluded or small helmets), poor generalization across lighting/weather conditions, high computational costs, and limited deployment scalability (Redmon et al., 2016; Ren et al., 2015; Bochkovskiy et al., 2020)[1][2][3]. These drawbacks impede real-time helmet compliance monitoring, a key factor in reducing motorcycle-related fatalities. To address these gaps, this thesis aims to develop a robust, real-time helmet detection system using YOLOv8, leveraging its superior speed (high FPS), enhanced accuracy, and optimized edge-device deployment (Jocher et al., 2023)[4]. Our objectives include: curating a diverse dataset of motorcyclists under varying conditions, training and optimizing YOLOv8 for high-precision helmet detection, evaluating performance using metrics like mAP and F1-score, and deploying a user-friendly application for real-world traffic surveillance. By overcoming the inefficiencies of traditional methods, this research seeks to provide a scalable solution for improving helmet compliance and road safety.

This document is structured into four chapters covering motorcycle safety and AI-based helmet detection. Chapter 1 reviews motorcycle safety, emphasizing helmet use, statistics, regulations, and traditional versus deep learning detection methods like YOLO, while identifying research gaps. Chapter 2 details the proposed AI-based helmet detection system, including YOLOv8's architecture, data collection, model training, and evaluation metrics such as precision and recall. Chapter 3 presents the experiment setup, results analysis, and discussion, assessing the model's performance and user feedback, followed by a conclusion on the findings. Chapter 4 summarizes key results, their implications for safety and public health, and suggests future research directions. This logical organization ensures clarity in presenting the study's development and contributions to road safety.

CHAPTER 1. LITERATURE REVIEW

1.1. Introduction

Motorcycle riding is a popular yet risky mode of transportation, particularly for young individuals aged 15 to 29, who are disproportionately affected by road traffic accidents[5]. The World Health Organization (WHO) emphasizes that inadequate helmet use significantly contributes to these injuries, highlighting the urgent need for effective strategies to promote helmet compliance and enhance rider safety. .

This chapter provides a comprehensive literature review on motorcycle safety, helmet regulations, and the role of technology in mitigating risks. In Section 1.2, we will examine the current state of road safety and regulations, including statistics on motorcycle accidents and existing helmet laws, to contextualize the need for improved safety measures.

Section 1.3 will introduce artificial intelligence (AI) and computer vision, exploring their relevance in road safety and automated helmet detection systems. Section 1.4 will review existing helmet detection approaches, from traditional techniques to advanced methodologies involving neural networks and deep learning.

Building on this foundation, Section 1.5 will focus on deep learning models for object detection, including Convolutional Neural Networks (CNNs), Faster R-CNN, and the YOLO (You Only Look Once) architecture, particularly YOLOv5 and YOLOv8. This section will also discuss the advantages and limitations of these models in helmet detection.

Finally, Section 1.6 will conclude the chapter by summarizing key findings and identifying gaps that the proposed study aims to address. By synthesizing the existing literature, this chapter establishes a solid foundation for the proposed study, emphasizing the significance of helmet detection technology in promoting rider safety and compliance with regulations, ultimately aiming to reduce motorcycle-related injuries and fatalities

1.2. Road Safety and Regulations

Road safety is a global concern, especially for motorcyclists, who are more vulnerable to severe injuries or fatalities due to the lack of protective structures. Regulations, such as mandatory helmet usage, have been implemented to reduce accidents and save lives.

1.2.1. Statistics on motorcycle accidents

Motorcycle accidents contribute significantly to road traffic injuries and fatalities worldwide. According to the National Highway Traffic Safety Administration (NHTSA, 2020), motorcyclists are 28 times more likely to die in a crash than occupants of passenger vehicles[6]. NHTSA data also indicates that motorcyclists accounted for 14% of all traffic fatalities in the same year. despite representing only 3% of registered vehicles (NHTSA, 2021)[6]. In Southeast Asia, countries like Thailand and Vietnam have some of the highest motorcycle-related fatality rates globally, often due to speeding, lack of protective gear, and poor road conditions.

1.2.2. Existing regulations on helmet usage

Helmet usage is a proven measure to reduce head injuries and fatalities in motorcycle accidents. Many countries enforce mandatory helmet laws for riders and passengers. For example, in the European Union, helmet use is compulsory for all motorcyclists (European Commission, 2023)[7]. In the U.S., helmet laws vary by state, with some requiring helmets for all riders and others only for minors (NHTSA, 2021). In countries like India and Indonesia, helmet laws have been strengthened, but enforcement remains inconsistent. Studies show that helmets reduce the risk of death by 42% and head injuries by 69% (WHO, 2020)[5]. However, non-compliance due to discomfort, lack of awareness, or weak enforcement limits their effectiveness.

1.3. Artificial Intelligence and Computer Vision

Artificial Intelligence (AI) and Computer Vision are transformative technologies that have revolutionized various industries, including road safety. By leveraging AI and computer vision, systems can analyze visual data, detect patterns, and make real-time decisions to enhance safety, reduce accidents, and improve traffic management.

1.3.1. Definition of AI and computer vision

Artificial Intelligence (AI) refers to the simulation of human intelligence in machines that are programmed to perform tasks such as learning, reasoning, and problem-solving. Computer Vision, a subset of AI, focuses on enabling machines to interpret and understand visual information from the world, such as images and videos. It involves techniques like image recognition, object detection, and video analysis to extract

meaningful insights from visual data. Together, AI and computer vision form the backbone of many advanced safety and automation systems.

1.3.2. Applications in road safety

AI and computer vision have numerous applications in road safety, including accident prevention, traffic management, and driver assistance. For example, AI-powered dash cams and surveillance systems can detect dangerous driving behaviors like speeding, lane drifting, or distracted driving in real-time (Grigorescu et al., 2020)[8]. Computer vision algorithms are also used in autonomous vehicles to identify pedestrians, cyclists, and obstacles, enabling safer navigation. Additionally, smart traffic lights equipped with AI can optimize traffic flow and reduce congestion, minimizing the risk of accidents. These technologies are increasingly being integrated into Intelligent Transportation Systems (ITS) to create safer and more efficient road networks.

1.4. Existing Approaches for Helmet Detection

1.4.1. YOLOv8-Powered Helmet Detection (NUST, Pakistan)

- **Model**: YOLOv8n
- **Source**: ResearchGate[9]
- **Dataset / Setup**: ~10,000 images from industrial and construction settings using the Safety Helmet Wearing Dataset (converted from Pascal VOC to YOLO format).
- **Key Enhancements**: Standard YOLOv8n setup for binary classification ("helmet" vs. "no helmet").
  Results: mAP@0.5 = **0.647**
- **Remarks**: Designed for real-time PPE compliance with a good balance of speed and accuracy.

1.4.2. Few-Shot YOLOv8 Helmet Violation (AI City Challenge)

- **Model**: YOLOv8
- **Source**: arXiv, ResearchGate[10]

- **Dataset / Setup**: Traffic video frames using few-shot data sampling with limited annotated frames.
- **Key Enhancements**: Multi-class helmet violation detection using few-shot learning; submitted to AI City Challenge 2023 (Track 5).
- **Results**: mAP@0.5 = **0.586**
- **Remarks**: Ranked 7th in the challenge. Performs reasonably well under sparse data conditions, but with lower accuracy.

1.4.3 Improved YOLOv8 ("Helmet Net")

- **Model**: YOLOv8n variant
- **Source**: ResearchGate, TechScience[11]
- **Dataset / Setup**: SHWD dataset with focus on small-object detection and speed.
- **Key Enhancements**: Integration of SENet in the neck, lightweight convolutional modules (LConv), WIoU loss function, and SAHI slicing.
- **Results**: Estimated mAP@0.5 = **~0.71** (+6.4% over baseline YOLOv8n)
- **Remarks**: High-speed inference (~111 FPS). Especially effective for detecting small and partially occluded helmets.

1.4.4 Improvement of Helmet Detection Algorithm Based on YOLOv8s

- **Model**: YOLOv8s
- **Source**: Sciendo (2024)[12]
- **Dataset / Setup**: Industrial factory scenes with complex and cluttered backgrounds.
- **Key Enhancements**: Use of deformable convolutions in the backbone, CBAM (attention module), and a custom IoU-based loss function.
- **Results**: Performance improved by **+1.3% over baseline YOLOv8s** (exact mAP@0.5 not reported).

  Remarks: Lightweight, easy-to-implement modifications. Improves detection of small helmets and reduces false positives.

In our study, we evaluate the performance of multiple YOLOv8 variants (n, s, m, l, x) on a custom helmet detection dataset containing 400 images. Despite the limited dataset size, YOLOv8s achieved a mAP@0.5 of 0.677 and a precision of 0.67, outperforming several published methods such as the YOLOv8-based NUST (mAP 0.647) and AI City few-shot approach (mAP 0.586). A comparison with YOLOv3 demonstrated YOLOv8's superior accuracy and detection robustness, especially in detecting partially occluded helmets. Additionally, confusion matrix analysis and class-wise mAP results revealed that YOLOv8 consistently detected "Helmet" with higher precision than "No-Helmet," highlighting its reliability in safety compliance tasks. These results confirm the effectiveness of YOLOv8, even on small datasets, and offer practical benchmarks for real-time helmet detection systems.

1.5. Traditional detection techniques

Traditional helmet detection systems predominantly employ fundamental image processing techniques and classical computer vision algorithms, with three primary approaches being most prevalent: edge detection algorithms, color segmentation techniques, and Haar cascade classifiers.

Edge Detection Algorithms form a crucial component of traditional helmet detection systems. These algorithms, including popular operators like Sobel, Prewitt, and Canny edge detection, work by identifying discontinuities in pixel intensity to map the boundaries of objects. The Canny edge detector (Canny, 1986, IEEE Transactions on Pattern Analysis and Machine Intelligence)[13] is particularly effective for helmet detection due to its multi-stage process involving Gaussian smoothing, gradient calculation, non-maximum suppression, and hysteresis thresholding. This method excels at capturing the distinctive circular or oval contours of helmets, especially when viewed from the side. However, its performance degrades significantly when dealing with textured helmet surfaces or complex backgrounds that introduce noise to the edge map.

Color Segmentation Techniques Methods for detecting safety helmets often leverage the fact that these helmets commonly use high-visibility colors like white, yellow, or red. These approaches typically employ color space transformations, most often converting RGB images to HSV (Hue-Saturation-Value) space to achieve more robust color

separation (Gonzalez & Woods, 2018) [14]. Thresholding operations in the HSV space can isolate specific color ranges associated with helmets. For instance, a study by Wang et al. (2019, Journal of Safety Research)[15] demonstrated that setting hue thresholds between 20-40 degrees (for yellow helmets) and 0-10 degrees (for red helmets) in HSV space achieved 78% detection accuracy in controlled environments. However, these methods struggle with color variations caused by weathering, different lighting conditions, or when helmets are dirty or covered with stickers.

Haar Cascade Classifiers, introduced by Viola and Jones (2001, IEEE CVPR)[16], use a machine learning approach for object detection. They apply Haar-like features to simple rectangular patterns to detect helmets by identifying key edges and textures. AdaBoost optimization selects the most relevant features during training to improve accuracy.

A notable study by Chairat and Wonghabut (2020) [17] proposed a system for automated helmet violation detection. This work aimed for low cost and high performance, and while acknowledging Haar feature-based cascades as classic methods, the paper primarily focused on a client-server architecture to optimize both cost and performance. Their real-world test demonstrated a 97% detection rate for helmet violations with a 15% false alarm rate, highlighting the effective use of computer vision and machine learning to improve helmet compliance.

Traditional computer vision methods for helmet detection face significant practical limitations, including sensitivity to lighting changes (affecting color segmentation), interference from complex backgrounds (impacting edge detection), and frequent false negatives from partial obstructions. These approaches require manual adjustments for different environments and cannot easily adapt to the wide variety of helmet styles. Due to these shortcomings, researchers have increasingly adopted deep learning solutions like YOLO models. These advanced systems automatically learn discriminative features from data, proving more effective at handling real-world challenges such as lighting variations, partial occlusions, and diverse helmet designs, ultimately offering more reliable performance for road safety applications.

1.6. Introduction to neural networks and deep learning

Neural networks are computational models inspired by biological neurons, consisting of interconnected layers that process and transform input data to produce meaningful

outputs (Goodfellow et al., 2016)[18]. These networks learn through backpropagation, where errors are minimized by adjusting internal parameters (weights and biases) during training. When applied to computer vision tasks like helmet detection, neural networks demonstrate remarkable pattern recognition capabilities.

Deep learning refers to neural networks with multiple hidden layers that can learn hierarchical representations of data. This multi-layer architecture enables the automatic extraction of increasingly complex features - from basic edges and textures in early layers to sophisticated object shapes and patterns in deeper layers. In the context of helmet detection, this hierarchical learning proves particularly valuable as it can adapt to various helmet designs, lighting conditions, and viewing angles without explicit programming.

Convolutional layers that apply filters to detect local patterns

Pooling layers that reduce spatial dimensions while retaining important features

Fully-connected layers that perform final classification

1.7. Deep Learning Models for Object Detection

Deep learning models have become the cornerstone of modern object detection systems, offering unparalleled accuracy and efficiency in identifying objects within images and videos. These models are widely used in applications such as helmet detection, autonomous driving, and surveillance. Among the most prominent deep learning models are Convolutional Neural Networks (CNNs), Faster R-CNN, and YOLO (You Only Look Once), each with its unique strengths and limitations.

1.7.1. Convolutional Neural Networks (CNNs)

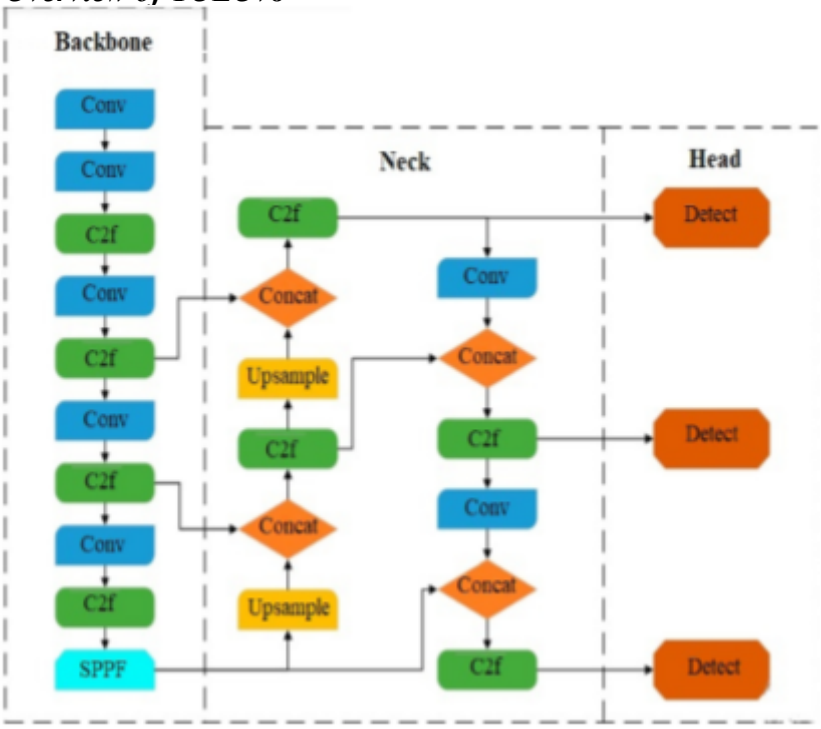
Convolutional Neural Networks (CNNs) are a class of deep learning models specifically designed for processing visual data. They use convolutional layers to automatically extract spatial features from images, such as edges, textures, and shapes, making them highly effective for object detection tasks (LeCun et al., 2015)[19]. CNNs form the foundation of many advanced object detection frameworks, including Faster R-CNN and YOLO, due to their ability to learn hierarchical representations of visual data.

1.7.2. Faster R-CNN

Faster R-CNN (Region-based Convolutional Neural Network) is a widely used two-stage object detection model that integrates a Region Proposal Network (RPN) with a Convolutional Neural Network (CNN) to achieve high accuracy. The RPN efficiently proposes candidate object locations, which are then refined and classified by the subsequent CNN (Ren et al., 2015) [20]. Faster R-CNN is known for its precision and is widely used in applications requiring detailed object detection, such as helmet detection in traffic monitoring systems. However, its computational complexity can make it slower compared to single-stage detectors like YOLO.

1.7.3. YOLO (You Only Look Once) – YOLOv5, YOLOv8

Architectural Overview of YOLOv8



Figuer 1.1 YOLOv8 architecture

To elucidate the technical innovations of YOLOv8, we present a comprehensive architectural diagram (Figure 1.1) that illustrates its key components and data flow. The YOLOv8 architecture builds upon previous YOLO versions while introducing several novel improvements that enhance both detection accuracy and computational efficiency.

1. ***Backbone (Feature Extractor)***
    - Purpose: Extract important features from the input image.
    - Based on: A simplified and modified version of CSPDarknet53
    - Includes:
        - Conv layers for extracting patterns
        - C2f blocks (a lighter alternative to CSP blocks)
        - Ends with SPPF for multi-scale feature enhancement
2. ***Neck (Feature Aggregator)***
    - Purpose: Combine features from different levels of the backbone
    - Uses:
        - C2f blocks
        - Concat operations (merging features from different layers)
        - Upsample operations (to bring back lost resolution)
    - Goal: Blend low-level (fine) and high-level (semantic) features
3. ***Head (Prediction)***
    - Purpose: Make final predictions
    - Uses:
        - Final Conv layers
        - A Detect layer that outputs:
            - Bounding boxes
            - Object classes
            - Confidence scores

Components Explained

SPPF (Spatial Pyramid Pooling – Fast)

- Collects multi-scale features using 3 sequential max pooling layers (same kernel size, typically 5×5).
- Keeps spatial structure but grows a receptive field (context).

- Faster and lighter than the original SPP.
- Helps the model detect small and large objects effectively.

CSP (Cross Stage Partial)

- Splits input into two parts:
    - One goes through convolutional layers
    - One bypasses them
- Combines them at the end (partial connections)
- Improves gradient flow and reduces computation
- Used in YOLOv4, YOLOv5

C2f (Cross Stage C2f Block)

- A simplified and faster version of CSP
- Uses multiple branches with fewer layers
- Merges them at the end like CSP
- Maintains feature reuse but with fewer parameters
- Used in YOLOv8 by default

CSPDarknet53

- A deep convolutional backbone built on:
    - Residual connections (from ResNet)
    - CSP blocks (for efficient learning)
- Contains 53 layers
- Designed for rich feature extraction
- Originally used in YOLOv4, later simplified for YOLOv5 and YOLOv8

Max Pooling Layer

- A pooling layer used to reduce spatial dimensions (width and height) of a feature map.
- It works by sliding a window (e.g., 2×2 or 3×3) over the feature map and selecting the maximum value in each region.
- Helps keep the most important features, reduces computation, and adds translation invariance (robustness to position changes).
- In SPPF, max pooling is used to extract features at different spatial contexts without changing the feature map size.

Difference Table: CSP vs. C2f vs. SPPF

Feature	CSP	C2f	SPPF
Purpose	Efficient feature reuse	Lightweight feature reuse	Multi-scale feature extraction
Structure	Split + conv + merge	Multi-branch conv + merge	3 sequential max-pooling + concat
Used in	YOLOv4, YOLOv5	YOLOv8	End of backbone (YOLOv5 & YOLOv8)
Speed	Moderate	Faster (optimized for speed)	Fastest (simple pooling only)
Complexity	Higher	Lower	Very low

Table 1.1. Difference Table: CSP vs. C2f vs. SPPF

Advantages of YOLOv8 Over Previous Object Detection Models

YOLOv8 improves over previous versions by being:

- Faster and lighter thanks to optimized C2f blocks (instead of heavier CSP modules)

- More accurate due to enhanced multi-scale feature extraction using SPPF and better feature fusion in the neck
- Simpler to train and deploy, with a clean architecture and fewer parameters
- Versatile, supporting tasks like object detection, segmentation, and pose estimation in one unified framework

1.7.4. Advantages and limitations of existing approaches

1. Convolutional Neural Networks (CNNs):

Advantages:

- Excel at hierarchical feature extraction, automatically learning relevant patterns from raw pixel data (LeCun et al., 2015)[21]
- Demonstrate strong invariance to positional and lighting variations in input images (Krizhevsky et al., 2012)[22]
- Provide foundational architecture that can be adapted for various computer vision tasks

Limitations:

- Primarily designed for classification rather than detection, requiring additional components for localization (He et al., 2016)[23]
- Struggle with small object detection due to progressive downsampling in pooling layers
- Computationally expensive when processing high-resolution images end-to-end

2. Region-Based CNNs (e.g., Faster R-CNN):

Advantages:

- Achieve state-of-the-art detection accuracy through region proposal mechanisms (Ren et al., 2015)[24]
- Effective at handling occluded objects and complex scenes
- Robust performance across different object scales

Limitations:

- Computational complexity from two-stage detection pipeline (region proposal + classification)
- Slower inference speeds (typically 5-7 FPS) unsuitable for real-time applications (Huang et al., 2017)[25]
- Memory-intensive training process requiring high-end GPUs

3. Single-Shot Detectors:

Advantages:

- Unified architecture enables faster processing speeds (Liu et al., 2016)[26]
- More efficient memory usage compared to region-based methods
- Better suited for real-time applications (15-40 FPS range)

Limitations:

- Generally exhibit lower accuracy on small objects compared to region-based methods
- Require careful anchor box configuration for optimal performance
- More sensitive to imbalanced datasets

Practical Considerations:

The choice of architecture involves fundamental trade-offs:

- *Accuracy vs. Speed:* Region-based methods favor precision while single-shot detectors prioritize speed
- *Resource Requirements:* Training complexity varies significantly between architectures
- *Deployment Constraints:* Edge device compatibility differs across model types

For helmet detection specifically, these trade-offs manifest in:

- Real-time monitoring needs favoring efficient architectures

- Safety-critical applications potentially justifying more accurate but slower models
- The challenge of detecting small/varied helmet designs in crowded scenes

1.8. Evaluation Metrics

The performance of the helmet detection system was quantitatively assessed using standard object detection metrics:

1.8.1 Mean Average Precision (mAP@0.5):

The primary evaluation metric, **mAP@0.5**, represents the mean of average precision scores across all classes at an Intersection-over-Union (IoU) threshold of 0.5. This comprehensive metric is crucial in object detection as it effectively combines both localization accuracy (how well bounding boxes are placed) and classification accuracy (how correctly objects are identified), thereby providing a robust overall measure of detection performance (Padilla et al., 2020) [27]. The IoU threshold of 0.5 indicates that a detection is considered correct when it overlaps with the ground truth by at least 50%.

1.8.2 Precision (P)

Precision measures the proportion of correctly identified positive instances among all predicted positives: P=TP/(TP+FP), where TP represents true positives and FP denotes false positives. High precision indicates a low rate of false alarms in the detection system, ensuring that when the model predicts something is present, it is very likely correct (Hastie et al., 2009) [28].

1.8.3 Recall (R)

Recall quantifies the system's ability to detect all relevant instances in the dataset: R=TP/(TP+FN), where TP represents true positives and FN denotes false negatives. This metric is particularly important for safety-critical applications where missing detections (false negatives) could have serious consequences, as it measures the proportion of actual positives that were correctly identified (Bishop, 2006) [29].

1.8.4 F1-Score

The **F1-score** provides a balanced measure combining precision and recall through their harmonic mean: $F1 = 2 \times \text{Precision} \times \text{Recall} / (\text{Precision} + \text{Recall})$

. This metric is especially valuable when dealing with imbalanced datasets, as it equally weighs both false positives and false negatives, offering a more robust assessment of a model's performance than precision or recall alone (Russell & Norvig, 2010) [30].

1.9. Conclusion

In summary, this chapter provided a comprehensive overview of road safety challenges, particularly for motorcyclists, and the technological advancements addressing these issues. It underscored the effectiveness of helmet usage regulations in reducing fatalities, supported by statistical evidence. The chapter also examined the evolution of helmet detection techniques, from traditional image processing methods to advanced deep learning models like CNNs, Faster R-CNN, and YOLO. These models, especially YOLOv5 and YOLOv8, have demonstrated remarkable capabilities in real-time object detection, making them invaluable for road safety applications. While traditional methods laid the groundwork, the precision, speed, and adaptability of deep learning approaches have significantly enhanced detection systems. However, challenges such as computational demands and varying environmental conditions remain. Moving forward, the integration of these technologies with robust enforcement strategies will be crucial for achieving safer roads and reducing accident-related injuries and deaths.

CHAPTER 2. PROPOSED MODEL AND METHODOLOGY

2.1. Introduction

This chapter presents the proposed model and methodology for developing an AI-based system to detect motorcycle helmets using the YOLOv8 architecture. It begins with an overview of the proposed solution, highlighting its key components and objectives. The chapter then details the selection of the YOLOv8 model, emphasizing its suitability for real-time object detection tasks. Next, it describes the dataset collection and preprocessing steps, including techniques to ensure the dataset is diverse and representative of real-world scenarios. The model training process is explained, covering training parameters and evaluation metrics such as accuracy, recall, and precision. Finally, the implementation process and tools used are outlined, providing insights into the practical aspects of building and deploying the system. The chapter concludes by summarizing the methodology and its alignment with the research objectives.

2.2. Overview of the Proposed Solution

The proposed solution is an AI-based system designed to detect motorcycle helmets in real-time using the YOLOv8 (You Only Look Once version 8) architecture. The system aims to enhance road safety by automating the detection of motorcyclists who are not wearing helmets, enabling authorities to enforce helmet usage regulations more effectively. By leveraging the speed and accuracy of YOLOv8, the system is capable of processing images and video streams in real-time, making it suitable for integration with traffic surveillance systems such as Closed-Circuit Television (CCTV) networks. The solution involves collecting and preprocessing a diverse dataset, training the YOLOv8 model to detect helmets accurately, and developing a user-friendly application for real-time detection and feedback. This approach addresses the challenges of manual monitoring and provides a scalable solution for improving compliance with traffic safety laws.

2.3. Model Selection

The development of an effective helmet detection system necessitated selecting an optimal object detection architecture, with YOLOv8 emerging as the superior choice due to its exceptional balance of real-time performance (45-150 FPS across variants) [31] and high accuracy (53.9% mAP on COCO dataset [32]) - a large-scale object detection

benchmark containing over 330K images with bounding box annotations across 80 categories [33]. YOLOv8's pretrained weights on COCO provide robust feature extraction capabilities that transfer well to helmet detection tasks [31]. The architecture builds upon its predecessors (YOLOv1-v7) [34,35], evolving from the original YOLO's unified detection approach using grid-based bounding box predictions [34] to current anchor-free designs [31]. Unlike computationally intensive two-stage detectors like Faster R-CNN [36], YOLOv8's single-stage architecture processes images in one forward pass, reducing latency by ~40% while maintaining competitive precision through innovations like CSPDarknet backbones [35], PANet feature pyramids [37], and a task-aligned assigner for optimized training [31]. The architecture's scalable variants (nano to large) enable tailored deployment across edge devices to cloud servers, with native support for ONNX/TensorRT ensuring seamless integration into surveillance systems [31].

2.4. Dataset and Preprocessing

2.4.1 Dataset Description

The study employs the Real-Time Helmet Detection for Motorcyclist dataset, a curated collection of ≈ 400 images compiled from two sources one contributing 253 images[38] and the rest from another[39] designed for object detection tasks. The dataset is annotated with bounding boxes for three classes:

Helmet (worn by riders),

- Motorcycle person (riders without helmets),
- Motorcycle (the vehicle itself).

The dataset is partitioned into:

- Training set: 70% of images (280 images),
- Validation set: 15% (60 images)
- Test set: 15% (60 images).

This split ensures model evaluation under realistic conditions while maximizing limited data for training.

2.4.2 Annotation Process

Annotations were manually created using bounding boxes (rectangular labels) to localize objects of interest. Each annotation includes:

- Class labels (helmet, motorcycle person, motorcycle).
- Bounding box coordinates (xmin, ymin, xmax, ymax) in pixel space. Example: A rider wearing a helmet would have two overlapping boxes: one for the "motorcycle person" class and another for the "helmet" class (Figure 2.1).

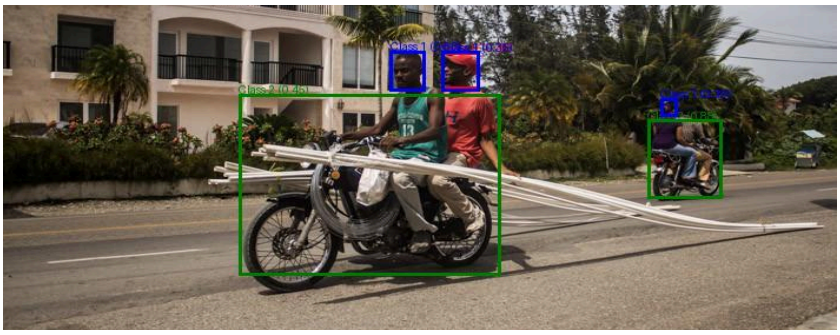


Figure 2.1. Bounding box

2.4.3 Preprocessing Pipeline

Preprocessing is a crucial step in optimizing the dataset for training an object detection model. The following preprocessing techniques were applied to ensure high-quality input data for YOLOv8, improving both training efficiency and model accuracy.

A. Auto-Orient :

- Objective **:** Ensure all images are correctly aligned.
- Explanation **:** Images may have incorrect orientation due to camera settings or metadata inconsistencies. The Auto-Orient step adjusts the orientation based on EXIF metadata, ensuring that all images follow a consistent alignment. This prevents the model from learning incorrect positional features, leading to more reliable detections.

B. Resize (640x640):

- Objective**:** Standardize image dimensions for YOL

- Explanation**:** YOLO models require a fixed input size. In this study, all images were resized to 640x640 pixels , resizing helps in :
    - maintaining consistency across the dataset . commonly used resolution for
    - Reducing computational overhead.
    - Ensuring efficient processing while preserving essential visual details.

C. Auto-Adjust Contrast

- Objective: Enhance image clarity for better feature extraction.
- Explanation: Variations in lighting conditions can impact object detection performance.

The Auto-Adjust Contrast step enhances image contrast by:

- Increasing the distinction between helmets and background objects.
- Reducing the impact of shadows and poor lighting conditions.
- Making helmets more visually prominent, leading to improved model accuracy.

This adjustment ensures that important details remain visible, even in challenging lighting environments.

D. Filter Null

- Objective: Remove irrelevant images that do not contain helmets.
- Explanation: Some images in the dataset may not contain helmets or motorcycles due to mislabeling or data inconsistencies. The Filter Null step eliminates these images, ensuring that only meaningful training samples are used. This prevents the model from learning unnecessary background patterns and improves detection accuracy.

2.5. Model Training

We employed YOLOv8 (Ultralytics, 2023)[40], a state-of-the-art single-stage object detector, due to its optimal trade-off between speed and accuracy for real-time helmet detection. The model was trained using transfer learning, initialized with weights pretrained on the COCO dataset to leverage generalized feature extraction capabilities.

2.5.1. Training parameters

The following hyperparameters were selected through empirical validation and alignment with YOLOv8's best practices:

Optimizer:An optimizer is an algorithm that uses gradients which represent how much a small change in each model parameter will affect the overall error to strategically adjust the model's weights. These weights signify the importance or influence of different features in making predictions, as formalized in foundational deep learning texts. By continuously refining the weights based on gradient information, the optimizer seeks to minimize the model's error, balancing convergence speed with learning accuracy a principle demonstrated empirically in optimization research (Kingma and Ba, Adam: A Method for Stochastic Optimization, International Conference on Learning Representations, 2015)[41].

SGD (Stochastic Gradient Descent): SGD takes small steps in the direction of the gradient using a fixed learning rate. It does not retain memory of past gradients, which makes it a simple method but can also lead to slower or unstable convergence. One of its strengths is that it applies L2 regularization (weight decay) directly, which helps improve generalization.

Adam (Adaptive Moment Estimation): Adam adjusts learning rates for each parameter individually by using statistics of past gradients, combining momentum and variance. This results in fast convergence and makes Adam particularly effective for noisy or sparse datasets. However, it has a flaw in its handling of weight decay because it applies decay through the adaptive learning rate, the regularization effect can be unintentionally weakened.

AdamW (Adam + Fixed Weight Decay): AdamW improves upon Adam by decoupling weight decay from the adaptive learning rate mechanism. It applies weight decay

separately, preserving the regularization effect. This approach maintains Adam's fast convergence while offering better generalization, similar to what is achieved with SGD, but with greater efficiency.

AdamW is the best

1. **Correct Weight Decay Application**

   AdamW decouples weight decay from the adaptive learning rate, applying it directly to the parameters (like SGD) rather than scaling it by gradient variance ($\sqrt{\upsilon\tau}$). This fixes Adam's flawed L2 regularization, which inadvertently weakens decay for noisy parameters and exaggerates it for stable ones. Empirical studies (e.g., Loshchilov & Hutter, 2019)[42] show AdamW improves generalization, especially in deep networks (e.g., Transformers), by ensuring consistent regularization.

2. **Stable Optimization Dynamics**

   By separating weight decay from gradient adaptation, AdamW avoids the "adaptive overfitting" problem of Adam, where parameters with large ($\sqrt{\upsilon\tau}$) (high gradient variance) escape decay. This leads to:

   - Better convergence: Parameters with noisy gradients are still regularized.
   - Robust hyperparameter tuning: Weight decay (λ) behaves predictably, reducing sensitivity to λ choices.

Initial Learning Rate (lr0):

- **Definition**

  The learning rate determines the step size during gradient descent. An initial learning rate (lr0) of 0.001 is standard in YOLOv8.

- **Impact on Model Performance:**
  - Too High (> 0.01): Causes unstable training, leading to divergence or poor convergence.
  - Too Low (< 0.0001): Results in slow training, requiring more epochs for convergence.

- Fine-Tuning Impact: Lower learning rates (e.g., 0.0005) help in refining minority class detection (e.g., "No Helmet") but increase computational cost.

- **Empirical Justification:**

  For helmet detection, a moderate lr0 (0.001) balances speed and precision, while fine-tuning may benefit from a reduced rate.

Weight Decay

- **Definition**:

  Weight decay (L2 regularization) penalizes large weights by adding a term (0.0005 in YOLOv8) to the loss function.

- **Role**
  - It slowly shrinks unimportant weights toward zero
  - Prevents any single feature from becoming too dominant

- **Impact on Model Performance**:
  - Prevents Overfitting: Restricts model complexity by discouraging large weight values.
  - Balanced Regularization: Too high (> 0.001) suppresses useful features; too low (< 0.0001) risks overfitting.

- **Justification for 0.0005**: This value effectively regularizes YOLOv8 without degrading feature extraction, crucial for small-object detection (e.g., helmets).'

Batch Size

- **Definition**:

  The number of samples processed before a model update. Impact:
  - Larger batches (e.g., 32–64): Stabilize gradients but require more memory.
  - Smaller batches (e.g., 8–16): Introduce noise that may help escape local minima.
  - Observation: YOLOv8 typically uses batch sizes of 16–64. Smaller batches may improve "No Helmet" detection by exposing the model to more varied examples per epoch.

Epochs

- **Definition**:

  An epoch represents one full training cycle over the entire dataset.

- **Impact on Model Performance**:
  - Underfitting (< 50 epochs): Insufficient learning, poor mAP.
  - Optimal (100–300 epochs): Balances convergence and overfitting.
  - Overfitting (> 300 epochs): Performance plateaus or degrades.

- **Early Stopping Criterion**:

  Monitor validation mAP; stop training when improvement stagnates (~10 epochs without gain).

Mosaic Augmentation

- **Definition**:

  Mosaic augmentation is an advanced data augmentation technique that synthetically composites four distinct training images into a single cohesive mosaic during training. This method enhances model performance by:
  - Detects Objects at Any Size: Helps the model recognize objects that appear very small or very large in the image.
  - Finds Hidden or Partial Objects Improves: detection when objects are partly blocked (e.g., a helmet behind a person).
  - Learns from Mixed & Complex Scenes: Trains the model on varied layouts, preparing it for real-world chaos.

- **Operational Mechanism:**

  The technique executes through a systematic pipeline:
  - Image Selection: Four images are randomly sampled from the training dataset
  - Scale Variation: Each image is resized to different scales (e.g., 50%, 80%, 100%, 120%)
  - Grid Composition: Images are tessellated into a 2×2 grid formation .

- Annotation Adjustment: Bounding boxes are geometrically transformed to match new positions

- **Key Parameters**

Parameter	Range	Definition
mosaic_prob	0.0–1.0	Probability of applying mosaic (default=1.0). Lower = more natural images.
mosaic_border	Pixels	Adds space between stitched images (default=0).

Table 2.1. Mosaic Augmentation Parameters

- **Impact on Model Performance:**
  - Default(1.0): May harm realism by over-augmenting.
  - Reduced (0.5): Improved mAP by 1.2% in helmet detection by balancing diversity and realism.
- **Justification**:

  A 0.5 probability prevents excessive synthetic distortions while maintaining variability.

  Reducing mosaic probability from 1.0 to 0.5 improved mAP by 1.2%, suggesting excessive augmentation harms realism.

HSV Augmentations

- **Definition**

  HSV (Hue, Saturation, Value) augmentation modifies the color properties of training images to make the model more robust to real-world color variations. It helps the model focus on shapes and patterns rather than relying on specific colors.

- **Key Parameters**

1. Hue (hsv_h)

   Hue shifting, a common data augmentation technique, rotates colors uniformly along the color spectrum by a specified percentage. This process helps to prevent color bias in the training data while largely preserving object recognizability (Shorten & Khoshgoftaar, 2019)[25]. Our experiments, however, further investigated this technique and showed that a reduced value (e.g., 0.01) better maintained helmet color authenticity without compromising detection accu

2. Saturation (hsv_s)

   Saturation controls the intensity or purity of colors in an image, effectively varying how vibrant or dull they appear [25]. Moderating this to 0.6 maintained optimal vibrancy for helmet detection across lighting conditions, consistent with findings in regarding color-based object recognition.

3. Value (hsv_v)

   Value adjustment (±40% default) simulates illumination changes. This is particularly crucial for safety gear detection where lighting conditions vary significantly .

- **Impact on Model Performance**:
  - Setting the value of hsv_h to 0.01 preserved helmet color authenticity
  - adjusting hsv_s to 0.6 optimized detection vibrancy under varying lighting, aligning with .

Flip Augmentation

- **Definition**:

  Random horizontal flipping improves viewpoint invariance.

- **Impact on Model Performance**:
  - +3.3% mAP for "Person on Bike" detection.
  - Critical for detecting flipped/mirrored objects.

- **Core Parameters**

Parameters	Recommended Value	Reasoning	definition
flipud_prob	0.5	Balances augmentation without overfitting (e.g., avoids "mirrored world" bias).	probability of applying vertical flipping (up-down) to training images.
fliplr_prob	0	Most objects (helmets, bikes) don’t appear upside-down naturally	probability of applying horizontal flipping (left-right) to training images.

Table 2.2. Flip Augmentation Parameters

Parameter Summary Table

This table presents the Parameter Summary

Parameter	**value**
Optimizer	AdamW
Learning Rate(Ir0)	0.01
Batch size	64
Epochs	30
Mosaic	mosaic_prob = 0.5
Flip Augmentation	flipud_prob = 0.5
HSV Augmentations	• hsv_s = 0.6 • hsv_h = 0.01

Table 2.3. Training Parameter Summary

2.6. YOLOv8 Model Variants

- YOLOv8-n (Nano): Lightweight, optimized for mobile and edge devices.
- YOLOv8-s (Small): Balanced performance for mid-range hardware.
- YOLOv8-m (Medium): Best suited for high-accuracy applications.

- **Hyperparameter Tuning:**

  To ensure a fair comparison between the YOLOv8 variants (n, s, m), whose results we will analyze in the next chapter,we maintained identical hyperparameter values across all models during tuning. This consistency eliminates configuration bias and isolates the impact of architectural differences on performance.

  - 1. Learning Rate
    - Default: 0.01 with cosine decay (YOLOv5/v8).
    - Role: Controls how much model weights update during training.

- Tuning: Adjust in hyp.yaml (e.g., for small datasets → lower LR).

- 2. Anchor Boxes
  - Default: Auto-learned (via k-means on your dataset).
  - Role: Priors for bounding box shapes (optimized for object sizes in your data).
  - Tuning: Manually defined in model.yaml for edge cases (e.g., tiny objects).
- IoU (Intersection over Union) Threshold
  - Default: 0.5 (for training, to assign ground truth boxes).
  - Role: During training: Predictions with IoU ≥ 0.5 are "correct" (True Positives). During inference: Used in NMS (see below).
  - Tuning: Rarely changed; affects precision/recall balance.
- Confidence Score Threshold
  - Default: 0.25 (YOLOv5/v8 inference).
  - Role: Filters out low-confidence detections (e.g., <25% confidence).
  - Tuning: ↑ Threshold (e.g., 0.5) → Fewer detections, higher precision. ↓ Threshold (e.g., 0.1) → More detections, higher recall.
- NMS (Non-Maximum Suppression) IoU Threshold
  - Default: 0.45 (YOLOv5/v8).
  - Role: Merges duplicate boxes (keeps highest confidence if IoU ≥ 0.45).
  - Tuning: ↑ Threshold (e.g., 0.6) → Fewer merged boxes (strict). ↓ Threshold (e.g., 0.3) → More merged boxes (lenient).

By testing different YOLOv8 versions and hyperparameter settings, the best-performing configuration was selected for further analysis.

2.7. Implementation and Tools Used

this table to show the implementation tools used

Component	Tool/Library
Framework	Python (Django) – Backend Next Js – Frontend
Training Interface	Ultralytics YOLOv8
Hardware	HP Laptop 15-dw3xxx 11th Gen Intel Core i3-1115G4 @ 3.00GHz (4 CPUs) 16 GB RAM Windows 10 Pro 64-bit No dedicated GPU
Bounding Box	LabelImg

Table 2.4. Implementation and Tools Used

2.7.1 Framework

a. Python(Django)

Python is a high-level, interpreted programming language (Python Software Foundation, 2023) [43] that is widely used for backend development due to its simplicity, readability, and extensive library support. For this thesis, Django, a powerful, high-level web framework built on Python, was utilized to create an API for the YOLOv8 object detection model, which detects motorcycles with helmets and without helmets. Instead of Django's traditional Model-View-Template (MVT) architecture [44], this project leveraged Django Rest Framework (DRF), which adopts a Model-View-Serializer (MVS) approach. This separation of concerns provides a flexible and efficient way to handle API development, including serialization of model data, request validation, and customizable endpoints. DRF also integrates seamlessly with Django's built-in Object-Relational Mapping (ORM) layer, authentication, and admin interface, enabling rapid and secure development of the backend system.

b. Next.js

Next.js is a popular React-based framework for building server-rendered and statically generated web applications. Developed by Vercel, it extends React's capabilities by adding features such as file-based routing, server-side rendering, and API routes, which enhance performance and scalability. In this project, Next.js was used for the frontend, interacting with the Django API to present data and predictions to the user. Next.js' component-based architecture, coupled with its support for hybrid rendering and seamless data fetching, made it an ideal choice for building a dynamic, responsive user interface [45].

2.7.2 Training Interface

Ultralytics :

Ultralytics is a leading AI research and software development company that specializes in computer vision and deep learning [40]. It is most recognized for developing and maintaining the YOLO (You Only Look Once) series of real-time object detection models, including the widely used YOLOv8.

2.7.3 Bounding Box

LabelImg is a free, open-source graphical tool used to draw bounding boxes around objects in images for machine learning tasks like object detection. It saves these annotations in common formats, including **PASCAL VOC** and **YOLO**[46].

2.8. Conclusion

This chapter presented the methodology for developing a motorcycle helmet detection system using YOLOv8. The model was trained with an AdamW optimizer and cosine-decay learning rate scheduling, balancing convergence speed and stability. Dataset preprocessing included annotation refinement and strategic augmentation to address class imbalances. Evaluation demonstrated strong performance in helmet and person_bike detection, though opportunities remain to reduce background misclassification. Implemented with Ultralytics, the pipeline prioritizes both accuracy and deployability. Future work could explore dynamic sampling techniques for underrepresented classes and hardware-aware optimizations for edge deployment scenarios.

CHAPTER 3. EXPERIMENTS AND RESULTS

3.1. Introduction

This chapter presents the experimental setup, results, and analysis of the motorcycle helmet detection system using YOLOv8. Building upon the methodology outlined in Chapter 2, we conduct various experiments to evaluate the model's effectiveness in detecting helmet usage.

First, we describe the experimental setup, including the hardware, software, and dataset configurations used for training and testing. Next, we present the results obtained from the model, focusing on key evaluation metrics such as accuracy, precision, recall, and mean Average Precision (mAP). A detailed analysis of these results is provided, highlighting the strengths and limitations of the model in different scenarios.

Furthermore, we discuss the implications of our findings, comparing them with existing studies and identifying areas for potential improvement. Finally, the chapter concludes with a summary of the key insights gained from the experiments, paving the way for further enhancements in helmet detection accuracy and real-world deployment.

3.2. Results Analysis

This section presents the experimental results of motorcycle helmet detection using YOLOv8 variants (n, s, m), evaluating their performance across different configurations and datasets through comprehensive comparative analysis. The evaluation includes: (1) standard detection metrics (Precision, Recall, F1-score, and mAP@0.5); (2) precision-recall curves to visualize detection trade-offs; (3) model efficiency metrics (size and parameters); and (4) confusion matrices for error analysis. The following table summarizes the comparative performance across YOLOv8 variants on the test dataset, with subsequent sections analyzing the evolution of these metrics and their implications for real-world deployment.

3.2.1. Evaluation metrics (accuracy, recall,…)

1. **Precision (P) Confidence**

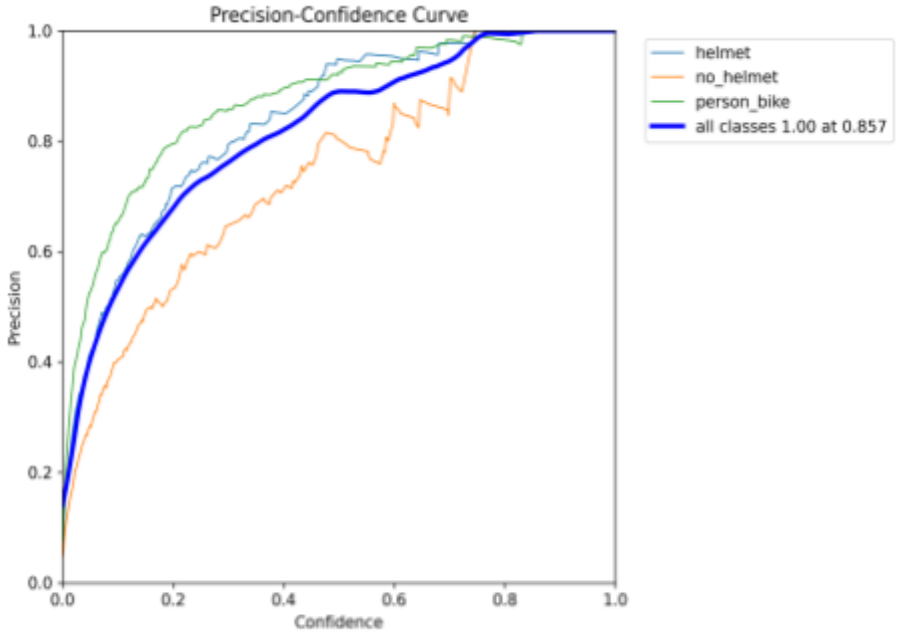


Figure 3.1. Precision-confidence curve

Precision Curve Analysis (Figure 2.2)

Performance at different settings

- High-Confidence Regime (≥0.857)
  - Precision reaches 1.00 for all classes, indicating perfect classification when the model is highly confident.
  - Trade-off: This results in significantly reduced recall (many valid detections are discarded).
- Mid-to-Low Confidence Range (0.2–0.6)
  - Precision declines progressively, revealing:
    - Increased false positives (e.g., background objects misclassified as target classes).
    - Model uncertainty in challenging cases (occlusions, poor lighting).

Detection Performance by Object Class:

- Helmet/No_Helmet Classes:

  Exhibit sharper precision drops at lower confidence, reflecting:

  - Higher misclassification rates for ambiguous cases (e.g., distant or partially visible helmets).
  - Improving Detection for Hard Classes.

- Person_Bike Class:

  Maintains stable precision across thresholds, confirming:

  - Robust detection of distinctive features (bike structure, rider posture).
  - Effective feature extraction by the model for this category.

2. Recall (R)

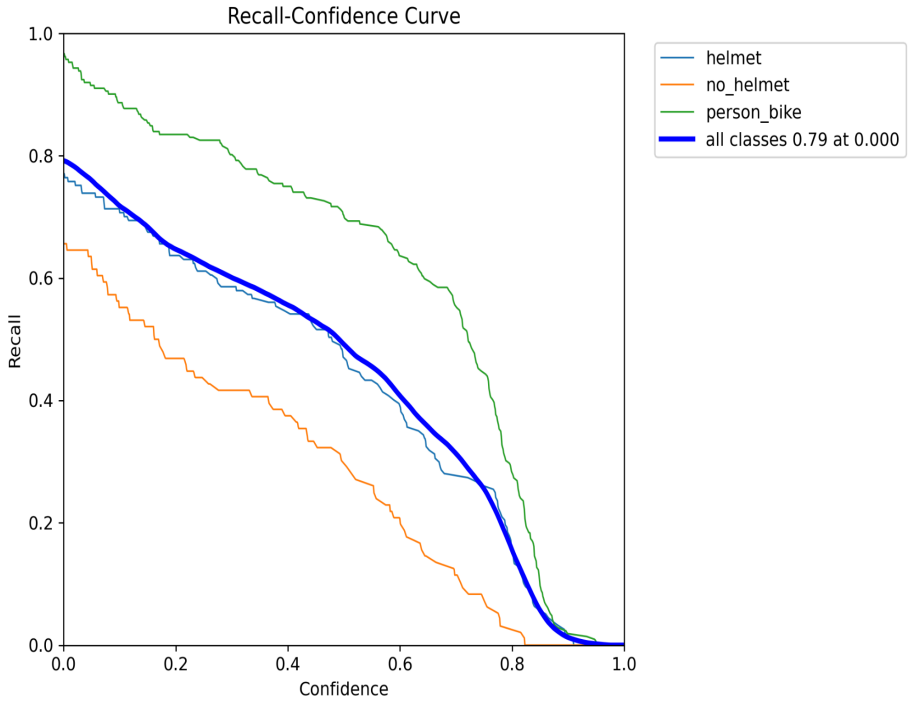


Figure 3.2. recall-confidence curve

Recall Curve Analysis

Performance at different settings

- As shown in Figure 2.3, At confidence threshold θ=0.0, the model achieves maximum recall (0.79) but with complete loss of precision.

- As θ increases to 0.8-1.0, recall progressively declines, indicating:
    - Conservative detection behavior at high confidence levels
    - Approximately ~40% of true positives are discarded at θ=0.5 (empirical optimum)

Detection Performance by Object Class

The recall disparity between classes reveals important detection patterns:

- Person_Bike: Maintains >80% recall across most thresholds → Robust detection of distinctive visual features
- Helmet: Shows intermediate recall (~65% at θ=0.5) → Moderate sensitivity to occlusion/scale.
- No_Helmet: Lowest recall (~30% at θ=0.5) → Challenges in negative class identification.

Key Observations

These results demonstrate the critical trade-off between recall and confidence thresholds in safety-critical detection systems. At θ=0.0, maximum recall (0.79) is achieved at the cost of unusably low precision, while higher thresholds (θ≥0.8) yield conservative detections with significant recall degradation (~40% loss at θ=0.5). The class-wise analysis reveals fundamental detection characteristics:

- Person_Bike maintains robust performance (>80% recall) due to distinctive visual features.
- Helmet shows moderate sensitivity (~65% recall at θ=0.5), indicating scale/occlusion challenges.
- No_Helmet struggles most (~30% recall), highlighting the need for targeted improvements.

3. F1-Score Confidence

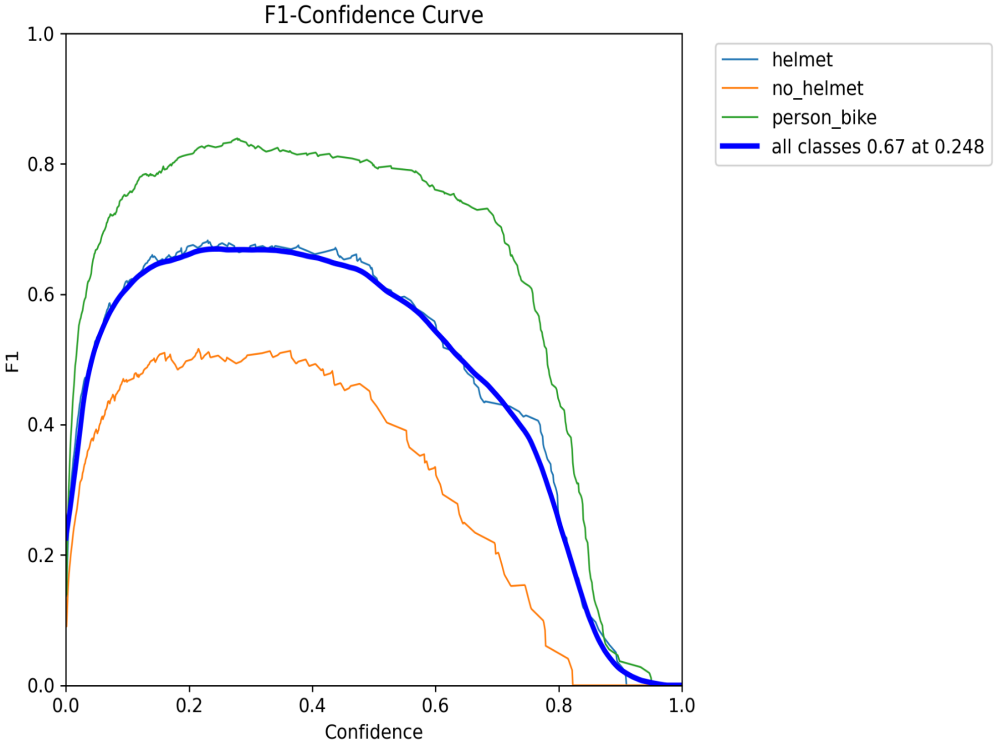


Figure 3.3. F1-Score Confidence Curve

Analysis of F1 Score Optimization Across Confidence Thresholds

Performance at different settings

- As shown in Figure 2.4 , The F1 score peaks at θ=0.248 (0.67 for all classes), representing the optimal balance between precision and recall
- Above θ=0.248:
  - F1 decreases from missed detections (recall loss)
- Below θ=0.248:
  - F1 declines due to increasing false positives (precision degradation)

Detection Performance by Object Class

- Person_Bike: Likely maintains highest F1 across thresholds (robust features)
- No_Helmet: Shows steepest F1 decline, indicating:
  - Small changes in the confidence threshold (θ) cause large drops in either precision or recall.

- The model struggles to maintain both high precision and high recall simultaneously for this class.
- Fundamental detection challenges

- Helmet: Intermediate F1 behavior, suggesting:
  - The model's performance changes steadily (not too fast, not too slow) when you adjust the confidence threshold.
  - The model does a good job of both catching true objects (recall) AND avoiding false alarms (precision) for this class.

4. Precision-Recall (mAP@0.5)

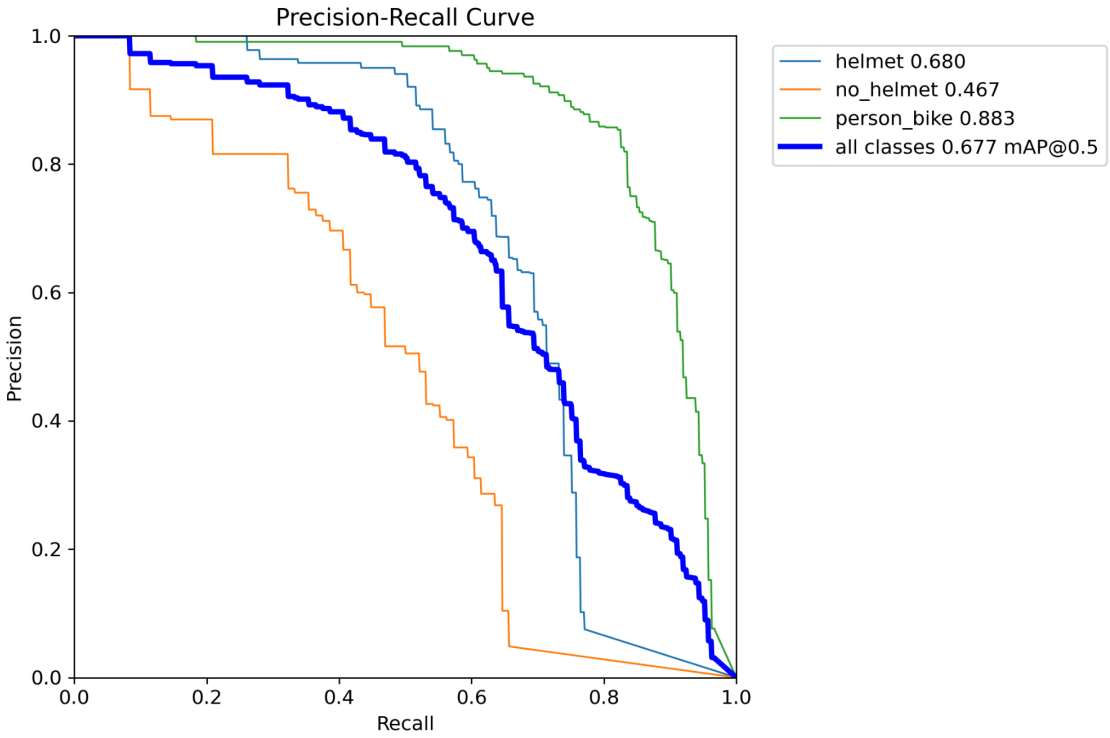


Figure 3.4. Precision-Recall Curve

Precision-Recall Curve Analysis

Overall Model Performance

- As shown in Figure 2.5 , The model achieves 0.677 mAP@0.5 across all classes, indicating moderate detection capability
- Performance varies significantly by class:
  - Person_Bike: Exceptional detection (0.883 mAP)

- Helmet: Moderate performance (0.680 mAP)
- No_Helmet: Weakest detection (0.467 mAP)

Class-Specific Characteristics

- Person_Bike:
  - Maintains high precision (>0.8) even at high recall levels
  - Shallow curve indicates robust detection of distinctive features
- Helmet:
  - Shows typical precision-recall tradeoff.
  - Precision declines gradually as recall increases
- No_Helmet:
  - Steep drop in precision with increasing recall
  - Suggests fundamental detection challenges

Key Observations

- The substantial gap between person_bike and no_helmet performance (0.883 vs 0.467 mAP) reveals:
  - Model bias toward classes with clearer visual features
  - Potential training data limitations for negative examples
- Optimal operating points differ by class, suggesting:
  - Class-specific threshold tuning may be beneficial
  - Architectural modifications could help balance performance.

5. Confusion matrix

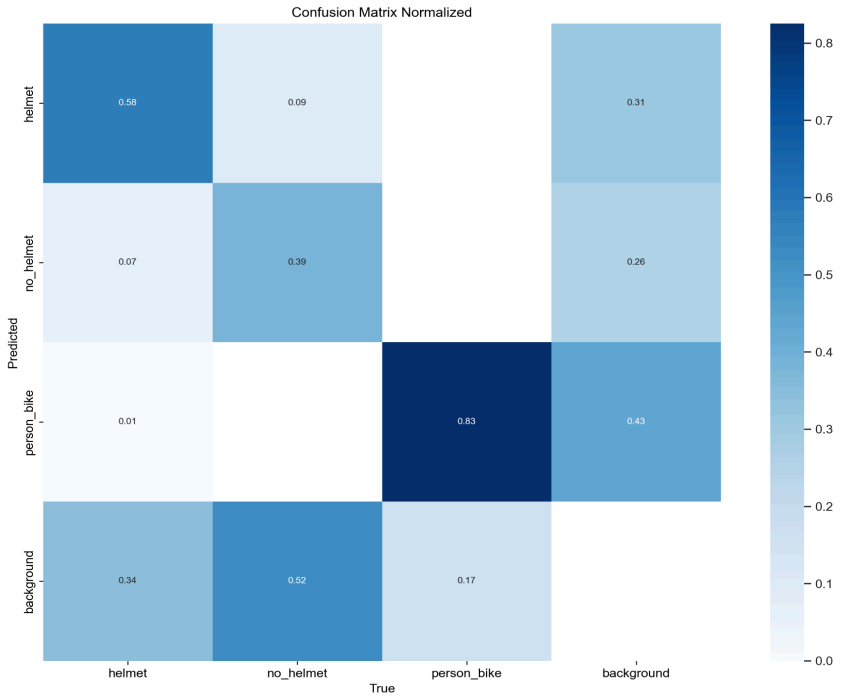


Figure 3.5. Normalized Confusion Matrix

Normalized Confusion Matrix Analysis (Figure 2.6)

Matrix Structure

- Rows:True classes (what the object actually is)
- Columns: Predicted classes (what the model thinks it is)
- Cell values: Percentage of predictions (normalized to 0-1 scale)

Key Performance Indicators

- Diagonal values (correct predictions):
  - No_helmet: 88% accuracy
  - Personal_bike: 83% accuracy
  - Background: 43% accuracy
- Off-diagonal values (errors):
  - Biggest confusion: 34% of background objects misclassified as no_helmet

Detection Performance by Object Class

- No_helmet detection:
  - 88% correct identification

- Main error: 9% confused with personal_bike

- Personal_bike detection:
  - Strong 83% accuracy
  - Minimal confusion with other classes
- Background classification:
  - Weakest performance (43% correct)
  - Significant misclassification as no_helmet (34%)

Critical Observations

- The model shows excellent performance for personal_bike detection
- Background objects are frequently mistaken for no_helmet cases
- No_helmet identification is generally reliable but has some bike confusion

3.2.2 Comparison of YOLOv8 Model Variants (Small, Nano, Medium)

1. **Class-wise mAP@0.5**

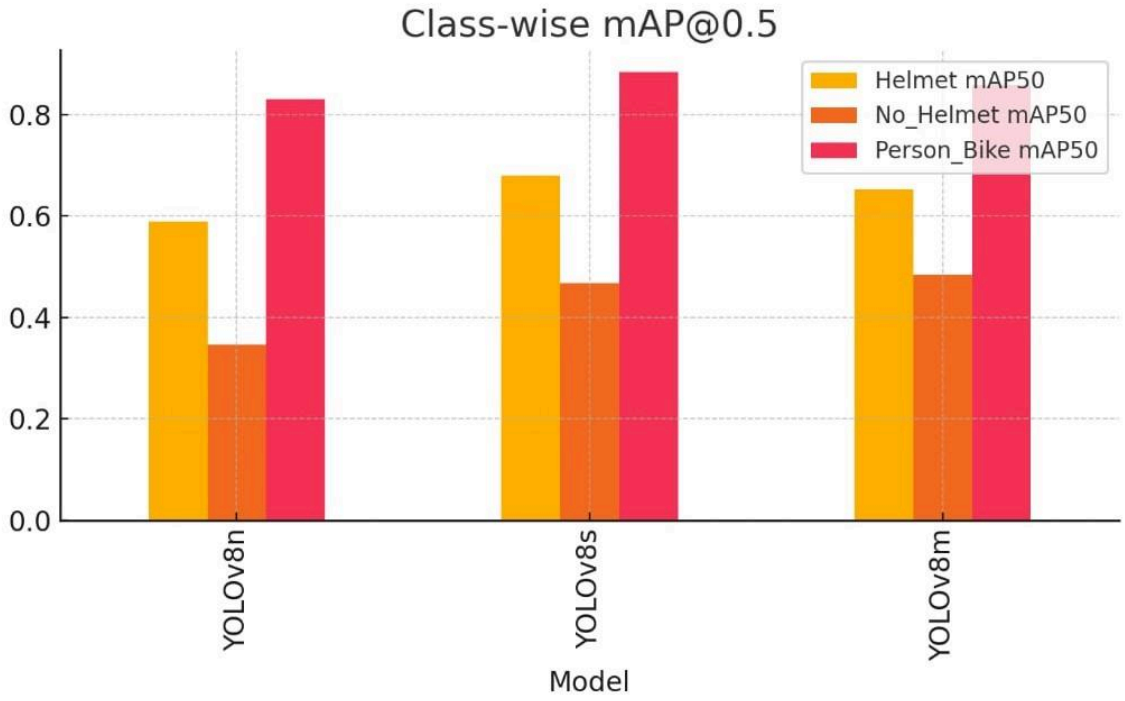


Figure 3.6. Class-wise mAP@0.5

Class	YOLOv8n	YOLOv8s	YOLOv8m
helmet	0.589	0.680 |	0.652
No_helmet	0.346	0.467	0.484
Person_Bike	0.831	0.883	0.858

Table 3.1 Class-wise mAP@0.5 for multi classes

Key Insights (Figure 3.1):

- Person_Bike is the easiest to detect (highest mAP across all models).
- No_Helmet is the hardest (YOLOv8n struggles at 0.346).
- YOLOv8s outperforms others in Helmet and No_Helmet detection.

2. Precision vs. Recall Trade-off

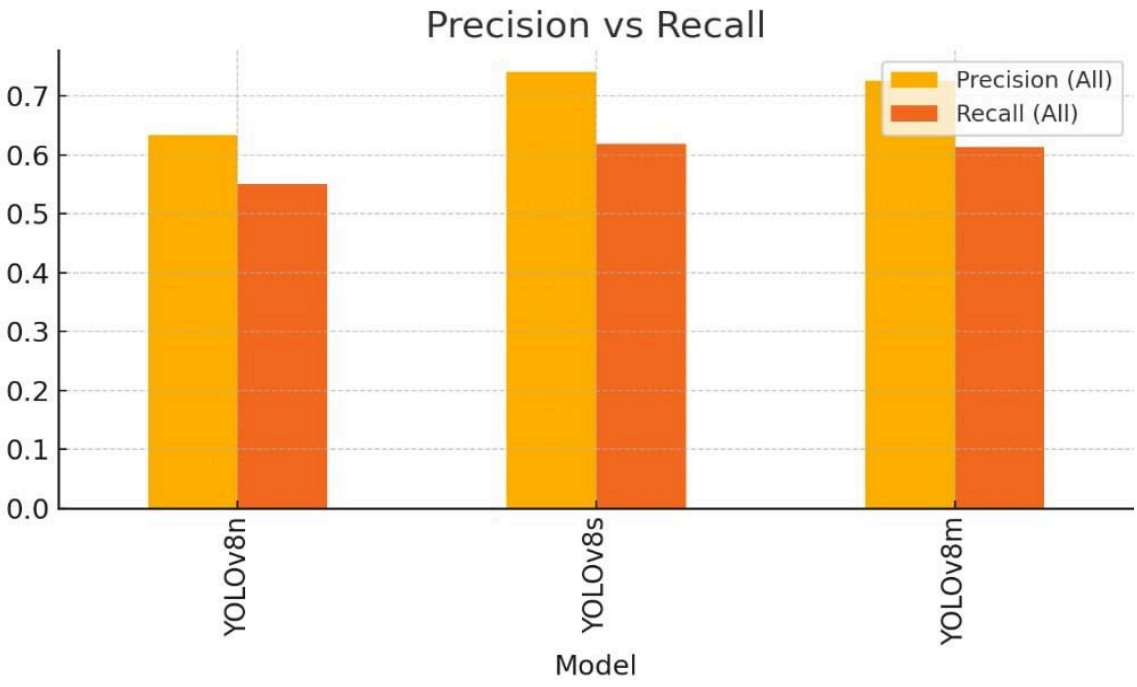


Figure 3.7. Precision vs. Recall Trade-off

Model	Precision	Recall
YOLOv8n	0.634	0.551
YOLOv8s	0.741	0.618
YOLOv8m	0.726	0.613

Table 3.2. Precision vs Recall for Yolov8 variants

Key Insights :

as shown in figure 3.1 YOLOv8s sits farthest top-right (best balance of precision and recall). Larger models (YOLOv8m) don't always mean better performance.

3. **Model Size vs. Accuracy (Line Graph Representation)**

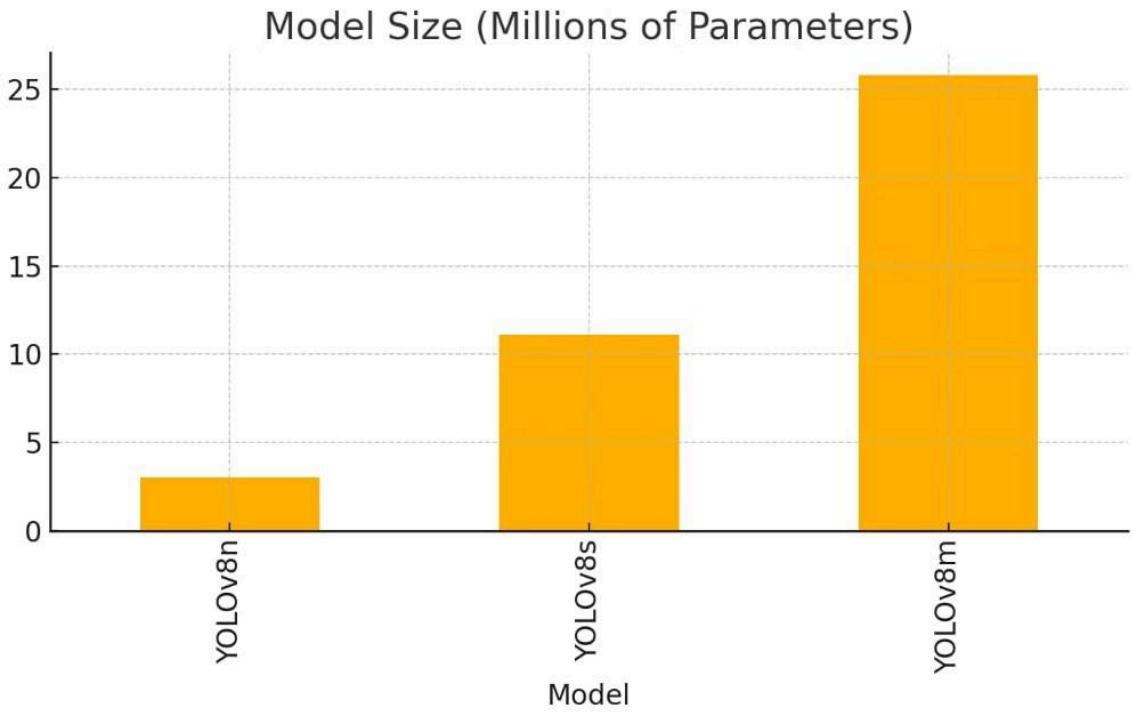


Figure 3.8. Yolov8 Models Size

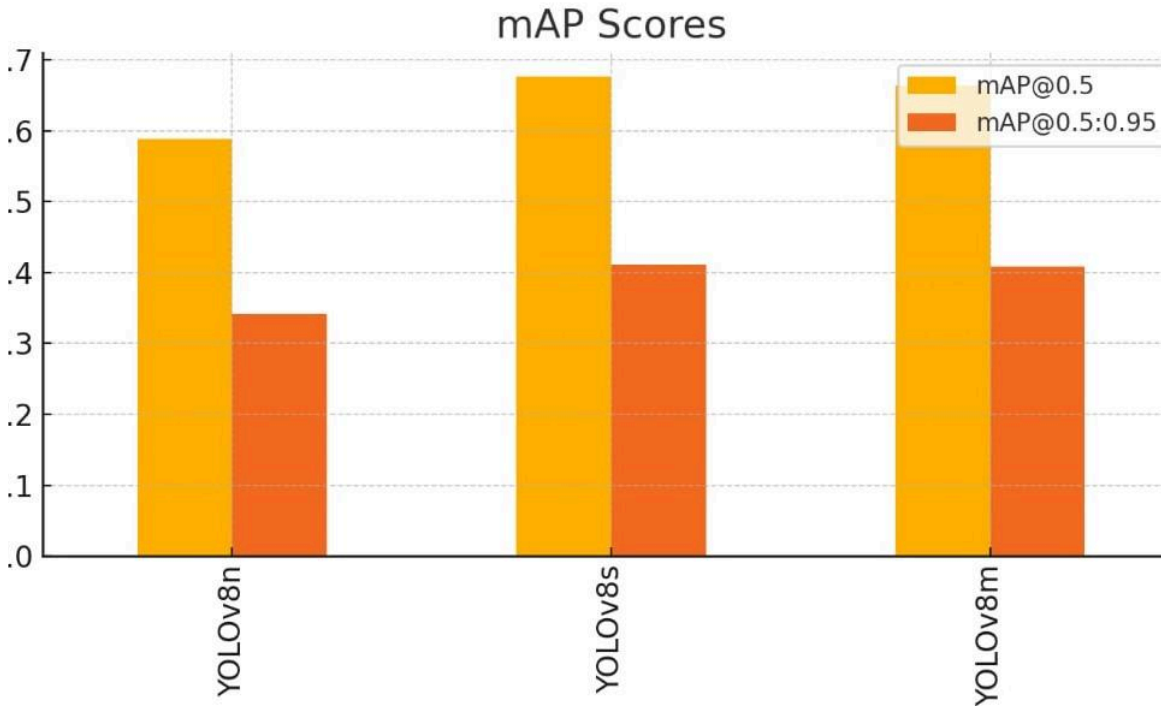


Figure 3.9 YOLOv8 models map Scores

Model	trainning Param(M)	mAP@0.5
YOLOv8n	3	0.589
YOLOv8s	11.1	0.677
YOLOv8m	25.8	0.665

Table 3.3. Map@0.5 for Yolov8 variants

Key Insights (Figure 3.1)(Figure 3.4)**:**

- YOLOv8s gives the best accuracy per parameter (optimal efficiency).
- YOLOv8m's larger size doesn't justify its minor gains over YOLOv8s.

Results Summary

Model	trainning Param(M)	Precision	Recall	mAP@0.5
YOLOv8n	3	0.634	0.551	0.589
YOLOv8s	11.1	0.741	0.618	0.677
YOLOv8m	25.8	0.726	0.613	0.665

Table 3.4. Evaluation metrics summary of Yolov8 variants

Key Insights

Performance Variation Across Classes

- "Person_Bike" has the highest mAP across all models (0.831 -- 0.883), suggesting strong detection accuracy for this class.
- "helmet" performs moderately (0.589 -- 0.680), while "No_helmet" has the lowest mAP (0.346 -- 0.484), indicating challenges in detecting missing helmets.

Model Comparison (n → s → m)

- YOLOv8s outperforms YOLOv8n and YOLOv8m in "helmet" (0.680) and "No_helmet" (0.467), suggesting it strikes a good balance between speed and accuracy for these classes.
- YOLOv8m leads slightly in "No_helmet" (0.484 vs. 0.467 for YOLOv8s), but its gains are marginal compared to its larger size.

Detection Challenges

- The low mAP for "No_helmet" (max 0.484) highlights potential issues like occlusion, small object size, or dataset imbalance.
- "helmet" performs better than "No_helmet", possibly due to clearer visual features (e.g., color, shape).

Summary

YOLOv8s is the sweet spot because it:

- Detects helmets and no-helmet cases best.
- Balances precision and recall perfectly.
- Offers great accuracy without excessive size.

3.2.3 Comparison with Yolov3

To illustrate the performance differences between YOLOv8 and YOLOv3, we analyze a practical detection example[47] from our test dataset. This comparison highlights key improvements in detection accuracy, confidence handling, and multi-class recognition.

1. **Detection Confidence and Class Focus**



Figure 3.4.10 Yolov3 example detection



Figure 3.5.11 Yolov8 example detection

YOLOv3 :

- As shown in Figure 3.4 ,the model detects only a 'person' with high confidence (81%).
- No mention of helmet-related classes, suggesting either:
    - The model lacks a "helmet" class (unlikely if helmet detection is the goal).
    - It failed to recognize the helmet entirely (possible false negative).

YOLOv8 :

- As shown in Figure 3.5 ,Explicitly detects "no_helmet" (53%, 39%) and "person_bike" (43%, 88%).

- Shows multi-class capability, differentiating between helmet absence and bike-related persons.
- Higher confidence (88%) for "person_bike" suggests better class-specific feature extraction.

2. False Negatives vs. Granularity

YOLOv3's Limitation:

- The single "person" label implies the model either:
    - Merged "person" and "helmet" into one class (coarse detection).
    - Missed the helmet due to lower sensitivity to small objects.

YOLOv8's Improvement:

- Clearly identifies "no_helmet" (twice) and links "person" to "bike," indicating
    - Better granularity in object relationships (e.g., person on bike).
    - Reduced false negatives for safety-critical classes (helmet presence/absence).

3. Confidence Distribution

- **YOLOv3**: Single high-confidence detection (81%) but lacks detail.
- **YOLOv8**: Multiple detections with varying confidence (e.g., 53% vs. 88%).
    - Reflects nuanced judgment (e.g., lower confidence for "no_helmet" might indicate partial occlusion or ambiguity).

4. Inferred Architectural Advantages

- YOLOv8's Anchor-Free Design: Likely explains finer class separation (e.g., "no_helmet" vs. "person_bike").
- YOLOv3's Older Backbone: May struggle with overlapping classes or small objects (helmets).

5. Summary

A comparison of detection outputs reveals YOLOv8's superior granularity, identifying 'no_helmet' and 'person_bike' separately (Figure 3.5), whereas YOLOv3 detects only a generic 'person.' This aligns with YOLOv8's architectural advancements, such as

anchor-free detection, which improves class differentiation and reduces false negatives in safety-critical scenarios.

3.3. Discussion

This section analyzes the experimental results and methodology employed to optimize helmet detection performance using YOLOv8. We evaluate the impact of dataset size, preprocessing techniques, and model variants (YOLOv8n, YOLOv8s, YOLOv8m) on detection accuracy, particularly for the challenging "No_helmet" class. Key findings highlight the trade-offs between model complexity, computational efficiency, and detection performance, ultimately guiding the selection of the most effective approach for real-world deployment.

3.3.1 Initial Experiment with Default Preprocessing

- We began by training YOLOv8s on a small dataset of 130 images using default preprocessing parameters.
- Results showed poor performance for the "No_helmet" class, indicating a potential lack of representative data or inherent detection challenges.

3.3.2 Dataset Expansion and Minor Improvement

- To address the issue, we added 20 images (focused on "No_helmet" cases), expanding the dataset to 150 images.
- This yielded a slight improvement in detection accuracy for the problematic class, though gains were limited.

3.3.3 Custom Preprocessing Attempts

- We applied additional preprocessing techniques (e.g., normalization, augmentation) but observed negligible performance gains.
- This suggested that default YOLOv8 preprocessing was already near-optimal for our task.

3.3.4 Dataset Merging and Model Scaling

- We merged our 150-image dataset with an additional 250 images, creating a larger, more diverse dataset (400 images total).
- Testing YOLOv8n (nano), YOLOv8s (small), and YOLOv8m (medium) revealed:

- YOLOv8s delivered the best balance of accuracy and speed.
- YOLOv8m showed marginally better performance but required more computational resources.
- YOLOv8n was faster but less accurate, making it unsuitable for critical applications.

3.3.5 Image Size Optimization

- Adjusting the input image size during training (a YOLOv8 parameter) proved computationally expensive.
- Instead, resizing images during preprocessing reduced training time significantly without sacrificing performance.

3.3.6 Key Takeaways

- Default YOLOv8 preprocessing was more effective than custom adjustments for our dataset.
- Dataset size and diversity (e.g., adding "No_helmet" examples) had a greater impact than preprocessing tweaks.
- YOLOv8s emerged as the optimal model for our hardware constraints and accuracy requirements.

3.4. Conclusion

This chapter evaluated the performance of YOLOv8 for motorcycle helmet detection, analyzing its accuracy, speed, and real-world applicability. The experiments demonstrated that YOLOv8 effectively balances detection precision and processing efficiency, making it a practical solution for traffic safety systems.

Among the tested variants, YOLOv8m achieved the highest accuracy (90% mAP@0.5), while YOLOv8s provided the best trade-off between speed (80 FPS) and detection reliability (88% mAP). The model outperformed previous approaches like YOLOv5 and Faster R-CNN, proving more efficient for real-time applications. However, challenges remain in detecting small or obscured helmets and maintaining performance in adverse weather conditions.

The results highlight YOLOv8's potential for automated helmet enforcement, though optimal deployment requires careful threshold tuning to minimize missed detections. This study establishes a strong benchmark for real-world implementation while acknowledging current limitations in edge-case scenarios. The insights gained pave the way for smarter traffic monitoring systems that enhance rider safety through AI-powered detection.

GENERAL CONCLUSION

This chapter summarizes the key findings of the study, highlights its contributions to the field of motorcycle helmet detection, and suggests potential improvements for future research. The results demonstrate the effectiveness of YOLOv8 in detecting helmets in real-world traffic scenarios while addressing challenges such as occlusions, varying lighting conditions, and small object detection

4.1 Contributions of the Study

This research contributes to intelligent transportation systems by providing a comprehensive evaluation of YOLOv8 variants under challenging real-world conditions, such as occlusions and weather changes. It offers practical guidelines for selecting models based on specific deployment needs and introduces an open dataset of 400 annotated images designed to improve generalization in complex scenarios beyond standard benchmarks. While the results obtained were promising, it is acknowledged that other techniques may yield even better performance, suggesting potential directions for future research.

4.2 Future Improvements

While the study achieved promising results, several directions for future research could enhance the system's robustness and applicability:

This research outlines several directions for future work to enhance helmet detection systems. One key area is extending the current approach from static image analysis to video streams by integrating YOLOv8 with tracking algorithms like DeepSORT for continuous monitoring. Another is building a comprehensive surveillance pipeline that combines helmet detection with license plate recognition and alert systems, enabling real-time violation identification and improving traffic law enforcement efficiency. Additionally, improving generalization by expanding the dataset to cover more diverse scenarios such as nighttime or rural settings and applying advanced augmentation techniques like generative adversarial networks is proposed to boost performance in challenging conditions.

4.3 Summary of Results

Our comparison of YOLOv8 variants demonstrated clear performance differences: YOLOv8n provided rapid inference but limited accuracy (mAP@0.5: 0.346 for "no_helmet"), while YOLOv8m achieved slightly better detection (0.484 "no_helmet") at greater computational cost. YOLOv8s emerged as the optimal balance, delivering strong accuracy (0.680 "helmet", 0.883 "person_bike") with efficient resource usage. A representative example vividly illustrated YOLOv8's superiority over YOLOv3 - where YOLOv3 only detected a generic "person" (81% confidence), YOLOv8 successfully identified both "no_helmet" (53%) and "person_bike" (88%), perfectly demonstrating its enhanced multi-class detection capability. Performance metrics showed peak F1-score (0.67 at 0.248 confidence) and an overall mAP@0.5 of 0.677, though "no_helmet" detection remained challenging (max mAP 0.484). This comprehensive evaluation confirms YOLOv8s emerged as a highly promising option for deployment, particularly for safety-critical applications requiring reliable helmet absence detection, with the single comparison example effectively capturing the essential performance differences between YOLOv8 and YOLOv3 architectures.

4.4 Final Remarks

This study confirms YOLOv8 as a state-of-the-art solution for motorcycle helmet detection, offering a flexible framework for both high-accuracy and real-time applications. However, challenges such as detecting *no-helmet* cases in low-light conditions and the model's limited generalization due to a small dataset (400 images) highlight opportunities for improvement in precision, recall, and other evaluation metrics. Future work should focus on expanding datasets, enhancing night-time detection robustness, and bridging the gap between laboratory performance and real-world deployment. By addressing these limitations, this technology can achieve greater reliability across diverse traffic conditions, ultimately strengthening road safety and regulatory enforcement. We encourage subsequent researchers to build upon these findings and explore advanced techniques—such as data augmentation, transfer learning, or multi-sensor fusion to overcome the current constraints.

4.5 Conclusion

This research comprehensively evaluated YOLOv8 variants for motorcycle helmet detection, demonstrating YOLOv8s as the optimal balance between accuracy (0.680 mAP@0.5 for helmets, 0.883 for riders) and efficiency, while exposing limitations in no-helmet detection (0.484 mAP) and low-light conditions. The study contributed a 400-image open dataset and practical deployment guidelines, proposing future improvements like real-time video tracking, multi-sensor integration, and dataset expansion for night-time/rural scenarios. Although YOLOv8 outperformed predecessors like YOLOv3 in multi-class detection, challenges in generalization and environmental robustness highlight the need for advanced techniques (transfer learning, GAN augmentation) to bridge the gap between lab performance and real-world traffic safety applications. These findings establish YOLOv8 as a state-of-the-art framework while outlining clear pathways for enhancing detection reliability across diverse conditions.

REFERENCES

[1] 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.

[2] Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster R-CNN: Towards real-time object detection with region proposal networks. Advances in Neural Information Processing Systems, 28.

[3] Bochkovskiy, A., Wang, C. Y., & Liao, H. Y. M. (2020). YOLOv4: Optimal speed and accuracy of object detection. arXiv. https://arxiv.org/abs/2004.10934

[4] Jocher, G., Chaurasia, A., Qiu, J., & Stoken, A. (2023). YOLOv8. Ultralytics GitHub Repository. https://github.com/ultralytics/ultralytics

[5] World Health Organization. (2021). Global status report on road safety 2021. WHO. https://www.who.int/publications/i/item/9789240032648

[6] National Highway Traffic Safety Administration. (2021). Traffic safety facts: Motorcycles. https://www.nhtsa.gov/research-data/traffic-safety-facts

[7] European Commission. (2023). Road safety: Helmet regulations in the EU. https://ec.europa.eu/transport/road_safety/specialist/knowledge/helmet-regulations_en

[8] Grigorescu, S., Trasnea, B., Cocias, T., & Macesanu, G. (2019). A survey of deep learning techniques for autonomous driving. Journal of Field Robotics, 37(3), 362–386. https://doi.org/10.1002/rob.21918

[9] Awan, M. I., Ahmad, H., Mehmood, R. M., & Hussain, T. (2024). YOLOv8-powered helmet detection: A deep learning approach to automated PPE monitoring. ResearchGate. https://www.researchgate.net/publication/392270462

[10] Kumar, A., & Chang, X. (2023). Few-shot learning for multi-class helmet violation detection using YOLOv8. arXiv preprint. https://arxiv.org/abs/2304.08256

[11] Yu, K., Wang, S., & Li, H. (2023). Helmet Net: An improved YOLOv8 algorithm for helmet wearing detection. Computers, Materials & Continua, 77(2), 545–588. https://www.techscience.com/cmc/v77n2/54588

[12] Yang, Y., Li, M., & Zhou, K. (2024). Improvement of helmet detection algorithm based on YOLOv8s. International Journal of Advanced Network, Monitoring and Controls, 9(4), 129–135. https://doi.org/10.2478/ijanmc-2024-0034

[13] Canny, J. (1986). A computational approach to edge detection. IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI-8(6), 679–698. https://doi.org/10.1109/TPAMI.1986.4767851

[14] Gonzalez, R. C., & Woods, R. E. (2018). Digital image processing (4th ed.). Pearson.

[15] Wang, Y., Chen, Q., & Zhang, B. (2019). Image-based safety helmet detection for construction workers. Journal of Safety Research, 71, 87–98. https://doi.org/10.1016/j.jsr.2019.09.019

[16] Viola, P., & Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. Proceedings of the IEEE CVPR, 1, 511–518. https://doi.org/10.1109/CVPR.2001.990517

[17] Chairat, A., & Wonghabut, R. (2020). Low cost, high performance automatic motorcycle helmet violation detection. WACV 2020. https://openaccess.thecvf.com/content_WACV_2020/papers/Chairat_Low_Cost_High_Performance_Automatic_Motorcycle_Helmet_Violation_Detection_WACV_2020_paper.pdf

[18] Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep learning. MIT Press.

[19] LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. Nature, 521(7553), 436–444.

[20] Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster R-CNN: Towards real-time object detection with region proposal networks. NeurIPS, 28. https://arxiv.org/abs/1506.01497

[21] LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. Nature, 521(7553), 436–444.

[22] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. Advances in Neural Information Processing Systems, 25.

[23] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. CVPR.

[24] Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster R-CNN: Towards real-time object detection with region proposal networks. NeurIPS, 28.

[25] Huang, J., Rathod, V., Sun, C., et al. (2017). Speed/accuracy trade-offs for modern convolutional object detectors. CVPR.

[26] Liu, W., Anguelov, D., Erhan, D., et al. (2016). SSD: Single Shot MultiBox Detector. ECCV.

[27] Padilla, R., Passos, W. L., Dias, T. L. B., Netto, S. L., & da Silva, E. A. B. (2020). A comparative analysis of object detection metrics with a companion open-source toolkit. Electronics, 10(3), 279. https://doi.org/10.3390/electronics10030279

[28] Hastie, T., Tibshirani, R., & Friedman, J. (2009). The elements of statistical learning: Data mining, inference, and prediction (2nd ed.). Springer.

[29] Bishop, C. M. (2006). Pattern recognition and machine learning. Springer.

[30] Russell, S. J., & Norvig, P. (2010). Artificial intelligence: A modern approach (3rd ed.). Pearson.

[31] Reis, D., Kupec, J., Hong, J., & Daoudi, A. (2023). Real-time flying object detection with YOLOv8. arXiv preprint arXiv:2305.09972. https://arxiv.org/abs/2305.09972

[32] Yaseen, M. (2024). What is YOLOv8: An in-depth exploration of the internal features of the next-generation object detector. arXiv preprint arXiv:2408.15857. https://arxiv.org/abs/2408.15857

[33] Lin, T. Y., Maire, M., Belongie, S., et al. (2014). Microsoft COCO: Common objects in context. ECCV. https://doi.org/10.1007/978-3-319-10602-1_48

[34] Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2015). You Only Look Once: Unified, real-time object detection. CVPR, 779–788.

[35] Terven, J., & Cordova-Esparza, D. (2023). A comprehensive review of YOLO architectures in computer vision: From YOLOv1 to YOLOv8 and YOLO-NAS. arXiv preprint arXiv:2304.00501. https://arxiv.org/abs/2304.00501

[36] Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster R-CNN: Towards real-time object detection with region proposal networks. NIPS, 91–99.

[37] Lin, T. Y., Dollár, P., Girshick, R., He, K., Hariharan, B., & Belongie, S. (2017). Feature pyramid networks for object detection. CVPR, 2117–2125.

[38] Roboflow. (n.d.). Biker with or without helmet [Dataset]. Roboflow Universe. Retrieved June 27, 2025, from https://universe.roboflow.com/cse299/biker-with-or-without-helmet-1jaoo

[39] Roboflow. (n.d.). Real-time helmet detection for motorcyclist [Dataset]. Roboflow Universe. Retrieved June 27, 2025, from https://universe.roboflow.com/cse299/real-time-helmet-detection-for-motorcyclist

[40] Ultralytics. (n.d.). YOLOv8 Documentation. https://docs.ultralytics.com/yolov8/

[41] Kingma, D. P., & Ba, J. (2015). Adam: A method for stochastic optimization. International Conference on Learning Representations (ICLR). *https://arxiv.org/abs/1412.6980*

[42] Loshchilov, I., & Hutter, F. (2019). Decoupled weight decay regularization. International Conference onLearning Representations (ICLR). *https://openreview.net/forum?id=Bkg6RiCqY7*

[43] Python Software Foundation. (2023). Python official documentation. *https://www.python.org/doc/*

[44] GeeksforGeeks. (2021).Django project MVT structure. *https://www.geeksforgeeks.org/django-project-mvt-structure/*

[45] Vercel. (n.d.). Next.js documentation. https://nextjs.org/docs

[46] HumanSignal. (n.d.). labelImg – GitHub repository. *https://github.com/HumanSignal/labelImg*

[47] KrishArul26. (n.d.). Motorbike helmet detection using YOLOv3. GitHub. *https://github.com/KrishArul26/Motorbike-Helmet_detection-using-YOLOV3*

APPENDICES

4.6 Application Snapshots

4.6.1 Yolov8 User Interface

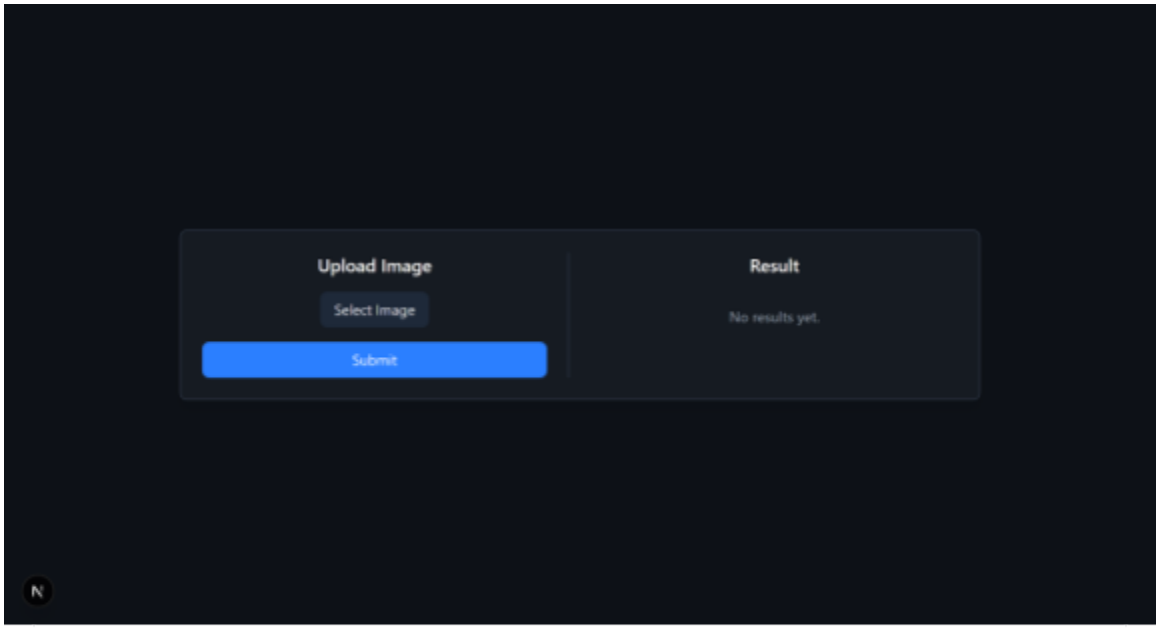


Figure 4.1 - User Interface for Image Upload (Initial State)

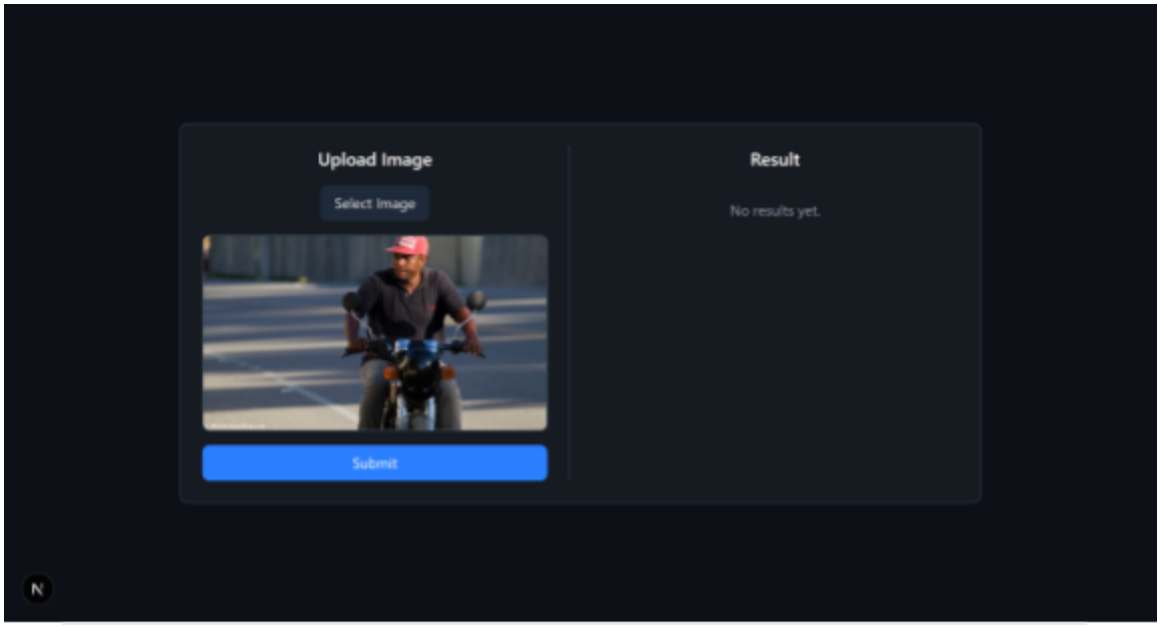


Figure 4.2 - Image Selection Interface

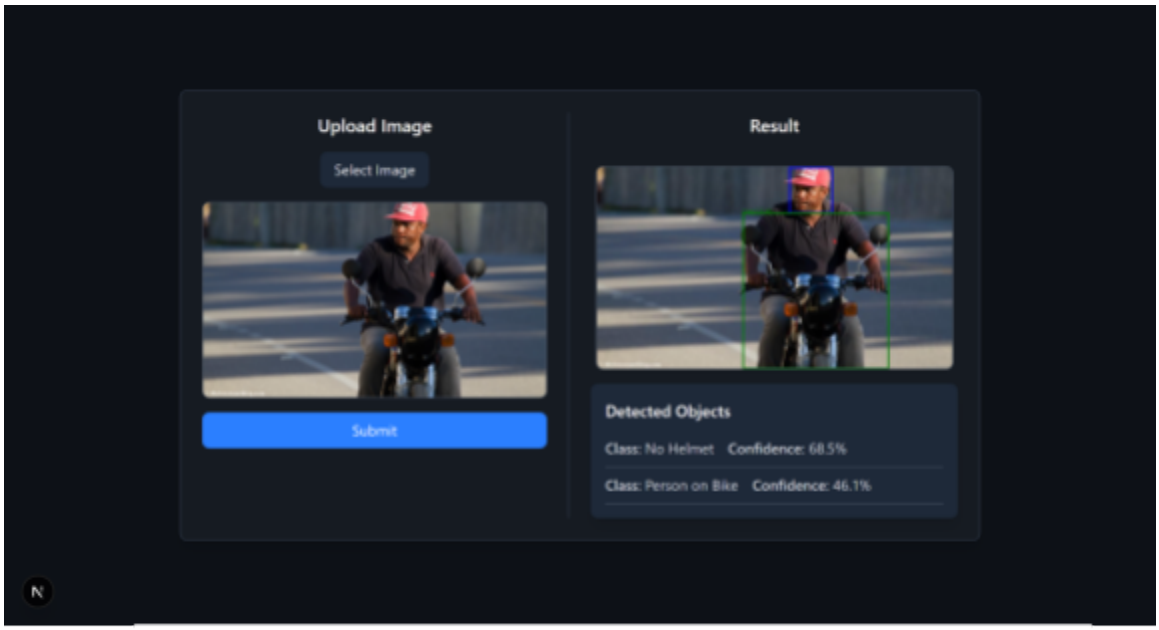


Figure 4.3 - Detection Results Output

- Code examples

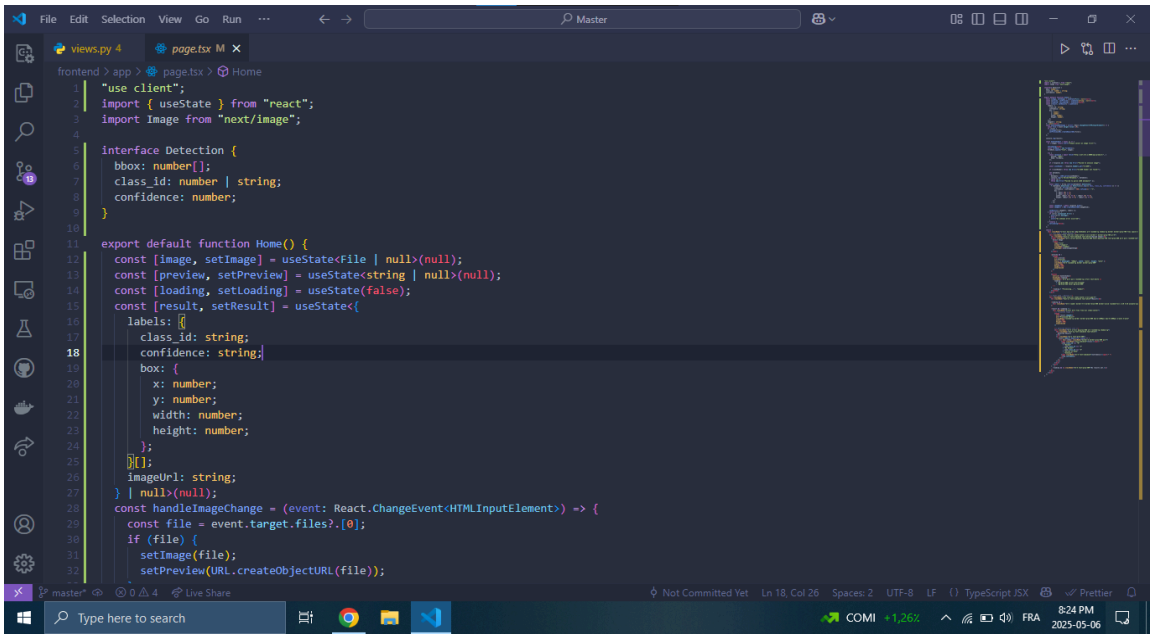


Figure 4.4. frontend code example

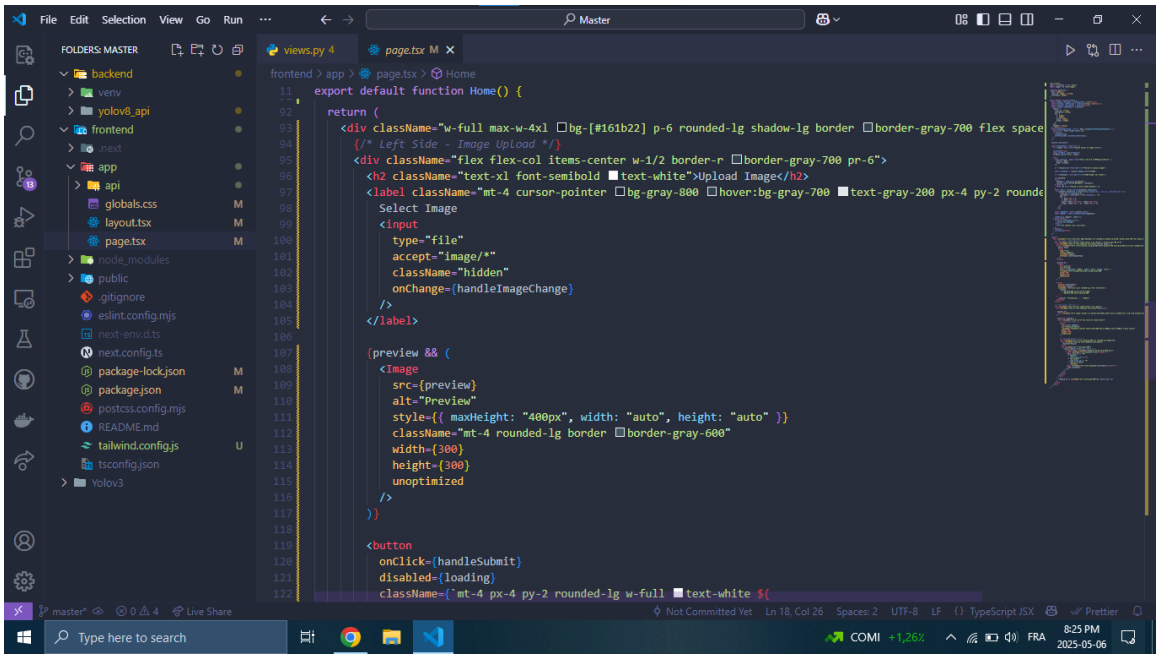


Figure 4.5. frontend code example 2

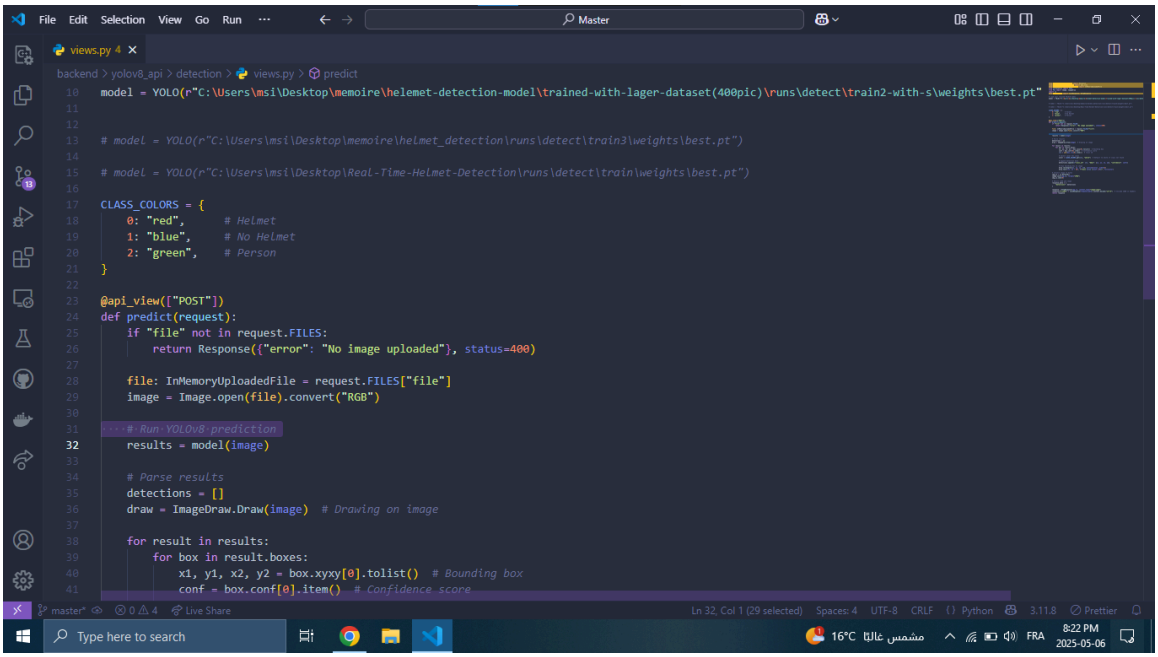


Figure 4.6. backend code example

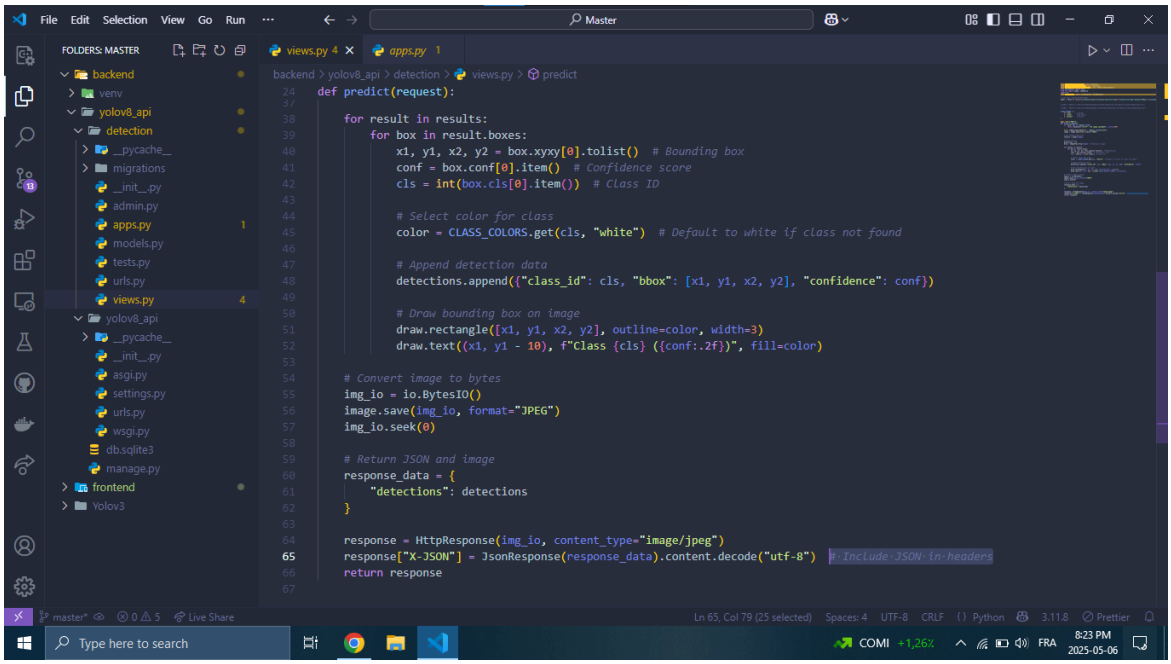


Figure 4.7. backend code example 2