UNIVERSITY OF KASDI MERBAH OUARGLA

FACULTY of New Information Technologies and Communication

DEPARTMENT of Electronic and Telecommunication

ACADEMIC MASTER Thesis

Domain : Science and Technology

Faculty : Electronics

Specialty : Electronics of Embedded Systems

Presented by :

❑ Benhammouda Salsabil

❑ Aoun Douaa

Topic

Deep Learning-Based Car Damage Detection using YOLOv8

Submit Date : June 2026

before the jury :

❑ Mr. President UKM OUARGLA

❑ Mr. Examiner UKM OUARGLA

❑ Mr. Azeddine Benlamoudi Supervisor UKM OUARGLA

❑ Mr.lyes Benaissa Supervisor University Del Salento

University year : 2025/2026

Abstract

This thesis presents a deep learning-based approach for automatic car damage detection, classification, and segmentation using computer vision techniques. In recent years, deep learning has emerged as a powerful solution for automating visual inspection tasks, especially in the automotive sector where accurate damage assessment is essential for insurance, maintenance, and evaluation processes. The proposed work focuses on identifying and localizing different types of vehicle damages, including scratches, dents, broken glass, cracked parts, and damaged lights. To achieve this, state-of-the-art deep learning models such as YOLO, and DETR are implemented and analyzed. These models are widely used in object detection and image segmentation tasks due to their high performance and adaptability. The system is trained and evaluated using a custom dataset composed of Algerian car images, which reflects real-world conditions and adds a local context to the study. This dataset includes various damage annotations that allow for both detection and segmentation tasks. The main objective of this research is to compare the performance of different models using standard evaluation metrics such as mean Average Precision (mAP) and Intersection over Union (IoU). Furthermore, the study investigates how the use of a local dataset influences model accuracy and robustness. Ultimately, this work aims to contribute to the development of an efficient and practical system for automated vehicle damage assessment that can be applied in real-world scenarios.

Keywords— Deep Learning, Computer Vision, Car Damage Detection, Object Detection, Image Segmentation, YOLO, DETR, mAP, IoU, Transfer Learning, Algerian Dataset.

ملخص

يقدم هذا العمل نظامًا ذكيًا للكشف الآلي عن أضرار السيارات باستخدام تقنيات الرؤية الحاسوبية (Computer Vision) والتعلم العميق (Deep Learning)، بهدف المساهمة في تطوير حلول دقيقة وفعالة لدعم عمليات تقييم الأضرار في مجالات التأمين، مراكز الصيانة، والخبرة التقنية، خاصة مع تزايد الاعتماد على الأنظمة الذكية في أتمتة المهام المعقدة.

تعتمد المنهجية المقترحة على إنشاء مجموعة بيانات مخصصة تحتوي على صور لسيارات جزائرية تم جمعها ومعالجتها خصيصًا لهذا البحث، ثم تنظيمها وفق تنسيق (Common COCO in Objects Context)، وهو من أكثر التنسيقات استخدامًا في مهام كشف الأجسام. تضم هذه المجموعة ست فئات رئيسية من الأضرار الشائعة في المركبات، وهي: الخدوش (Scratch)، الانبعاجات (Dent)، التشققات (Crack)، الزجاج المكسور (Broken Glass)، المصابيح المكسورة (Broken Light)، والإطارات المثقوبة (Flat Tire). وقد خضعت البيانات لعمليات دقيقة تشمل الوسم (Annotation)، التنظيف، والتحقق، ثم تقسيمها إلى مجموعات تدريب، تحقق، واختبار لضمان جودة التجارب وموثوقية النتائج.

من أجل تنفيذ مهام الكشف، تم اعتماد ومقارنة نموذجين حديثين من نماذج كشف الأجسام، هما YOLO (You Look Only Once) المعروف بسرعته العالية وقدرته على العمل في الزمن الحقيقي، وDETR (Detection Transformer) الذي يعتمد على بنية المحولات (Transformers) ويوفر قدرة قوية على فهم العلاقات المكانية داخل الصورة واكتشاف الأجسام بطريقة شاملة. كما تم توظيف تقنيات التعلم بالنقل (Transfer Learning) والضبط الدقيق (Fine-Tuning) للاستفادة من النماذج المدربة مسبقًا وتحسين أدائها على البيانات المحلية.

تم تقييم أداء النماذج باستخدام مجموعة من المقاييس القياسية المعتمدة في مجال كشف الأجسام، تشمل الدقة (Precision)، الاسترجاع (Recall)، ومتوسط الدقة (mAP)، بالإضافة إلى تحليل منحنيات الخسارة

تؤكد هذه الدراسة أن تقنيات الذكاء الاصطناعي والتعلم العميق توفر حلولًا واعدة لبناء أنظمة ذكية للكشف الآلي عن أضرار السيارات، مع إمكانية تطبيقها في الواقع العملي. كما تفتح المجال لأعمال مستقبلية تهدف إلى تحسين الأداء من خلال توسيع قاعدة البيانات، زيادة تنوع العينات، اعتماد نماذج أحدث، ودمج تقنيات تقسيم الصور (Image Segmentation) للحصول على دقة أعلى في تحديد الأضرار..

الكلمات المفتاحية: أضرار السيارات، التعلم العميق، كشف الأجسام، YOLO، DETR، الرؤية الحاسوبية، COCO، التعلم بالنقل، Precision، Recall. mAP

Resumé

Ce travail présente un système intelligent de détection et de localisation des dommages de véhicules à l'aide des techniques de vision par ordinateur et d'apprentissage profond. L'approche proposée repose sur la création d'une base de données personnalisée d'images de véhicules algériens, annotée au format COCO, comprenant six catégories principales de dommages : rayure (scratch), bosse (dent), fissure (crack), vitre brisée (broken glass), phare cassé (broken light) et pneu crevé (flat tire). Pour la détection des dommages, des modèles modernes de détection d'objets, notamment YOLO et DETR, sont utilisés afin d'assurer une localisation précise des différents types de défauts. Le système adopte une approche de détection directe basée sur l'image, permettant une analyse efficace de scènes réelles complexes. Les performances des modèles sont évaluées à l'aide de métriques standard telles que la Précision (Precision), le Rappel (Recall) et la Précision Moyenne Moyenne (mAP - mean Average Precision). Les résultats expérimentaux démontrent la capacité des modèles étudiés à détecter et à localiser les dommages de véhicules dans diverses conditions réelles. Les résultats obtenus sont globalement satisfaisants et confirment le potentiel des techniques d'apprentissage profond pour la mise en œuvre de systèmes automatisés de détection des dommages automobiles. Ils mettent également en évidence l'importance de la qualité des données dans l'amélioration de la robustesse des modèles. Des améliorations futures pourront être envisagées grâce à l'extension et à la diversification de la base de données.

Keywords— : dommages de véhicules, vision par ordinateur, apprentissage profond, détection d'objets, YOLO, DETR, COCO, apprentissage par transfert, précision, rappel, mAP

ACKNOWLEDGMENTS

In the Name of Allah, the Most Gracious, the Most Merciful. All praise is due to Allah, through whose blessings good deeds are completed and by whose guidance objectives are achieved. Peace and blessings be upon Prophet Muhammad, his family, and his companions. I would like to express my sincere gratitude to my supervisors, Dr. Azeddine Benlamoudi and Dr. Ilyes Benaissa, for their rigorous academic supervision, valuable guidance, continuous support, and insightful advice, which greatly contributed to the completion and quality of this work. I would also like to extend my deepest appreciation to the Dean of the Faculty, Dr. Benarabi Bilal, for his support, guidance, and academic leadership, which helped create a motivating environment that facilitated the accomplishment of this work. My sincere thanks also go to all the professors who accompanied me throughout my academic journey, for their dedication to teaching and for their essential role in shaping my scientific knowledge and academic development. I am profoundly grateful to my beloved family, especially my parents, for their unconditional support, constant encouragement, and great patience, which, after Allah, were the main driving force behind reaching this stage. Finally, I would like to thank everyone who contributed to this work, directly or indirectly, even through a kind word or sincere advice.

DEDICATIONS

I dedicate this thesis to: To myself To that soul which stumbled but did not fall, which endured hardships and remained patient through a long journey, until it reached this moment by the grace of God.

TTo my father My first support and role model in life, who never stopped guiding and supporting me, and whose presence was a source of my strength and perseverance.

To my mother The heartbeat and refuge of the soul, who never ceased to pray for me and to surround me with love and tenderness, whose prayers were the secret of my success after God.

To my siblings… My companions in this journey and my source of support and reassurance, who shared both hardship and joy, and were always by my side in every step.

And to everyone who supported me, even with a kind word, a sincere prayer, or a simple encouragement… I extend my deepest gratitude and appreciation to all of you.

With sincere love, appreciation, and gratitude to all.
Benhammouda Salsabil
Aoun Douaa

Contents

Abstract i

Dedication iv

List of Abbreviations xiii

General Introduction 1

1 Literature Review 4

1.1 Computer Vision and Object Detection 4

1.1.1 Definition of computer vision 4

1.1.2 Definition and methodology of object detection 5

1.2 Object Detection Evolution 8

1.3 YOLO-Based Object Detection 14

1.3.1 Overview of YOLO architecture 14

1.3.2 YOLO Real-time detection principle 15

1.3.3 Advantages of YOLO models 17

1.3.4 Overview of Transformer-Based architecture 17

1.3.5 Transformer-Based Real-time detection principle 18

1.3.6 Advantages and limitations of Transformer-Based models 18

1.4 Challenges 19

1.5 Car Damage Detection in Previous Research 20

1.5.1 Detection and Classification Approaches 20

1.5.2 Datasets, Challenges, and Research Limitations 21

2 Methodology 24

2.1 Dataset Collection 24

2.2 CARDD Dataset . . . 27
2.3 COCO Format . . . 28
2.4 Roboflow Platform and Dataset Preparation . . . 30
2.4.1 Roboflow Workspace Setup . . . 30
2.4.2 Dataset Uploading and Management . . . 31
2.4.3 Annotation Process on Roboflow . . . 31
2.4.4 Annotation Guidelines . . . 32
2.4.5 Roboflow Preprocessing . . . 32
2.4.6 Roboflow Data Augmentation . . . 33
2.4.7 Dataset Export . . . 34
2.5 Models . . . 34
2.5.1 Model Selection Criteria . . . 34
2.5.2 YOLO-Based Object Detection . . . 35
2.5.3 DETR . . . 35
2.5.4 Faster R-CNN . . . 36
2.5.5 Comparison Between YOLO , Faster R-CNN and DETR . . . 36
2.5.6 Training Strategy . . . 37
2.5.7 Training Parameters . . . 37
2.5.8 Model Output . . . 38
2.6 Evaluation . . . 39
2.6.1 Evaluation Objective . . . 39
2.6.2 Evaluation Metrics . . . 39
2.6.3 Quantitative Comparison . . . 40
2.6.4 Qualitative Evaluation . . . 41
2.6.5 Failure Case Analysis . . . 41

3 Results and Discussion 42

3.1 Dataset Analysis . . . 42
3.1.1 Dataset Summary . . . 42
3.1.2 Damage Classes . . . 43
3.1.3 Scratch . . . 43
3.1.4 Dent . . . 44
3.1.5 Crack . . . 45

3.1.6 Broken Glass . . . 45
3.1.7 Broken Lamp . . . 46
3.1.8 Flat Tire . . . 46
3.1.9 Class Distribution Analysis . . . 47
3.1.10 Impact of Class Imbalance . . . 48
3.2 Training Results . . . 49
3.2.1 Training Configuration Summary . . . 49
3.2.2 YOLO Training Results . . . 49
3.2.3 DETR Training Results . . . 50
3.2.4 Faster R-CNN Training Results . . . 51
3.2.5 Training Loss Analysis . . . 51
3.2.6 Overfitting and Underfitting Analysis . . . 52
3.3 Quantitative Evaluation . . . 52
3.3.1 Evaluation Metrics . . . 52
3.3.2 Overall Performance Comparison . . . 53
3.3.3 Precision Analysis . . . 53
3.3.4 Recall Analysis . . . 53
3.3.5 Mean Average Precision Analysis . . . 53
3.4 Per-Class Evaluation . . . 54
3.4.1 Per-Class Performance . . . 54
3.4.2 Scratch Detection . . . 54
3.4.3 Dent Detection . . . 54
3.4.4 Crack Detection . . . 55
3.4.5 Broken Glass Detection . . . 55
3.4.6 Broken Lamp Detection . . . 55
3.4.7 Flat Tire Detection . . . 55
3.5 Qualitative Evaluation . . . 55
3.5.1 Correct Detection Examples . . . 55
3.5.2 Missed Detection Examples . . . 56
3.6 Model Comparison . . . 57
3.6.1 Accuracy Comparison . . . 57
3.6.2 Speed Comparison . . . 57

3.6.3 Best Model Selection . . . 58
3.7 Error Analysis . . . 58
3.7.1 Common Error Types . . . 58
3.7.2 Causes of Detection Errors . . . 59
3.7.3 Effect of Algerian Environmental Conditions . . . 59
3.8 Discussion . . . 59
3.8.1 Interpretation of Results . . . 59
3.8.2 Impact of Dataset Size . . . 60
3.8.3 Impact of Annotation Quality . . . 60
3.8.4 Practical Relevance . . . 60
3.8.5 Limitations . . . 61
3.9 Chapter Summary . . . 61

4 Economic Feasibility Study **63**

Economic Feasibility Study **64**
4.1 Introduction . . . 64
4.2 Added Value . . . 64
4.3 Target Market in Algeria . . . 65
4.4 Competitive Analysis . . . 65
4.5 Strategic Partnerships . . . 66
4.6 Economic Model . . . 66
4.7 Conclusion and Economic Findings . . . 66

General Conclusion and Future Work **68**

Bibliography **70**

List of tables

1.1 Chronological Comparison of YOLO Variations and Key Technical Enhancements 11

2.1 Defined Car Damage Categories in the Custom Dataset. 25
2.2 Main Characteristics of the CARDD Dataset 28
2.3 Training Hyperparameters and Optimal Experimental Configurations . . 38
2.4 Performance Evaluation Metrics and Technical Definitions 40

3.1 Class distribution of annotated instances in the Algerian car damage dataset 48
3.2 Overall Performance Comparison Between Detection Models. 53
3.3 Accuracy and Speed Comparison. 58

4.1 Added Value Pillars and Quantitative Impact 64
4.2 Target Market Segments 65
4.3 SWOT Analysis of the Project 65
4.4 Strategic Partnership Benefits 66

List of figures

1 Effect of image processing on a frog image, showing enhanced clarity and extracted features.[1] 1

1.1 Example of Bounding-Box detection 7
1.2 Object Detection using YOLO Algorithm[2] 9
1.3 Faster R-CNN architecture with Region Proposal Network (RPN) [3] . . 12
1.4 Architecture of the DETR model and its object detection pipeline [4] . . 13
1.5 Overview of the YOLO Architecture[5] 15
1.6 The YOLO detection system pipeline, illustrating the unified single-pass regression approach from an $S \times S$ grid to final detections [6]. 16

2.1 Stacked bar chart of the dataset class distribution. 26
2.2 Examples of the six damage categories in the CARDD dataset (dent, scratch, crack, glass shatter, lamp broken, and flat tire). 28
2.3 Examples of object annotations in the MS COCO dataset[7] 29

3.1 Example of scratch damage on a vehicle. 44
3.2 Example of dent damage on a vehicle 44
3.3 Example of crack damage on a vehicle. 45
3.4 Example of broken glass damage on a vehicle 46
3.5 Example of broken lamp damage on a vehicle 46
3.6 Example of flat tire damage on a vehicle 47
3.7 Distribution of car damage classes. 49
3.8 Dataset splitting architecture for car damage detection. 49
3.9 YOLOv11 training results displaying loss convergence curves and overall performance 50

3.10 RF-DETR training results displaying loss convergence curves and overall performance 51
3.11 Average Precision by class(YOLO) 54
3.12 Average Precision by class(DETR) 54
3.13 Examples of correct damage detections. 56
3.14 Examples of Missed Detection. 57
3.15 Common error types in vehicle damage detection 58
3.16 Causes of detection errors in vehicle datasets 59
3.17 Effect of Algerian environmental conditions on detection performance . . 60

List of Abbreviations

AI: Artificial Intelligence

YOLO: You Only Look Once

CNN: Convolutional Neural Network

R-CNN: Region-based Convolutional Neural Network

DETR: Detection Transformer

COCO: Common Objects in Context

HOG: Histogram of Oriented Gradients

SIFT: Scale-Invariant Feature Transform

mAP: Mean Average Precision

AP: Average Precision

CarDD: Car Damage Detection Dataset

F1-Score: Harmonic Mean of Precision and Recall

FP: False Positive

FN: False Negative

TP: True Positive

GIoU: Generalized Intersection over Union

IoU: Intersection over Union

JSON: JavaScript Object Notation

KNN: K-Nearest Neighbors

MLOps: Machine Learning Operations

NMS: Non-Maximum Suppression

NLP: Natural Language Processing

ROI/RoI: Region of Interest

SGD: Stochastic Gradient Descent

VOC: Visual Object Classes

BBR: Bounding Box Regression

SURF: Speeded-Up Robust Features

GPU: Graphics Processing Unit

RPN: Region Proposal Network

ROI: Region of Interest

SAE: Society of Automotive Engineers

GNSS: Global Navigation Satellite System

IMUs: Inertial measurement units

SSL: Solid-state LiDAR

FN: False Negatives (missed targets)

FP: False Positives (wrong detections)

IDSW: Identity Switches (when a tracker switches from one ID to another)

GT: Ground Truth object count

TP: True Positive

TN: True Negative

TPR: True Positive Rate

TNR: True Negative Rate

General Introduction

Humans possess an extraordinary ability to instantly recognize objects in an image, identify their locations, and understand their interactions. This remarkable capability stems from the efficiency and accuracy of the human visual system. Inspired by this ability, researchers in Artificial Intelligence (AI) aim to replicate similar visual understanding in machines through the field of Computer Vision.

Computer Vision enables machines to analyze and interpret visual information, allowing them to perform complex tasks such as object detection, image classification, scene understanding, and autonomous navigation. In recent years, the rapid advancement of Deep Learning techniques has significantly improved the performance of computer vision systems, making them more accurate, efficient, and capable of handling challenging real-world environments. Consequently, these technologies have found applications in numerous domains, including intelligent transportation systems, surveillance, healthcare, and automated inspection [6]. This figure demonstrates how image processing enhances visual clarity ,as shown in Figure 1.

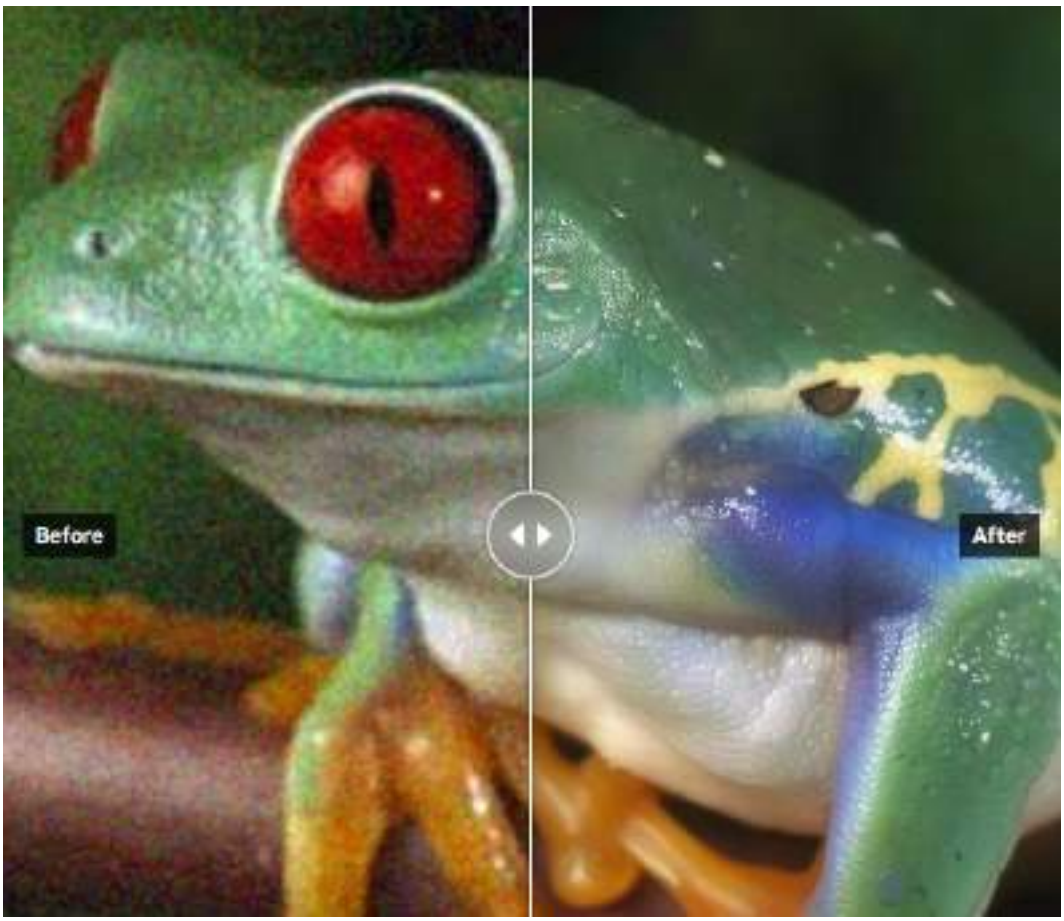


Figure 1: Effect of image processing on a frog image, showing enhanced clarity and extracted features.[1]

Among the various applications of computer vision, vehicle damage assessment has attracted considerable attention due to its importance in the automotive industry, insurance services, fleet management, and vehicle maintenance. Traditionally, damage inspection relies on manual evaluation performed by human experts. However, this process is often time-consuming, labor-intensive, costly, and subject to human errors and inconsistencies. Automated vision-based solutions offer an attractive alternative by enabling fast, objective, and reliable damage assessment from vehicle images.

Despite the remarkable progress achieved in object detection and image segmentation, vehicle damage analysis remains a challenging problem. Vehicle damages exhibit significant variations in appearance, size, shape, texture, and severity. Furthermore, environmental factors such as illumination changes, shadows, reflections, occlusions, complex backgrounds, and different camera viewpoints can negatively affect the performance of deep learning models. These challenges become even more pronounced when dealing with real-world vehicle images collected under unconstrained conditions.

Another important limitation is the availability of suitable datasets. Most publicly available vehicle damage datasets are either relatively small, imbalanced, or collected under controlled environments, which restricts their ability to generalize to real-world scenarios. In addition, these datasets rarely represent the characteristics of vehicles commonly found in Algeria. As a result, models trained on foreign datasets often suffer from domain shift when applied to local vehicle images, leading to degraded performance.

To address these limitations, this work focuses on the development of a customized dataset of Algerian vehicle images annotated according to the COCO format. The dataset includes six common damage categories: dent, scratch, crack, broken glass, flat tire, and broken light. Based on this dataset, several state-of-the-art deep learning models for object detection and instance segmentation are investigated and evaluated in order to identify effective solutions for robust vehicle damage assessment under realistic conditions.

The main objective of this research is to develop an automated computer vision framework capable of accurately detecting, classifying, and segmenting vehicle damages. Such a system can contribute to improving inspection efficiency, supporting insurance claim automation, facilitating vehicle condition assessment, and enabling intelligent fleet management solutions.

The main contributions of this work can be summarized as follows:

- Construction of a dedicated COCO-format dataset containing real-world images of Algerian vehicles with multiple damage categories. - Development of a structured annotation pipeline to ensure consistency and quality throughout the labeling process. - Training and evaluation of state-of-the-art object detection and instance segmentation models using standard metrics such as mAP and IoU. - Comparative analysis of different approaches to identify the most suitable model for vehicle damage assessment in realistic environments. - Investigation of the challenges associated with local vehicle damage detection and the impact of a domain-specific dataset on model performance.

The remainder of this dissertation is organized as follows. Chapter 1 presents the theoretical background and related work in computer vision, deep learning, object detection, and instance segmentation. Chapter 2 describes the proposed methodology, dataset construction process, and experimental setup. Chapter 3 presents the obtained results and performance evaluation. Finally, the conclusion summarizes the main findings and discusses potential directions for future work.

Chapter 1

Literature Review

This chapter reviews the literature related to object detection,explaining its evolution from traditional methods to modern approaches.It also addresses the main challenges faced by object detection systems,including background noise,difficulties in object classification,and challenges associated with small object detection.

1.1 Computer Vision and Object Detection

1.1.1 Definition of computer vision

Computer vision allows machines to understand their surroundings through digital image processing. In this way, they can recognize objects and navigate easily. To achieve this, computer science, artificial intelligence, and image analysis are combined to extract valuable information from images. This field plays an important role in many applications, such as robotics and mobile devices. Real-time computer vision has achieved remarkable results and opened new horizons for scientific research.However, there are some difficulties facing computer vision research. One of the biggest challenges is developing algorithms capable of accurately detecting objects, segmenting images, and classifying them. Each of these tasks requires a deep understanding of mathematics and computational logic. Furthermore, each presented challenge requires a different approach, which increases the complexity of the research. In order to develop artificial intelligence and provide practical solutions, researchers must combine theoretical knowledge and practical skills to make real progress in this field [8].

1.1.2 Definition and methodology of object detection

Identifying objects in images is fundamental for many computer vision tasks. It is used in instance segmentation, automatic image captioning, object tracking, and much more. The field of object detection has seen significant advancements in recent years thanks to deep learning techniques. These developments have led to significant breakthroughs and made object detection a research field of great interest. It is now used in real-world applications such as self-driving, robotic vision, and video surveillance [9].

To make these developments happen we group deep learning architectures into two types based on how they process information: two-stage models and single-stage models. Two-stage models, like Faster R-CNN work in two steps. First, they find areas in an image where objects are likely to be. Then they classify these areas focusing on getting results even if it takes a bit longer. Single-stage models, such as the YOLO family do object detection in one step. They look at the image and try to find objects and their classes all at once. This makes them much faster which is great for applications like checking and evaluating vehicle damage in time. We use two-stage models and single-stage models for needs. Two-stage models are good when accuracy is more important. Single-stage models, like YOLO are good when speed is key [10]. The YOLO model helps in vehicle damage inspection and assessment. The Faster R-CNN model is also used for object detection.

Deep learning for object detection

Deep learning has transformed object detection from hand-crafted features to end-to-end trainable systems that learn directly from data. Modern detectors can accurately locate and classify many object categories in real time and are used in security, medicine, autonomous driving, and robotics. Deep Convolutional Neural Networks (CNNs) learn hierarchical features ranging from low-level pixels to high-level semantics, effectively replacing traditional manual features like HOG and SIFT. This yields more robust and discriminative representations, especially when dealing with complex scenes, occlusion, and scale changes, while scaling exceptionally well with larger amounts of data. Recently, the integration of Transformers and attention mechanisms has further improved global context modeling, pushing object detection into a new era cite zaidi

Bounding-Box detection

In modern object detection, "bounding-box detection" usually means predicting precise rectangles (or oriented boxes) that localize objects, and training models so these boxes closely match ground-truth annotations. The papers focus on two things: how boxes are predicted/parameterized, and how loss functions and architectures make this regression more accurate and efficient. Bounding box regression refines a predicted box, initialized from an anchor or a reference point, to ensure it closely matches the target object, making it a crucial step in object localization[11]. Typical parameterizations utilize either the coordinates of the box corners `(x1, y1, x2, y2)` or its center and size `(x, y, w, h)` to regress the corresponding offsets relative to an anchor or a reference box[12]. While these parameterizations establish the foundational geometry of the boxes, maximizing localization accuracy heavily relies on the design of the loss function. Advanced IoU-based improvements such as GIoU, EIoU, Focal-EIoU, N-IoU, AIoU, MPDIoU, BIoU, SIoU, CFIoU, Shape-IoU, and DAPIoU each add specific penalty terms for distance between centers or corners, side-length mismatch, shape/scale, or direction/angle, aiming for faster convergence and higher mAP on COCO, VOC, VisDrone, SODA-D, and other standard benchmarks.In fact, several works demonstrate a consistent +1–4% increase or more in mAP by simply swapping the bounding box regression (BBR) loss, even when deployed on strong backbones such as YOLOv7, Mask R-CNN, and RetinaNet[13].Furthermore, some methods dynamically adapt the loss formulation to focus on hard or high-quality examples, utilizing focal-style regression parameters or specialized penalty terms that explicitly emphasize difficult objects and small targets.



Figure 1.1: Example of Bounding-Box detection

One-stage and two-stage detectors

Object detectors in deep learning are mainly divided into one-stage and two-stage architectures. They differ in how they generate candidate regions, and this leads to a core trade-off between speed and accuracy.

Two-stage detectors divide the object detection task into two sequential steps to achieve high localization accuracy. First, a region proposal network scans the input image to identify potential areas of interest where objects might exist. Then, a separate downstream network processes these candidate regions to determine the specific object class and fine-tune the bounding box coordinates. While this method, exemplified by frameworks like Faster R-CNN, excels at capturing small or complex objects, its two-step nature makes it computationally heavy and slower[10].

In contrast, one-stage detectors eliminate the region proposal step entirely, treating object detection as a direct, unified regression problem. These models process the entire image in a single forward pass, predicting both bounding box coordinates and class probabilities simultaneously. Frameworks such as the YOLO series utilize this streamlined architecture to achieve ultra-fast inference speeds suitable for real-time applications, while modern iterations have continuously optimized their designs to successfully close

the accuracy gap with two-stage designs[10].

1.2 Object Detection Evolution

Damage detection was a challenging task prior to the development of modern deep learning–based techniques. In the past,inspection was performed manually by experts or through simple image processing methods that relied on basic features such as edges,color variations,texture,and shape.However,these approaches often produced unsatisfactory results and faced several limitations

On the one hand,manual inspection was time-consuming and subject to variability between inspectors.On the other hand,traditional methods struggled to achieve accurate results due to several factors, including poor lighting conditions, the inability to detect small-scale damage, and sensitivity to variations in viewing angles.These limitations made it difficult to accurately localize damage.

Over time, object detection methods have evolved from traditional approaches such as HOG, SIFT, SURF, and KNN to modern approaches based on deep learning, with the aim of identifying damage locations and types within images, such as YOLO, Faster R-CNN, and DETR. Object detection is considered a fundamental task in computer vision[14].

YOLO models :

When humans look at an image, they can quickly determine what things are there, where they are, and how they interact. The human visual system is quick and precise, enabling us to carry out difficult activities like driving with minimal conscious thinking.Quick and precise object identification algorithms would make it possible for computers to operate automobiles without the need for specialized sensors, enabling assistive technology to provide human users with real-time scene information, and open the door to the development of responsive, general-purpose robotic systems.[6]. The YOLO algorithm was developed to address the limitations of traditional object detection methods. It is one of the most widely used algorithms for object detection because of its high processing speed. The name stands for 'You Only Look Once' indicating that the image is processed through the neural network a single time to simultaneously predict the location

and class of objects unlike traditional approaches that require multiple passes over the image. YOLO treats the detection task as a single regression problem,where it processes the entire image by dividing it into a grid of cells each cell predicting whether an object is present within it. If an object is detected, it outputs: The class of the object. The coordinates of its position. A confidence score. The object's position is typically represented by four values `(x, y, w, h)` where x and y denote the center of the object and w and h specify the width and height of the bounding box. Finally, the algorithm draws a bounding box around the damage as illustrated in Figure 1.2.

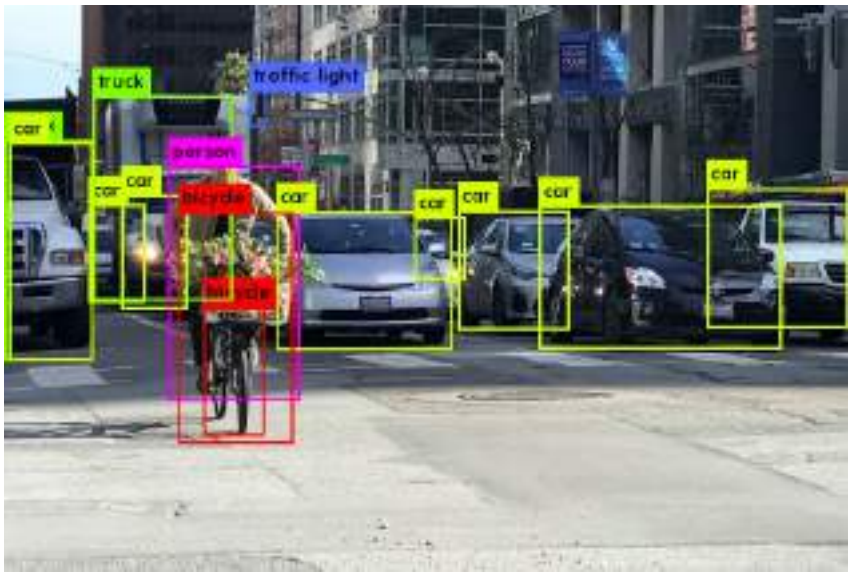


Figure 1.2: Object Detection using YOLO Algorithm[2]

The YOLO algorithm has changed a lot over the years with new versions. Each new version of the YOLO has made the YOLO better at finding objects accurately. This is why the YOLO is so popular and widely used for finding objects in pictures and videos. The first version of the YOLO algorithm, which is called YOLOv1 did things differently. The YOLOv1 looked at the picture just once to find objects and say what they are. This made the YOLOv1 algorithm very fast and good, for using in time. The YOLOv1 had some problems. It had trouble finding objects and saying exactly where objects are when they are close together[6]. To get around these problems YOLOv2 was made with some things that made it better at finding objects and more stable. This new version used some ways of doing things like anchor boxes and batch normalization, which made the model better at finding objects of all sizes and made training easier. YOLOv2 could also handle pictures and was better at finding important things, which made it more reliable.[15] On

YOLOv3 made the algorithm even better by using the Darknet-53 backbone network, which was really good at finding important things and still worked fast. One of the things about YOLOv3 was that it could find objects of all sizes, which meant it could find small things, medium things and big things more easily. This version of YOLOv3 was good because it was fast and accurate which made it useful for a lot of world things. YOLOv3 was an improvement, over YOLOv2. It was because of the YOLOv3 that object detection became more practical[16]. Subsequently, YOLOv4 had some tweaks that made it much faster and more accurate.It used the CSPDarknet setup and some smart ways to train like adding data and better optimization methods. These tweaks helped YOLOv4 work well even on computers that weren't super powerful which made it more useful in real-life situations[17]. Then YOLOv5 became very popular because it was easy to use worked and was simple. It was easy to set up. Could run on different devices and it still had great detection results. People started using YOLOv5 a lot in industries and research because it was flexible and could work in time. YOLOv5 was really good, at what it did and YOLOv4 was still a strong competitor YOLOv4 and YOLOv5 both had their strengths.The YOLO algorithm has gotten better with time. New versions like YOLOv7 and YOLOv8 have really improved how well they can find objects. They are more accurate. Can do things faster. They are also better at working with computers to see things[18]. They can do more than just find objects. They can also help with things like figuring out what is in a picture, finding the pose of something, and seeing how things are separated. The YOLO algorithm is very good at handling situations and complicated scenes. It can do things with precision. The YOLO algorithm has changed a lot over time. This has made it one of the models for finding objects with computers. The YOLO algorithm is used in things now. It is used in self driving cars, looking at pictures, watching people with cameras, robots finding damage and separating pictures. This is because The YOLO algorithm demonstrated high performance in object detection tasks due to its strong balance between speed and accuracy. As a result, YOLO has become one of the most widely used object detection models in modern computer vision applications.

Table 1.1: Chronological Comparison of YOLO Variations and Key Technical Enhancements

Version	Year	Backbone	Key Innovations & Technical Enhancements
YOLOv1	2016	Darknet-24	First single-stage detector; unified detection as a single regression task.
YOLOv2	2017	Darknet-19	Added Batch Normalization, Anchor Boxes, and high-resolution classifiers.
YOLOv3	2018	Darknet-53	Introduced multi-scale predictions and Binary Cross-Entropy loss.
YOLOv4	2020	CSPDarknet53	Integrated CSP connections, Mosaic augmentation, and Bag-of-Freebies.
YOLOv5	2020	Modified CSP	Native PyTorch framework with automated anchor box calculation.
YOLOv7	2022	Extended-ELAN	E-ELAN architecture featuring structural model re-parameterization.
YOLOv8	2023	C2f Module	Anchor-free detection baseline with native multi-task learning support.

Faster R-CNN Models :

In recent years, computer vision has experienced significant and rapid progress, largely driven by the development of deep convolutional neural networks (CNNs) [19]. and the availability of large-scale annotated datasets[20]. Although early region-based CNN approaches were computationally expensive sharing convolutional computations among region proposals has significantly reduced their cost[21].These deep learning approaches have led to remarkable improvements over traditional methods in several tasks, including image classification[19] ,object detection[22],[21], and various other recognition problems. Among these techniques, Faster R-CNN has become a potent framework for object identification, incorporating a Region Proposal Network (RPN) that greatly increases accuracy and speed. Two components make up our object detection system, Faster R-CNN.The first module suggests regions using a deep fully convolutional network.and the Fast R-CNN detector is the second module, which makes use of the suggested areas[21]. The Region Proposal Network (RPN) is a key innovation of Faster R-CNN. It is a fully convolutional network that takes an image of arbitrary size as input and generates a set of rectangular object proposals,each associated with an objectness score.This mechanism enables efficient end-to-end generation of region proposals within the Faster R-CNN

framework[21].

The RPN operates by sliding a small neural network over the feature map produced by the CNN backbone CNN.At each spatial location,it predicts multiple region proposals with different scales and aspect ratios,known as anchors.

For each anchor,the network outputs two main predictions: a binary classification score indicating whether the region contains an object (objectness score), and a regression offset used to refine the coordinates and dimensions of the bounding box.

During training, anchors are labeled as positive or negative based on their Intersection over Union (IoU) with ground-truth boxes.This supervision enables the network to learn which regions are likely to contain objects and how to adjust them accurately.

By sharing convolutional features with the detection network,the RPN significantly reduces computational cost and enables end-to-end training of the entire Faster R-CNN architecture.

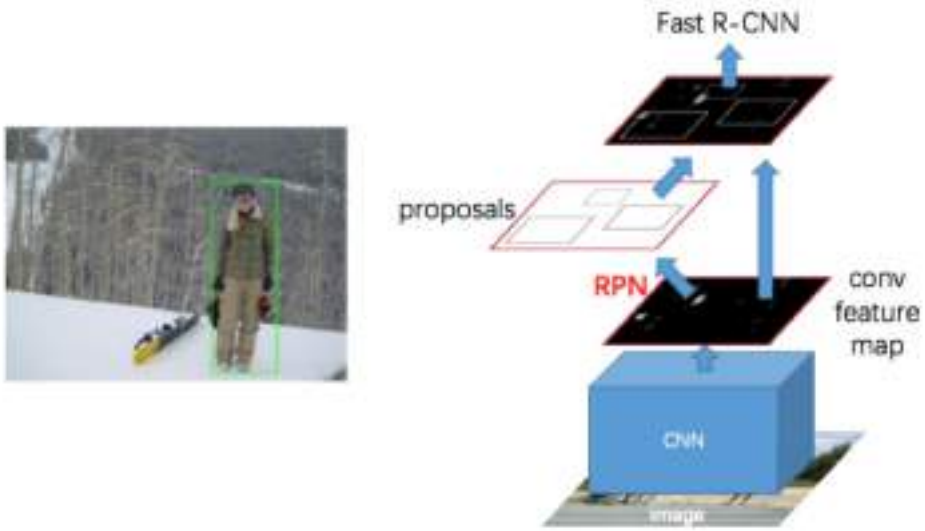


Figure 1.3: Faster R-CNN architecture with Region Proposal Network (RPN) [3]

As shown in Figure 2.2,Faster R-CNN generates region proposals, extracts feature maps using a CNN, applies ROI processing on each region, and then classifies each region into object categories with bounding box regression.

Fast R-CNN introduces the Region of Interest (ROI) Pooling layer to extract features from region proposals. A Region of Interest (ROI) refers to the most relevant regions in an image where objects are likely to be present. Instead of processing the entire image, the model focuses only on these selected regions. After generating region proposals, RoI Pooling converts each region into a fixed-size feature map, enabling uniform processing

by fully connected layers. This step is essential for linking region proposals with the final classification and bounding box regression stages[23]

DETR Models:

DETR (Detection Transformer) is a state-of-the-art object detection approach based on the Transformer architecture. It approaches object identification as a direct set prediction problem, in contrast to conventional methods like YOLO and Faster R-CNN. DETR predicts every item in an image in a single forward pass, eliminating the need for manually created components like anchor boxes or non-maximum suppression. This method streamlines the entire detection process and enables end-to-end training.

Two essential elements underpin direct set prediction in object detection: (1) a set-based loss function that requires a one-to-one matching between predicted outputs and ground-truth objects, usually accomplished through bipartite matching; and (2) a global optimization objective that guarantees all predictions are taken into account concurrently rather than separately.[4]. The DETR model's architecture and object detection process are depicted in the accompanying figure. The DETR model is made up of a pretrained

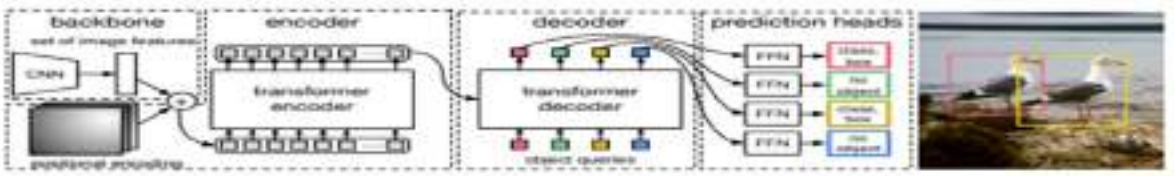


Figure 1.4: Architecture of the DETR model and its object detection pipeline [4]

CNN backbone, similar to ResNet, which produces a set of lower dimensional features. These features are then formatted into a single set and added to a positional encoding, before being fed into a Transformer, which consists of an encoder and a decoder. The decoder's output is then passed through a fixed number of Prediction Heads, each comprising feed forward networks. Each Prediction Head outputs a predicted bounding box and a class prediction.The bipartite matching loss is used to compute the loss. The model can successfully identify items and forecast their classes thanks to this procedure. The model can handle complicated situations and recognize many items at once thanks to the usage of Transformers and Prediction Heads. The DETR model produces state-of-the-art performance in object identification applications by utilizing the advantages of both CNNs and Transformers[4, 24]

1.3 YOLO-Based Object Detection

1.3.1 Overview of YOLO architecture

To understand how the YOLO algorithm works, one must look at its architecture, which is primarily divided into three interconnected main components that work together to transform the input image into real-time accurate predictions. These components are:

- **The Backbone:** The backbone is responsible for receiving the raw input image and extracting its primary features. It relies on deep convolutional neural networks (CNNs) that gradually reduce the spatial dimensions of the image while effectively retaining and encoding the most critical visual patterns, edges, textures, and shapes[5].
- **The Neck:** Acting as a bridge between the feature extraction phase and the prediction phase, the neck is designed to merge and topologize feature maps extracted from different layers of the backbone, combining both shallow spatial details and deep semantic information. This multi-scale feature fusion is critical, as it enhances the model's multi-scale representations and significantly improves its capacity to detect objects of various dimensions, ranging from extremely small features to large structures within the scene[5].
- **The head:** The head constitutes the final component of the architecture, responsible for execution during the dense prediction stage. It processes and analyzes the enriched feature maps delivered by the neck to perform the definitive detection tasks. This phase simultaneously yields the precise spatial localization of objects via bounding boxes, assigns them to their respective categories through classification, and calculates a confidence score reflecting the model's certainty for each obtained result[5].

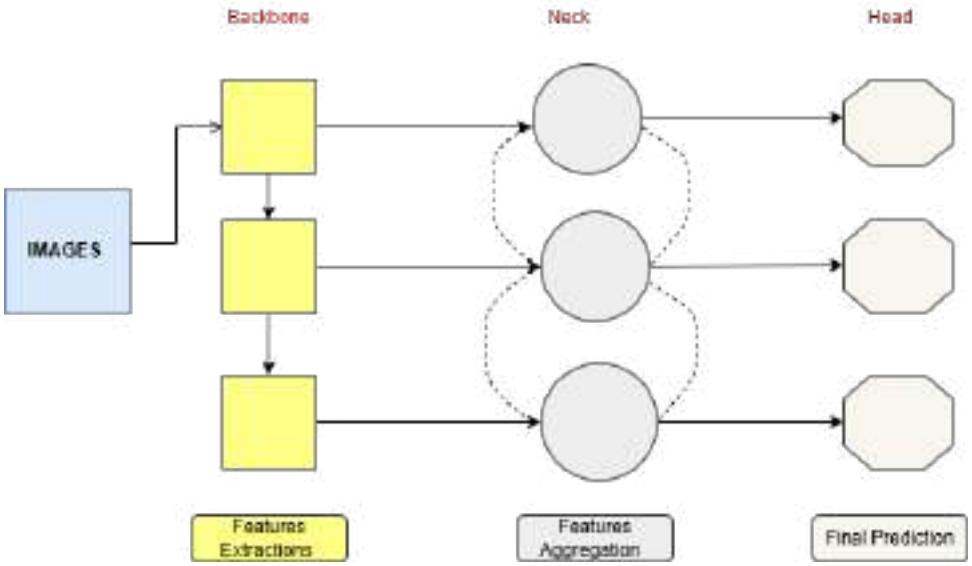


Figure 1.5: Overview of the YOLO Architecture[5]

As shown in Figure 1.5 ,the YOLO architecture is composed of three main components : the Backbone which extracts features from the input image. the Neck which fuses features at multiple scales, and the Head which predicts the bounding boxes ? class labels and confidence scores. these components work together to achieve accurte and real-time object detection.

1.3.2 YOLO Real-time detection principle

YOLO is a family of object detection models designed to detect and classify objects in images or video fast enough for real-time use, such as autonomous driving, surveillance, and robotics. Its key idea is to treat detection as a single prediction problem solved in one network pass, rather than a multi-stage pipeline. Specifically, YOLO's real-time capability comes from treating object detection as a single, end-to-end regression task over the whole image, using a one-stage CNN that predicts all boxes and classes concurrently. This unified design eliminates heavy multi-stage processing, thereby enabling high frame rates while maintaining competitive accuracy[6].

The YOLO framework is distinguished by its remarkable capacity for real-time detection, a capability that stems from three fundamental architectural principles originally introduced by Redmon et al. First, YOLO reformulates object detection as a single regression problem, wherein raw image pixels are mapped directly to bounding box coordinates and class probabilities in an end-to-end manner. Consequently, this approach

completely eliminates the need for a separate, computationally expensive region proposal pipeline. Secondly, the network operates through a single forward pass, where a unified convolutional neural network processes the entire image globally at once. This parallelized evaluation enables the model to perform feature extraction and output all bounding boxes alongside their respective class scores simultaneously, drastically reducing inference latency. Finally, the model relies on grid-based prediction, dividing the input image into a spatial grid. Each individual cell within this grid is dynamically assigned the responsibility of predicting bounding boxes and class probabilities for any object whose geometric center falls within its boundaries. Together, these interconnected principles achieve a highly synchronized alignment between spatial localization and categorical classification, granting YOLO its exceptional speed in processing visual data[6].This efficient detection process is visualized in Figure 1.6,which outlines the YOLO pipeline. it demonstrates the transition from the initial grid divison (S x S) to the generation of bounding boxes and class probabilities, culminating in the final detection results.

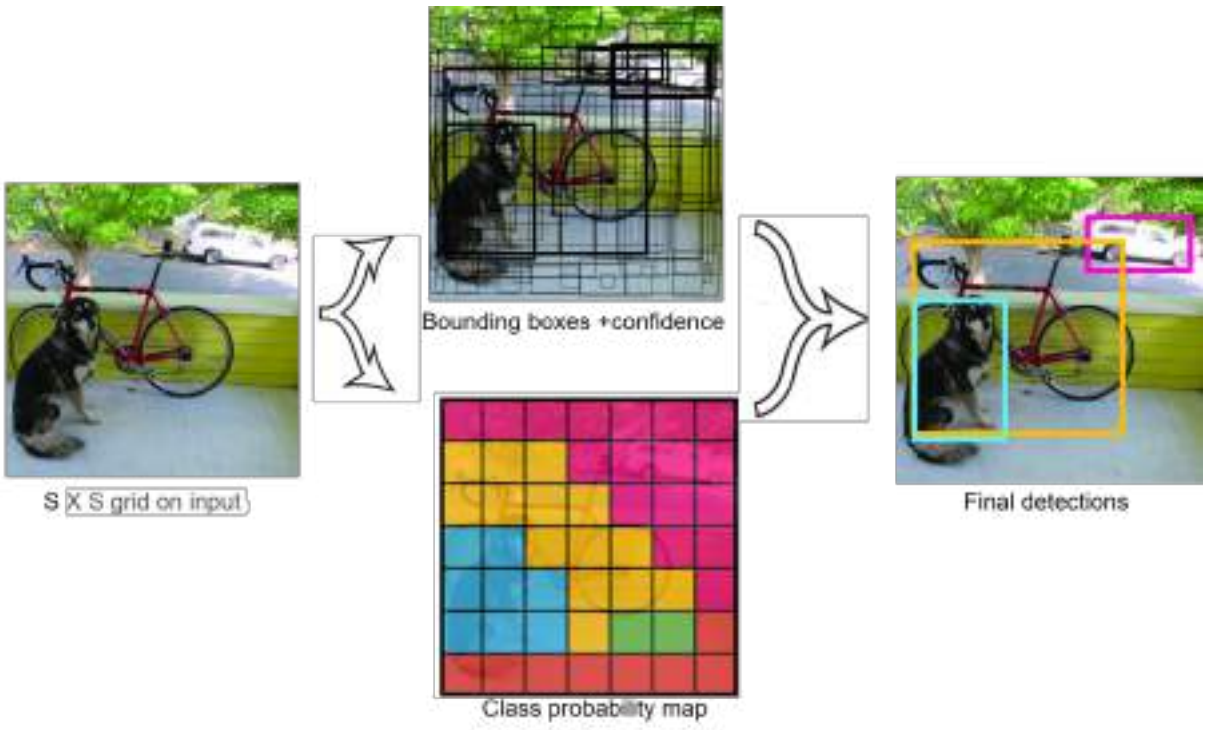


Figure 1.6: The YOLO detection system pipeline, illustrating the unified single-pass regression approach from an $S \times S$ grid to final detections [6].

1.3.3 Advantages of YOLO models

The architectural paradigm of the You Only Look Once (YOLO) framework introduces a fundamental trade-off between computational velocity and spatial precision. The primary advantage of YOLO models lies in their unified, single-shot detection principle. Unlike multi-stage detectors that utilize separate region proposal networks, YOLO frames object detection as a single regression problem, processing the entire image in a single forward pass. This global contextual reasoning enables exceptional inference speeds, making YOLO uniquely suited for real-time applications and seamless deployment on resource-constrained edge devices[6].

1.3.4 Overview of Transformer-Based architecture

The architectural paradigm of Transformers represents a monumental shift in deep learning, fundamentally substituting traditional recurrence and convolution with self-attention mechanisms. Today, these highly scalable frameworks underpin state-of-the-art models across natural language processing, computer vision, time-series analysis, and multimodal tasks. While contemporary literature establishes a foundational 'standard' Transformer architecture, recent research exhibits a distinct trajectory toward domain-specific adaptations. In visual recognition and object detection, these specialized modifications successfully optimize the sequence-based attention mechanism to process multi-dimensional grids, thereby bridging the operational gap between sequential modeling and spatial feature extraction. Ultimately, by centralizing self-attention as their core engine, Transformers have driven radical cross-domain performance enhancements and architectural evolution[25].

The architectural foundational framework of the standard Transformer is governed by three interconnected structural dimensions. First, it employs a distinct encoder-decoder structure, establishing an end-to-end sequence-to-sequence mechanism where stacked encoding layers map input tokens into high-dimensional latent vectors, which are subsequently projected by the decoding layers into target output tokens. Second, the processing engine relies heavily on self-attention and multi-head attention mechanisms; each layer dynamically computes rich contextual interactions across all token pairs globally, integrated alongside position-wise feed-forward networks, residual connections, and

layer normalization to stabilize deep gradient propagation. Third, because this parallelized structure inherently lacks recurrent loops, positional encodings—utilizing either fixed sinusoidal functions or learned embeddings—are explicitly injected into the input representations to preserve vital spatial and sequential topology[26].

Across domains, Transformer-based architectures share a common backbone: stacked encoder/decoder layers built from multi-head self-attention, feed-forward networks, residual connections, normalization, and positional encodings. On top of this, numerous variants adjust architecture depth, attention structure, and modality integration to improve efficiency and accuracy for specific tasks.

1.3.5 Transformer-Based Real-time detection principle

The core principle of Transformer-based real-time object detection relies on eliminating classical handcrafted anchors and redundant post-processing, such as Non-Maximum Suppression (NMS), to minimize inference latency. The pipeline operates by first passing the input image through an efficient Convolutional Neural Network (CNN) backbone to extract multi-scale feature maps. These visual features are then processed by a streamlined Transformer encoder-decoder, where a fixed set of learned dynamic vectors, known as object queries, scan the entire image in parallel. By performing direct set-prediction via bipartite matching, each query independently communicates with the global context to predict object categories and bounding boxes simultaneously. This hybrid approach successfully achieves high-speed, real-time performance while maintaining superior localization accuracy, making it highly suitable for resource-constrained edge deployment[27].

1.3.6 Advantages and limitations of Transformer-Based models

The integration of Transformer-based models into computer vision offers distinct advantages along with specific limitations. Primarily, its multi-head self-attention mechanism enables global contextual modeling, capturing long-range spatial dependencies across the entire image simultaneously, which traditional CNNs miss. Furthermore, modern variants eliminate handcrafted anchors and redundant Non-Maximum Suppression (NMS) pipelines, achieving direct set-prediction with low inference latency ideal for edge deployment. However, due to the absence of inductive biases, these models are inherently data-hungry and exhibit slower convergence rates during training. Additionally,

the standard self-attention layer scales with quadratic computational complexity, which induces significant memory overhead and can lead to spatial degradation, occasionally limiting the precise localization of highly fine-grained, micro-scale targets.

1.4 Challenges

Despite the significant progress made by modern models in object detection, it remains a complex task and faces a set of challenges. Among these challenges are the following:

- Detecting small objects is one of the main challenges faced by object detection models, especially when these objects occupy a very small area within the image. In the YOLO model, the image is divided into a grid, and each cell is responsible for predicting objects within its region. However, when an object is very small, a single grid cell may not contain sufficient information to represent it accurately, which can lead to missed detections or low confidence in predictions. In the faster R-CNN model, small objects are lost during the pooling operations. Despite using the region proposal mechanism, this issue remains one of the common problems. In the DETR (Detection Transformer) model, the attention mechanism is based on (global attention), which tends to focus on large and clear regions in the image, while ignoring small objects and sometimes overlooking fine details. As a result, detecting small objects becomes one of the main challenges faced by this model.
- Object detection models have a time with complex or cluttered backgrounds. This is because the extra visual information can get in the way of the model's ability to find what it is looking for. In the YOLO model, background noise can be a problem. It can confuse the grid cells. Make the model think something is there when it is not. This can lead to mistakes in where the model thinks an object is or what the object is. The Faster R-CNN model also has trouble with environments. This can make the Region Proposal Network find a lot of things that are not really there. This makes the model work harder. Makes it less accurate at finding objects. The DETR model has a problem. If the background is cluttered the model can get distracted. It starts paying attention to things that are not important instead of the object it is supposed to find. This makes it harder for the DETR model to

tell what is an object and what is background. As a result, the model finds things that are not really there, which is not what we want the YOLO model, the Faster R-CNN model, and the DETR model to do.

- lass imbalance is a common problem in machine learning and object detection, and it occurs when some classes are heavily represented in the dataset while others appear very rarely. This imbalance causes the model to become biased toward the majority classes, as it learns more from them during training, while the minority classes are not well learned. As a result, the model may perform well overall but poorly on rare categories. In object detection datasets such as the COCO dataset, this issue is very common because some objects like persons or cars appear frequently, while other objects appear much less often. This leads the model to prioritize common classes and ignore or misclassify rare ones, which reduces the overall detection performance, especially for underrepresented categories.

1.5 Car Damage Detection in Previous Research

1.5.1 Detection and Classification Approaches

Vehicle Damage Classification

Previous research in this field primarily focuses on vehicle damage classification, utilizing binary classification techniques to effectively distinguish between damaged and undamaged vehicles. Beyond this initial sorting, these methodologies aim to categorize general types of damage by leveraging advanced image classification models based on Convolutional Neural Networks (CNNs). Ultimately, the primary objective of these frameworks is to support rapid, automated assessments of vehicle conditions, thereby streamlining and simplifying the entry-level inspection processes within the transportation and insurance sectors [28].

Car Part Detection

The proposed system extends beyond mere damage classification to precisely isolate specific vehicle components, such as doors, bumpers, hoods, lights, and windshields. The significance of this phase lies in its capacity for spatial localization of the affected areas

on the vehicle's bodywork, which directly enhances the overall accuracy of the subsequent damage analysis. From a practical standpoint, this precise identification facilitates the automation of repair cost estimation and drives the development of intelligent, streamlined solutions for vehicle inspection within the automotive insurance sector[29].

Scratch, Dent, Crack, and Broken-Part Detection

The automated vehicle inspection system expands on part identification by actively classifying diverse surface anomalies, including scratches, dents, cracks, and broken components. However, robust detection remains highly challenging due to the inherently small scale and irregular geometries of these damages. Furthermore, environmental and structural noise (such as ambient lighting fluctuations, intricate shadows, surface debris, and specular reflections) can severely compromise identification accuracy. Consequently, the deployed deep learning models must possess advanced high-resolution feature extraction capabilities to effectively discern genuine structural anomalies from misleading visual artifacts.

Damage Severity Estimation

Integrating Damage Severity Estimation into the automated pipeline is paramount for evaluating vehicle degradation across distinct levels, typically spanning from minor to severe. This classification serves as a cornerstone for insurance claim processing, empowering underwriting firms to rapidly quantify both the financial liability and the structural impact resulting from an accident. By automating this evaluative phase, the system drastically accelerates claim verification cycles and mitigates the inherent subjectivity associated with manual human appraisal. Consequently, this optimization streamlines the strategic decision-making process, precisely determining whether a vehicle is economically viable for repair or should be designated as a total loss.

1.5.2 Datasets, Challenges, and Research Limitations

Datasets Used in Previous Work

Previous research in this field primarily focuses on vehicle damage classification, utilizing binary classification techniques to effectively distinguish between damaged and

undamaged vehicles. Beyond this initial sorting, these methodologies aim to categorize general types of damage by leveraging advanced image classification models based on Convolutional Neural Networks (CNNs). Ultimately, the primary objective of these frameworks is to support rapid, automated assessments of vehicle conditions, thereby streamlining and simplifying the entry-level inspection processes within the transportation and insurance sectors.

Annotation Types in Existing Datasets

Regarding the types of annotations in existing datasets, prior literature exhibits a diverse range of labeling methodologies specifically tailored to distinct computer vision tasks. These approaches vary from coarse, image-level labels designated for standard classification to precise bounding boxes utilized in object detection frameworks. To achieve more granular spatial localization, datasets increasingly incorporate polygonal annotations and semantic segmentation masks[**?**]. Ultimately, these annotations are standardized into widely accepted academic and industry formats, most notably COCO and YOLO, to facilitate efficient and structured deep learning model training.

Main Challenges Reported in Previous Studies

The main challenges documented in prior literature reveal critical obstacles that significantly hinder a model's ability to generalize effectively. First, vehicle damages frequently manifest as small-scale and irregularly shaped deformations, making feature extraction inherently difficult. This issue is further compounded by environmental and structural noise, such as specular reflections on metallic vehicle surfaces, poor ambient lighting, and low-resolution imagery. Additionally, computer vision frameworks routinely struggle with occlusion and the partial visibility of affected areas. Beyond these environmental factors, intrinsic dataset complexities impose severe learning bottlenecks; these include high inter-class similarity, where distinct damage categories appear visually identical, alongside severe class imbalance between highly frequent damage types and rare, critical distortions.

Limitations of Existing Studies

An examination of the limitations within current literature highlights several critical research gaps. The primary constraint stems from a heavy reliance on small-scale benchmarks or proprietary and restricted datasets, which inherently confines the scope of investigation to a narrow range of damage categories. Moreover, these frameworks often lack the ecological and visual diversity characteristic of real-world scenarios, leading to fragile model performance when deployed outside controlled settings. Notably, prior studies exhibit a significant absence of localized evaluations within specific regional vehicle environments, revealing weak cross-border generalization capabilities when subjected to varying road infrastructures and diverse atmospheric factors across different countries.

Research Gap

In the context of the Algerian ecosystem, these global limitations manifest as distinct research and operational challenges. Primarily, there is a severe lack of specialized Algerian vehicle damage detection datasets, underscoring an urgent need for locally collected and precisely annotated imagery that reflects domestic fleet demographics and road conditions. Consequently, evaluating existing state-of-the-art object detection models on native Algerian data is critical to establishing realistic performance baselines. To address these data engineering bottlenecks, utilizing advanced MLOps platforms, such as Roboflow, becomes paramount for streamlined dataset preparation, robust cloud-based annotation, and standardized model evaluation.

Chapter 2

Methodology

This chapter presented the methodology used for car damage detection and segmentation, including the collection and preparation of a custom Algerian car damage dataset, preprocessing techniques, model selection, training procedures, and evaluation methods.

2.1 Dataset Collection

The Algerian automotive market is characterized by a unique vehicle fleet and diverse environmental backgrounds, which often leads to a significant domain shift and reduced precision in standard deep learning models. To address this challenge, we systematically constructed a customized, domain-specific dataset tailored to the Algerian context. Through this effort, we aim to bridge the gap in localized data availability and enhance the robustness of vehicle damage detection. This section details our data collection sources, the defined damage categories, and the dataset distribution used for model training and evaluation.

During the data collection phase, significant challenges were initially encountered due to strict privacy regulations and confidentiality concerns raised by the insurance experts regarding customer data. To overcome this obstacle and strictly comply with ethical research standards, a rigorous preprocessing step was implemented before any analysis: all vehicle license plates and identifying personal features were carefully blurred or removed. This process fully guaranteed complete anonymity for vehicle owners while preserving the essential visual features required for training the object detection models.

Following the resolution of these privacy constraints, the primary image data for

this research was successfully acquired through a direct collaboration with professional automotive insurance experts operating in the southern regions of Algeria, specifically in Touggourt and Ouargla. This localized data collection ensures that the dataset reflects realistic, officially documented car accidents within a unique geographical and climatic context. Collecting images from these specific areas introduces distinctive environmental factors, such as intense desert lighting, high solar reflection on vehicle bodies, and dust conditions, which present realistic and rigorous challenges for modern computer vision model

The constructed dataset comprises a total of 608 high-resolution images meticulously gathered and filtered from the insurance experts' records. To ensure precise damage localization and classification, six distinct categories of vehicle damage were defined based on the most frequent insurance claims and repair scenarios. These classes include: Scratch, Dent, Crack, Broken Glass, Broken Lamp, and Flat Tire. To provide a clear overview of the data structure, Table 3.1 outlines the defined classes and their corresponding descriptions. To evaluate the generalization ability of the trained object

Table 2.1: Defined Car Damage Categories in the Custom Dataset.

Class Name	**Description**
Scratch	Surface marks on the vehicle body or paintwork
Dent	Deformations and depressions on metallic panels
Crack	Structural splits in plastic parts (e.g., bumpers)
Broken Glass	Shattered or cracked windshields and windows
Broken Lamp	Smashed or damaged headlights and taillights
Flat Tire	Deflated or punctured wheels

detection models and prevent overfitting, the collected dataset was split into three independent subsets following standard machine learning engineering practices. The splitting ratio was defined as 70% for training, 15% for validation, and 15% for testing. The exact distribution of the 608 images across these sets is detailed as follows:

- Training Set (70%): Comprises 425 images, which are utilized to optimize the model parameters and learn the structural features of each damage category.
- Validation Set (15%): Comprises 91 images, used during the training process to monitor performance, tune hyperparameters, and prevent overfitting.
- Testing Set (15%): Comprises 92 images, reserved exclusively for the final evaluation to measure the models' accuracy on unseen real-world scenarios.

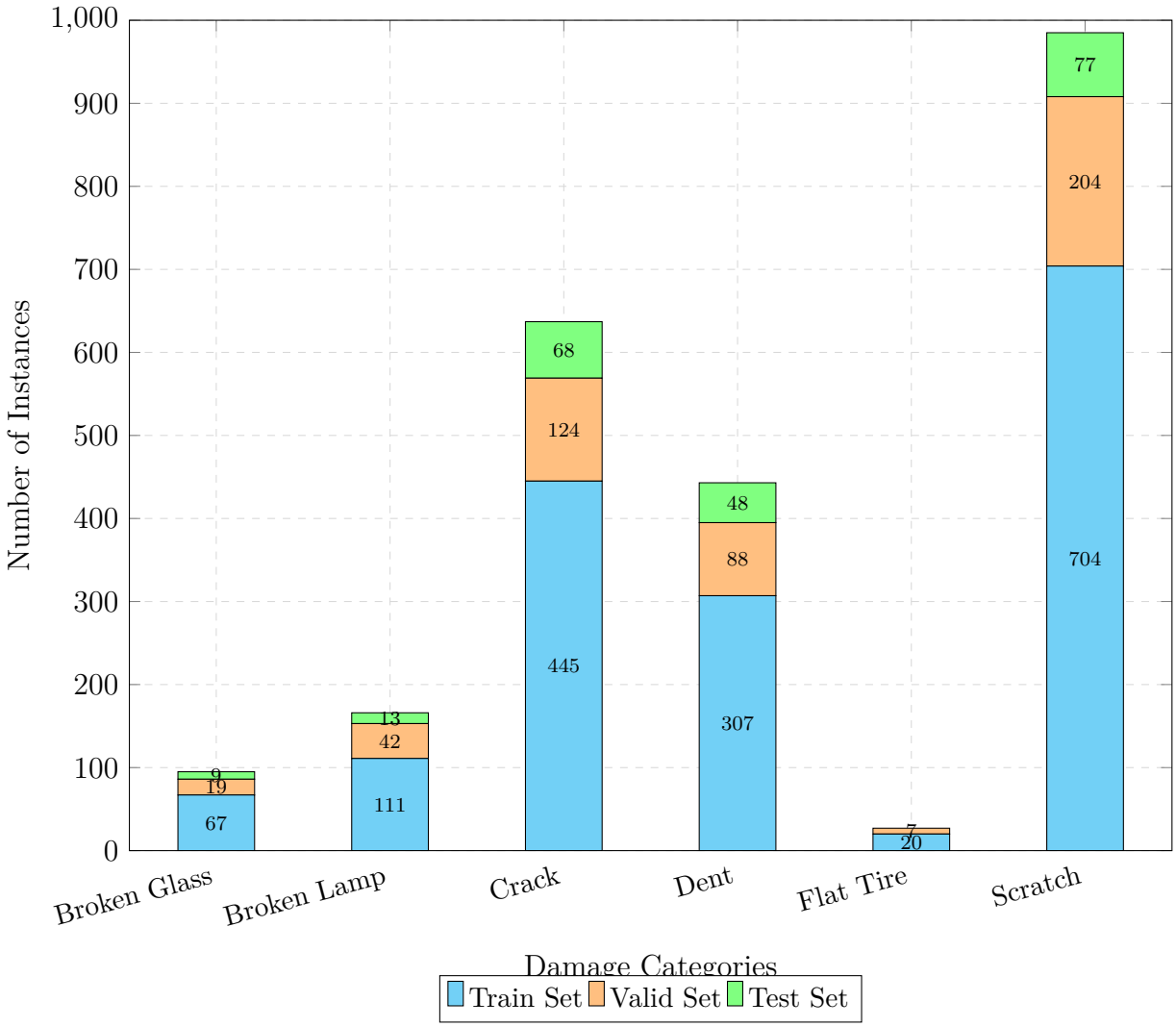


Figure 2.1: Stacked bar chart of the dataset class distribution.

The graph clearly shows that *Scratch* and *Crack* have the highest number of instances in the dataset. This distribution is very logical for two practical reasons:

First, in everyday car accidents and urban driving, minor damages like scratches and small cracks are the most common events. Severe damages, such as a totally broken lamp or a flat tire, happen much less frequently. Second, from an annotation perspective, a single car accident usually causes multiple separate scratches and cracks across the vehicle's body. Since our chart counts the total number of bounding boxes (instances) and not the number of images, these repetitive small damages naturally multiply the numbers for the *Scratch* and *Crack* classes. This provides the model with enough data to learn and detect these very common vehicle defects.

2.2 CARDD Dataset

3.1.2 CARDD Dataset

In addition to the custom Algerian dataset, the Car Damage Detection Dataset (CARDD) was utilized as a public benchmark dataset for vehicle damage detection. CARDD was developed to address the lack of large-scale publicly available datasets dedicated to automatic car damage assessment and has become an important resource for computer vision research in the automotive insurance domain. The dataset contains 4,000 high-resolution vehicle images with more than 9,000 annotated damage instances distributed across six damage categories[30]. These annotations support both object detection and segmentation tasks through detailed region-level labeling. CARDD provides a diverse collection of vehicle damage scenarios captured under different real-world conditions making it suitable for training and evaluating deep learning models[30]. Owing to its scale annotation quality and public availability, CARDD is widely used as a benchmark dataset for assessing the performance and generalization capability of modern object detection architectures[30]. In this work CARDD was employed as a reference dataset for model pretraining and comparative evaluation before fine-tuning on the custom Algerian vehicle damage dataset. The following figure presents sample images from the CARDD dataset along with their annotated damage regions.

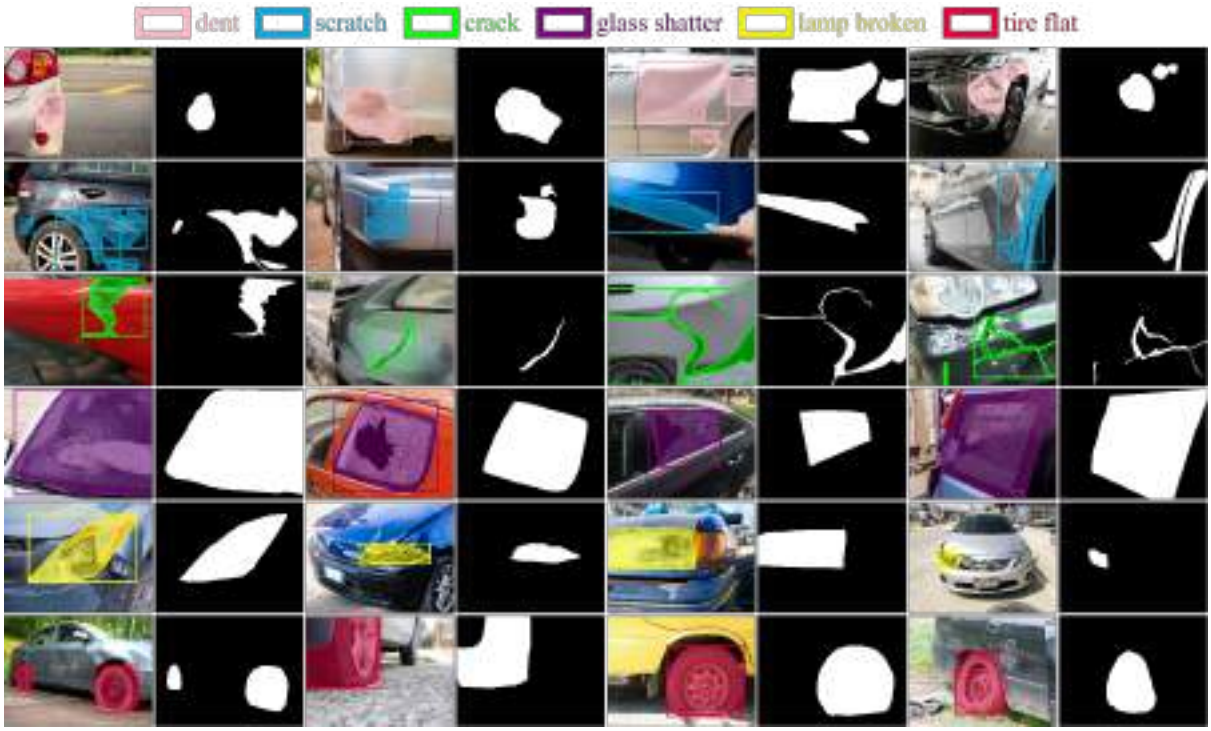


Figure 2.2: Examples of the six damage categories in the CARDD dataset (dent, scratch, crack, glass shatter, lamp broken, and flat tire).

The following table presents the main characteristics of the CARDD dataset used in this study.

Table 2.2: Main Characteristics of the CARDD Dataset

Property	**Value**
Number of Images	4000
Number of Instances	$>$ 9000
Number of Classes	6
Tasks Supported	Detection and Segmentation
Annotation Type	Region-level Annotations

2.3 COCO Format

The famous COCO (Common Objects in Context) dataset is a fundamental reference in the field of deep learning, featuring 91 distinct target categories, of which 82 categories are the most complete in terms of data, each exceeding 5,000 training samples. The dataset cumulatively includes approximately 2.5 million instances identified within 328,000 separate images. The COCO dataset does not provide random expansion in the number of categories, but rather prioritizes deeper repetition of models for each cate-

gory, unlike traditional datasets such as ImageNet. This feature has led deep learning architectures to achieve strong accuracy in 2D bounding boxes and segmentation masks. Moreover, the main feature of COCO lies in its dense layout distribution, resulting in an average of 7.7 symbolic annotations (labels) per scene, which clearly surpasses PASCAL VOC (2.3) and ImageNet (3.0). This structural richness forces neural networks to capture deep contextual cues and geographical and spatial associations between the classified targets and the surrounding background elements[7]. As shown in Figure 2.3, the structural complexity and the high density of annotations in COCO provide a superior framework for object detection tasks.



Figure 2.3: Examples of object annotations in the MS COCO dataset[7]

Since the COCO format provides high accuracy in 2D localization, we have chosen it as the official format for our data to ensure precise detection of car damages in complex environments. Detecting car damages is one of the challenging tasks in the field of computer vision, as images are often captured in various and cluttered real-world

environments, such as streets with dense shadows and sunlight reflections on the metallic structures of vehicles. By relying on the COCO format (via the structured JSON file), detection models such as YOLO and DETR can better recognize different damage patterns and distinguish them from the cluttered background details.

Technically, the COCO format organizes all the dataset information within a unified JSON file, which acts as a central database linking each car image to its damage labels. Instead of scattering the information across multiple files, the JSON structure integrates the metadata through four interconnected components; it begins with the info section, which stores general project details such as the year and data description, followed by the categories field, which defines the types of damages to be detected (such as dents, scratches, and cracks) by assigning a unique identifier to each category. This schema is directly linked to the images section, which lists each composed image along with its name and dimensions. Finally, the system links all these elements together through the annotations component, which accurately matches the image ID with its corresponding damage ID, and specifies the exact spatial location of the damage using standard bounding box coordinates `[x, y, width, height]`[7]. Through this cohesive structure, detection models like YOLO and faster R-CNN can smoothly read the files to learn to accurately locate car damage with high efficiency.

2.4 Roboflow Platform and Dataset Preparation

2.4.1 Roboflow Workspace Setup

To start the practical implementation, a custom project was created within the Roboflow platform specifically designed for the dataset of car damage detection in Algeria. The project environment was explicitly defined as a custom object detection procedure, which aligns with the goal of locating and identifying specific structural defects in vehicles. And within this workspace, the settings included configuring the six distinct damage categories to ensure accurate classification during the bounding box annotation phase. After this initial setup, all the raw images that were successfully collected were uploaded to the Roboflow cloud repository. Finally, to ensure the statistical integrity of the training and evaluation process, the entire dataset was systematically organized and divided into three separate sections (training, validation, and testing).

2.4.2 Dataset Uploading and Management

To start the data management and purification phase, all the collected images were officially imported into the Roboflow platform. During this phase, the data was meticulously cleaned and filtered. Images of poor quality, low resolution, irrelevant content, or duplicate scenes were carefully removed. This step was important to ensure data quality and prevent the models from learning inaccurate information. The Roboflow platform was used to manage and track different data versions. The same structure of the core categories was maintained in each release. The structured data releases served as a precise framework, enabling a fair comparison between different preprocessing settings and data configurations throughout the experimental period.

2.4.3 Annotation Process on Roboflow

To ensure high-quality reference data, we manually identified and framed all damage areas in the images we collected using bounding boxes directly on the Roboflow platform. This manual process was essential for teaching the models the precise visual characteristics of vehicle defects.

During the labeling phase, we assigned each bounding box to one of the six predefined damage categories. This strict assignment of categories prevented any confusion in the labels and ensured the consistency of data classification.

The focus was on the visible and localized damage areas instead of the entire car structure. By isolating only the affected areas, we trained the models to ignore the intact parts of the car and focus on the defects.

The bounding boxes were drawn as tightly and precisely as possible around each affected area. These narrow boxes are crucial because they prevent the inclusion of background pixels or undamaged parts of the car, reducing noise during model training.

Finally, we implemented a quality control step where we meticulously reviewed ambiguous or unclear damages before final approval. This additional step ensured the correct labeling of questionable images or their exclusion, maintaining a reliable and clean dataset.

2.4.4 Annotation Guidelines

To ensure the highest standards of data quality, we focused on a single framework for visible car damages. This means we did not allow the models to predict hidden or invisible defects. Moreover, we ensured that large undamaged areas were not included within the bounding boxes. Our goal is to make the models learn from the damages themselves, not from the undamaged parts of the car. In the presence of multiple defects, we used separate bounding boxes for each damaged area. This helped the model understand each case individually. To avoid errors, we focused on selecting the most accurate label for each type of damage. This reduced the overlap between categories and increased the model's accuracy. We also chose not to frame damages that are not clear, hidden, or too ambiguous. We wanted to protect the training data from any distorted data. Finally, we ensured that the labels were consistent across all images. This helped the model learn more efficiently and achieve accurate results.

2.4.5 Roboflow Preprocessing

To properly prepare the dataset for the training phase, we applied automatic correction to ensure there are no issues with image rotation resulting from differences in imaging devices. This ensures that all the characteristics and details of the car appear correctly in their spatial layout.We also resized all the images to uniform dimensions to meet the requirements of object detection models like YOLO and Faster R-CNN. These models require fixed pixel sizes to function correctly. This step was important because it ensured that all the images had the same shape and size before being input into the network. This helped facilitate feature extraction. Additionally, this step helped reduce unnecessary differences resulting from the varying resolutions of the original images. This prevents the models from being biased toward larger images or losing details in smaller images. Finally, the same settings have been accurately maintained across all versions of the dataset. This ensures a fair and reliable comparison between our experiments, allowing us to evaluate the models based solely on their performance.

2.4.6 Roboflow Data Augmentation

Enhancing the generalization capabilities of our models, data augmentation techniques were applied to artificially increase the diversity of the dataset by creating new variations of existing images. This technique was particularly useful in reducing overfitting during model training, ensuring that the networks not only memorize the training samples but instead learn strong and generalized features.

The augmented images were designed to simulate the variations of real-life conditions in Algerian environments, such as differences in lighting conditions, dust, and the strong sunlight characteristic of local roads.

The geometric and optical transformations were carefully selected to avoid any unrealistic distortion of the damages, thus ensuring that the artificial changes remain physically reasonable and do not alter the actual semantic meaning of the car defects.

Geometric Augmentations

To introduce strong spatial changes to our dataset, we implemented specific geometric transformations, including horizontal flipping, small rotations, scaling, cropping, and translation. The application of horizontal flipping allowed the models to detect damage regardless of whether it appeared on the left or right side of the car, while the introduction of small rotations simulated different camera angles. Moreover, resizing and cropping were used to vary the area and focus of the target regions, helping the networks to identify car defects from different distances and zoom levels. Finally, the displacement shifted the position of the cars within the frame, ensuring that the feature extraction process remained independent of the precise location of the damage. Together, these geometric transformations forced the models to learn the fixed structure of car defects instead of memorizing fixed positions.

Photometric Augmentations

To make our models highly flexible toward changing environmental factors and non-ideal shooting conditions, we applied a comprehensive set of photometric transformations, including brightness adjustment, light exposure change, contrast modification, blur simulation, and noise addition. Using brightness adjustment allowed us to simulate the extreme lighting changes that cars face on the roads, while light exposure adjustment

modified the effect of artificial camera shuttering to replicate excessive midday sunlight or dark shadows. Additionally, contrast adjustment was implemented to alter the distinction between light and dark areas on the car's surface, helping the network to identify fine textures of damage even in low-visibility conditions. To account for unexpected real-world imaging problems, blur simulation was integrated to replicate distortions caused by camera shake or vehicle movement, teaching the network to recognize defects even in the absence of sharp edges. Finally, noise was added to simulate pixel-level graininess characteristic of low-quality smartphone sensors or poor visibility. By incorporating these visual modifications, we ensured that the models strictly focus on the structural features of the car damage instead of being misled by changing lighting conditions or deteriorating image quality.

2.4.7 Dataset Export

Once the preprocessing and data augmentation stages were completed, the final dataset was exported from the Roboflow platform to begin the model training process. To meet the specific input requirements of the frameworks we chose, the export was implemented in two distinct formats: the COCO format, which relies on JSON annotations for architectures like Faster R-CNN, and the original YOLO format, specifically designed for YOLO-based object detection models. Each exported dataset package was meticulously organized to include raw image files and their corresponding annotation files. In the end, this organized export method was crucial as it ensured smooth compatibility, alignment of structured data, and seamless integration with our specified detection architectures.

2.5 Models

2.5.1 Model Selection Criteria

Once the preprocessing and data augmentation stages were completed, the final dataset was exported from the Roboflow platform to begin the model training process. To meet the specific input requirements of the frameworks we chose, the export was implemented in two distinct formats: the COCO format, which relies on JSON annotations for architectures like Faster R-CNN, and the original YOLO format, specifically designed for

YOLO-based object detection models. Each exported dataset package was meticulously organized to include raw image files and their corresponding annotation files. In the end, this organized export method was crucial as it ensured smooth compatibility, alignment of structured data, and seamless integration with our specified detection architectures.

2.5.2 YOLO-Based Object Detection

YOLO models were selected because of their strong real-time detection capability, making them highly efficient for dynamic environments. This efficiency stems from the fact that YOLO performs object localization and classification in a single stage, treating detection as a single regression problem[6]. During inference, the model predicts bounding boxes, class labels, and confidence scores simultaneously from a single forward pass. This specific architecture is highly suitable for detecting visible car damages such as scratches, dents, cracks, broken glass, broken lamps, and flat tires. Furthermore, due to their optimized balance of speed and computational footprint, YOLO-based models are efficient for possible future deployment in web or mobile applications, providing a practical foundation for automated vehicle inspection.

2.5.3 DETR

DETR (Detection Transformer) is a transformer-based object detection model selected to explore a modern end-to-end approach for vehicle damage detection. It treats object detection as a direct set prediction problem combining a CNN backbone with an encoder-decoder transformer to capture global relationships in the image. During inference it directly predicts a fixed set of object queries representing bounding boxes and class labels without using region proposal networks or non-maximum suppression which simplifies the pipeline. DETR is effective in handling complex scenes with multiple or overlapping damages due to its ability to model global context but it generally requires more training time and larger datasets to perform well. In this study, it was used for comparison with YOLO-based and Faster R-CNN models.

2.5.4 Faster R-CNN

We chose Faster R-CNN as a two-stage object detection model to provide a strong benchmark in our experiment. In this system, the first stage uses a Region Proposal Network (RPN) that scans the feature maps extracted by the backbone network to generate high-quality region proposals that are likely to contain the target objects. Then, these candidate regions are processed using a Region of Interest (RoI) pooling layer, allowing the second stage to efficiently classify these proposed regions into specific damage categories and simultaneously determine their bounding box coordinates. Since this process decouples the location search from the classification process, Faster R-CNN is widely known for its strong localization performance, making it particularly effective in identifying complex, small, or irregular boundaries of car defects. In the end, we used it as a comparative model against YOLO-based detection to rigorously evaluate the trade-offs between two-stage accuracy

2.5.5 Comparison Between YOLO , Faster R-CNN and DETR

Three object detection frameworks were investigated for vehicle damage detection, each representing a distinct methodological paradigm. YOLO-based models follow a single-stage detection strategy, enabling efficient real-time inference through simultaneous localization and classification, though with potential limitations in detecting small or overlapping objects. Faster R-CNN adopts a two-stage approach, where region proposals are generated and subsequently refined, resulting in superior localization accuracy at the expense of increased computational cost. DETR introduces an end-to-end transformer-based architecture that models global image context through attention mechanisms and directly predicts a set of object queries, eliminating the need for region proposal generation and post-processing steps. However, it typically requires larger datasets and longer training durations for optimal convergence. Collectively, these models provide a comprehensive trade-off analysis between computational efficiency, localization accuracy, and architectural complexity.

In short, comparing both models helps in evaluating the trade-off between inference speed and detection accuracy. This allows us to have a comprehensive understanding of their performance and suitability. This is especially important within the framework of

our vehicle inspection.

2.5.6 Training Strategy

To ensure a fair and consistent evaluation, the models were trained using the same splits of the training, validation, and test data across all experiments. To leverage previous feature representations, pre-trained weights were used when available, and transfer learning techniques were systematically applied to significantly improve the models' performance on our custom and limited dataset.

The training process is technically carried out by rotating the data in a forward pass, where the input passes thru the network layers to generate predictions for bounding boxes and class labels. The model then calculates the loss function—which includes location loss, classification loss, and confidence score—to measure the error rate compared to the true values. Based on this value, the error is backpropagated thru the network using backpropagation to update the model's weights via the optimization algorithm to gradually reduce the errors. Throughout this phase, the validation set is relied upon at the end of each training epoch to monitor the learning curves and check the progression of loss and accuracy to prevent overfitting. In the end, the test set was strictly retained and used only for the final evaluation to measure the models' ability to generalize to completely new data.

2.5.7 Training Parameters

To start training the machine learning model, it is essential to set some basic configurations. First, we need to determine the size of the image we will be working with, so that we become capable of analyzing uniform data. Additionally, we need to determine the batch size, which represents the number of images that will be processed at each time, and the number of times the training process will be repeated. These settings are known as the number of training epochs.

To ensure the stability of the model during training, the learning rate must be adjusted. This rate controls how efficiently the model learns from the data. Choosing the appropriate optimization algorithm also plays an important role in updating the internal weights of the model.

When we move to the inference and evaluation stage, we rely on the confidence

threshold and the intersection over union threshold to assess the accuracy of the predictions. The confidence threshold is defined by our level of trust in the model's results, while the intersection threshold measures the degree of alignment between the predictions and the actual values. In this way, we can filter the final predictions and ensure better defect detection accuracy.

Table 2.3: Training Hyperparameters and Optimal Experimental Configurations

Hyperparameter	**Description**	**Value**
Input Image Size	The dimensions to which all images are resized to ensure uniform data analysis.	640 × 640
Number of Epochs	The total number of complete passes the training algorithm makes through the entire dataset.	50
Batch Size	The number of images processed simultaneously in a single forward and backward pass.	16
Learning Rate	The step size taken during weight updates to ensure stable convergence of the model.	0.01
Optimizer	The specific optimization algorithm used to update the network's internal weights.	Adam / SGD
Confidence Threshold	The minimum probability score required to accept a model's prediction as a valid detection.	0.25
IoU Threshold	The intersection over union criterion used to measure box alignment and filter overlapping predictions.	0.45

2.5.8 Model Output

To comprehensively evaluate the performance of the trained models, detailed output data is generated for each sample analyzed during the inference stage. The spatial boundaries of the damages are precisely determined within the image using predicted bounding boxes, and the expected damage class is assigned to each box. The confidence score for each prediction is calculated, determining the model's certainty regarding the accuracy of the localization and classification tasks together. The final detection results, which integrate bounding boxes, class labels, and confidence scores, are displayed and visualized directly on the test images to facilitate the qualitative assessment of the system's performance.

2.6 Evaluation

2.6.1 Evaluation Objective

To rigorously evaluate the effectiveness of the programs, we need to conduct a final phase in this study. At this stage, we will evaluate the ability of each model to detect car damages in Algeria. We will conduct this evaluation on new images that have not been seen before. This will help us measure the models' ability to generalize in an unbiased manner. We will evaluate the accuracy of damage classification and the precision of locating it. We will compare the expected and actual images. In this way, we will be able to determine the best-performing model. And we won't stop there; we will also evaluate the performance of the models in other ways. We will look for common errors that occur during detection. We will also look into false positives, where non-existent damages are detected. This will give us a broader understanding of the behavior of each program[31].

2.6.2 Evaluation Metrics

To evaluate the performance of object detection models, we must rely on reliable evaluation metrics. First, we measure the prediction accuracy, which determines whether the positive predictions are correct or not. Then we measure the recall rate, which indicates the model's ability to detect all actual damages. To achieve a balanced evaluation, we calculate the F1-score, which is the harmonic mean between precision and recall. To determine the accuracy of spatial location, we use the intersection over union. This metric determines the degree of overlap between the predicted bounding boxes and the actual bounding boxes. Additionally, we calculate the average precision and overall average precision to analyze the detection quality across different damage categories. Finally, we must evaluate the time feasibility and computational efficiency of each model. We measure the inference time under the same standardized conditions. This allows us to compare the performance between different models and choose the best among them[31].

Table 2.4: Performance Evaluation Metrics and Technical Definitions

Metric	**Technical Description**	**Mathematical Representation**
Precision	Measures the accuracy of positive predictions, representing the ratio of true positive detections to all predicted positives.	$\text{Precision} = \frac{\text{TP}}{\text{TP+FP}}$
Recall	Measures the model's ability to find all relevant cases, representing the ratio of true positives to all actual ground truths.	$\text{Recall} = \frac{\text{TP}}{\text{TP+FN}}$
F1-score	The harmonic mean of Precision and Recall, providing a balanced measure of the model's overall detection performance.	$\text{F1} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision+Recall}}$
IoU	Evaluates the spatial overlap between the predicted bounding box (B_p) and the ground truth bounding box (B_{gt}).	$\text{IoU} = \frac{\text{Area}(B_p \cap B_{gt})}{\text{Area}(B_p \cup B_{gt})}$
AP	Computes the area under the Precision-Recall curve for a single specific object class.	$\text{AP} = \int_0^1 P(R)\, dR$
mAP	The average of the Average Precision (AP) calculated across all designated damage categories in the dataset.	$\text{mAP} = \frac{1}{N} \sum_{i=1}^{N} \text{AP}_i$
Inference Time	The latency or time taken (typically in milliseconds) by the model to process a single input image and output final predictions.	Measured in ms per image

.

2.6.3 Quantitative Comparison

To determine the true performance of the software architectures, we conducted a comprehensive comparison between the YOLO and Faster R-CNN models using the same dataset. We evaluated the performance of each model using metrics such as mean average precision, precision, recall, and F1-score[31]. These metrics help us comprehensively evaluate detection accuracy and localization. Additionally, we carefully examined the computational efficiency and inference speed to assess the suitability of each model for real-world applications. In the end, we chose the model that achieves the best balance between prediction accuracy and computational efficiency was selected, with the detailed comparative results presented in Chapter 3.3.2.

2.6.4 Qualitative Evaluation

To fully complete the quantitative evaluation, specific steps for qualitative analysis have been established, aiming to conduct a thorough examination of the prediction process carried out by the trained models. This visual inspection includes a careful study of the correct detections to ensure the reliability of the network's performance. Furthermore, a plan has been developed to identify missed detections and false positive predictions through a comprehensive review process. Moreover, localization errors, such as imprecise bounding boxes, are meticulously examined, and cases of confusion between visually similar categories are analyzed through an in-depth study. The detailed visual results, including prediction samples and the challenges faced by the model, are thoroughly discussed in Chapter 4.

2.6.5 Failure Case Analysis

To achieve a comprehensive understanding of the practical limitations of the implemented object detection architectures, a methodological framework was established to analyze specific failure cases and critical edge situations. This analysis focuses on several critical challenges observed during the model evaluation phase. First, fine structural challenges are examined, such as small scratches that the model consistently overlooks due to their minimal pixel footprint. Secondly, environmental and superficial interferences are assessed, including specular reflections and complex shadows that are misidentified as structural damage, cracks, or dents, along with surface dirt that is frequently confused with real scratches. Thirdly, the impact of image quality degradation is evaluated, focusing on how low-lighting conditions affect overall detection quality and the impact of blurriness on bounding box localization accuracy. Finally, data-related constraints are examined to understand how rare and underrepresented categories exhibit inherently weaker detection performance compared to well-represented categories.

Chapter 3

Results and Discussion

This chapter presents the experimental results obtained after training and evaluating the selected object detection models on the Algerian car damage dataset. The analysis includes dataset statistics, quantitative evaluation, qualitative prediction results, model comparison, and failure case analysis.

3.1 Dataset Analysis

3.1.1 Dataset Summary

The experiments in this chapter are conducted on a custom Algerian car damage dataset specifically constructed for object detection in real-world automotive inspection scenarios. The primary purpose of this dataset is to enable the training and evaluation of deep learning models capable of automatically localizing and classifying visible damage on vehicles operating under Algerian environmental conditions. In total the dataset comprises 608 annotated images of passenger vehicles collected in outdoor settings reflecting typical accident and inspection contexts encountered in Algerian cities and suburban areas.

The dataset is split into three disjoint subsets to support a rigorous experimental protocol: 426 images are used for training ,122 images are reserved for validation, and 60 images are held out for final testing. The training set is employed to learn the model parameters, while the validation set is used to tune hyperparameters monitor convergence, and prevent overfitting. The test set remains unseen during model development and is used exclusively in this chapter to provide an unbiased assessment of the final detection

performance.

Each image is annotated for object detection using axis-aligned bounding boxes that tightly enclose the visible damage regions. The dataset includes six damage categories: Scratch, Dent, Crack, Broken Glass, Broken Lamp and Flat Tire. These classes were selected to cover a broad range of common external damages relevant to insurance assessment and repair planning. Annotations were created and managed using the Roboflow platform and exported in both COCO and YOLO formats, ensuring compatibility with widely used object detection frameworks and facilitating reproducible experimentation.

3.1.2 Damage Classes

Autonomous Driving refers to the integration of advanced technologies that enable vehicles to operate with minimal or no human intervention. The development of Autonomous Vehicles (AVs) is progressively bridging the gap between traditional human-driven cars and fully self-driving systems, with the potential to eliminate the need for a human driver entirely. This progression spans a continuum of automation levels, encompassing technologies designed to assist drivers and make decisions on their behalf. Notable examples include crash warning systems, adaptive cruise control (ACC), lane-keeping assistance, and automated parking systems [32].

3.1.3 Scratch

Scratches typically appear as thin, elongated, and often low-contrast marks on painted surfaces. They may vary in length and orientation and can sometimes be confused with reflections, dust streaks or background textures.Figure 3.1 illustrates such a case,where a subtle scratch on the vehicle's suface is visually similar to environmental reflections.



Figure 3.1: Example of scratch damage on a vehicle.

3.1.4 Dent

Dents by contrast, correspond to localized geometric deformations of metal or plastic panels, often characterized by subtle changes in shading, curvature, and specular highlights. Detecting dents requires sensitivity to shape and illumination variations rather than explicit texture changes.Figure 3.2 depicts such a deformation on a vehicle's rear panel,where the subtle changes in curvature and shading highlight the typical appearance of a dent.



Figure 3.2: Example of dent damage on a vehicle

3.1.5 Crack

Cracks usually manifest as narrow fractures in bumpers, plastic parts, or glass components; they can be irregular and branching, and may have very small width relative to their length, which makes them challenging to detect at lower resolutions.Figure 3.3 illustrates this complexity, showing a narrow, irregular crack on a vehicle bumper that demonstrates the difficulty of isolating such fine features from the surrounding surface.



Figure 3.3: Example of crack damage on a vehicle.

3.1.6 Broken Glass

Broken Glass refers to shattered or significantly fractured windshields, windows, or mirrors. Visually, this class is associated with fragmented patterns, sharp edges, and irregular transparency or reflections.As presented in Figure 3.4,the distinct fragmented patterns and sharp edges of the fractured windshield provide a clear example of how such visual features are categorized within the dataset.



Figure 3.4: Example of broken glass damage on a vehicle

3.1.7 Broken Lamp

Broken Lamp corresponds to visible damage on headlights or taillights, including broken covers, missing pieces, or internal fractures that alter the usual lamp appearance. Figure 3.5 provides a clear example of this damage,showing a broken lamp cover with visible structural fractures.



Figure 3.5: Example of broken lamp damage on a vehicle

3.1.8 Flat Tire

Flat Tire denotes tires that have visibly lost air pressure and exhibit deformation at the contact area with the ground sometimes accompanied by partial rim exposure.Figure 3.5 illustrates this condition, showing the characteristic shape of a tire that has lost its pressure.



Figure 3.6: Example of flat tire damage on a vehicle

These six classes are particularly important for automated vehicle inspection systems because they directly impact safety assessment, repair cost estimation, and insurance decision-making. A reliable detection of such damages can therefore support both operational workflows and objective, consistent evaluations of vehicle condition.

Transitioning from the qualitative description of damage types, the next subsection analyzes how these classes are distributed in the dataset and how this distribution may influence the learning behavior of deep object detection models

3.1.9 Class Distribution Analysis

The distribution of annotated instances across the six damage classes is not uniform in the Algerian car damage dataset.Scratch represents the most frequent class, containing the highest number of annotated instances while Crack is also highly represented with a large amount of training samples. In contrast, Flat Tire is the least represented category containing only a limited number of annotations. The remaining classes, including Broken Glass, Broken Lamp, and Dent, appear with intermediate frequencies. This unequal distribution indicates a clear class imbalance within the dataset. During training, the deep learning model is exposed more frequently to dominant classes such as Scratch and Crack than to minority classes like Flat Tire. Consequently, the model may learn more robust feature representations for highly represented categories while struggling to generalize effectively for rare classes. Such imbalance can negatively influence detection performance particularly for underrepresented categories leading to lower recall and Average Precision (AP) values. Therefore the class distribution is an important factor to

consider when interpreting the experimental results and evaluating the robustness of the proposed models.

Table 3.1: Class distribution of annotated instances in the Algerian car damage dataset

Damage Class	**Instances**	**Percentage (%)**
Scratch	985	41.87
Crack	637	27.07
Dent	443	18.83
Broken Glass	95	4.04
Broken Lamp	166	7.06
Flat Tire	27	1.15
Total	2353	100

This table confirms a clear class imbalance, where Scratch and Crack dominate the dataset while Flat Tire remains significantly underrepresented.

3.1.10 Impact of Class Imbalance

Class imbalance has a direct and often significant impact on the performance of object detection models.In the context of the present dataset, frequent classes such as Scratch and Crack achieve better performance because they have a large number of training samples, which helps the model learn effectively. In contrast, rare classes such as Flat Tire, Broken Glass, and Broken Lamp have limited data, which weakens the model's ability to learn their features and leads to a decrease in performance, especially lower Precision, Recall, and AP values, particularly for the Flat Tire class. Therefore, the results should be interpreted with consideration of class imbalance as better performance on frequent classes does not necessarily indicate a stronger model but may simply reflect the larger number of training samples. Similarly poorer performance on rare classes is often due to limited data rather than model limitations..Figure 3.7 illustrates the class distribution within our dataset, where Scratch represents the majority class at 41.87%, followed by Crack at 27.07% providing a clear overview of the data composition.

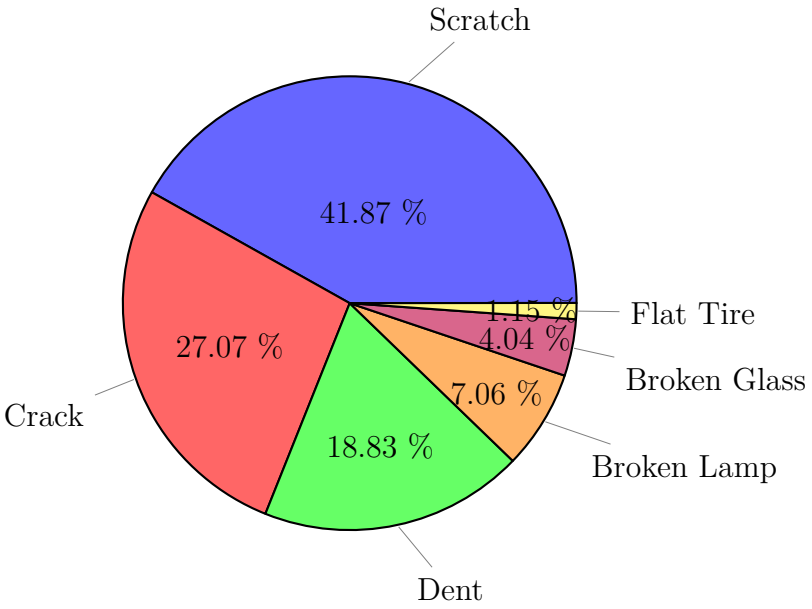


Figure 3.7: Distribution of car damage classes.

3.2 Training Results

3.2.1 Training Configuration Summary

The car damage detection models were trained in a supervised object detection setting using annotated images collected from the CARDD dataset and the custom Algerian vehicle damage dataset[33][34].

Dataset (608 Images)
⌞**Train (70%)** → 426 Images
⌞**Validation (20%)** → 122 Images
⌞**Test (10%)** → 60 Images

Figure 3.8: Dataset splitting architecture for car damage detection.

3.2.2 YOLO Training Results

The YOLO model achieved effective detection performance in the car damage dataset and demonstrated strong localization capability for visible damages such as dents, scratches, and broken glass [35].The following figure illustrates the training results for the YOLO model.

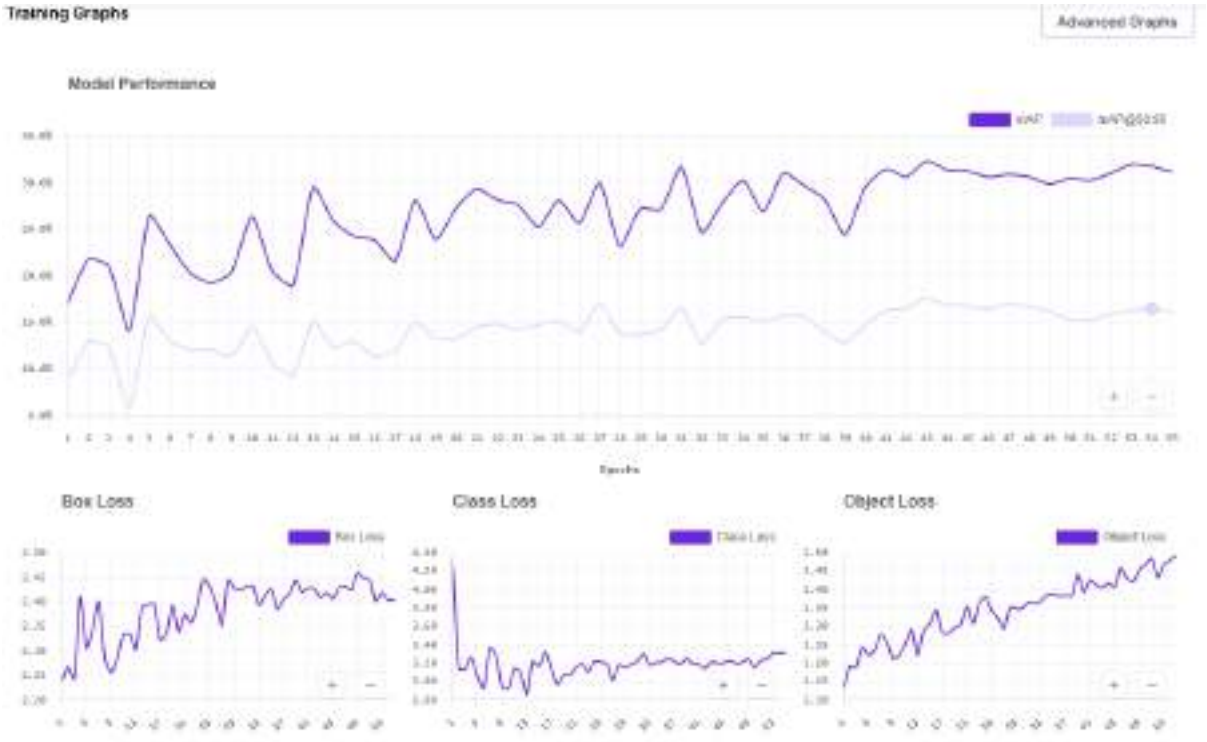


Figure 3.9: YOLOv11 training results displaying loss convergence curves and overall performance

3.2.3 DETR Training Results

The DETR model achieved superior detection performance on the car damage dataset and demonstrated excellent localization capability for various types of visible damages including dents, scratches, and broken glass, outperforming the YOLO model in overall detection accuracy and robustness. The training results and loss convergence curves for the DETR model are presented in the following figure.

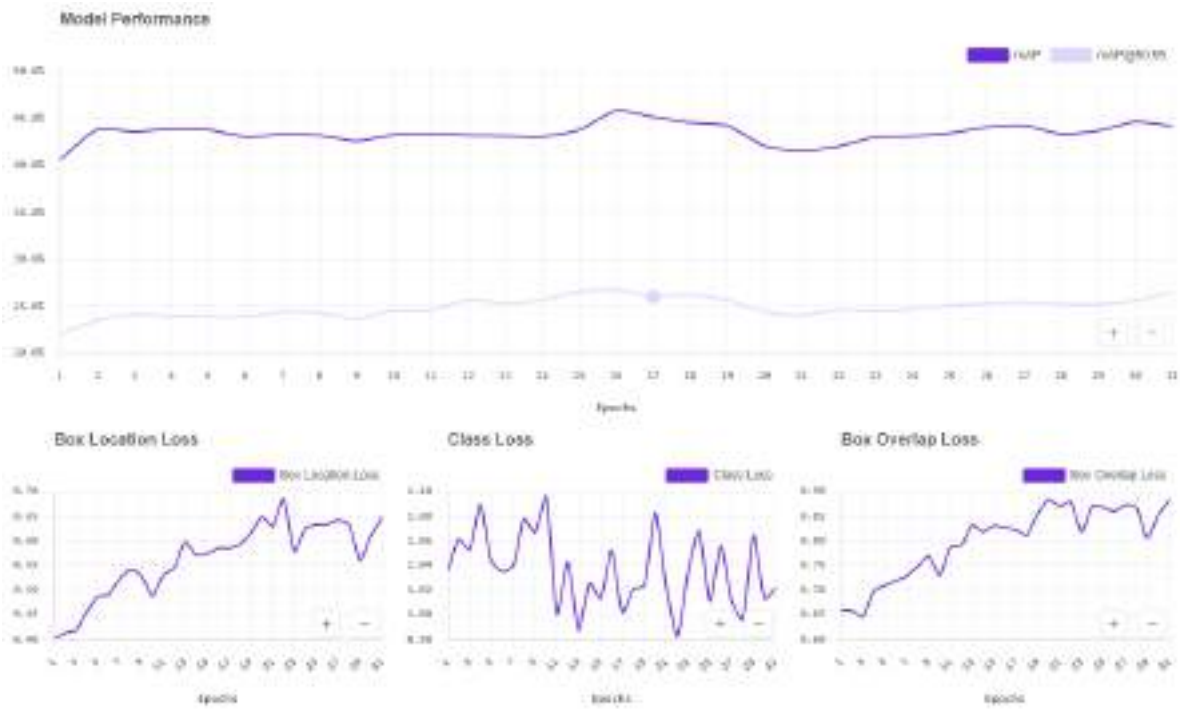


Figure 3.10: RF-DETR training results displaying loss convergence curves and overall performance

3.2.4 Faster R-CNN Training Results

Faster R-CNN–based detectors trained for infrastructure and surface damage typically achieve higher localization accuracy and recall than single-stage models, especially for small, thin, or blurry defects such as cracks and potholes [36][37][38]. Reported results on damage-related datasets show F1-scores and recall often exceeding 0.6–0.9, with good alignment between predicted and ground-truth bounding boxes [37]. However, these gains come at the cost of greater computational complexity and longer inference times compared to YOLO-type models, which can constrain real-time automotive applications[36].

3.2.5 Training Loss Analysis

Training and validation loss curves are central to analyzing learning dynamics in object detection models. For detectors such as YOLO ,DETR and Faster R-CNN, the total loss typically combines classification, bounding box regression, and sometimes objectness components[39][40]. A steadily decreasing training loss accompanied by a similarly decreasing, smoothly converging validation loss indicates effective optimization and good generalization[41][33]. In contrast, divergence between the two curves—such as training

loss continuing to drop while validation loss plateaus or increases—signals overfitting, suboptimal learning rate schedules, or insufficient regularization[33][42][43].

3.2.6 Overfitting and Underfitting Analysis

Overfitting and Underfitting Analysis In car damage detection with relatively small, custom datasets, models are prone to both underfitting and overfitting. Underfitting may occur when the network capacity, training time, or feature representation are insufficient to capture complex damage patterns, leading to high losses on both training and validation sets[42]. Overfitting, manifested by low training loss but significantly higher validation loss and degraded test mAP, is common when labeled data are limited or class distributions are imbalanced [33][42]. Data augmentation and transfer learning from large-scale sources help mitigate overfitting by increasing sample diversity and providing richer initial feature representations, thereby improving generalization for damage classes with few examples[44].

3.3 Quantitative Evaluation

3.3.1 Evaluation Metrics

The YOLO and DETR detectors were evaluated using standard object detection metrics: Precision: ratio of correctly detected damages to all predicted damages (penalizes false positives). Recall: ratio of correctly detected damages to all ground-truth damages (penalizes missed detections). F1-score: harmonic mean of precision and recall, summarizing detection quality. IoU (Intersection over Union): overlap between predicted and ground-truth bounding boxes, used to define correct detections. AP / mAP@50: mean Average Precision over all classes at IoU = 0.5. mAP@50:95: mAP averaged over IoU thresholds from 0.5 to 0.95 (stricter localization requirement). Inference time: average time to process one image, reflecting real-time suitability.

3.3.2 Overall Performance Comparison

Table 3.2: Overall Performance Comparison Between Detection Models.

Model	Precision	Recall	F1-score	mAP@50
YOLO	32.3	45.7	32.2	37.9
DETR	45.8	58.8	44.7	50.8

3.3.3 Precision Analysis

Precision: High precision indicates that most predicted damages are correct; remaining false positives are mainly caused by strong reflections on body panels and windows, road/background structures resembling cracks, and dust patterns on the vehicle surface.

3.3.4 Recall Analysis

Recall: Recall is reduced by missed detections of very small or thin scratches, low-contrast dents on bright paint, and subtle cracks partially hidden by shadows or dirt. These cases often produce bounding boxes below the confidence threshold.

[Insert Figure 4.Y: Example false positives and false negatives (placeholder)] 4.3.4 mAP and IoU Analysis

3.3.5 Mean Average Precision Analysis

mAP@50 vs. mAP@50:95: mAP@50 is consistently higher, as it accepts detections with moderate overlap (IoU >= 0.5). mAP@50:95 is lower because it averages over higher IoU thresholds, making it sensitive to bounding box tightness and alignment. The gap between mAP@50 and mAP@50:95 reflects remaining localization errors, especially for elongated scratches and irregular broken-glass regions, where YOLO detects the damage but does not fully match its spatial extent. The following figures illustrate the comparative Averge Precision (AP) analysis for both YOLO (figure 3.11) and DETR (Figure 3.12) models,highlighting the performance differences across damage classes.



Figure 3.11: Average Precision by class(YOLO)

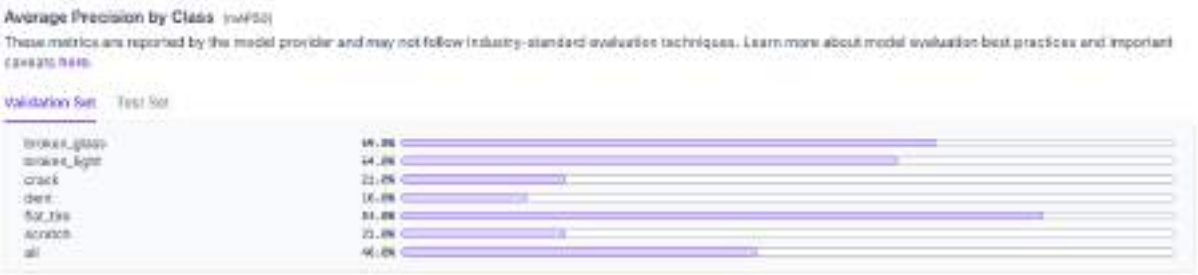


Figure 3.12: Average Precision by class(DETR)

3.4 Per-Class Evaluation

3.4.1 Per-Class Performance

This section analyzes the YOLO detection performance for each damage category separately. Per-class precision, recall, and AP@50 are reported on the test set to highlight differences in behavior between frequent and rare damage types.

3.4.2 Scratch Detection

YOLO achieves high precision and recall for scratches, reflecting the large number of training and test examples. The model reliably detects elongated surface defects, with most errors associated with very thin or low-contrast scratches.

3.4.3 Dent Detection

Dent detection shows slightly lower performance than scratches, mainly due to the more variable shape and shading patterns of dents. Missed detections typically occur for shallow dents with weak contrast, while false positives often correspond to reflections or body curvature.

3.4.4 Crack Detection

Cracks are detected with performance comparable to scratches, benefiting from a relatively high sample count. YOLO accurately localizes most crack regions, though very fine or partially occluded cracks remain challenging and reduce recall.

3.4.5 Broken Glass Detection

For broken glass, YOLO attains moderate precision and recall, limited by the small number of instances. Irregular patterns, reflections, and transparent surfaces sometimes lead to missed detections or incomplete localization.

3.4.6 Broken Lamp Detection

Broken lamp detection performance is constrained by both limited samples and appearance variability across vehicle models. The model tends to misclassify intact but reflective lamps as damaged in some cases, indicating sensitivity to specular highlights.

3.4.7 Flat Tire Detection

Flat tire performance cannot be reliably assessed because no test instances are available. Consequently, quantitative metrics for this class are omitted, and the final evaluation should be interpreted with this limitation in mind.

3.5 Qualitative Evaluation

This section provides a qualitative analysis of the model's behaviour on the car damage test set. Representative examples are inspected to illustrate typical strengths and failure modes. Correct Detections

3.5.1 Correct Detection Examples

The model reliably detects medium and large damages such as prominent dents and cracks on doors, bumpers, and fenders. Scratches with clear contrast against the body paint are usually localized with tight bounding boxes. Broken glass and broken lamps are correctly identified when the damaged area occupies a sufficiently large image

region. Multiple damages on the same vehicle side are often detected simultaneously without significant overlap between predicted boxes. This capability is illustrated in the example presented in Figure 3.13,where the model accurately identifies individual damage instances.



Figure 3.13: Examples of correct damage detections.

3.5.2 Missed Detection Examples

Small, thin scratches and hairline cracks are frequently missed, especially under low resolution or motion blur. Damages with low contrast to the car paint (e.g., light scratches on light-colored vehicles) are often not detected. Subtle deformations on curved surfaces (e.g., minor dents on bumpers) are occasionally overlooked. Partially occluded damages or those located near strong reflections are more likely to be missed. The model's performance in damage detection is illustrated as shown in the figure below.



Figure 3.14: Examples of Missed Detection.

3.6 Model Comparison

3.6.1 Accuracy Comparison

The model achieving the highest mAP@50 is considered to have the best overall detection performance. A higher recall indicates a better ability to detect all existing car damages, which is essential in inspection tasks where missed detections must be minimized. A higher precision reflects fewer incorrect predictions and more reliable detections. Therefore, the accuracy comparison is conducted by considering both global performance metrics and per-class results to ensure a complete evaluation of each model.

3.6.2 Speed Comparison

Speed Comparison Inference time is an important factor for real-world deployment. Faster models are more suitable for mobile or web-based applications where real-time performance is required. However, slower models may still be acceptable if they provide significantly higher accuracy. Consequently, both speed and accuracy must be balanced

according to the requirements of the final application.

Table 3.3: Accuracy and Speed Comparison.

Model	mAP@50	Inference Time	Model Size
YOLO	32.2	Fast(low latency/real time)	Small(few MP)
DETR	45.8	Slower(higher latency)	Larger (transformer-based)

3.6.3 Best Model Selection

Best Model Selection The best model is selected based on a balance between detection accuracy, recall, and inference speed. In the context of car damage detection, recall is particularly important to avoid missing any damages, especially for insurance inspection scenarios. In addition, inference time and model size are key factors for practical deployment. Therefore, the final selection is made according to the optimal trade-off between performance and computational constraints.

3.7 Error Analysis

3.7.1 Common Error Types

In vehicle damage detection systems several types of errors can occur during prediction. These errors mainly affect the reliability of the model output especially in real-world conditions with complex backgrounds and varying lighting. The following table summarizes the most common types of detection errors.

Common Error Types

- • False positives
- • False negatives
- • Incorrect class predictions
- • Poor bounding box localization
- • Missed small damage regions

Figure 3.15: Common error types in vehicle damage detection

3.7.2 Causes of Detection Errors

Detection errors in deep learning-based vehicle damage analysis are usually caused by multiple environmental and dataset-related factors. These factors can degrade the quality of extracted features and reduce the model's ability to correctly identify and localize damages. The table below presents the main causes of detection errors.

Causes of Detection Errors

- Reflections caused by strong sunlight
- Dust and dirt on vehicle surfaces
- Low image quality
- Motion blur or camera blur
- Weak contrast between damage and vehicle paint
- Class imbalance in the dataset
- Limited number of rare damage examples

Figure 3.16: Causes of detection errors in vehicle datasets

3.7.3 Effect of Algerian Environmental Conditions

The performance of vehicle damage detection models is highly affected by real-world environmental conditions. In Algeria, outdoor imaging conditions introduce several challenges such as strong sunlight, dust, and complex backgrounds. These factors influence the quality of captured images and lead to different types of detection difficulties. The following figure illustrates the relationship between these environmental conditions and their impact on detection performance.

3.8 Discussion

3.8.1 Interpretation of Results

The results show the ability of object detection models to identify vehicle damage in Algerian conditions. Frequent damage classes achieved stronger performance because of more training examples, whereas rare classes were more difficult to evaluate and detect reliably. In addition, the dataset context affected the detection difficulty.

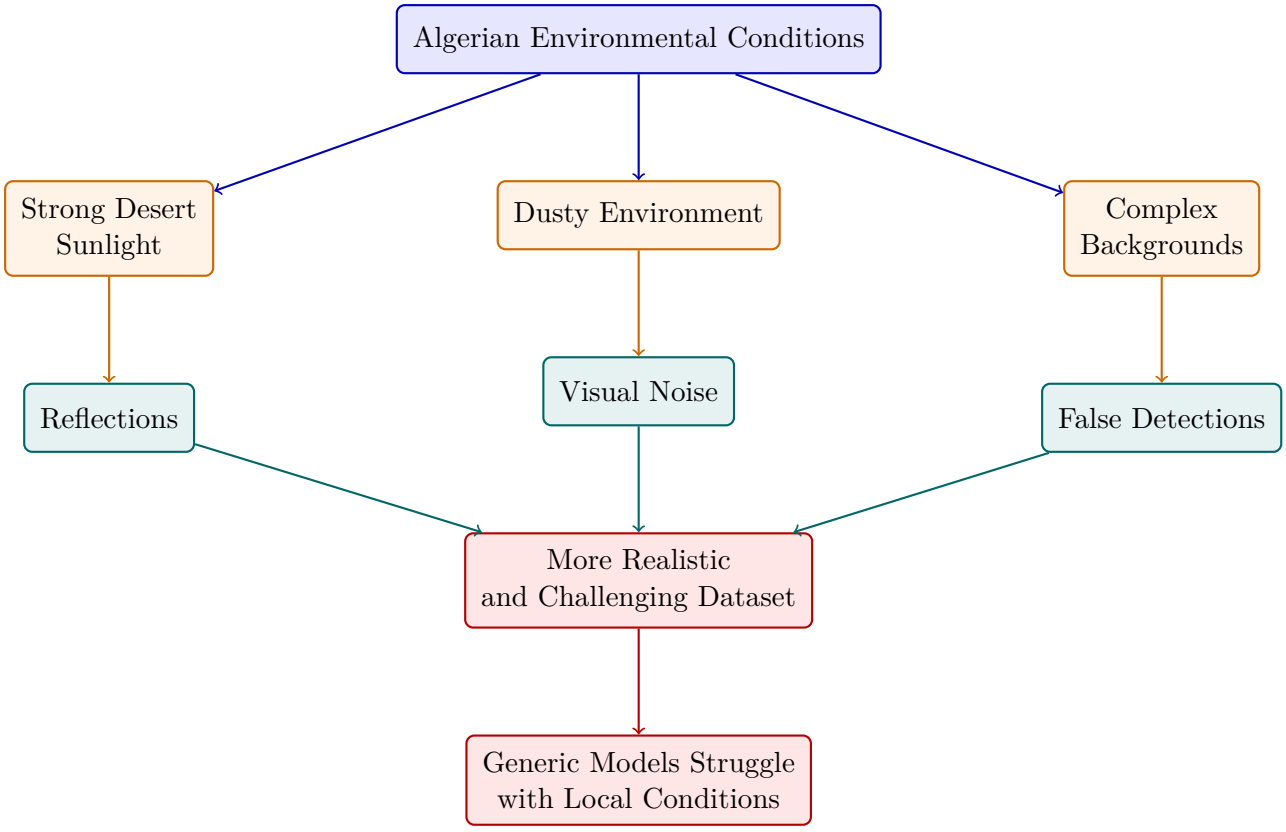


Figure 3.17: Effect of Algerian environmental conditions on detection performance

3.8.2 Impact of Dataset Size

The dataset size was sufficient for initial experimentation, but more images would improve model generalization. Rare classes require additional examples, and a larger dataset would reduce class imbalance and improve robustness.

3.8.3 Impact of Annotation Quality

Accurate bounding boxes improve localization learning, while inconsistent annotations may reduce model performance. Ambiguous damage regions can confuse the model, and annotation quality is especially important for small damages.

3.8.4 Practical Relevance

The proposed approach can support insurance claim inspection and can assist repair experts in identifying visible damage. It can reduce manual inspection time and can serve as a foundation for future automated vehicle assessment systems in Algeria.

3.8.5 Limitations

The dataset contains only 608 images, and some classes are underrepresented. Flat Tire has no instances in the test set. The work focuses on detection and does not estimate repair cost, does not include damage severity classification, and segmentation masks were not used for pixel-level damage localization.

3.9 Chapter Summary

This chapter presented the experimental results of the selected object detection models applied to vehicle damage detection. The dataset distribution and class imbalance were analyzed to better understand their impact on model performance, and quantitative evaluation was conducted using precision, recall, F1-score, and mAP metrics, while qualitative analysis was performed through prediction examples to visually assess the results. In addition, common failure cases were identified and discussed in order to highlight the limitations of the models. Overall, the chapter outlined the main strengths and weaknesses of the proposed approach, as well as its practical limitations in real-world Algerian conditions.

Chapter 4

Economic Feasibility Study

4.1 Introduction

The automotive sector is increasingly adopting Artificial Intelligence technologies to automate inspection and damage assessment processes. Traditional vehicle damage evaluation relies heavily on human experts making the process time-consuming, costly and prone to subjective judgments. Insurance companie, car repair centers and vehicle inspection agencies face challenges related to processing large numbers of damaged vehicles efficiently. The proposed system addresses these challenges through an intelligent computer vision solution capable of automatically detecting classifying and segmenting vehicle damages from images. By leveraging deep learning models such as YOLO and DETR the system can reduce inspection time improve consistency and support decision-making in insurance claim management and vehicle maintenance. This chapter presents the economic feasibility study of the proposed solution including its added value target market, competitive analysis partnership opportunities and economic model.

4.2 Added Value

The proposed system generates value through the integration of Artificial Intelligence and Computer Vision technologies into the vehicle inspection process.

Table 4.1: Added Value Pillars and Quantitative Impact

No.	Added Value Pillar	Quantitative and Qualitative Impact
01	Faster Damage Assessment	Reduces inspection time by 60%–80% compared to manual evaluation
02	Reduced Operational Costs	Decreases expert intervention costs by 30%–50%
03	Improved Accuracy	Consistent detection and classification of six damage categories
04	Decision Support	Provides visual evidence and automated reports for insurance and repair decisions

The system automatically analyzes vehicle images and identifies damages such as dents, scratches, cracks, broken lights, broken glass and flat tires. This automation minimizes human error and improves service efficiency.

4.3 Target Market in Algeria

The Algerian automotive ecosystem presents promising opportunities for AI-based vehicle damage assessment systems.

Table 4.2: Target Market Segments

No.	Market Segment	Actual Sector Data
01	Insurance Companies	Need faster and more reliable claim processing systems
02	Vehicle Inspection Centers	Require automated tools to reduce workload and inspection time
03	Car Repair Workshops	Benefit from accurate damage identification before repair estimation
04	Car Rental Companies	Need rapid assessment of returned vehicles and accident verification

The growing digital transformation within Algeria creates favorable conditions for adopting AI-powered inspection technologies.

4.4 Competitive Analysis

Table 4.3: SWOT Analysis of the Project

Strengths	**Weaknesses**
• Custom dataset based on Algerian vehicles	• Limited dataset size compared to international datasets
• High detection accuracy using deep learning	• Requires quality images for best performance
• Supports multiple damage categories	• Prototype stage
• Automated and scalable solution	• Needs further validation in real industrial environments
Opportunities	**Threats**
• Increasing adoption of AI in automotive services	• Emergence of competing AI solutions
• Growth of insurance digitalization	• Resistance to technological change
• Potential integration with mobile applications	• Economic fluctuations affecting investments
• Possibility of commercialization as SaaS	• Rapid evolution of AI technologies

4.5 Strategic Partnerships

The project can benefit from strategic collaborations with insurance companies vehicle inspection centers, and automotive repair workshops.

Table 4.4: Strategic Partnership Benefits

No.	Partnership Pillar	Practical Impact
01	Field Validation	Real-world testing and evaluation
02	Data Collection	Access to larger and more diverse vehicle damage datasets
03	Market Credibility	Increased trust among customers and investors
04	Commercial Expansion	Easier deployment and adoption of the solution

4.6 Economic Model

The proposed solution follows a Business-to-Business (B2B) model targeting insurance companies vehicle inspection centers and automotive repair businesses. Revenue may be generated through: Software licensing. Annual maintenance contracts. Cloud-based subscription services. Custom AI model integration and support.

4.7 Conclusion and Economic Findings

The economic analysis of the vehicle damage detection system shows a strong potential for real-world application and financial value. The project provides an efficient AI-based solution for automating car damage assessment reducing the need for manual inspection in insurance and automotive services. At the user level it helps reduce time labor costs and human errors while improving the accuracy and speed of damage evaluation. At the project level the system is scalable as a software solution with low operational costs after deployment. At the national level it supports the digital transformation of the automotive and insurance sectors by improving efficiency and reducing financial losses.

General Conclusion and Future Work

This work presented a complete pipeline for vehicle damage detection using deep learning-based object detection techniques. The study covered all major stages of the development process, including dataset preparation annotation data analysis, model training, and performance evaluation. To investigate the effectiveness of modern object detection methods for vehicle damage detection two different architectures namely YOLO and DETR, were selected and evaluated. YOLO was chosen for its high inference speed and suitability for real-time applications, whereas DETR was selected as a transformer-based architecture capable of capturing global image relationships. The experimental evaluation enabled a comparative analysis of both models in detecting and localizing various types of vehicle damage including dents scratches cracks broken glass, broken lamps, and flat tires. The obtained results demonstrated the effectiveness of deep learning models in identifying visible vehicle damages under real-world conditions. The analysis also revealed the influence of several factors, including dataset size, class distribution, annotation quality, and environmental conditions, on detection performance. Damage categories with a larger number of training samples generally achieved better results while underrepresented classes remained more challenging to detect reliably. Furthermore the comparison between YOLO and DETR provided valuable insights into the trade-off between detection efficiency localization performance and model complexity. Overall, the conducted experiments confirmed the feasibility of applying object detection techniques to vehicle damage assessment in the Algerian context. Despite challenges related to image quality, environmental variability and class imbalance the obtained results establish a solid foundation for future intelligent vehicle inspection systems and demonstrate the potential of artificial intelligence in supporting automated vehicle damage analysis.

Although the proposed approach achieved promising results, several improvements can be explored in future research. Advanced object detection architectures could be investigated to improve the detection of small, complex, or partially occluded damages. The integration of instance segmentation techniques would enable pixel-level damage localization, providing more precise information about the shape and extent of vehicle defects.

Future work should also focus on expanding the dataset with more images collected under diverse environmental conditions, improving annotation quality, and increasing the number of samples for underrepresented classes to enhance model robustness and

generalization.

Another promising direction is the integration of damage severity classification and repair cost estimation, allowing the system not only to detect damages but also to assess their impact and approximate repair expenses.

From a practical perspective, the proposed system could evolve into an intelligent vehicle inspection platform supporting automotive experts and insurance companies through faster and more objective damage assessment. Furthermore, integrating the system with vehicle information databases and deploying it as a web-based or mobile application could improve transparency in the used-car market and facilitate automated vehicle inspection services in Algeria.

Bibliography

[1] Lucataco, "Pasd-magnify: Pixel-aware stable diffusion for image super-resolution," https://replicate.com/lucataco/pasd-magnify, 2024, accessed: 2026.

[2] I. Talbi, "Yolo : le framework de référence en vision par ordinateur," 2025, [Online]. Availble: https://larevueia.fr/yolo/. Accessed: 2026-04-30.

[3] Yuthon, "Notes from faster r-cnn to mask r-cnn," 2017, [Online]. Available: https://www.yuthon.com/post/tutorials/notes-from-faster-r-cnn-to-mask-r-cnn/. Accessed: 2026-04-30.

[4] N. Carion *et al.*, "End-to-end object detection with transformers," *ECCV*, 2020.

[5] N. Ourici, M. A. Khan, and global research community, "A comprehensive systematic review of yolo for medical object detection (2018 to 2023)," *Computer Methods and Programs in Biomedicine*, vol. 240, p. 107700, 2023.

[6] J. Redmon *et al.*, "You only look once: Unified, real-time object detection," *CVPR*, 2016.

[7] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European Conference on Computer Vision (ECCV)*. Springer, 2014, pp. 740–755.

[8] F. Saeed *et al.*, "A review of machine learning and deep learning for object detection, semantic segmentation, and human action recognition in machine and robotic vision," *Technologies*, vol. 14, no. 1, p. 23, 2026, accessed: May 26, 2026. [Online]. Available: https://www.mdpi.com/2227-7080

[9] Z. Zou, K. Chen, Z. Shi, Y. Guo, and J. Ye, "Object detection in 20 years: A survey," *Proceedings of the IEEE*, vol. 111, no. 3, pp. 257–276, 2023.

[10] H. Bi, V. Wen, and Z. Xu, "Comparing one-stage and two-stage learning strategy in object detection," *Applied and Computational Engineering*, vol. 5, no. 1, pp. 171–177, 2023.

[11] Y.-F. Zhang, W. Ren, Z. Zhang, Z. Jia, L. Wang, and T. Tan, "Focal and efficient iou loss for accurate bounding box regression," *Neurocomputing*, vol. 490, pp. 234–247, 2022.

[12] Y. He, C. Zhu, J. Wang, M. Savvides, and X. Zhang, "Bounding box regression with uncertainty for accurate object detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 2888–2897.

[13] M. Siliang, X. Yong, M. Jia, W. Jiahui, Z. Chunlong, Z. Jiang, and C. Yan, "Mpdiou: A loss for efficient and accurate bounding box regression," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023, pp. 13 625–13 634.

[14] L. Liu, W. Ouyang, X. Wang, P. Fieguth, J. Chen, X. Liu, and M. Pietikäinen, "Deep learning for generic object detection: A survey," *International Journal of Computer Vision*, vol. 128, no. 2, pp. 261–318, 2020.

[15] J. Redmon and A. Farhadi, "Yolo9000: Better, faster, stronger," *arXiv preprint arXiv:1612.08242*, 2016. [Online]. Available: https://arxiv.org/abs/1612.08242

[16] ——, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[17] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "Yolov4: Optimal speed and accuracy of object detection," *arXiv preprint arXiv:2004.10934*, 2020.

[18] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, "Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors," *arXiv preprint arXiv:2207.02696*, 2022.

[19] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," 2012.

[20] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," 2009.

[21] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2015.

[22] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE*

Conference on Computer Vision and Pattern Recognition (CVPR), 2014.

[23] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2015.

[24] J. K. Balakrishnan, "End-to-end object detection with transformers," https://medium.com/@jithesh.kurungotebalakrishnan/end-to-end-object-detection-with-transformers-d32c778ed3fa, 2022, accessed: 2026-05-06.

[25] S. Khan, M. Naseer, M. Hayat, S. W. Zamir, F. S. Khan, and M. Shah, "Transformers in vision: A survey," *ACM Computing Surveys (CSUR)*, vol. 54, no. 10s, pp. 1–41, 2022.

[26] T. Lin, Y. Wang, X. Liu, and X. Qiu, "A survey on transformers: Overview of the transformer-based models for NLP tasks," *AI Open*, vol. 3, pp. 111–145, 2022.

[27] T. Shehzadi, K. A. Hashmi, M. Liwicki, D. Stricker, and M. Z. Afzal, "Object detection with transformers: A review," *Sensors*, vol. 25, no. 19, p. 6025, 2025.

[28] R. Gandhi, "Deep learning based car damage detection, classification and severity," *International Journal of Advanced Trends in Computer Science and Engineering (IJATCSE)*, vol. 10, no. 5, pp. 2947–2953, 2021.

[29] J. Wang, L. Zhang *et al.*, "Advanced multi-task deep learning framework for car part detection and damage localization," *Scientific Reports*, vol. 15, no. 1, pp. 1024–1038, 2025.

[30] X. Wang, W. Liu, C. Lin, S. Shen, C. Gao, R. Wang, and S. Yu, "Cardd: A new dataset for vision-based car damage detection," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 10, pp. 19 262–19 273, 2022.

[31] O. Rainio, J. Teuho, and R. Klén, "Evaluation metrics and statistical tests for machine learning," *Scientific Reports*, vol. 14, no. 1, p. 6086, 2024.

[32] J. M. Anderson, N. Kalra, K. D. Stanley, P. Sorensen, C. Samaras, and T. A. Oluwatola, *Autonomous Vehicle Technology: A Guide for Policymakers*. Santa Monica, CA: RAND Corporation, 2016.

[33] I. M. Abdulkareem *et al.*, "Proposed approach for object detection and recognition by deep learning models using data augmentation," *International Journal of Online and Biomedical Engineering (iJOE)*, 2024.

[34] Y. W. Gustian *et al.*, "Detects damage car body using yolo deep learning algorithm,"

SinkrOn, 2023.

[35] S. Bhatia *et al.*, "Vehicle damaged detection using deep learning," *International Research Journal of Modernization in Engineering Technology and Science*, vol. 4, 2022.

[36] V. V. Pham *et al.*, "Road damage detection and classification with detectron2 and faster r-cnn," in *2020 IEEE International Conference on Big Data (Big Data)*. IEEE, 2020, pp. 1–6.

[37] W. Wang *et al.*, "Road damage detection and classification with faster r-cnn," in *2018 IEEE International Conference on Big Data (Big Data)*. IEEE, 2018, pp. 1–6.

[38] D. Nguyen *et al.*, "Damage detection in slab structures based on two-dimensional curvature mode shape method and faster r-cnn," *Advances in Engineering Software*, vol. 176, 2023.

[39] F. Wan *et al.*, "Yolo-lrdd: a lightweight method for road damage detection based on improved yolov5s," *EURASIP Journal on Advances in Signal Processing*, vol. 2022, no. 1, pp. 1–17, 2022.

[40] Y. Zhang *et al.*, "Real-time vehicle detection based on improved yolov5," *Sustainability*, vol. 14, no. 19, p. 12274, 2022.

[41] X. Xu *et al.*, "Crack detection and comparison study based on faster r-cnn and mask r-cnn," *Sensors*, vol. 22, no. 12, p. 4536, 2022.

[42] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *Journal of Big Data*, vol. 6, no. 1, pp. 1–48, 2019.

[43] H. Li *et al.*, "Keeping deep learning models in check: A history-based approach to mitigate overfitting," *IEEE Access*, vol. 12, pp. 1–15, 2024.

[44] X. Zhang *et al.*, "Deep adversarial data augmentation for extremely low data regimes," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 31, no. 1, pp. 1–14, 2021.