

City Research Online

City St George's, University of London

Citation: Hogan, M. J. (2025). Explainable AI for object detection from autonomous vehicles. (Unpublished Doctoral thesis, City, University of London)

This is the accepted version of the paper.

This version of the publication may differ from the final published version. To cite this item please consult the publisher's version.

Permanent repository link: https://openaccess.city.ac.uk/id/eprint/34798/

Copyright and Reuse: Copyright and Moral Rights remain with the author(s) and/or copyright holders. Copies of full items can be used for personal research or study, educational, or not-for-profit purposes without prior permission or charge, unless otherwise indicated, provided that the authors, title and full bibliographic details are credited, a hyperlink and/or URL is given for the original metadata page and the content is not changed in any way. For full details of reuse please refer to City Research Online policy.

City Research Online: http://openaccess.city.ac.uk/ publications@citystgeorges.ac.uk

Explainable AI for Object Detection from Autonomous Vehicles

Maxwell John Hogan

Supervisor: Prof. Nabil Aouf

This dissertation is submitted for
the degree of Doctor of Philosophy



School of Science & Technology

Department of Engineering

February 2025

Contents

List of Acronyms **xvi**

Declaration **xviii**

Abstract **xix**

Acknowledgements **xxi**

1 Introduction and Background **1**

1.1 Motivation 1

1.2 Thesis Outline 4

1.2.1 Objective 4

1.2.2 Outline and Contributions 4

1.3 Publications and Submitted Manuscripts 6

1.4 Theoretical Concepts 7

1.4.1 Autonomous Systems 7

1.4.2 Foundations of Automatic Perception in Autonomous Systems 8

1.4.3 Artificial Intelligence 12

1.4.3.1 Machine Learning 13

1.4.3.2 Deep Learning 14

1.4.3.3 Optimisation 16

1.5 Object Detection from Autonomous Vehicle Platforms 16

1.6 Transparency and Explainability in Deep Learning 21

1.6.1 Properties of Explanations 22

1.6.2 Local Explanations with Saliency Maps 24

1.6.3 Perturbation Based Explainers 26

1.6.4 Backpropagation-Based and Gradient-Based Explainers . . . 29

1.6.5 SHAP and DeepSHAP 35
1.7 Evaluating Explainers and Explanations 36

2 Development of Software Tools and Datasets 40
2.1 Introduction and Motivation 40
2.2 Available Datasets 44
2.2.1 Existing XAI Datasets and Validation Procedures 44
2.2.2 The VisDrone Dataset 49
2.3 Sensor System Architecture and Software Tools 51
2.3.1 Sensor Rig 52
2.3.2 Capture Software 54
2.3.3 XAI Labelling Software 55
2.4 XAI dataset for Ground-based Drones 58
2.4.1 Environment 58
2.4.2 Additional Descriptive Labels 60
2.4.3 Bounding Box Distribution 63
2.5 Deep Object Detection Evaluation 67
2.6 Wrapping Game Analysis 67
2.7 Summary 71

3 Grad-CAM Based Explainers for Object Detection from Drone Platforms 73
3.1 Introduction 73
3.2 Methodology 75
3.2.1 Tile Dataloader for Small Object Detection 75
3.2.2 Grad-CAM with YOLOv5 76
3.2.3 Detection Constrained Grad-CAM with YOLOv5 78
3.3 Aerial Object Detection Results 80
3.3.1 VisDrone Challenge Metrics 80
3.3.2 Tile Loader Appraisal 82
3.3.3 Timing Characteristics 86
3.4 Validation of Grad-CAM Based Explainer 87
3.4.1 Model Dependency Test 88
3.4.2 Data Dependency Test 89
3.4.3 Wrapping Game Analysis 90

3.4.4 Diagnosing Deep Detector Reasoning with Grad-CAM 93
3.5 Summary 95

4 Explainable Object Detection for Autonomous Vehicles using KernelSHAP 99
4.1 Introduction and Motivation 99
4.2 Methodology 102
4.2.1 Extracting Features 103
4.2.2 Similarity Metric 105
4.2.3 Creating the Saliency Map 106
4.3 Experimental Setup 107
4.4 Qualitative Analysis 110
4.4.1 Identifying the Deliberate Bias in SHAP Maps 110
4.4.2 Identifying Failure Modes 111
4.4.3 Comparing Model Behaviours 112
4.5 Quantitative Evaluation 113
4.5.1 The Pointing Game 114
4.5.2 Evaluation of Bias Detection 116
4.5.3 Deletion and Insertion 117
4.5.4 Wrapping Game Analysis 120
4.6 Summary 121

5 DetDSHAP: Explainable Object Detection for Uncrewed and Autonomous Platforms with Shapley Values 123
5.1 Introduction and Motivation 123
5.2 Related Work 126
5.3 Methodology 129
5.3.1 DeepSHAP for Object Detection (DetDSHAP) 129
5.3.2 DetDSHAP Pruning Framework 134
5.4 Experimental Results 136
5.4.1 Wrapping Game Analysis 139
5.4.2 Pruning Toy Models 140
5.4.3 Pruning Performance 143
5.4.4 Post-Pruning Network Performance Quality 150
5.4.5 Pruning Verses Design 155

5.5 Summary 156

6 Conclusion **158**

6.1 Overview 158

6.2 Summary and Discussion 158

6.3 Future Work 162

6.3.1 Extending the XAI-AV Dataset 162

6.3.2 Local to Global Explanations 162

6.3.3 Advancing Pruning Framework 163

List of Figures

1.1 City St George's Autonomous Drone Platform[19]. 7

1.2 Chronological progression of key advancements in image perception, highlighting major feature extraction techniques and the early adoption of deep learning for classification and object detection. This timeline focuses on foundational contributions leading to modern deep learning approaches, which are further discussed in Section 1.5. . . . 9

2.1 Figure of the capture vehicle - a Renault Twizy. The sensors utilised include; left and right thermal cameras, a 32 channel Lidar, and an active RGBD camera. 41

2.2 This figure shows the bounding box statistics for the coco dataset. The left subplot shows the width and height distribution of the bounding boxes as a percentage of the image dimensions. The right subplot shows the centroid location distribution of bounding boxes in pixels. Histograms in both subplots provide additional insight into the frequency of widths, heights, and locations. 47

2.3 This figure shows the bounding box statistics for the instances of "Car" the Kitti dataset. The left subplot shows the width and height distribution of the bounding boxes, whereas, right subplot shows the centroid location distribution of bounding boxes. Histograms in both subplots provide additional insight into the frequency of widths, heights, and locations. 48

2.4 This figure shows the bounding box statistics for the instances of "Pedestrian" in the Kitti dataset. The left subplot shows the width and height distribution of the bounding boxes, whereas, right subplot shows the centroid location distribution of bounding boxes. Histograms in both subplots provide additional insight into the frequency of widths, heights, and locations. 49

2.5 Scatter plot showing the width and height of the bounding boxes in the VisDrone training set in pixels. The top histogram shows the occurrences of a given width, while the right side histogram shows the occurrence of a given height. The plot is limited to 400 pix, although there are some boxes wider than this 50

2.6 Examples from the VisDrone Detection Dataset[148] 51

2.7 Overview of the sensor system architecture, showing the Intel NUC capture PC connected to various sensors, including thermal cameras, Light Detection and Ranging (LiDAR), RGBD camera, and IMU. Additional power was required for some sensors, this is indicated in the diagram, others were able to powered from USB connection to the Capture PC. 52

2.8 This illustration shows the visual sensor data available in the proposed dataset. All the sub-figures show sensor feeds taken from the same frame in time. 53

2.9 Functional overview of the software threads, in which each cycle is composed of three steps. Step 1: the central thread sends a trigger message to all sensor threads simultaneously. Step 2: Each thread will take the most recent frame and save it to the filesystem. Step 3: The threads send a message back to the central thread to notify that the save is complete. 54

2.10 This figure shows the timing characteristics of the developed capture software. 56

2.11 This diagram outlines the process flow for labeling datasets using the developed software. It covers the stages from inputting RGB images, processing through AI and human agents, to the final output of labels which are stored in XML format. 57

2.12 This figure demonstrates the interface for creating the semantic labels on a given instance. The left image shows the initial guesses produced by the AI agent in white. The right image shows the refined labels by the human agent. Also indicated are some corrections that are typically required to be made by the human. 57

2.13 On the left portion of this figure is an image taken from our sensor rig while the right side shows the semantic label for this image. . . . 58

2.14 This figure shows an extract from the page that the human agent uses to set the Additional Descriptive Labels (ADLs). In this example the user is labelling the white van marked with the orange box. A full overview of the available labels are shown in Fig. 2.16. 58

2.15 This figure shows the routes taken during data acquisition where the green marker indicates the starting location and the red the finish. These have been produced using OpenStreetMap[153] 59

2.16 This figure shows a tree diagram which illustrates the links of the our ADLs. Each ADL is indicated in green. The class labels belonging to 'Body Shape' and 'Vehicle Colour' and their distribution can be found in Fig. 2.17. The Class labels for Observable Facet and their distribution can be found in Fig. 2.18. 60

2.17 This matrix shows the distribution of the vehicle descriptions. 61

2.18 This graphic shows the distribution of observable vehicle facets in our dataset. The label of 'Not Applicable' refers to situations where there is too much occlusion or the vehicle is too far away for the observer to determine. 62

2.19 This graphic shows the states of the Traffic lights in our dataset. . . 62

2.20 Comparison of bounding box sizes for the 'Vehicle' and 'Pedestrian' classes in the proposed dataset. The left subplot shows the width and height distributions for vehicles, while the right subplot shows the same for pedestrians. Histograms along the top and right provide further detail on the occurrences of specific width and height values. 63

2.21 Comparison of bounding box centroid locations for the 'Vehicle' and 'Pedestrian' classes in the proposed dataset. The left subplot shows the distribution of centroid locations for vehicles, while the right subplot shows the same for pedestrians. Histograms provide insight into the frequency of centroid locations along the width and height of the image. 64

2.22 This figure shows the bounding box statistics for instances of Traffic Light in the proposed dataset. The left subplot shows the width and height distribution of the bounding boxes in pixels. The right subplot shows the centroid location distribution of bounding boxes. Histograms in both subplots provide additional insight into the frequency of widths, heights, and locations. 65

2.23 These plots show the outcome of the Wrapping Game analysis on the DetDSHAP explainer developed in this study. 69

3.1 Illustration of an image with the tiles overlaid, numbered 0 to 5. The groundtruth bounding boxes are shown as solid, slightly transparent, rectangles. 76

3.2 Illustration of an image with the tiles overlaid, numbered 0 to 5. The groundtruth bounding boxes are shown as solid, slightly transparent, rectangles. 77

3.3 Illustration of the proposed Grad-CAM based algorithm to generate heatmaps from a Yolov5 model. Step 1: The input image and target class are selected a priori. Step 2: Network conducts a forward pass on the input image to produce predicted boxes, at the same time feature maps from each detection layer are stored. Step 3: Backpropagation occurs from the encoded predictions and Gradients are calculated. Step 4: Heatmaps are generated for each detection layer from the stored feature maps and calculated gradients. 78

3.4 Saliency maps generated from the convolution layers prior to each detection layer. The explicant in this case was YOLOv5l trained on the VisDrone dataset. The target class was 'car'. 79

3.5 Saliency maps for the 'Truck' class score, and objectness score for the final detection layer. Seven trucks were detected on this layer, these have been identified by the black bounding boxes. 79

3.6 This figure illustrates the individual saliency maps that can be generated with the bounding box-constrained Grad-CAM. The maps for the vehicles are shown along the bottom, and the maps for the Pedestrians are shown along the right side. 80

3.7 Detection restrained Saliency maps showing the attention of the deep network. These Saliency Maps are created by summing the maps for the individual objects. The right column displays raw maps from the shallow, middle, and deep detection layers, while the left column shows these maps overlaid on the input image. 81

3.8 Plots to show the performance of YoloV5 at different object scales. . 83

3.9 Plots to show the performance of CenterNet-ResNet50 at different object scales. 84

3.10 Confusion matrix illustrating the model's performance without the Tile Loader. 85

3.11 Confusion matrix illustrating the model's performance with the Tile loader 86

3.12 Table showing the Cascading randomisation on YOLOv5. 88

3.13 Top row: Grad-CAM feature maps for 'car' on the three detection layers. Bottom row: Images that were masked using the Grad-CAM maps to obscure important features. 90

3.14 The Graph shows the number of matched boxes in blue, and the number of unmatched boxes in red, for each class after salience-weighted perturbation is applied. 91

3.15 Change in confidence score due to salience-weighted perturbation. . 92

3.16 Plots showing the Wrapping Game analysis of the proposed Grad-CAM framework with YOLOv5 on the summer set. (a) analysis across all instances in the dataset. (b) analysis considering the model's confidence in a given instance. 93

3.17 Plots showing the Wrapping Game analysis of the proposed Grad-CAM framework with YOLOv5 on the winter set. (a) analysis across all instances in the dataset. (b) analysis considering the model's confidence in a given instance. 93

3.18 This figure shows the two samples used in the diagnosis of the deep detector's reasoning which was conducted using Grad-CAM. The bounding boxes predicted by the deep network have been plotted. 94

3.19 Saliency maps showing the attention of the deep network on sample 1 highlighting the network's attention for the class 'Vehicle'. 96

3.20 Saliency maps showing the attention of the deep network on sample 2 highlighting the network's attention for the class 'Vehicle'. 97

3.21 Saliency maps showing the attention of the deep network on sample 1 highlighting the network's attention for the class 'Pedestrian'. . . . 97

3.22 Saliency maps showing the attention of the deep network on sample 2 highlighting the network's attention for the class 'Pedestrian'. . . . 98

3.23 Saliency maps showing the attention of the deep network trained on sample 2. The first row displays the overlaid maps from the shallow, middle, and deep detection layers, while the second row shows the corresponding raw maps, highlighting the attention for the class 'Traffic Light'. 98

4.1 An example of the KernelSHAP explanation. (a) Shows the image with the target bounding box in blue. (b) Shows the saliency map that indicates the important parts of the object that caused the network to predict this box. 101

4.2 Overview of the proposed KernelSHAP framework. 102

4.3 (a) shows the feature boundary in white for the bounding box for the bus in yellow. (b) shows the features that are extracted using SLIC on the left, and an example of a perturbed image on the right. . . . 103

4.4 Examples of masks applied to the image with the nearest matched bounding box. 105

4.5 Saliency Map created using the proposed method showing the regions of the bus that are important to the Deep Neural Network (DNN). . 107

4.6 Loss plots for the training and validation sets during training of the biased and control model. Only the training set for the biased model contained the artificial bias, although both models were evaluated on the same validation set. 108
4.7 An example from the biased dataset showing the marker over the top of cars. Note the black van on the right side of the image does not have a marker. 109
4.8 Input image with the groundtruth bounding box in orange, with the marker present. 111
4.9 SHAP map generated for the biased model of a car with the bias present. 112
4.10 SHAP map generated for the control model of a car with the bias present. 113
4.11 Saliency maps generated from the biased model on the predicted bounding box without the bias present in the input. 114
4.12 Saliency maps generated from the biased model on the groundtruth bounding box without the bias present in the input. 115
4.13 The saliency maps generated of an object of class 'bus' using the groundtruth bounding box 116
4.14 Image with the groundtruth bounding box used to compare the behaviour of different architectures. 117
4.15 SHAP map comparison for Faster R-CNN and YOLOv5. The first row displays raw and overlay SHAP maps for Faster R-CNN, and the second row shows the corresponding maps for Yolov5. 118
4.16 Plot showing the accuracy of the KernelSHAP framework at identifying the bias when considering different Intersect over Union (IoU) thresholds, and three scales of bounding boxes 119
4.17 Plots showing the similarity in relation to (a) the amount of pixels removed from the image, and (b) the amount of pixels added back to an image. 119
4.18 Plots showing the Wrapping Game analysis of the proposed KernelSHAP framework with YoloV5. (a) analysis across all instances in the dataset. (b) analysis considering the model's confidence in a given instance. 121

4.19 Plot showing the Wrapping Game analysis of the proposed KernelSHAP framework, with Faster R-CNN for all instances in the dataset. . . . 122

5.1 The domain of the present work. While literature exists on one or the intersection of two domains represented by each circle, this study introduces a connection between all three. 125

5.2 Generalised pruning framework [162] 127

5.3 Example of SHapley Additive exPlanations (SHAP) map generated using the approach introduced in this work with an image in the proprietary dataset. (L) The input image with the predicted box. (C) The SHAP map superimposed over the input image. (R) The SHAP map and the colour bar. 129

5.4 This figure illustrates the SHAP maps that can be generated for individual objects in the input image. The maps for the vehicles are shown along the bottom and the maps for the Pedestrians are shown along the right side. 131

5.5 An example of multi-class explanation. (a) Shows the image with the target bounding boxes of a person and a motorcycle. (b) Shows the area of interest around the bounding box, (c) The SHAP map generated for the motorcycle, (d) map for the rider. 132

5.6 This figure shows an overview of the YOLOv5 architecture. The figure also shows various SHAP maps extracted at different layers. 133

5.7 Examples from the three datasets used in the experiments: Simple AD (left), Visdrone (center), and COCO2017 (right). The COCO2017 images show objects of interest for this study. 138

5.8 Plots showing the Wrapping Game analysis of the proposed DetD-SHAP framework with YoloV5 for the Summer Set. (a) analysis across all instances in the set. (b) analysis considering the model's confidence in a given instance. 140

5.9 Plots showing the Wrapping Game analysis of the proposed DetD-SHAP framework with YoloV5 for the Winter Set. (a) analysis across all instances in the set. (b) analysis considering the model's confidence in a given instance. 141

5.10 Plots showing the Wrapping Game analysis of the proposed YOLO LRP framework with YoloV5 for the Summer Set. (a) analysis across all instances in the set. (b) analysis considering the model's confidence in a given instance. 141

5.11 Toy training sets used to fit the simple model. The decision boundaries are superimposed over each plot. 142

5.12 Toy validation sets used to evaluate the toy models after pruning. . . 142

5.13 DeepSHAP pruning at 33%. 143

5.14 LRP pruning at 33%. 143

5.15 DeepSHAP pruning at 50%. 144

5.16 LRP pruning at 50%. 144

5.17 This figure displays the pruning trend when using DetDSHAP as a criterion to prune YOLOv5s trained on proprietary self-driving car dataset. 145

5.18 This figure displays the pruning trend when using LRP as a criterion to prune YOLOv5s trained on the proprietary self-driving car dataset. 146

5.19 This figure displays the pruning trend when using DetDSHAP as a criterion to prune YOLOv5l trained on the Visdrone dataset. 147

5.20 This figure displays the pruning trend when using LRP as a criterion to prune YOLOv5l trained on the Visdrone dataset. 148

5.21 This figure displays the pruning trend when using DetDSHAP as a criterion to prune YOLOv5s trained the Coco2017 dataset. 149

5.22 This figure displays the pruning trend when using LRP as a criterion to prune YOLOv5s trained the Coco2017 dataset. 150

5.23 This figure depicts the precision-recall curves before and after pruning for the YOLOv5s that was trained on the self driving dataset. 151

5.24 This figure depicts the precision-recall curves before and after pruning for the YOLOv5m that was trained on the Coco2017 dataset. The solid lines denote the average trend for all classes in their respective dataset, and the dotted lines represent the best and worst cases across all classes. 152

5.25 This figure depicts the precision-recall curves before and after pruning for the YOLOv5l that was trained on the Visdrone dataset. The solid lines denote the average trend for all classes in their respective dataset, and the dotted lines represent the best and worst cases across all classes. 153
5.26 Confusion matrix of the unpruned YOLOv5l network trained on the Visdrone Dataset. 154
5.27 Confusion matrix of the YOLOv5l network trained on the Visdrone Dataset after pruning has been applied. 155
5.28 This plot shows the performance quality and structure efficacy of the pruned YOLOv5m model compared to two other models released by [72]. Training and evaluation are conducted on the COCO2017 dataset. 157

List of Acronyms

AD	Autonomous Driving
ADAS	advanced driver assistance systems
ADL	Additional Descriptive Label
ANN	Artificial Neural Network
API	Abstract Programming Interface
AUC	Area Under Curve
AV	Autonomous Vehicles
BVLOS	Beyond Visual Line of Sight
CAA	Civil Aviation Authority
CAM	Class Activation Mapping
CNN	Convolutional Neural Network
CRP	Contrastive Relevance Propagation
CUDA	Compute Unified Device Architecture
DfT	Department for Transport
DNN	Deep Neural Network
FLOPs	Floating point OPerations per Second
FoV	Field of View
FPN	Feature Pyramid Network
FPS	Frames per Second
GAP	Global Average Pooling
GFLOPS	Giga Floating Point Operations Per Second

GPS	Global Positioning System
GPU	Graphics Processing Unit
IG	Integrated Gradients
IMU	Inertial Measurement Unit
IoU	Intersect over Union
LiDAR	Light Detection and Ranging
LRP	Layer-wise Relevance Propagation
mAP	mean average precision
NMS	non-maximum suppression
PCA	Principal Component Analysis
PFPN	Panoptic Feature Pyramid Network
ReLU	Rectified Linear Unit
SHAP	SHapley Additive exPlanations
SLAM	Simultaneous Localisation and Mapping
SSD	Single Shot Detector
UAV	Uncrewed Aerial Vehicle
VLOS	Visual Line of Sight
XAI	Explainable Artificial Intelligence
YOLO	You Only Look Once

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements.

Maxwell Hogan

February 2025

Abstract

To tackle the complex tasks required for autonomous vehicle operations, robust perception capabilities often depend on multiple deep neural networks (DNNs). However, the opaque nature of DNN detection algorithms presents challenges in transparency, leading to unpredictable behaviour in critical applications, such as scene and object recognition for uncrewed aerial vehicles (UAVs), and self-driving cars. This research aims to bridge this transparency gap by developing explainability methods that enhance trust in networks used on autonomous vehicles.

This study addresses three major limitations in current literature. First, state-of-the-art object detection networks exhibit reduced performance with aerial imagery from UAVs, highlighting the need for robust solutions. Second, most existing explainability techniques focus on image classification, rather than object detection, leaving an important gap. Third, a lack of standardised validation methods for explanations presents many ongoing challenges.

To address these issues, this thesis introduces three novel explainability frameworks for deep object detection. The first framework adapts Grad-CAM for the YOLOv5 detector, generating explanations for class scores, objectness scores, and bounding box coordinates, while evaluating real-time performance. Building on these insights, the second explainer, a KernelSHAP-based framework, introduced a model-agnostic approach to explain object detection across architectures. Finally, the DetDSHAP framework offers a propagation-based method that, not only calculates contributions of individual pixels to a predicted bounding box, but also how discrete units of the DNN played a role in the predictions. The DetDSHAP was employed to optimise model performance through pruning.

Additionally, a novel "Wrapping Game" approach is proposed to validate the reliability of explainers in high-stakes edge cases, providing a measure for the discriminative power of explanations. This work is further supported by the develop-

ment of the XI (eXplainable Intelligence) Autonomous Driving dataset, tailored to autonomous vehicle challenges, which enables rigorous testing of explainability techniques in real-world scenarios. Together, these contributions form a comprehensive framework to enhance the interpretability of deep object detection models, ensuring autonomous vehicle systems are both effective and trustworthy.

Acknowledgements

I would like to extend my sincere gratitude to my supervisor and mentor, Prof. Nabil Aouf, for his unwavering guidance, encouragement, and invaluable feedback. The knowledge I have gained through our many technical discussions has been instrumental in shaping the ideas presented in this thesis, and has contributed massively to my personal growth as a researcher.

I am also deeply grateful to the Robotics, Autonomy and Machine Intelligence group at City, St George's, for their invaluable support and motivation throughout this journey.

I gratefully acknowledge the funding provided by the Defence Science and Technology Laboratory, which has been instrumental in supporting my PhD research.

I would also like to acknowledge, in no particular order, Youngseob Dae, Chuyao Wang, Mohammed Baira, and Xi Guo for their contribution to developing the XAI-AV dataset that is presented in this thesis.

I would like to express my heartfelt gratitude to my family, particularly my parents, Paul and Stacey Hogan, for their endless encouragement, understanding, and support over the past four years. Their assistance with proofreading and unwavering belief in me has been invaluable throughout. Lastly, I want to acknowledge my sister, Brontë (1998-2011), whose memory I carry with me every day and whose presence continues to inspire me.

Chapter 1

Introduction and Background

In this introductory chapter, the reader is provided the current context from which this work arises. They will be made aware of modern autonomous vehicles and how deep learning is beginning to play an important role in how Autonomous Vehicles (AV) perceive their environment, and objects around them, in order to undertake the tasks required of them. Within this important context, this chapter will then present the reader with the role that explainable artificial intelligence can play in the development of trustworthy automatic perception algorithms for autonomous vehicles.

1.1 Motivation

Machine learning has become synonymous with high performance in numerous tasks, driving innovation across many real-world applications. However, due to the inherent "black-box" nature of many machine learning models, particularly DNNs, their behaviour can be difficult to predict or interpret. When failures occur, identifying the cause is challenging, raising issues of trust and safety, particularly for deploying DNNs in real-world scenarios. Moreover, in some settings, regulations may require transparency in algorithmic decision-making processes [1]. Consequently, there has been considerable growth in the field of Explainable Artificial Intelligence (XAI), with the development of techniques that can help identify network biases [2], assist in network debugging [3], and offer alternative evaluation methods [4].

In the UK, the Civil Aviation Authority (CAA) is drafting regulations for Uncrewed Aerial Vehicle (UAV) operating Beyond Visual Line of Sight (BVLOS) due to the expected rapid growth in demand for such operations. This mode of opera-

tion is a core component of a recently released strategy for the modernisation of UK airspace [5]. The CAA's strategy emphasises moving from segregated to integrated, unsegregated airspace where BVLOS-capable drones will operate alongside manned aircraft without special provisions. Currently, this integration is limited by technological gaps in collision avoidance and airspace management, and conventional aircraft often cannot easily visually detect small UAV. Similarly, drones currently lack the reactive capabilities of human pilots to avert certain potential collisions. For now, segregated airspace is necessary to maintain safety.

The strategy envisions a phased approach, beginning with technology trials in managed airspace, to address these limitations. New advancements, particularly in detect-and-avoid systems, are prioritised to improve safety. As policies evolve and technologies mature, unsegregated BVLOS operations are expected to become feasible, with XAI-enhanced detection systems playing a key role in establishing the robustness and trustworthiness needed for UAV guidance systems.

The UK Department for Transport (DfT) code of practice for automated vehicle trials echoes these requirements, emphasising the need for intelligible data presentation [6][7]. For autonomous systems transparency in data interpretation is essential, requiring XAI to validate and secure AI-based systems used in UAVs and ground vehicles. The safe operation of these systems within regulatory frameworks, coupled with accessible insights for both operators and regulators, is essential to advancing autonomous capabilities across UK airspace and public roads.

Additionally, deep learning models face unique vulnerabilities to adversarial attacks and other manipulations. These attacks exploit inputs deliberately designed to mislead the model, which poses significant risks in high-stakes environments. As Molnar notes in [8], machine learning models, especially DNNs, can be manipulated by adversarial examples, underscoring the importance of interpretability in mitigating these risks. Moreover, as models become integral to safety-critical operations, they also become entry points for cybersecurity threats, emphasising the need for proactive interpretability measures to understand and anticipate model behaviour under potentially adverse conditions.

While explainability is often seen as a means of improving trust and security in machine learning models, recent research highlights that XAI methods themselves are susceptible to adversarial attacks. Adversaries can manipulate explanations through various means, including adversarial examples, data poisoning, model ma-

nipulation, and backdoor attacks. These attacks allow an explanation to be altered while the model's predictions remain unchanged, making them difficult to detect. For instance, an adversarially modified input can mislead saliency maps into highlighting irrelevant regions, concealing the true reasoning behind a detection. Similarly, model manipulation attacks can distort attribution methods such as SHAP or Grad-CAM, making an inherently biased model appear fair and transparent. The risks posed by such attacks extend beyond model auditing, as malicious actors could exploit XAI vulnerabilities to conceal security threats in safety-critical systems, such as autonomous vehicles or medical diagnostics [9].

Given these concerns, it is crucial to develop more robust interpretability techniques, as a single explanation method can be easily manipulated. By introducing ensemble explanations, where multiple XAI methods are cross-validated, the reliability of model interpretations can be significantly improved. By leveraging multiple saliency map methods for object detection, a more reliable and resilient understanding of model behaviour can be achieved, mitigating the risks associated with adversarial attacks on explanations. Additionally, standardised validation protocols and attack-resistant benchmarks are needed to assess the robustness of explainability methods in adversarial settings. Establishing such benchmarks is crucial to detect and mitigate adversarially manipulated explanations, ensuring that XAI methods remain robust under small perturbations or targeted attacks [9].

This study centres on object detection from an autonomous vehicle platform using 2D optical sensors. Object detection is critical to autonomous vehicle perception, providing essential information for tasks like obstacle avoidance, navigation, and dynamic scene understanding [10][11]. Saliency maps are a form of local explanation well-suited for image data. Saliency maps illustrate how each pixel, or group of pixels, contributes to a specific prediction. Saliency-based explainers can be divided into perturbation-based and propagation-based approaches. Perturbation methods, which treat the network as a closed box, offer high portability, making it easier to swap or modify networks. This approach ensures compatibility with any network architecture, eliminating trade-offs between explainability and performance. In contrast to perturbation-based methods, propagation-based methods tend to provide more detailed causal information from within the network, and generally provide the explanation faster. In this thesis, both approaches will be explored and contrasted.

While substantial research has focused on XAI techniques for classification tasks [12][4][13][14], object detection remains under-explored, particularly with regard to the unique challenges posed by autonomous platforms. As previously discussed, onboard object detection applications have grown considerably, and are vital for bringing autonomous vehicle platforms 'into the wild' to engage in the range of tasks required in urban applications, and in other theatres such as search and rescue, automatic target recognition, etc. Each of these safety-critical applications stands to benefit from XAI advancements.

1.2 Thesis Outline

1.2.1 Objective

Beyond the initial motivations, the overarching aim of this thesis is to develop explainability techniques tailored to the unique challenges of object detection in autonomous vehicle environments. These environments are characterised by cluttered, dynamic scenes and the added complexity of perceiving small objects. The methods presented here strive to create explanations that, not only reflect the model's true decision-making process, but also remain interpretable and accessible to human operators. In particular, this research has three main objectives:

1. To enhance transparency in deep neural network (DNN) object detection for autonomous vehicles by developing and evaluating explainability methods that improve interpretability, without compromising performance in real-time applications.

2. To address the limitations of existing explainability techniques for object detection by extending current methods beyond image classification, ensuring they are tailored to object detection networks.

3. To expand upon validation methodologies for explainability, that measure the discriminative power and reliability of explanations under edge-case conditions.

1.2.2 Outline and Contributions

The structure of the thesis along with each chapter's contribution are presented as follows:

Chapter 1: Introduction and Background

This chapter establishes the motivation and context for research into XAI with a focus on autonomous vehicle platforms, particularly drones and autonomous cars. It reviews the role of deep learning in enabling autonomous perception, and explores the need for transparency in AI models deployed in safety-critical scenarios. Key theoretical concepts are introduced, including autonomous systems, AI, and object detection, setting a solid foundation for the reader.

Chapter 2: Development of Software Tools and Datasets

This chapter details the creation and adaptation of tools and datasets tailored for the XAI research conducted in this work. This includes the design and implementation of a custom sensor rig for City St George's AV platform, and a new dataset, the XAI-AV dataset, to support object detection tasks under varied conditions. Furthermore, it introduces the "Wrapping Game" evaluation metric to assess the effectiveness of explainers produced for object detection models. The chapter contributes software and methodologies essential for validating XAI techniques on autonomous vehicle platforms.

Chapter 3: Grad-CAM Based Explainers for Object Detection from Drone Platforms

This chapter explores the adaptation of the Grad-CAM framework so that it can be utilised to explain objects detected in UAV-captured imagery with the goal of providing near real-time explanations. It includes the development of a tile-based dataloader designed to enhance the detection of small objects. Extensive validation, including the use of the "Wrapping Game," examines Grad-CAM's effectiveness in capturing model attention.

Chapter 4: Explainable Object Detection for Autonomous Vehicles using KernelSHAP

This chapter presents a novel adaptation of KernelSHAP for object detection. By addressing the challenge of portability, this approach facilitates compatibility across diverse model architectures. Quantitative and qualitative assessments verify the ability of this KernelSHAP adaptation to reliably allocate feature importance in

object detection, and showcases its flexibility for various deep learning models.

Chapter 5: DetDSHAP Explainable Object Detection for Autonomous Platforms with Shapley Values

This chapter introduces DetDSHAP, an extension of DeepSHAP, specifically designed for object detection tasks. The development builds upon findings in previous chapters to develop a precise and informative explainer. Moreover, the link between Deep network compression and Explainability is explored.

Chapter 6: Conclusion

The final chapter summarises the contributions made throughout the thesis, emphasising the advancements in explainability for Autonomous System perception capabilities. A discussion of the broader implications of the research addresses the challenges and potential in applying XAI to autonomous platforms. The chapter concludes by outlining possible avenues for future work.

1.3 Publications and Submitted Manuscripts

Conferences

1. M. Hogan and P. N. Aouf, "Towards real time interpretable object detection for uav platform by saliency maps," in *2021 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 2021, pp. 1178–1183

2. M. Hogan, N. Aouf, P. Spencer, and J. Almond, "Explainable object detection for uncrewed aerial vehicles using kernelshap," in *2022 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, 2022, pp. 136–141

3. M. Hogan and P. N. Aouf, "Explainable dataset for ground based drones in urban environments," in *2024 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, Accepted for publication, 2024

Journals

1. M. Hogan and P. N. Aouf, "Detdshap: Explainable object detection for uncrewed and autonomous drones with shapley values," *Trust through eXplAIn-*

ability (XAI), Robustness and Verification of Autonomous Systems, 2025, Manuscript in submission

1.4 Theoretical Concepts

This section introduces the fundamental concepts that underpin the contributions made throughout this thesis. It begins by providing an explanation of autonomous systems, focusing on mobile robots, particularly UAVs and Autonomous Driving (AD) systems. The discussion will then lead on to automatic perception algorithms, where it is established that the state-of-the-art are based upon DNNs.

1.4.1 Autonomous Systems

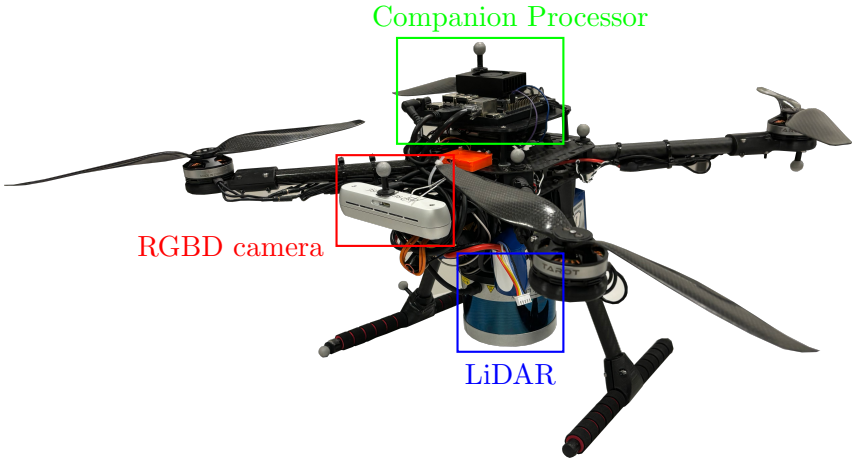


Figure 1.1: City St George's Autonomous Drone Platform[19].

An autonomous system is a self-operating machine capable of performing tasks without direct human intervention, relying on a combination of sensors, software, and control algorithms to perceive its environment, make decisions, and act accordingly. For mobile robots, this often includes navigating through complex environments, avoiding obstacles, and achieving specific goals, such as transporting goods, or monitoring an area.

An example of an autonomous aerial system, otherwise referred to as an UAV, is shown in Fig. 1.1. This quadcopter platform typically includes Global Positioning System (GPS) and Inertial Measurement Unit (IMU) to enable basic autonomous flight, such as way-point missions. However, due to current restrictions on BVLOS operations, such drones are generally required to operate within Visual Line of Sight

(VLOS) operations. Without additional sensors, the platform would be limited to basic tasks, such as package delivery.

The platform shown in Fig. 1.1 has been equipped with a sensor suite, including a Velodyne 32-line puck LiDAR [20], and an Intel RealSense D455f RGB-Depth camera [21]. These additional sensors enable the platform to perform more advanced tasks, such as aerial mapping, surveillance, and search and rescue missions.

To extend the onboard computing capabilities of UAV platforms, a companion processor can be mounted to embed more advanced guidance and control algorithms. The platform in Fig. 1.1, utilises the Nvidia Jetson Orin Nano [22], equipped with an Nvidia Ampere Graphics Processing Unit (GPU) featuring 1024 Compute Unified Device Architecture (CUDA) cores and 32 Tensor Cores. This setup allows the system to operate independently, making decisions on flight paths, obstacle avoidance, and target detection, even in dynamic environments.

Similarly, ground based autonomous systems, such as self-driving cars, can be equipped with comparable sensors along with advanced compute capabilities, allowing them to detect road conditions, recognise traffic signals, and identify other vehicles or pedestrians. The autonomous driving software processes this data in real-time, enabling the vehicle to navigate safely and efficiently without human input. An example of such a system in shown in Fig. 2.1.

Despite the recent advancements in embedded hardware, including the rapid progress in Nvidia's Jetson product range, the desired capabilities of autonomous systems remain limited in terms of computational power and performance. This restricts the feasibility of high-complexity computations, and is especially pronounced in applications requiring real-time responsiveness.[23]

For the autonomous systems discussed here, they are expected to be able to handle a range of scenarios and uncertainties, requiring robust perception and decision-making, to ensure safe and reliable operation. As will be discussed in subsequent sections, deep learning promises to vastly enhance the capabilities of autonomous systems, if such algorithms can be deployed safely.

1.4.2 Foundations of Automatic Perception in Autonomous Systems

As established, autonomous systems like UAV and self-driving cars, rely on a combination of sensors and software to operate independently to navigate through complex

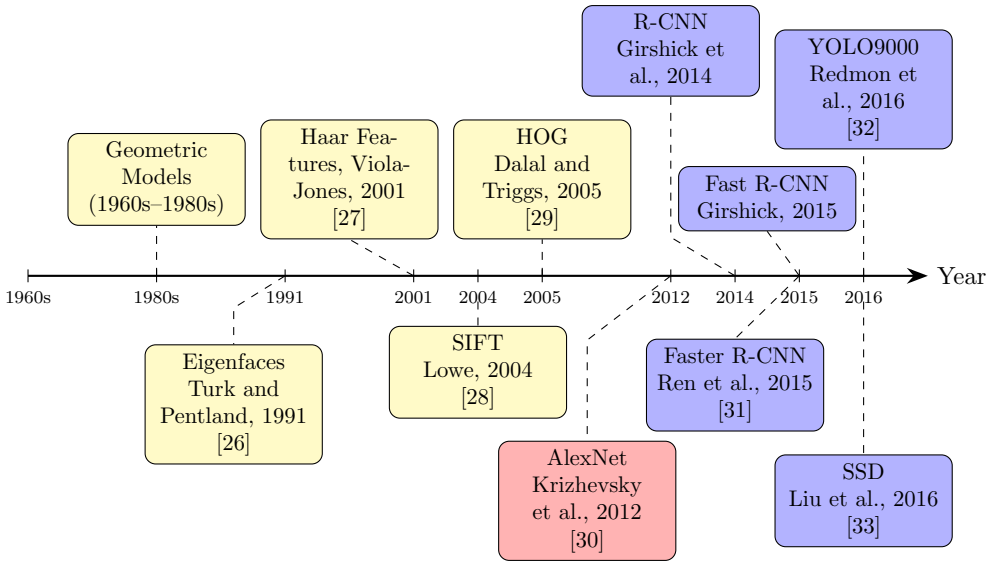


Figure 1.2: Chronological progression of key advancements in image perception, highlighting major feature extraction techniques and the early adoption of deep learning for classification and object detection. This timeline focuses on foundational contributions leading to modern deep learning approaches, which are further discussed in Section 1.5.

environments, and execute tasks without direct human input. It is not possible for the autonomous system to infer complex information, such as the presence of an object and its location, from raw sensor data alone. Hence, the sensor feeds of the system will be enhanced with multiple automatic perception algorithms as part of its software payload. Most applications use simple visual spectrum cameras due to their low cost and wide availability. However, there are vast bodies of work that also employ sensors of different modalities, such as thermal imagery [24], or 3D sensors like LiDAR [25].

This section traces the historical progression of automatic perception methods, from early human-defined feature extraction techniques to the emergence of deep learning. The selected milestones in Fig. 1.2 highlight key transitions that shaped modern perception systems. While not exhaustive, this timeline emphasises breakthroughs that laid the groundwork for current state-of-the-art models, which will be explored in Section 1.5, once a solid foundation has been established for the reader.

In this section we provide the reader with It can be considered that the field of automatic perception emerged in the 1960s, primarily relying on human-engineered features. Early approaches, known as geometric models, represented objects as sets of simple shapes, such as edges, lines, and polygons. These models laid foundational work in computer vision, demonstrated in applications like fingerprint classification [34] and satellite imagery analysis [35].

However, these models faced limitations in complex scenarios, as noted by Mundy's review [36]. Challenges included low contrast at object boundaries, cluttered backgrounds, and occlusions. Zisserman et al. achieved reasonable success in multi-object recognition within moderately cluttered scenes, by introducing class-specific geometric constraints for feature grouping and recognition [37]. However, even with these advancements, performance gains remained constrained by environmental variability, and the challenges posed by occlusions and viewpoint changes.

To address these limitations, appearance-based models emerged, focusing on visual features, such as texture, colour, and images gradients, to enhance robustness. A prominent example was the Eigenfaces method for face recognition, introduced by Turk and Pentland in the early 1990s [26]. Their approach used Principal Component Analysis (PCA) and represented faces as linear combinations of basis images. However, this methodology still required specific viewing conditions for accuracy.

Given the limitations of geometric and appearance-based techniques, these methods were ultimately incompatible with the demands of modern autonomous systems. It is recognised that the modern evolution of object detection only began in the past quarter-century, and can be divided into two eras: traditional object detection (pre-2014), and deep learning-based detection (post-2014) [38][11].

Handcrafted feature descriptors were widely used during the traditional object detection era. A substantial breakthrough came from P. Viola and M. Jones, who proposed a real-time human face detection algorithm that was several orders of magnitude faster than previous methods [27]. They employed Haar features, which utilise rectangular regions to capture edge information across an image. Combined with integral images, Haar features made real-time applications feasible, even with limited computing resources.

More advanced feature descriptors followed, including the Scale-Invariant Feature Transform (SIFT), introduced by David Lowe [28], and the Histogram of Oriented Gradients (HOG), developed by N. Dalal and B. Triggs [29]. These descriptors were instrumental in object recognition, as they provided robustness to scale, rotation, and moderate viewpoint changes. Both descriptors achieved prominence in the traditional era and beyond. SIFT, for example, was proposed as part of a framework for feature-based target recognition for UAVs [39]. HOG, initially developed for pedestrian detection, played a crucial role in this application for many years [40][41][42][43][44].

During this time, machine learning algorithms processed these extracted features. Support Vector Machines (SVMs) and decision trees became standard for classifying or detecting objects based on feature sets, providing a structured, data-driven approach to interpretation. In particular, AdaBoost, used by Viola and Jones in their face detection method [27], selected the most informative features, forming an efficient cascade classifier by combining weak learners (simple classifiers) into a robust ensemble.

The introduction of AlexNet in 2012 [30] marked a paradigm shift in object detection, as convolutional neural networks (Convolutional Neural Networks (CNNs)) replaced the need for hand-engineered features. This pivotal transition is highlighted in Fig. 1.2 with a red marker, indicating its significance in the shift toward deep learning-based perception. The timeline further distinguishes the evolution of CNN-based object detection models, shown in blue, to the prevalence of such architectures post-2012.

This innovation marked a turning point, as CNNs bypassed the need for hand-engineered features, making object detection more scalable, accurate, and adaptable to complex environments. From this point forward, object detection methods would increasingly rely on deep learning to drive performance improvements. These new techniques are indicated in the timeline with blue boxes. For the reader's understanding, deep learning and machine learning will be further defined in Section Section 1.4.3.

In the years following AlexNet, numerous CNN-based models were developed to address both classification and localisation, transforming object detection into a field dominated by deep architectures. Models like R-CNN (Region-based CNN), Fast R-CNN, and Faster R-CNN, introduced frameworks for detecting and classifying objects within an image by integrating region proposal networks. Faster R-CNN, in particular, significantly reduced the computational overhead, allowing for even faster object detection [31].

Another key advancement was the development of single-shot object detectors, such as YOLO[32] (You Only Look Once) and Single Shot Detector (SSD)[33]. Unlike the region-based approaches, these models processed the entire image in a single pass, achieving near real-time speeds while maintaining high accuracy. YOLO's ability to predict both bounding boxes and class labels simultaneously has made it a widely used approach in real-time applications, especially for autonomous vehicles

and robotics.

These deep learning-based methods marked a paradigm shift in object detection, as they could learn complex features directly from data, handle multiple objects in an image, and generalise across varied conditions and environments. By leveraging large annotated datasets, deep neural networks brought unprecedented accuracy and robustness to object detection, establishing the foundation for contemporary autonomous perception systems. Further to the discussion here, these types of datasets are expanded upon in Chapter 2, including the XAI-AV dataset that was developed in the course of this thesis.

In the era of deep learning, the features extracted from images became increasingly abstract, as the operations within deep learning models were concealed behind numerous layers and complex operations. This inherent opaqueness of deep networks poses a significant barrier to deploying state-of-the-art perception algorithms in safety-critical scenarios, such as those considered in this work.

1.4.3 Artificial Intelligence

The concept of Artificial Intelligence is a well-established across science, technology, engineering, and mathematics disciplines, but has recently gained widespread attention in popular media, often in loosely defined terms. While various approaches to defining AI have been proposed throughout history, a human-centred perspective is arguably the most relatable - one that focuses on creating machines capable of performing tasks typically done by humans, with the ultimate goal of exceeding human performance.

In this section two branches of AI that are pertinent to the objective of this thesis are presented. The first, Machine Learning (ML), is defined as a field of AI that enables computers to learn from data without being explicitly programmed. In the current era of big data, ML has become essential for processing and analysing vast amounts of information, automating complex tasks in a wide range of applications, from language processing, to image recognition.

The second branch, Deep Learning (DL), is a specialised subset of ML that uses complex neural network architectures, allowing models to learn directly from raw data. As discussed in Section 1.4.2, before deep learning architectures, such as AlexNet[30], revolutionised the field, traditional approaches required developing custom feature extractors for each task. In contrast, deep learning models automat-

ically learn feature representations across multiple layers, enabling them to handle more complex data inputs with minimal preprocessing.

Within the broader category of machine learning, Artificial Neural Networks (ANNs) are particularly relevant, as they serve as the foundation for deep neural networks. Inspired by the human brain, ANNs are composed of interconnected layers of nodes, or neurons, that process data in ways that emulate biological neurons. In the last part of this section optimisation is defined. Optimisation is essential in AI, particularly in training ML and deep learning models, as it underpins the learning process itself. In the training of ANNs, optimisation algorithms adjust the model's weights to reduce the error between the network's predictions and the true output [45].

ANNs, with their ability to learn complex patterns from large datasets, paved the way for modern deep learning architectures, which stack multiple layers to form DNNs. These DNNs have driven advancements in AI by achieving high accuracy in complex tasks, such as, image classification and object detection, tasks that previously relied on handcrafted rules and feature extraction.[30][46][47]

1.4.3.1 Machine Learning

A popular definition of machine learning is available from Tom Mitchell[48], and is as follows: a computer program is said to learn from experience E with respect to some class of tasks T, and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E. There exist many different kind of machine learning which are more or less appropriate, depending on the nature of the parameters T, P and E.

Machine learning methodologies can be divided into two primary categories: supervised and unsupervised learning [49]. What separates the two is that supervised learning methodologies seek to learn a functional relationship between inputs x and outputs y using a labelled set of input-output pairs, D, as defined in Eq. (1.1). Here, D represents the training set, and N denotes the number of training examples.

$$\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N} \tag{1.1}$$

In the context of object detection, x is visual sensor data and y is a set of descriptors for each object in that sensor's Field of View (FoV), which includes

the coordinates representing the space that the object occupies, and a class label defining the object's type i.e. ['car', 'pedestrian' ...]. Like the underlying ML model, these descriptors are dependent on the parameters T, P and E.

The second type of methodologies are unsupervised learning. These type of ML attempt to directly isolate key patterns present in the input data, without access to the output labels y, to validate its predictions. As such, this approach is often referred to as knowledge discovery [49].

Unsupervised learning more closely resembles the way humans and animals typically learn. It is also more versatile than supervised learning, as it does not rely on a human expert to label the data manually. Traditionally, it has been looked at unfavourably given that labelled data can be expensive to acquire. However, since the proliferation of many large scale datasets, this factor has become less impactful [50][51][10]. Additionally, supervised learning typically yields very little information, certainly not enough to reliably estimate the parameters of complex models[49].

1.4.3.2 Deep Learning

Building upon the foundational principles of machine learning, deep learning is a subfield focused on enabling models to learn hierarchical representations from data through deep neural networks. Unlike traditional machine learning methods, which often require manual feature engineering, deep learning models automatically extract complex features across multiple layers, making them well-suited for high-dimensional data, like images and language.

This approach has had a transformative impact on various fields, particularly computer vision and natural language processing, where deep learning models achieve impressive adaptability and performance gains [30, 47]. By allowing models to learn directly from large amounts of unstructured data, deep learning reduces the need for predefined feature sets, enhancing the ability to generalise across diverse datasets.

CNNs have become one of the most transformative architectures in the deep learning landscape, and are particularly effective for image and video data processing. Unlike traditional neural networks, CNNs exploit the spatial structure of data through key principles: local connections, shared weights, and pooling layers. Each layer within a CNN progressively learns more abstract features of the input data. For example, the initial layers typically capture low-level features like edges, while deeper layers assemble these into more complex motifs and object parts. This hierar-

chical feature learning enables CNNs to recognise patterns invariant to translations or distortions, which is essential for robust image recognition and classification tasks.

CNNs operate through multiple convolutional and pooling layers, where the convolutional layers detect feature patterns within local regions of the data, using filters that slide across the input. These shared filters allow CNNs to recognise patterns regardless of their location in the image, a property known as translation invariance. Pooling layers further enhance this invariance by downsampling the feature maps, reducing their dimensionality, while retaining the most salient information. These architectural properties allow CNNs to generalise effectively across diverse applications, from facial recognition to autonomous vehicle perception, making them highly adaptable to complex visual tasks[47].

Since the groundbreaking AlexNet model, deep networks have steadily grown deeper, allowing them to capture increasingly complex patterns in data. A significant advancement came with ResNet, which introduced residual blocks. These are structures that allow information to skip layers, enabling the model to retain important features across different levels of abstraction [52]. This architectural shift paved the way for more proposals of deep networks with non-sequential pathways, encouraging networks to become deeper and more resilient to the vanishing gradient problem. However, as these networks grew deeper, the learned features became more abstract and less intuitively interpretable, resulting in layers that encode unhumanly complex patterns. These advancements, while improving performance, have also made understanding and interpreting these models a challenging endeavour.

Formal safety verification poses significant challenges in balancing system robustness with the real-time demands of operations expected from Autonomous Systems. Robustness is one of the key metrics to measure how stable a neural network's outputs are under random noises, external perturbation, or adversarial attacks to its inputs, a known vulnerability in neural networks that can severely impact safety in AV systems. As noted by Chen et al.[53], developing more efficient and rigorous verification techniques is essential for progressing towards safer neural network-driven perception systems, especially given their complexity in interpreting dynamic environments. This underscores the need for dedicated explainability techniques to bridge the gap between model reasoning and human understanding.

1.4.3.3 Optimisation

During the training of an ANN, optimisation algorithms make adjustments to the model's weights to reduce the error between the predictions and the true output. This adjustment is achieved through iterative updates, typically guided by an objective function, or loss function, which quantifies the accuracy of the model's predictions [49]. The loss function can be defined by $L(\theta)$, where θ represents the model's weights.

To minimise $L(\theta)$, a common optimisation method called gradient descent is used. Gradient descent iteratively adjusts the weights in the direction of the negative gradient of the loss function, aiming to find the values of θ that minimise $L(\theta)$. Formally, gradient descent can be defined by equation 1.2. Where η is the learning rate which sets how big a step is taken during each iteration.

$$\theta_{t+1} = \theta_t - \eta \nabla_\theta L(\theta) \tag{1.2}$$

Popular optimisation techniques, such as Stochastic Gradient Descent (SGD)[54], Adam[55], and RMSprop[56], have become standard due to their effectiveness in refining model parameters across complex, high-dimensional datasets [57].

1.5 Object Detection from Autonomous Vehicle Platforms

This section broadens the scope of object detection especially for autonomous ground and aerial platforms, and on the various state-of-the-art DNN based architectures. The discussion of these various architectures will focus mostly on what is referred to as the 'head' of the detector. This is the part of the detector which synthesises the prediction, and it is what sets it aside from other tasks of image recognition. Hence, this is where special consideration needs to be taken when developing new explainers.

Object detection and recognition in computer vision is the multi-goal task of both classifying and localising objects within the field of view of an imaging system. Typically, object detection is not used as a standalone solution but rather plays a critical role within the broader perception pipeline of an autonomous system. Tasks such as obstacle avoidance, and scene understanding rely on accurate

object detection to interpret the environment effectively. Similarly, effective path planning is essential for autonomous systems operating in complex settings, where object detection enables the generation of safe and efficient route decisions [58][59]. Object detection has also been proposed on autonomous aerial platforms for tasks such as agriculture [60], search and rescue [61], and defence applications, including Automatic Target Recognition (ATR) [25]. Given its broad applicability, developers often treat object detection as a discrete component, focusing on its optimisation and integration within larger autonomous systems.

In the Visdrone-DET 2018 results paper [62], some of the most effective algorithms used Feature Pyramid Networks (FPN) [63]. This type of feature extractor produces multi-scale feature maps using a bottom-up and top-down pathway. The bottom-up pathway is the standard convolutional neural network (CNN) feature extractor—as feature maps propagate, semantic richness increases, but spatial resolution decreases. The top-down pathway consists of reconstruction layers and lateral connections to earlier parts of the network. The reconstruction layers perform upsampling, while the lateral connections merge upsampled high-level features with low-level feature maps, preserving both semantic meaning and spatial details. The result is a hierarchical feature pyramid, which enhances detection across various object sizes. The challenge coordinators speculate that this property makes FPN well suited for detecting objects of diverse scales.

A further extension of FPN is the Panoptic Feature Pyramid Network (PFPN) [64], which enhances the feature pyramid structure for segmentation tasks. While standard FPN is primarily used for object detection, PFPN integrates instance and semantic segmentation to achieve panoptic segmentation—a unified framework for detecting both countable objects (e.g., cars, people) and uncountable regions (e.g., roads, sky). PFPN incorporates an additional segmentation branch, enabling pixel-wise predictions alongside bounding box detections, making it particularly effective for dense scene understanding. However, despite its advantages, PFPN has computational drawbacks that may limit its applicability in real-time systems, such as autonomous vehicles[65][66].

At the time You Only Look Once (YOLO) was first introduced by Redmon et al.[32], the typical approach used by developers was to take an existing DNN-based classifier and perform multiple inferences on snippets of the entire image - such as a sliding window [67], or region proposal [68]. These approaches were

slow and inefficient as multiple inferences had to take place before the entire image had been evaluated. Moreover, these pipeline-based approaches had two further downsides. Firstly, it was hard to optimise because each component needed to be trained individually. Secondly, the classifier had the disadvantage of not having access to contextual information in the entire image.

YOLO was proposed as the first unified model approach where a single CNN is used to predict numerous bounding boxes and the class probabilities simultaneously. Being capable to perform detections in such a manner makes YOLO a very fast algorithm. Moreover, a unified model means that training is much simpler, and the detection performance can be optimised directly. Because the algorithm utilises the entire image, it can take advantage of contextual information, which the developers suggest allows the algorithm to make fewer false positives.

There have been subsequent releases of YOLO in [69], [70], [71] and [72]. These works presented some enhancements to the model that improved the detection performance. The most significant will be summarised here, as well as how the detection layers of YOLOv5 from Jocker et al.[72].

In [69], Redmon et al. modified YOLO to an anchor box approach, which is still used by YOLOv5. Anchor boxes - sometimes referred to as prior boxes - are essentially predefined blueprints of bounding boxes that the network will predict offsets from. A detection layer will split the image into $NbyN$ cells. Each cell is responsible for detecting an object if the centre of that object is within that cell. For each anchor box, a cell will predict a set of class scores, and an 'objectness' score, which can be used to distinguish between a true and false positive. In addition, the network will also predict two other properties of the box. Firstly, the offset between the bounding box centre and the top left corner of the cell. Secondly, horizontal and vertical scaling factors, which are used to re-scale an anchor box to fit an object. The most state-of-the-art version of YOLO use multiple detection layers which are responsible for making detections from multiple different scales. This allows the network to be better at detecting a larger span of object sizes. This approach is similar to Feature Pyramid Network (FPN), as such, there have been more YOLO style submissions to the recent Visdrone competitions.

SSD is not a popular algorithm in the Visdrone competitions. Nevertheless it is covered here as Hideomi et al.[73], proposed an XAI framework specific to this algorithm which will be discussed in a later section. SSD can be considered somewhat

similar to YOLO based architectures, as it is also built on top of a single convolution network, uses anchor boxes, and also requires post-processing with non-maximum suppression to remove duplicate boxes. What distinguishes SSD from YOLO is that it has two output layers that fork, with one for performing classification, and the other for performing localisation. Another distinction is that SSD uses VGG-16 [74] as a backbone. Whereas, YOLO networks use Darknet [71], which is a much more modern architecture.

There are many object detectors that are extensions of the DNN known as Region-based Convolution Network (R-CNN). This network was first introduced by Girshick et al.[68], as an alternative to the typical sliding window approach that was employed at the time. They proposed the use of a selective search algorithm to generate region proposals - subsections of the image - which could then be given a to classifier. Their technique would extract 2,000 regions that may contain an image.

There are two main problems with vanilla R-CNN. The first issue is that the classifier has to run 2,000 times, which results in long training times and real-time inference being unreachable. Secondly, the search algorithm is a fixed algorithm, therefore, may not be optimised for a given application. These issues lead Girshick et al.[75] to propose Fast R-CNN. This introduced the Region of Interest Pooling layer which takes a feature map from the last convolutional layer, and a list of region proposals, to produce a feature vector which is then refined using the Fast R-CNN head to yield the bounding boxes and class scores. The Fast R-CNN consists of two fully connected layers, one to manage each task, in a similar manner to SSD. The convolution operation only needs to be run once, resulting in sped up training and shorter inference time.

The downside of Fast R-CNN is that it still uses the selective search algorithm. Ren et al.[31] proposed Faster R-CNN which does away with the selective search algorithm. Instead, a separate Region Proposal Network (RPN) is used to predict the region proposals with various scales and aspect ratios. This new RPN has two main advantage. Firstly, the convolutional layers are shared across both the RPN and the Fast R-CNN head, resulting in a large speed boost. Secondly, the RPN allows Faster R-CNN to be trained end-to-end which makes training simpler, and allows the region proposal task to be optimised for the given dataset.

Despite recent advancements in algorithm design that emphasise speed, deploying deep detection networks on resource-constrained autonomous vehicle platforms

remains challenging [11]. In a comprehensive study, Huang et al.[76] explored the complexities of balancing speed, memory, and accuracy in convolutional object detection systems. Common strategies to optimise these models include simplifying the network by reducing input image size, decreasing the number of regional proposals (or anchor boxes), and employing simpler feature extractors. However, these adjustments often compromise performance, especially for small object detection, where finer details and higher resolution are essential for accurate inference.

However, Huang et al.[76] also identify pruning as a promising technique for achieving a balance between speed and accuracy, which could make deep object detectors more feasible on embedded systems. The authors highlight the need for further research into structured pruning techniques specifically tailored to object detection architectures, to address the computational constraints of autonomous platforms. This perspective is reinforced by Abhishek et al.[11], who emphasise that reducing network complexity is essential for real-time applications in autonomous vehicles, where constraints on latency, energy consumption, and processing power are significant. By removing less impactful weights, or entire filters, pruning allows for efficient inference, reducing both model size and computational demands, making it particularly advantageous for automotive edge devices. This thesis explores the intersection of pruning and explainable AI in Chapter 4.

The integration of machine learning and deep learning techniques in the perception and control of vehicular subsystems has led to a strong dependence on data from external systems and sensors. However, even minor uncertainties in this input data can result in unintended outcomes, compromising vehicle performance, and potentially leading to severe consequences. Developing automatic perception methods and control algorithms that are robust to these uncertainties is critical. In this context, XAI becomes essential - by enhancing the transparency of machine learning and deep learning models, XAI provides insights into a model's decision-making process, allowing developers to detect vulnerabilities, address biases, and improve reliability. This transparency not only aids in diagnosing performance issues caused by data uncertainties, but also contributes to safer, more dependable autonomous systems.

1.6 Transparency and Explainability in Deep Learning

In the context of deep neural networks, transparency refers to the ability to understand and explain the model's internal workings and decisions in human-comprehensible terms. As Molnar [8] suggests, models that are transparent offer greater "translucency" into their mechanisms, allowing for visibility into the relationships that drive predictions. However, DNNs are typically opaque due to their complex, layered architecture, making transparency difficult to achieve. This lack of transparency is especially concerning in safety-critical systems like autonomous vehicles, where interpretability becomes a necessity. As Doshi-Velez and Kim highlight in [77], interpretability serves as a critical safeguard, enabling users to verify a model's reasoning, even in "complex applications" where exhaustive testing is impractical. For such applications, transparency allows for human oversight, ensuring that a model's predictions align with ethical and safety standards, and addresses the "incompleteness" inherent in scenarios where every possible outcome cannot be fully accounted for.

In the context of XAI, an explanation will illustrate how specific input features of the data relate to the network's prediction on that data, in a manner that is comprehensible to a human. An explainer is the algorithm that generates these explanations [8]. In this scenario, the network whose behaviour will be explained is referred to as the explicant, and the recipient of the explanation is the explainee. While the explainee will typically be a human agent, in some cases the explainee might be a software system. Such a software system may use the explanation for objectives like error detection [78].

Explanations can be either global or local. Global explanations aim to comprehend the model as a whole by examining all of its features and learned weights. However, such explanations can fall short for models that behave differently for various feature combinations, as they may not represent local behaviour well [79]. Local explanations, in contrast, examine individual instances, making them effective for understanding complex behaviour on a case-by-case basis.

The terms interpretability and explainability are often used interchangeably in the literature, but recent works have highlighted important distinctions. Gilpin et al. [80] define interpretability as the science of understanding what a model does to reach a prediction, focusing on finding causal attributions of a prediction. In

contrast, explainability is the process of selecting and summarising the most relevant causes. Nielsen et al. [81] further clarify that interpretability is a passive quality of the model, referring to its intrinsic ability to attach human-understandable causes to outputs. Explainability, however, is an active characteristic, designed to clarify a machine learning model's inner workings for a human observer.

1.6.1 Properties of Explanations

This section will review works presenting the properties of an explanation. The purpose is to investigate what would be considered a good explanation, and how humans formulate explanations for other humans. The differences between interpretability and explainability will also be addressed in this section.

Miller [82] proposes that the ongoing development of explainable artificial intelligence would greatly benefit from building upon the vast pre-existing research in philosophy, psychology and cognitive science. To that end, they conduct an in-depth review of existing literature in the social sciences on the properties of "good" explanations. Their finding are summarised in the following paragraphs.

Good explanations are contrastive, in that they are sought in response to particular counter-factual cases. Miller states that when humans provide an explanation of a given event, they provide the cause relative to some other event that did not occur. This is a counterfactual explanation, which describes a causal situation in the form: "How does the prediction change if input X had been different?". This finds relevance in deep learning, as developers often care more about cases in which their algorithms contested the groundtruth, as opposed to cases when they are error-free. However, this may also present a challenge for XAI as a human might simply question "why X?" - leaving the counterfactual scenario unstated. Consequently, in some cases, the counterfactual scenario may have to be inferred by the explainer.

Explanations are often selected in a biased manner as humans do not typically provide all causes for an event as an explanation; Miller states that a human can't simulate back through all possible causes and evaluate their counterfactual cases. Humans will instead select what they believe are the most relevant causes through cognitive biases. Millers states that these cognitive biases can arise from how easily a human can mutate the cause to create the counterfactual scenario.

A cause's mutability can depend on its abnormality, its responsibility, the time it occurred in relation to the event, and its controllability. An abnormal cause is a

cause that had a small probability, but nevertheless happened, and eliminating it would greatly change the output. Abnormal causes are typically considered more relevant than normal causes, even if a normal cause might have greater than, or equal influence, on the event.

Miller argues that applying the aforementioned cognitive biases in selecting explanations from causes can improve human interactions with XAI. In a machine learning context, this means that features with the greatest influence might not be the best indicator for a good explanation - at least from the user's standpoint. This author's standpoint is that it will require the explainer, not only to appropriately identify causal information correctly, but then also to be able to generate a report presenting the causes that would make good explanations to the human.

Explanations are social, in that they are a transfer of knowledge, presented as part of an interaction between the explainer and explainee. The level of explanation provided is dependent on the domain for which the explainer is operating in, and the prior knowledge of the explainee. A good explainer should be able to be queried, should the explainee not be happy with the provided explanation. This ties in with another point from Miller, where they state that using statistical generalisation to explain why events occur to a human, can be unsatisfactory. Instead, statistical analysis would be best used to back up a causal explanation in the dialogue with the explainee.

In the context of XAI, this will require an explainer to provide two key functions. The first is to be queried by the explainee - such as producing a new explanation based on a hypothetical counterfactual example produced by the explainee. The second function requires the explainer to produce the appropriate level of detail to match the explainee's level of experience. In some situations, sparse explanations - explanations that use few features - may be superior when 'socialising' with the lay user. Whereas, a technical person may prefer statistical or empirical evidence from the explainer. This notion is shared by Chu et al.[83], who found that saliency maps alone can be unsatisfactory for many users.

Neilsen et al.[81] have produced a tutorial on gradient-based explainers that produce saliency maps. Saliency maps will be discussed in greater depth in the subsequent sections. Part of the tutorial by Neilsen et al. focused on best practices when selecting an explainer and what attributes they believe should be addressed. Unlike Miller's study, they stay within the realm of deep learning research.

The first attribution they identify as important for an explanation is faithfulness. This is the measure of how well the explainer is an accurate proxy of the model's decision-making process. This attribute is linked with that of fidelity which will be revisited here in section 1.7 and refers to how well the explainer is able to identify feature relevance. Secondly, they identify robustness, this is related to the stability of a saliency map against small perturbations in the input that might be present for natural reasons, but also might be the result of an adversarial attack.

They also identify attributes that one would accredit to the explainer rather than the explanation itself. The first is efficiency, which can be considered as the number of passes through the network required to synthesise the explanation. The second attribute, is that of implementation invariance, i.e. the explainer must produce the same explanation for two functionally equivalent models on the same input.

An effective explanation must not only clarify high-confidence predictions, but also illuminate lower-confidence instances where a model's decision-making may be less certain. Molnar [8] highlights that explainers must perform well across diverse model outputs, as low-confidence predictions are often where interpretability is most critical. In such cases, explanations can help detect potential model uncertainties and errors, offering insights into when and why a model's predictions may be less reliable. These instances are particularly valuable because, in real-world applications, lower-confidence predictions may represent cases with unusual or challenging data, where model missteps could have significant consequences. Therefore, for a robust AI system, the explanation framework should be equipped to handle and reveal insights across the full spectrum of model confidence levels, ensuring transparency and trustworthiness, even in challenging cases.

1.6.2 Local Explanations with Saliency Maps

In this thesis, the focus is on the domain of object detection on 2D images. As such, this section will discuss explainers that generate local explanations for image recognition algorithms. A common format for an explanation of a prediction inferred by an image recognition algorithm is a saliency map, also known as an attribution map. These types of local explanations show how each pixel, or group of pixels, contributed to a specific prediction. Hitherto, the majority of these types of explainers have primarily focused on the image classification task, with very few branching out into other tasks [84][73].

In addition to saliency-based approaches, alternative explanation strategies have also been explored for aerial images. For example, the work by [85] investigated feature visualisation techniques that generate images representing the learned features of a network unit—whether a single neuron or the entire network—by maximising the mean activation of that unit. However, these visualisations tend not to be human-friendly and fail to provide the localisation information that is critical for detection tasks [86].

Saliency maps have been used to analyse a classifier's localisation ability, which has been revealed to be learned implicitly. Indeed, Lai et al. [87] have used saliency maps to perform weakly supervised object detection. In contrast, object detection algorithms explicitly predict the localisation of objects as part of their output. One might assume that exploring the model's attention is unnecessary; however, Petsiuk et al. [84] have found that DNN-based object detectors can be reliant on contextual information present outside the bounding box. This finding demonstrates that a predicted box's causation and its location are not strictly linked. Hence, it remains pertinent to study a detector's saliency for this reason. Moreover, understanding which parts of an object within the bounding box are more and less important to a network can provide deeper insights into model behaviour.

Beyond addressing contextual dependencies, saliency maps are particularly well-suited for object detection as they align with the spatial nature of the task. Unlike classification networks, which produce a single label per image, object detectors simultaneously classify and localise multiple objects. Saliency maps provide an interpretable means of examining which regions within and beyond bounding boxes contribute to a detection, making them a natural choice over alternative explainability methods that do not capture spatial dependencies [88]. Furthermore, by revealing how detectors leverage background context, saliency maps help identify biases where models rely on spurious correlations, such as road texture in vehicle detection or sky colour in aeroplane detection.

Another key advantage of saliency maps is their ability to provide fine-grained explanations of model decisions, making them particularly useful for object detection tasks where precise feature-level understanding is essential. Unlike feature attribution techniques that operate at a high level, saliency maps reveal which specific pixels or object parts contributed to the model's confidence in its predictions. Simonyan et al. [89] demonstrated that saliency maps can highlight low-level edges and

textures, as well as high-level semantic features, providing an interpretable bridge between raw image input and model decisions. More recent studies [90][87] have reinforced this idea, showing that saliency-guided learning improves object detection, particularly in weakly supervised settings where bounding box annotations are limited. These fine-grained explanations are crucial not only for verifying correct object detection but also for diagnosing model errors, such as over-reliance on irrelevant background features or failure to focus on critical object parts, which can degrade model robustness [91].

Saliency maps also serve as a valuable diagnostic tool for validating model decisions and uncovering failure modes in detection networks. They allow researchers to verify whether an object detector is making decisions based on intrinsic object features rather than misleading elements in the background. For instance, if a detector highlights only shadows or reflections instead of object edges, this could indicate overfitting to non-robust features, leading to poor generalisation [91]. This capability is especially crucial in safety-critical applications, such as autonomous driving and medical imaging, where misinterpretations could lead to significant consequences.

There are two varieties of explainers that can be used to generate a saliency map. The first variety consists of perturbation-based explainers. These are model-agnostic techniques that treat the network as a closed box and only require access to the input and output of the network to produce explanations. The second variety consists of backpropagation-based explainers. These are not model-agnostic, as they require direct access to the model's layers. In some cases, the latter type of explainer imposes architectural constraints, necessitating modifications to the network before they can be utilised.

In the following two subsections, a review of explainers belonging to both varieties is provided. While most explainers have been designed for image classifiers, adaptations have been proposed for object detection. These adapted explainers focus on explaining individual detections rather than entire images, ensuring that explanations remain interpretable at the instance level.

1.6.3 Perturbation Based Explainers

This section will discuss the perturbation approach to generating saliency maps. The benefit of these type of explainers is that they only need access to the inputs and outputs of a model to provide the explanation. This means any model can be used

regardless of the architecture or components used. As will be discussed, this is not always the case when using backpropagation based explainers. This methodology treats the explicant as a closed box. Therefore, in this section the model being explained will be referred to as the closed box model. Here, three techniques are presented in detail: LIME [92], RISE [12] and KernelSHAP [93].

Local Interpretable Model-agnostic Explanations (LIME), from Ribeiro et al.[92], can be used to provide insights from any type of classifier - even those not based on DNNs. Their method works by training a surrogate interpretable model to represent a single instance. This surrogate model is trained on a new dataset consisting of perturbed samples. These perturbed samples are given to the closed box model and the prediction is used as the label. The way that the sample is perturbed depends on the type of data. For images, a method of perturbation for LIME might be to set groups of pixels to a uniform colour.

The problem in LIME that can arise with having fixed groups of pixels is that particular features can be missed. Randomised Input Sampling for Explanation (RISE) was proposed by Petsiuk et al.[12], in their approach they perturbed the images using randomly generated masks as a means to overcome the issue present with LIME. The final Saliency map is created using a weighted average of the random masks. Here, the weights are the output probabilities for the target class predicted by the closed box model. They evaluate their method using their Deletion and Insertion metrics, which will be discussed in detail in section 1.7. They use these metrics to support their method's ability over LIME and Grad-CAM [4].

RISE was adapted to D-RISE by Petsiuk et al.[84] to be used with detection style architectures. Their intention was to create an explainer to synthesise a saliency map that shows the influence of each pixel on the model in predicting a given box. Their method is consistent with RISE, except the weights for the weighted average are calculated using a new similarity metric. This similarity metric is meant to measure how different a bounding box predicted from a perturbed image is to the originally predicted bounding box. This metric takes into account not only the difference in class scores, but also the difference in location and shape.

SHapley Additive exPlanations (SHAP) was proposed by Lundberg et al.[93]. In their paper, they proposed several methods for estimating Shapley values from [94]. Shapley values come from coalition game theory, and represent how each feature (the player) contributes to the prediction (the payout). Shapley values have four

properties that make it the only attribution method that satisfies the definition of a fair payout [8].

The first property is efficiency, which guarantees that the contribution to the prediction is fairly distributed among the features. The second property is that of symmetry, which states if two features always contribute equally to every coalition, then they should receive the same payout. Third is the property of dummy - or the dummy player - which states that if a given feature does not change the prediction value, the shapley value is 0. The fourth property is that of additivity which considers a complex system that has multiple agents. The agents' payouts in each coalition should be the sum of the payments they would have received for that coalition if there was a separate game for each agent.

Molnar [8] conducted a review of many of the methods that are covered in this chapter. They state that due to the efficiency property of Shapley values, it may be the only metric that could provide a full explanation. They assert for situations where the law requires explainability, Shapley values may be the only legally compliant method.

In this report, two methods from [93] will be discussed, KernelSHAP and DeepSHAP. KernelSHAP uses the perturbation method to estimate the Shapley values, and thus will be covered in this section. KernelSHAP combines Shapley Values and LIME, where KernelSHAP represents Shapley value explanation as a linear model g, given by Eq. (1.3). Where M is the maximum coalition size and the number of features, and ϕ_j is the Shapley value for a given feature j. z' is the coalition vector and is described by Eq. (1.4), where 0 indicates an absent feature, and 1 indicates the feature is present.

$$g(z') = \phi_0 + \sum_{j=1}^{M} \phi_j z'_j \tag{1.3}$$

$$z'_k \in \{0,1\}^M, k \in \{1, ..., K\} \tag{1.4}$$

What separates KernelSHAP and LIME is the weighting of the instances in the regression model. In KernelSHAP, the sampled coalitions are weighted such that large coalitions (where few features are removed) and small coalitions (where few features are present) have the largest weights. The rationale is that more can be learned about the contribution of individual features when they are studied in

isolation. To achieve this weighting, Lundberg et al.[93] propose the SHAP kernel, defined in Eq. (1.5). The SHAP kernel is the sample weight given to each binary vector $z' \in \{0,1\}^M$ which represent whether a feature is present or not. The term $|z'|$ is the number of ones in z', and $\binom{M}{|z'|}$ is the number of ways to choose a subset of features of that size.

$$\pi_{x'}(z') = \frac{(M-1)}{\binom{M}{|z'|}|z'|(M-|z'|)} \tag{1.5}$$

As has been established, perturbation approaches only require access to the input and output of the network to generate a saliency map. It is very easy to deploy these types of explainers for a novice user, or anyone who has trained a model through an Abstract Programming Interface (API). It can be deployed quickly when experimenting with new architectures without needing to adapt them. It can also be used to investigate why an event occurred in a system that has already been deployed. However, these methods cannot be used to produce explanations in real time, for instance, in a way which would be beneficial to a remote pilot. Moreover, perturbing the input can risk introducing unwanted information which could produce misleading saliency maps.

1.6.4 Backpropagation-Based and Gradient-Based Explainers

Backpropagation-based explainers can be further divided into gradient based explainers, and attribution propagation based techniques. Gradient based explainers compute the gradient of the prediction with respect to the input features, and can be distinguished by how the gradient is computed. Here various explainers that use gradients are introduced. For each explainer provided there is a definition and an outline of the developer's motivation, accompanied by any weaknesses that have been expressed in other reviews.

Vanilla Gradients was introduced by Simonyan et al.[89] in 2013. It is one of the earliest of the saliency map methods; sometimes in the literature, it is referred to simply as 'saliency maps'. However, later gradient methods have been able to produce more concise and less noisy maps. That being said, this method has the practical benefits of being simple to implement, and the explanations produced can be quickly synthesised. Moreover, more recent state-of-the-art methods are often susceptible to other issues which will be discussed later in this section.

SmoothGrad from Smikov et al.[95] reduces noise by averaging over many saliency maps synthesised from copies of the same input with added Gaussian noise. Since SmoothGrad requires multiple forward propagation to generate the final saliency map, it can be considered that technique is a hybrid between perturbation and gradient based approaches. The fact it requires multiple inferences means that, like other perturbation approaches, it is slower and less efficient to produce an explanation.

The downside of most gradient-based approaches is that they are prone to the phenomenon of saturation. To understand saturation, consider the Rectified Linear Unit (ReLU) activation function, once the input is less than zero, the function will always output zero, regardless of how large the negative value is. Therefore, saturation could cause the explainer to underestimate the importance of input features [96].

Integrated Gradients (IG) was a technique presented by Sundararajan et al.[97], it is somewhat similar to Deeplift, another algorithm that will be discussed towards the end of this subsection. In order to overcome the phenomenon of saturation, both these explainers use a baseline to 'compare' a given instance to synthesise the explanation. Informally, IG calculates the average of all gradients, and then takes the element-wise product of this with the original image. Formally, IG uses a finite number of samples to approximate the integral. Another method that has been proposed to improve the sharpness of the saliency map, and overcome the problem of saturation, is Gradient*Input [98]. This is simply the element-wise product of the input and the gradients.

Adebayo et al.[99] conducted a review of saliency methods, in which they conducted a sanity check to establish each method's sensitivity to the learned weights and biases of the model. One such sanity check required comparing the saliency map of a trained model to that from a model where the weights had been randomly generated. They observed that, despite visual changes in the masks obtained from Gradient*Input and IG, the input structure was still clearly prevalent. They state that an analyst could mistake the patterns in the maps as legitimate. Nielsen et al.[81] define this phenomenon as input dominance.

Input dominance can occur with techniques that leverage the information present in the input features to synthesise the explanation. As the name suggests, this is when the saliency map relies too heavily on the input which does not depend on the

network. While this may produce more human interpretable explanations, it would not capture how the network processed the input to make its prediction [99].

In their tutorial on Gradient-Based explainers, Nielson et al. [81] found that many state-of-the-art gradient based techniques were prone to failure due to class agnostic behaviour. This is a phenomenon when there is very little difference in the generated maps for different class labels for the same image. Of the methods they examined, Grad-CAM was the only one not class insensitive.

Influenced by Class Activation Mapping (CAM) from [14], Selvaraju et al.[4] introduced Gradient-weighted Class Activation Mapping (Grad-CAM). In their paper, they demonstrate Grad-CAM, not only on image classification, but also on Image captioning and Visual Question Answering. The main advantage Grad-CAM has over CAM is that CAM can only be deployed on certain architectures which use Global Average Pooling (GAP) layers. They further analyse their methods using an automatic metric known as the "Pointing Game", which will be discussed further in section 1.7. Using this, they argue their method outperforms vanilla gradients in terms of interpretability and visualisations. Furthermore, They also performed human trials to show that Grad-CAM was able to discriminate between classes well - a finding also shared above by [81].

The main difference that distinguishes Grad-CAM and vanilla gradients is that the gradients are not computed all the way to the input. Instead, the gradient is only backpropagated to the last convolutional layer. The motivations the authors provide is twofold. Firstly, they state that the deeper layers in DNNs capture higher-level visual constructs with reference to [100][101]. Secondly, spatial information that is otherwise lost in the fully connected layers - of classifiers - is preserved by the convolutional layers. Hence, the last convolutional layer, before the fully connected layers, should be the best compromise between high-level semantics and spatial information.

Attribution propagation based explainers work by overriding the backpropagation protocols with their own ruleset. These include techniques such as Layer-wise Relevance Propagation (LRP), DeepLIFT[96] and DeepSHAP[93]. The literature surrounding attribution propagation will often refer to the magnitude of pixels in the saliency map as the 'Relevance' of a given pixel, and is used interchangeably with 'importance'.

As stated, LRP works by overriding the backpropagation protocols and imple-

menting its own ruleset. LRP was first introduced by Sebastian et al.[102]. However, subsequent works have introduced new rules in order to produce higher quality saliency maps including [103], [104], [105] and [73]. LRP uses the model's weights, and the neural activations created by the forward pass, to propagate relevance back from the output. It is important to understand the conservation property that LRP is subject to. This states that, whatever relevance is allocated to a given neuron, must be redistributed to the lower layer in equal amount.

The generic expression for calculating the relevance of a given neuron R_j is shown in Eq. (1.6), where R_k is the relevance of a neuron from the succeeding layer - the layer closer to the output. The determining factor for how the relevance is redistributed is z_{jk}, this can be considered a placeholder for whichever rule is applied. The denominator serves to enforce the property of conservation.

$$R_j = \sum_k \frac{z_{jk}}{\sum_j z_{jk}} R_k \tag{1.6}$$

To convey an understanding of some of the LRP rule system, outlined here are three rules that were employed in [105] to explain an image classifier - VGG-16 [74]. $LRP-0$[102] is the simplest rule (Eq. (1.7)), z_{jk} is the activation of neuron j, denoted by a_j, multiplied by the weight between neurons j and k. If this rule were applied to the entire network it is equivalent to Gradient*Input. They apply this rule on the deepest layers of VGG-16 - near the output - as they require a rule that is close to the function and its gradient. This is because the many concepts forming the different classes are entangled, due to the small number of neurons per class at these layers.

$$R_j = \sum_k \frac{a_j w_{jk}}{\sum_{0,j} a_j w_{jk}} R_k \tag{1.7}$$

The middle layers are characterised as having a more disentangled representation, but also spurious variations due to the stacking of many layers and weight sharing. Hence, they employ $LRP-\epsilon$ [102], denoted by Eq. (1.8), which add a small positive term (ϵ) in the denominator to absorb some relevance when the contribution of a succeeding neuron is weak, this results in only the most relevant factors being retained. Finally, in the shallowest layers - near the input - they use their own rule $LRP-\gamma$ (Eq. (1.9)) which favours the effect of positive contributions over

negative. Their reasoning is that this makes the explanation more human-friendly, by spreading relevance uniformly to an entire feature, rather than capturing the contributions of every pixel.

$$R_j = \sum_k \frac{a_j w_{jk}}{\epsilon + \sum_{0,j} a_j w_{jk}} R_k \tag{1.8}$$

$$R_j = \sum_k \frac{a_j (w_{jk} + \gamma w_{jk}^+)}{\sum_{0,j} a_j (w_{jk} + \gamma w_{jk}^+)} R_k \tag{1.9}$$

Tsunakawa et al.[73] introduce Contrastive Relevance Propagation (CRP), an extension of LRP, which they developed to provide the causal attribution for the detections made by SSD trained on the Pascal VOC 2012 dataset. As discussed in section 1.5, SSD has one fork for predicting the class, and one for predicting offsets for a set of predefined bounding boxes. To account for this, they have selected different propagation rules for each fork.

For the classification fork, they use the $z^+ - rule$, from Eq. (1.10), where x_i is the activation in the l-th layer and w^+ is the layer weights that are positive. Since this is an output layer, $R_k^{(l+1)}$ is the class vector with zero for all classes, except for the target class. This rule will apply more relevance to units that positively contribute to x_i. ‘

$${}^+R_{j\leftarrow k}^{(l,l+1)} = \frac{w_{jk}^+ x_j}{\sum_{j'} w_{j'k'}^+ x_{j'}} R_k^{(l+1)} \tag{1.10}$$

$${}^-R_{j\leftarrow k}^{(l,l+1)} = \frac{w_{jk}^- x_j}{\sum_{j'} w_{j'k'}^- x_{j'}} R_k^{(l+1)} \tag{1.11}$$

The localisation layers of SSD can predict both positive and negative offsets. Therefore, Tsunakawa et al. use sign based rule switching. If the prediction is positive then they apply the $z^+ - rule$ (Eq. (1.10)). Otherwise, they apply another rule which they define as defined as the $z^- - rule$. Shown in Eq. (1.11), this rule works contrary to $z^+ - rule$ by instead applying more relevance to units that negatively contribute to x_i.

Karasmanoglou et al.[106], later applied LRP to explain detections made by YOLOv5. They propose preprocessing the prediction prior to backpropagation to set the initial relevance. This preprocessing involves removing non-relevant information using max class selection, localisation by the bounding box, localisation by object

confidence, and then discarding the bounding box's dimensional data. This results in a single vector containing the class scores for the class of the object of interest for the cells where that object could be located.

Returning to Tsunakawa et al., they identify an issue that they describe as indistinguishable heatmaps, which arise when standard propagation rules are applied to SSD. This phenomenon is where the saliency maps for objects belonging to different classes look identical. Such a characteristic is unfavourable and makes the explainer untrustworthy. CRP will use a baseline in the middle layers where high-level features are stored. This baseline is acquired by repeatedly propagating the relevance from the classification layer for all classes, and then taking the average. The problem with this approach is that it makes CRP not very efficient.

The use of a baseline is not new and is often used to overcome the problem of vanishing gradients. This is the case in DeepLIFT (Deep Learning Important FeaTures) which was introduced by Shrikumar et al.[96]. In their work they refer to the baseline as the reference. The references for all neurons is based on a reference input that is propagated through the network. This input is selected based on the domain, and it is good practice to use multiple references to investigate a given prediction.

For example, they use a black - all zero - image as a reference for images in the MNIST [107] and the CIFAR10 [108]. While this worked well for MNIST, the saliency maps for CIFAR10 were challenging to interpret. As a counter-suggestion they suggested using a blurred version of the image which required more passes. Therefore, the effectiveness of DeepLIFT requires a designer to make careful consideration for the reference to produce 'good' explanations. To some degree this characteristic of DeepLIFT breaks the requirement for the explainer to be consistent, and can cause it to fail when exposed to new data.

Lundberg et al.[93] were able to exploit DeepLIFT to create DeepSHAP, a fast approximation algorithm for Shapley values. The benefit of using Shapley values has been stated above. If the model is fully linear, then the explainer produces the exact Shapley values. This explainer has already been applied by He et al.[109] to provide explanations for a DNN based autonomous navigation algorithm. With this explainer, they were able to produce saliency maps that illustrated features present in images from the 2D camera that were important to the network's decision making.

The challenge with using attention propagation based explainers is that they

require a user to rework the backpropagation protocols of a given deep learning framework to implement. This makes them less useful to a broader audience, such as those accessing a model through an API, or the lay-user who may not have the required knowledge to interpret information about the internal workings of DNNs [110]. Moreover, networks that are non-linear or have complex pathways - as is often the case with detection algorithms - will often break the rules required by these kinds of explainers.

Nonetheless, unlike perturbation based approaches, there are backpropagation based approaches that only require a single and forward pass to generate the saliency map. Hence, techniques like Grad-CAM could feasibly be used to perform real-time explanations, and this has been achieved by Young Jin k. et al.[111] for fire detection. Moreover, as has been stated, attribution propagation based explainers can be used to provide importance scores for units within the network which can be useful to an experienced developer.

1.6.5 SHAP and DeepSHAP

This section provides a concise overview of SHAP as introduced by Lundberg et al. [93]. They propose that any explanation for a model's prediction can be conceptualised as a surrogate model. This surrogate model serves as an interpretable approximation of the original model. As illustrated in Eq. (1.12), this surrogate model can be represented as a linear model g. In this expression, z' is a feature vector of size M of a given instance and the weights ϕ, are the contribution of each feature to a given outcome. Features may include elements from a table, pixels from an image, or units within a deep learning network.

$$g(z') = \phi_0 + \sum_{M}^{j=1} \phi_j z'_j \tag{1.12}$$

Lundberg et al. introduce KernelSHAP, this method is characterised as a perturbation-based technique that creates a dataset indicating the presence or absence of a specific feature z'_j , along with the corresponding predictions made by the network for each combination of features. The surrogate model utilised in this approach is a linear model that is trained using the dataset produced.

Lundberg et al.[93] successfully utilised DeepLIFT to develop DeepSHAP. They argue that incorporating an input can be interpreted as assigning it its actual value

rather than its reference value. Consequently, DeepLIFT can be regarded as an efficient approximation technique for the SHAP values ϕ. Chen et al.[112] propose using a background distribution to represent the reference value. They do this by obtaining SHAP values for each baseline in a group of samples and average over the resultant attributions. In [112], they prove that this method results in estimated SHAP values closer to the true SHAP values.

DeepSHAP has already been applied for UAV applications by He et al.[109] to provide explanations for a DNN based autonomous navigation algorithm. With it, they are able to produce saliency maps that illustrate what features that are present in images from the 2D camera are important to the deep network. Moreover, they are able to investigate how different units in their network influence the network outcome.

1.7 Evaluating Explainers and Explanations

This section addresses the challenges of evaluating explanations produced by XAI, drawing attention a recurring problem tackled in this thesis. As discussed briefly in section 1.6.1 an explanation can be considered a summary of relevant causal attributes. However, there is much debate over how to determine if what the saliency map is highlighting has actual causal significance, let alone if it can constitute an explanation. In addition, it is often quite challenging to disentangle errors made by the model from the errors made by the explainer during validation [97].

A significant defilement to the development of saliency map explainers is that design decisions are often made based on visual appeal on the image data. This issue was the motivation behind a recent study conducted by Adebayo et al.[99]. They found that some existing explainers produced saliency maps that were independent of the model's learned parameters and the input data. This would suggest that any such method would not be able to identify causal information in the input, and therefore not be adequate for synthesising explanations. They proposed several sanity checks for designers to perform to verify their saliency methods.

The first sanity check which they presented is what they refer to as the Model Parameter Randomisation Test. The test involves comparing the output of the saliency explainer on a trained model, with the output of the saliency method on a randomly initialised untrained model. The purpose of the test is to investigate the

sensitivity to the model's learned weights, if this is indeed the case, the two outputs should be substantially different.

The second sanity check is what they refer to as the Data Randomisation test. In this test a second model of the same architecture is trained on a dataset where the labels have been randomly re-assigned. They present the hypothetical scenario that, if an explanation did not change after randomly assigning diagnoses to CT scans, then the explanation failed to relate features within the CT scan to the correct diagnosis. Therefore, for an explainer to pass this check, it is expected that the saliency map generated on the second model will be significantly different to one generated on the control.

Tomsett et al.[110] were inspired by the sanity checks paper by Adebayo et al.[99] to investigate the reliability of current saliency metrics. They specifically focus on the property of fidelity - the ability for an explainer to appropriately discriminate between more and less relevant features in the input. To put it another way, fidelity is how well the explainer agrees with the way a model actually works. They point out that a major issue with most proposed metrics is that they only access the fidelity of individual explanations, as is the case in [99].

Most metrics to evaluate fidelity of individual saliency maps have hinged on using the saliency map to perturb the input while the researcher observes changes to the output [113][114]. Petsiuk et al.[12] proposed the Deletion and Insertion metrics to evaluate how well their method identifies the true cause of a deep classifier's decision, and later in [84] for deep detectors.

Deletion sequentially removes pixels, starting from the maximum value of the saliency map, while measuring how quickly the network's output deviates from the original prediction. It is expected that if the explainer does correctly allocate importance, the prediction will diverge quickly and then plateau when pixels become less important. Contrarily, Insertion measures how much the output converges as pixels are added back into the image. Here, the opposite effect is expected - that the prediction will approach the original prediction quickly as more more pertinent pixels are added back, and then plateau. The results are plotted and the final metrics are obtained from the Area Under Curve (AUC). Therefore, a low AUC for Deletion, and a high AUC for Insertion, are desired.

Tomsett et al.[110] state that, to be useful, metrics should have high statistical validity. However, this is not possible without groundtruth references - which there

are none for explanations [8].

Therefore, they present the following recommendations: New metrics should be compared directly with previous metrics that measure the same, or similar, properties. If a metric can be implemented in different ways - for example, pixel perturbation - then these metrics should be analysed under different implementations to establish their effects. Metrics should be analysed for how they might be "tricked" in different contexts. They tested a similar technique to Deletion, and found that it gave a high explainability score to maps produced by Sobel edge detection, despite not having any dependence on the explicant's processes. Finally, they state that metric developers should encourage their users to investigate and understand sources of variance in the metric scores. This should influence decisions about which saliency methods are appropriate for a particular model.

Other modes of assessment include validating the explainer by comparing it to human-annotated groundtruth. The theory is that a 'good' explanation should indicate that the model's attention aligns with a human's preconception. This type of assessment was employed by [115], and they refer to it as the 'Pointing Game'. To 'play' the Pointing Game, one must extract the maximum point in the saliency map. This point is then compared to the human groundtruth label. A 'hit' is considered if this maximum point is within the groundtruth, otherwise, it is considered a 'miss'. The Pointing Game accuracy is then calculated using Eq. (1.13). This methodology does not require the method to highlight the full extent of the object - such as is the case in the task of segmentation - and does not account for the model's classification accuracy.

$$Acc = \frac{\#Hits}{\#Hits + \#Misses} \tag{1.13}$$

The Pointing Game was extended by the creators of Grad-CAM [4] to also consider the classification accuracy. This is achieved by generating the maps for the top-5 class predictions, and adding the opportunity to reject any of these predictions below a given threshold. A hit is now considered if the visualisation correctly rejects the predictions which are not part of the groundtruth.

Zhang et al.[115] state the Pointing Game is designed to evaluate the discriminativeness of the explainer. However, this is incorrect, as a single point does not heavily support the notion that the explainer is capable of making fine distinctions.

Furthermore, since it relies on a human groundtruth, it also introduces other issues. Firstly, the test is reliant on the model's performance. Consequently, an instance is counted as a 'miss' if the model disagrees with the groundtruth. This would be unfair to the explainer, as it may still be correctly allocating correct causal information for the prediction.

The second issue with relying on the human groundtruth is that it may penalise the explainer for highlighting contextual information outside of the groundtruth [97]. The significance of contextual information in object recognition tasks, including object detection and semantic segmentation, has been established in [116][117]. Indeed, the authors of D-RISE found that on occasion, DNNs can utilise contextual regions outside of a bounding box when predicting that box.

Chapter 2

Development of Software Tools and Datasets

This chapter reviews the datasets relevant to this PhD thesis, beginning with an evaluation of existing datasets based on two key criteria: whether they provide examples from aerial or ground-based drone perspectives, and their suitability for explainable AI evaluation. The chapter then presents a novel dataset - the XI Autonomous Driving dataset - specifically developed to assess explanation techniques in 2D object detection from a ground-based drone in an urban environment. Additionally, it outlines the technical tools, software packages, and sensor rig used for data acquisition and processing of this dataset. Finally, the chapter introduces the 'Wrapping Game', a novel approach for evaluating the explainers proposed in this work.

2.1 Introduction and Motivation

Datasets are essential, not only for evaluating algorithms, but also for training deep learning models to perform effectively in real-world applications. In deep learning, a dataset is more than a collection of isolated, random pieces of information, it is a structured assembly of interconnected components that should accurately reflect the environment where the algorithm will be deployed. Machine learning algorithms, especially Deep Learning models, have become integral to AD systems, where they support critical tasks in perception [118], guidance [119], and odometry [120]. The recent advancements in AD systems owe much of their success to large-scale image datasets, which provide the rich data needed to train models that can generalise well



Figure 2.1: Figure of the capture vehicle - a Renault Twizy. The sensors utilised include; left and right thermal cameras, a 32 channel Lidar, and an active RGBD camera.

across varied conditions.

With the rapid adoption of machine learning in safety-critical fields, and the increasing complexity of these systems, there has been a growing emphasis on explainability to illuminate the inner workings of learned models. This, among other factors, has driven substantial growth in post-hoc explanation methods for deep learning aimed at image understanding [99][121][122]. The XI Autonomous Driving dataset proposed here is specifically designed for autonomous systems and captures diverse scenarios that mimic real-world complexities, such as environmental variability, lighting changes, and object diversity. By incorporating these real-world elements, this dataset offers a comprehensive foundation for both training and evaluation, while enhancing the robustness and interpretability of models in complex and dynamic scenarios.

The wider community has made claims that explanations may be used to potentially satisfy regulatory requirements [123][124], assist practitioners debug their model [125], and search for biases or irregular behaviour in their algorithms [126][16].

In the fields of image processing, pattern and object recognition, pixel attribution - or saliency maps methods - are a popular approach to investigating contextual information behind a prediction made by a DNN. Nonetheless, Adebayo et al.[99], found that some existing explainers produced saliency maps that were independent of the model's learned parameters and the data. This would suggest that any such method would not be able to identify causal information in the input, and therefore not be adequate for synthesising explanations.

Often decisions made by developers are based on pre-existing assumptions lacking empirical basis - a significant obstacle to the development of genuine Explainable AI. Without a means to effectively evaluate these explainers, they undermine the core concept of explainability, risk misleading development, and exacerbate the dangers of deploying DNNs in the wild. To meet this goal, within this chapter, is proposed the very first dataset aimed at the development and validation of explainable scene and object recognition for autonomous ground based vehicles.

While there do exist metrics to access the quality of a given explanation technique, there is no strong consensus in the literature on which approach should be used for a definitive assessment. Tomsett et al.[110] investigated the property of fidelity in XAI - the ability for an explainer to appropriately discriminate between more and less relevant features in the input. Most metrics to evaluate fidelity of individual saliency maps have hinged on using the saliency map to perturb the input while the researcher observes changes to the output[113][114]. Petsiuk et al. proposed the Deletion and Insertion metrics to evaluate fidelity of XAI approaches, for use in classification[12], and object detection tasks[84]. Other modes of assessment include validating the explainer by comparing it to human-annotated groundtruth. This type of assessment was employed by [115], and they refer to it as the 'Pointing Game'.

The aforementioned techniques will be applied, where best appropriate, in subsequent chapters for the purpose of evaluating the explainers that are proposed in this thesis. In addition, in this work the concept of explainability is introduced directly into AD at the dataset level. To the best of this author's knowledge, there does not exist a work that utilises attempts to provide a groundtruth for explainability on real-world data. Instead, the standard approach is to generate a synthetic dataset where the causal information is selected a-priori [127][128][129][130]. Therefore, the datasets used in these works are not representative of realistic image dataset.

Section 2.2.1 contains a literature review of existing XAI-based datasets, in which two main approaches that authors currently take were identified. The first is to provide a human-annotated 2D localisation label that can be used with techniques, such as the Pointing Game. The second is provide a means to intentionally bias the data to cue the explainer. The second approach has only really been applied to tabular data. Here, in addition to a major object class label, there are ADLs provided, these can be used to create purposely biased image sets to evaluate new vision-based explainers. In addition these ADLs, provide users with a better understanding of the proposed dataset, and make developers aware of biases in the data.

In addition to object detection, this dataset will support the task of end-to-end planning for autonomous driving agents. This is a crucial area that attracts a lot of research interest, and presents significant challenges. Conducting real-world training for self-driving agents is unsafe and expensive, which is why many researchers choose to experiment in simulated environments. Although these experiments have yielded promising results, there is always a performance gap due to the difference on the input representation between simulator and the real world. The dataset proposed here, aims to bridge this gap by providing data for sim-to-real scene understanding. Existing datasets for autonomous driving include semantic segmentation information, allowing vision models to understand the scene. However, these datasets describe each individual class of objects, which is beneficial for semantic segmentation tasks, but not optimal for sim-to-real applications. The complexity of the real world and simulation environments differs significantly.

This issue is considered here by abstracting complex real-world scenes into drivable lanes, non-drivable lanes, and obstacles with explainable information. These are all critical elements that self-driving algorithms rely on. This abstraction provides an efficient way for sim-to-real scene understanding, utilising methods such as contrastive learning. Focusing on these essential components can improve the transferability of models trained in simulation to real-world scenarios.

Section 2.2.1 contains a brief literature review of currently available datasets, comparing them to the one proposed here. It was found that there is a significant lack of AD datasets featuring the London metropolitan area, despite Transport for London (TfL) releasing a statement in support of the capital as a global testbed for autonomous vehicle innovation, and expressing its commitment to engaging with AD deployment within the city [131]. To address this gap, a bespoke platform was

developed to gather new data. The platform, shown in Fig. 2.1, is discussed in detail in the central part of this chapter.

To address the limitations of the Pointing Game, this chapter introduces the Wrapping Game as a validation procedure. For this method, semantic segmentation datasets can be used to evaluate how discriminative a proposed explainer can be of the object of interest. Here, the Wrapping Game is used to assess and compare the proposed DetDSHAP explainer to another explainer based on LRP introduced by Karasmanoglou et al.[106]. The DetDSHAP explainer will be introduced in Chapter 5.

2.2 Available Datasets

2.2.1 Existing XAI Datasets and Validation Procedures

Recent works have approached the validation of saliency methods by using the pixels' relevance as an object detection signal [115][130]. Arras et al. [130] propose CLEVR-XAI, a synthetic visual question-answering dataset with generated masks representing the explanation groundtruth. The mask is generated solely on the target's pixels, while the rest are defined as background. They also introduce two metrics that utilise the generated masks to evaluate a given saliency method: Relevance Mass Accuracy, and Relevance Rank Accuracy. The first is computed as a ratio of the sum of relevance within the groundtruth over the total relevance across the entire input. The latter is the number of high relevance scores within the groundtruth, divided by the area of the groundtruth.

The ADL approach proposed here is inspired by OpenXAI, introduced by Agarwal et al. [127] as a package to evaluate the quality of explanations generated by attribution-based explanation methods. They provide eleven evaluation metrics, some taken from [132] including six that can be used within their synthetic dataset, which are explained here. These metrics compute the fraction of top-K features common between the explanation and the groundtruth (similar to Relevance Rank Accuracy in [130]), considering their ranking order, sign agreement, and correlation. While they cover a broad range of applications, their approach is limited to tabular datasets for both discrete and continuous problems, and they do not extend their approach to the computer vision domain. Therefore, it would not be possible to apply their techniques directly to image data.

Mir'o-Nicolau et al. [128] extended the work performed in [127] to image processing without moving into the real domain. Their data is very simple, consisting of generated images containing basic patterns and shapes. Most of the synthetic datasets that were investigated in this study only considered relatively easy tasks, which could provide misleading validation results. To ensure that real data can be used in the same validation procedures as Agarwal et al. and Mir'o-Nicolau et al., ADLs was used so that the data could be represented in a tabular format with human-friendly image descriptors.

In contrast to these synthetic datasets, Kim et al. [133] utilised a subset of the Berkeley DeepDrive dataset (BDD-X) to generate textual explanations for autonomous driving decisions. This real-world dataset provides human-annotated driving behaviours, such as identifying why a vehicle slows down at a red light. While this method grounds explanations in the internal reasoning of the model, it lacks the relevance to object detection tasks, which are the focus of the evaluation methods proposed here.

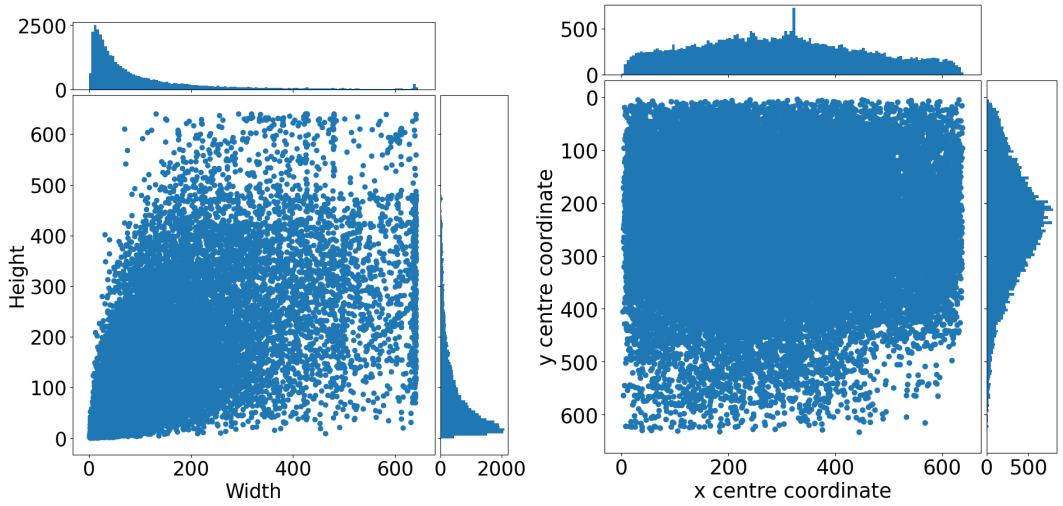
Localisation-based evaluations alone may be misleading, as they assume the model's decision is based solely on the object itself, not its context or background, which cannot always be ensured in real-world image datasets [130]. On the other hand, applying synthetically weighted datasets provides no guarantee that models trained on synthetic data will adhere to groundtruth explanations [127][134]. To properly vet an explainer, it must be evaluated using more than one procedure. Therefore, as part of the proposed dataset, both semantic segmentation and ADLs are provided to broaden the possible avenues for validation.

This section will present an analysis of other datasets that are intended to be utilised by autonomous and unmanned vehicles in urban environments, and establish what the special requirements are for datasets covering this modality. In Table 2.1, a summary of both large and small scale datasets within this criteria is made available. The main focus of the dataset proposed in the latter sections of this chapter is on object detection, and drivable lane recognition. In the proposed dataset access to a GPS/IMU system, LiDAR, and thermal cameras, is also made available. These additional sensors make it possible to expand into other tasks in the future, such as Visual odometry, Simultaneous Localisation and Mapping (SLAM), 3D estimation, and multi-model detection. Hence, datasets considering these applications will also be addressed to a limited degree.

Dataset	#Cat	Occ lbl	Sem lbl	Ori lbl	AD Pers	Dataset Size	Data Collection Method
VisDrone[62]	10	✓				10,209	RGB images by UAV
LabelMe[135]	183					11,845	Crowdsourced via web-based tool
ETHZ Pedestrian [136]	1				✓	2,293	Vehicle-mounted stereo cameras
Coco[50]	80	✓	✓			328,000	Crowdsourced via web-based tool
Daimler[137]	1	✓			✓	15,560 (pedestrian samples)	Vehicle-mounted camera
Caltech Pedestrian[138]	1	✓			✓	250,000 frames	Vehicle-mounted camera
ACDC[139]	19	✓	✓	✓	✓	4006	Vehicle-mounted camera
EPFL Multi-view Car[140]	20			✓		2000	Controlled indoor settings
CityScapes[141]	30	✓	✓	✓	✓	25,000	Vehicle-mounted camera
Kitti Detection[51]	8	✓	✓	✓	✓	14,999	Vehicle-mounted stereo cameras and LIDAR
BDD100K[142]	10	✓	✓		✓	100,000 (video clips)	Collected via dashboard cameras
Proposed Dataset	5	✓	✓	✓	✓	10,000	Vehicle-mounted camera, stereo thermal, and LIDAR

Table 2.1: The above table shows a comparison of similar object detection datasets, considering: the number of object categories (**#Cat**), presence of occlusion labels (**Occ lbl**), availability of semantic labels (**Sem lbl**), inclusion of orientation labels (**Ori lbl**), relevance to autonomous driving (**AD Pers**), total dataset size, and the data collection method which include the types of sensors used where applicable.

The task of object detection requires both the localisation and classification of objects within a scene. It is a trending topic in the computer vision community, and has various industrial applications such as autonomous driving, anomaly detection, face detection, and behavioural recognition. The PASCAL VOC dataset [143][144] is one of the pioneering works in generic object detection, which is designed to provide a benchmark for multiple tasks beyond object detection, as well as image classification, object segmentation, person layout, and action classification. The dataset contains 20 object classes and 21,738 images. ImageNet [145] builds upon the PASCAL VOC dataset, scaling up the number of object classes and images by several orders of magnitude, with 1,000 object classes and 1,431,167 annotated images. The MS



(a) Bounding box sizes.

(b) Bounding box centroid locations.

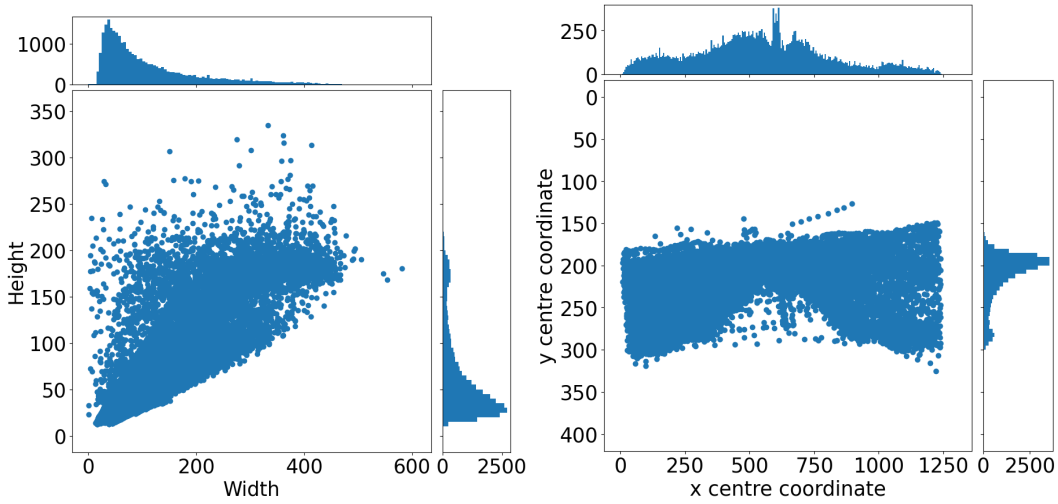
Figure 2.2: This figure shows the bounding box statistics for the coco dataset. The left subplot shows the width and height distribution of the bounding boxes as a percentage of the image dimensions. The right subplot shows the centroid location distribution of bounding boxes in pixels. Histograms in both subplots provide additional insight into the frequency of widths, heights, and locations.

COCO dataset from Lin et al. [50], contains more than 328,000 images with 2.5 million manually segmented object instances. It has 91 object categories with an average of 27.5k instances per category.

From Fig. 2.2b, it is evident that the size and shape of the bounding boxes in this dataset are generally spread out. That being said, the majority of the bounding boxes are under 120 pixels across the diagonal, with an average value of 101.5 pixels. From Fig. 2.2a, it can be seen that the centroids are generally distributed around the images. However, the majority are clustered around the centre of the image, and more are found in the upper area of the image than in the lower portions.

For comparison, within Fig. 2.3 and Fig. 2.4 is plotted the bounding box characteristics for the classes of "Car" and "Pedestrian" in the Kitti dataset. The most notable characteristic is that all the objects are located between 150 and 350 pixels from the top of the image. This is because the objects of interest would not be found in the sky, or too close to the capture vehicle. This information reveals that models trained on generic datasets will, more than likely, be less suited for deployment with AD perspectives. Moreover, as most object detectors have used these datasets as a benchmark, the architectures themselves may not be optimised for the application.

It is important to note that, other than KITTI, many of the datasets in Table 2.1 that were acquired in outdoor environments used fewer than six sequences and at



(a) Bounding box sizes.

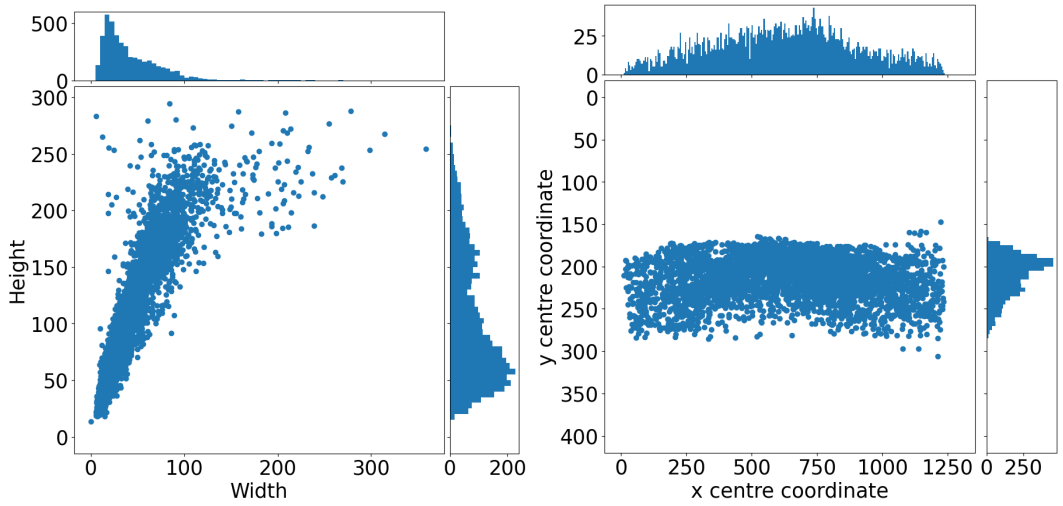
(b) Bounding box centroid locations.

Figure 2.3: This figure shows the bounding box statistics for the instances of "Car" the Kitti dataset. The left subplot shows the width and height distribution of the bounding boxes, whereas, right subplot shows the centroid location distribution of bounding boxes. Histograms in both subplots provide additional insight into the frequency of widths, heights, and locations.

most contained 6.4 km of acquisition. The Ford Campus dataset [146] employed a modified F-250 truck and included Lidar and visual data. Similar to the proposed dataset, the Ford Campus dataset used two sequences and included a distance of 5.1 km, which is comparable to the distance captured in the proposed work. The proposed dataset includes two different sequences with loop closure (which can be split into more sub-sequences) and provides access to 10,136 frames across the two sequences.

Looking to the number of categories used in object detection, the current datasets vary between relatively few, into the hundreds. The proposed dataset uses five primary categories of objects ["Drivable lane", "Non-drivable terrain", "Static obstacle", "Pedestrian", "Car"]. This is more than the number released in the initial version of KITTI and in the datasets proposed by [147], [137] and [136]. The latter three are only concerned with pedestrian detection.

Similar to the work by Mustafa et al. [140], the proposed dataset includes labels that indicate the facet of the vehicle being viewed by the observer. Their dataset consists of 2,000 images covering 20 different car models, with the 360-degree view divided into classes representing approximately 3 to 4 degrees each, resulting in roughly 100 facet classes per car model. In contrast, the proposed dataset represents vehicle facets with only eight possible classes, as higher fidelity is not necessary at this stage. A key difference is that the dataset presented here was not gathered



(a) Bounding box sizes.

(b) Bounding box centroid locations.

Figure 2.4: This figure shows the bounding box statistics for the instances of "Pedestrian" in the Kitti dataset. The left subplot shows the width and height distribution of the bounding boxes, whereas, right subplot shows the centroid location distribution of bounding boxes. Histograms in both subplots provide additional insight into the frequency of widths, heights, and locations.

in a controlled environment like theirs, which was acquired at a car show. This controlled, indoor setting enabled Mustafa et al. to achieve such fine granularity more easily, whereas the proposed dataset had to contend with the challenges of real-world, less controlled conditions.

Russell et al.[135], introduce LabelMe which includes an image dataset for object detection and recognition. For some of their objects they have also provided additional contextual labels. However, these are generated by a user on the fly, hence their labels are less specific, and there is no consistency across the dataset. This will make it much more challenging to extract specific clusters of image data.

Of the Datasets listed in Table 2.1, only the proposed dataset contains a significant sample from a major British city. Indeed, because there does not exist a great abundance of data collected within the London metropolitan area in existing works, there is a greater need for data acquired within this environment, particularly with the presence of the traditional London Taxi and New Routemaster buses. This data is provided by this work.

2.2.2 The VisDrone Dataset

This section introduces the VisDrone detection dataset[148][10], a dataset which is used extensively throughout this thesis to evaluate the developed explainers on aerial

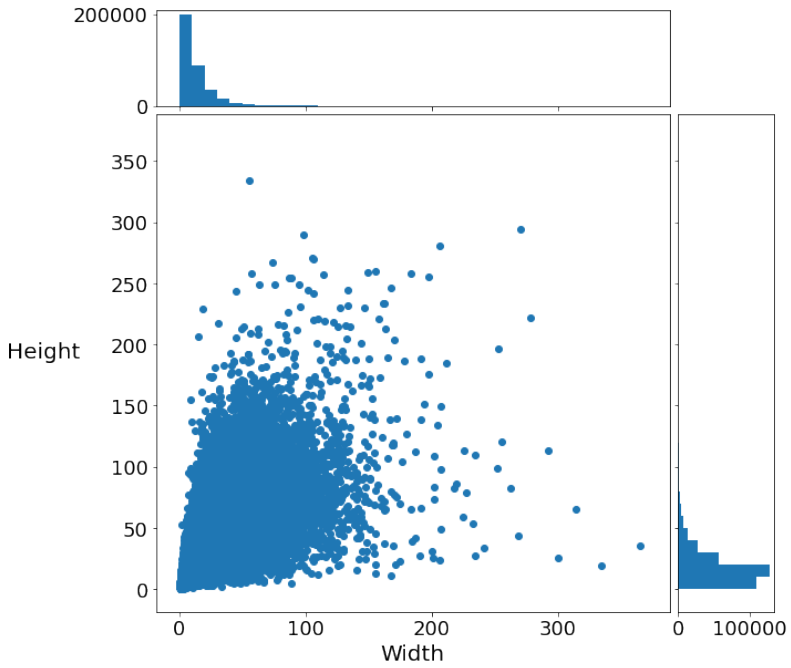


Figure 2.5: Scatter plot showing the width and height of the bounding boxes in the VisDrone training set in pixels. The top histogram shows the occurrences of a given width, while the right side histogram shows the occurrence of a given height. The plot is limited to 400 pix, although there are some boxes wider than this

imagery. The VisDrone competition was initiated in response to the lack of large-scale datasets featuring drone-captured scenes. Their detection dataset comprises 10,209 static aerial images from various cities across China [10]. Each image has been manually annotated with bounding boxes for 10 predefined classes: [pedestrian, person, car, van, bus, truck, motorbike, bicycle, awning-tricycle, and tricycle]. A key challenge of this dataset is the prevalence of small objects, as illustrated in Fig. 2.5, where most bounding boxes measure under 50x50 pixels. Additionally, objects are often occluded by background elements like buildings or foliage, or by other objects of interest. The images are also densely populated, containing numerous objects.

As is the case with imagery from the perspective of an autonomous road vehicle, performing object detection on images captured from the perspective UAV adds unique challenges, including viewpoint change, dynamic scales, occlusion and background clutter[10][149][148][150]. To illustrate this, some samples have been provided in Fig. 2.6 where it can also be seen that this dataset also contains night-time conditions.

This dataset presents two challenges for attention based explainers that traditional saliency approaches will likely struggle to overcome. Firstly, the size of the objects in the image means that only a small amount of pixels are used to represent

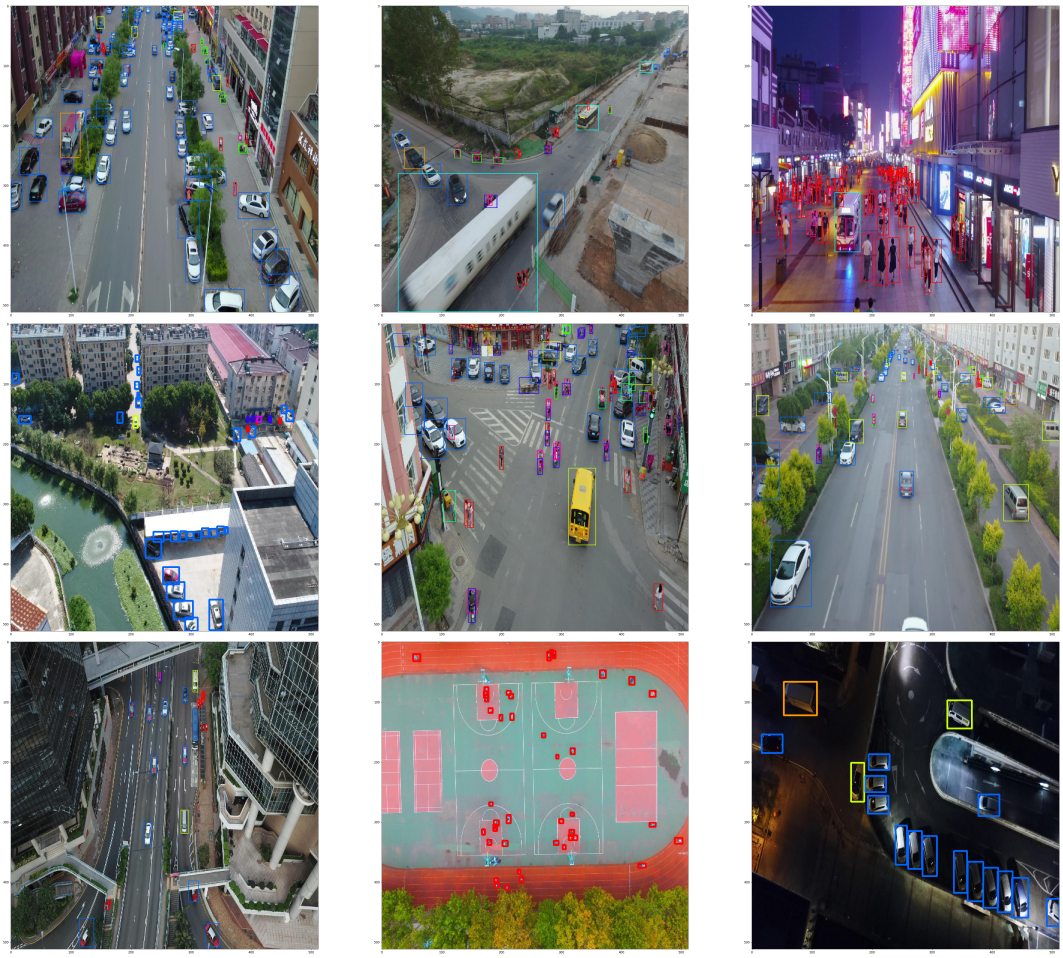


Figure 2.6: Examples from the VisDrone Detection Dataset[148]

a single object, and thus a saliency method may struggle to convey the importance of these small features. Secondly, the density of the objects within the image, which can often be found clustered together as seen in Fig. 2.6, will make it more challenging for an explainer to effectively disentangle causal information for neighbouring objects.

2.3 Sensor System Architecture and Software Tools

This section provides an overview of the sensor setup and custom software solutions developed to manage data collection and annotation. This setup, shown in Fig. 2.1, integrates multiple synchronised sensors. To handle the unique data requirements of these sensors, custom software solutions were developed: the Capture Software (Section 2.3.2) manages real-time data synchronisation across sensors, while the XAI Labelling Software (Section 2.3.3) provides a semi-automated platform for human agents to refine AI-generated annotations. Together, these tools enable efficient data

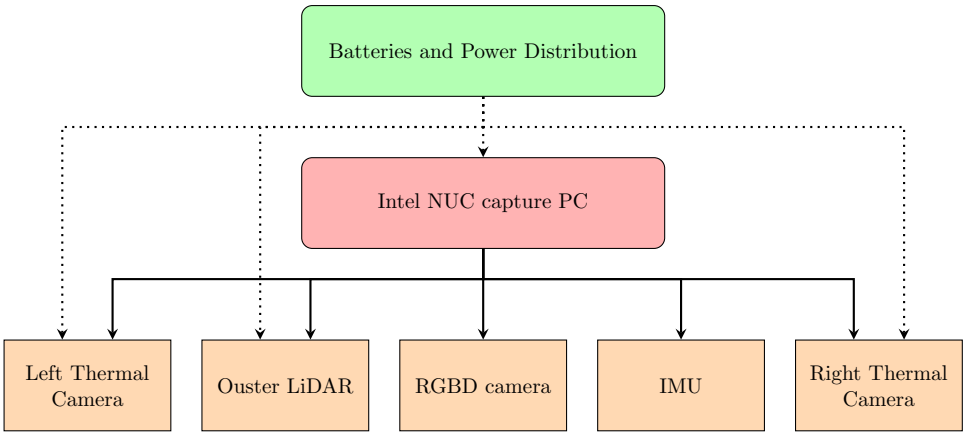


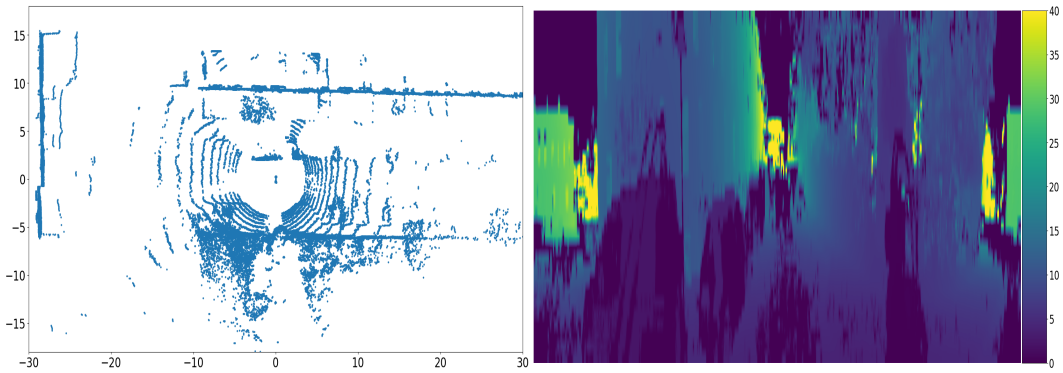
Figure 2.7: Overview of the sensor system architecture, showing the Intel NUC capture PC connected to various sensors, including thermal cameras, LiDAR, RGBD camera, and IMU. Additional power was required for some sensors, this is indicated in the diagram, others were able to powered from USB connection to the Capture PC.

capture and consistent labelling, forming a robust foundation for the dataset.

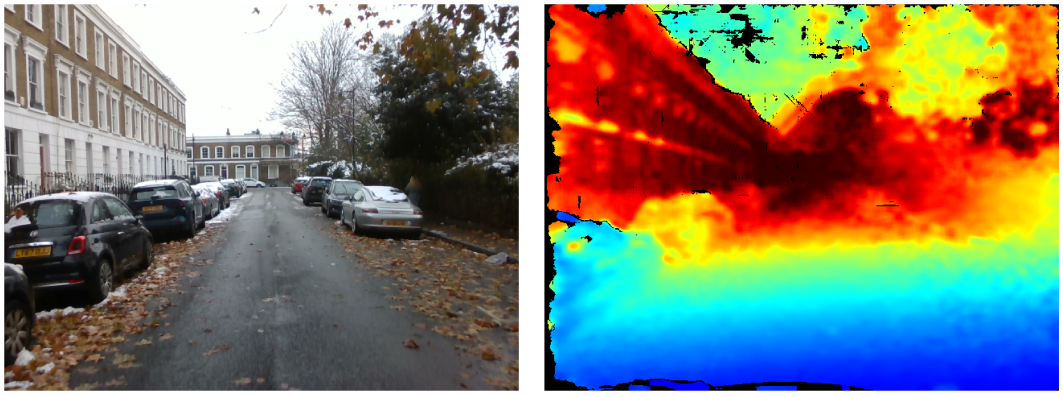
2.3.1 Sensor Rig

Figure 2.1 shows the placement of the sensors on the vehicle used for the data collection. There is also available a functional diagram in Fig. 2.7. While the main focus in this work is explainability on visible light imagery, additional sensor data was collected, all of which were synchronised at 10 frames per second. The developed rig included an ouster OS1 32 channel Lidar, two Flir Vue Pro R thermal cameras, an intel realsense D435i RGBD camera, and a high-resolution Inertia Measurement Unit. Figure 2.8 shows an instance from the available dataset with all the perception sensor data. The Flir Vue Pro R cameras have a FoV of $69^o \times 56^o$ and operate in the Spectral Band $7.5 - 13.5\mu m$. These required additional hardware including a frame-grabber for each camera to capture $690 \times 490pix$ images from these thermal cameras on the analogue video output at $10Hz$.

From the Intel Realsense camera both Depth and RGB images are made available, within Fig. 2.8b is shown frames from both these sensor modalities. The Depth module is a stereoscopic sensor that uses an active IR emitter, has a FoV of $87^o \times 58^o$, while the maximum range of the depth image is $3m$ - hence the requirement of a LiDAR. This camera was configured to output normalised $1280 \times 720pix$ Depth images at 30Hz. The RGB camera has a FoV of $69^o \times 42^o$ and a sensor resolution of 2MP. The LiDAR is a 32 Channel Ouster OS1 rev D, which is configured with a horizontal resolution of 1024. The point clouds are stored in csv format: $[x, y, z]$.



(a) Ouster LiDAR sensor data with the top-down view of the point cloud displayed in the left plot and the depth image created from that same point cloud displayed in the right plot. The colour bar along the side of the right plot shows the distance in meters.



(b) Data available data in the proposed dataset from the Intel Realsense camera. These include the RGB image and the Depth image. Note that the maximum range of this sensor is 6m and as such may not be suitable for many AD activities. This made it necessary for the inclusion of the LiDAR in our sensor suite.



(c) Demonstration of thermal images available in the proposed dataset from the left and right Flir cameras.

Figure 2.8: This illustration shows the visual sensor data available in the proposed dataset. All the sub-figures show sensor feeds taken from the same frame in time.

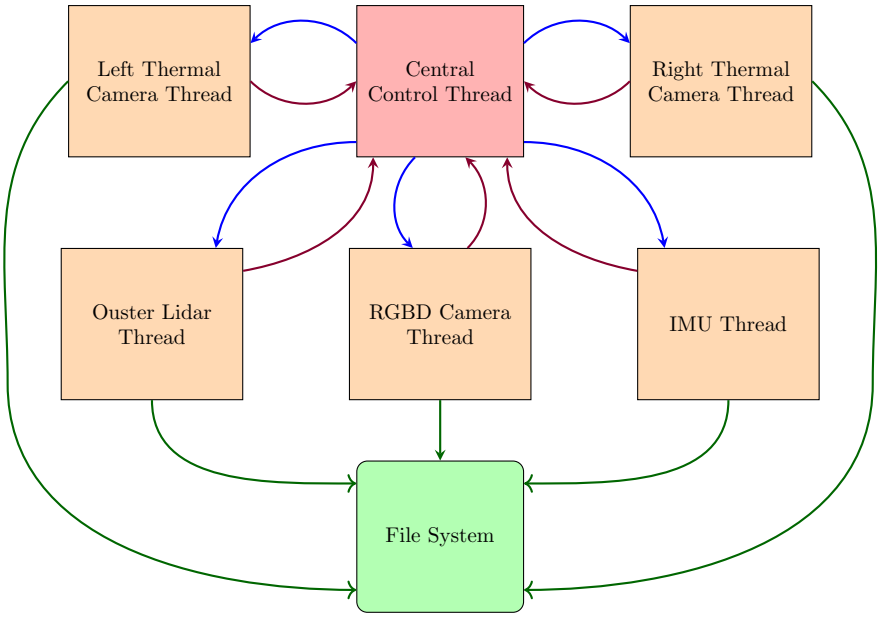


Figure 2.9: Functional overview of the software threads, in which each cycle is composed of three steps. Step 1: the central thread sends a trigger message to all sensor threads simultaneously. Step 2: Each thread will take the most recent frame and save it to the filesystem. Step 3: The threads send a message back to the central thread to notify that the save is complete.

2.3.2 Capture Software

Effectively integrating the sensors presented a significant challenge due to variations in their driver packages, and the unique data handling mechanisms implemented by each driver. Another challenge was managing the substantial sensor payload, particularly from the 32-channel Lidar, which generates 6,464 bytes per packet [151]. These inconsistencies and data demands required a robust solution that could handle each sensor's data stream independently, while maintaining synchronisation across the entire system.

To address this, the capture software was developed in C++ to leverage its performance capabilities and support multi-threading, which was essential given the lack of hardware synchronisation among the sensors. Multi-threading enables each sensor's data stream to be processed in real-time and independently, minimising latency and ensuring that each frame is captured simultaneously across sensors. This approach mitigates issues of time drift and data misalignment, which are essential for creating coherent multi-modal datasets.

As shown in Fig. 2.9, the software's functional overview illustrates the multi-threaded approach for capturing synchronised frames from each sensor in the rig. Each sensor operates within its own dedicated thread, enabling continuous data capture without blocking other processes. The Central Control Thread initiates

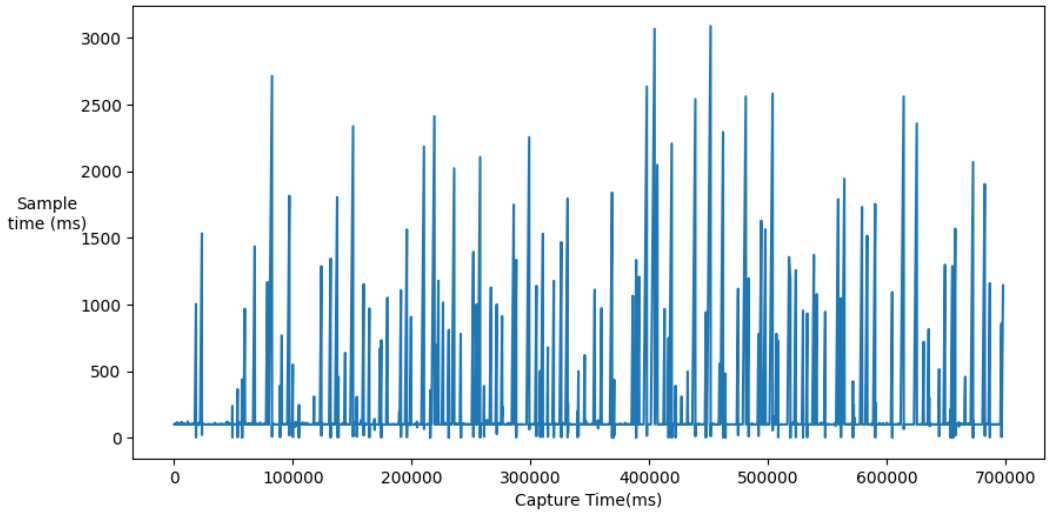
each capture cycle by sending a trigger message to all sensor threads simultaneously, ensuring synchronisation across all devices. Following this, each sensor thread saves its most recent frame to the File System, and then sends a completion message back to the Central Control Thread. This coordinated cycle allows the system to handle high data rates and maintain precise synchronisation, crucial for generating accurate, multi-modal datasets.

The capture times are recorded in the IMU data text files, and Fig. 2.10 shows the plotted sample times. While the sample interval does occasionally deviate from the target 10 Hz, with the worst instances reaching up to three seconds between samples, the vast majority of samples are consistently captured within the targeted 100 ms interval. For both summer and winter captures, the average sample interval was 112 ms and 123 ms, respectively, which is within acceptable limits for our purposes. The larger delays are attributed to instances when the buffer for saving files fills up, causing a temporary pause in data capture.

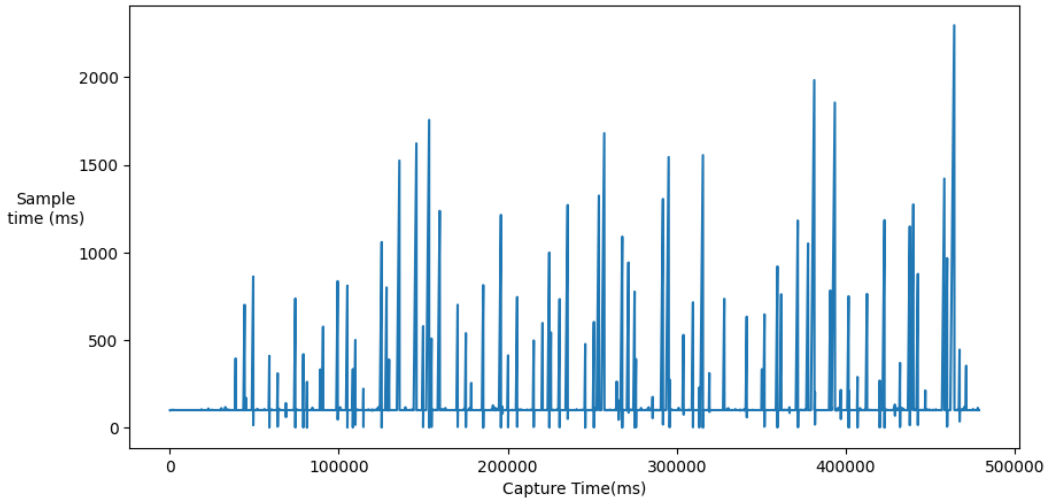
2.3.3 XAI Labelling Software

This section outlines the procedure for creating the semantic labels required for the proposed dataset and introduces the tool developed to facilitate this process. The tool was developed as a web-accessible platform using the open-source Django framework[152] to facilitate collaborative labelling among multiple human agents. This design choice allows for remote access and supports a consistent labelling process across users. The tool operates in a semi-automated fashion, where AI-generated labels are provided as initial predictions that human agents then refine, ensuring accurate and contextually appropriate annotations. Figure 2.11 illustrates the labelling interface, which incorporates features to streamline the workflow, enhance consistency, and maintain data quality across contributors.

The labelling process operates in a semi-automated fashion. Initially, each image is processed by an AI agent, which produces an initial guess of the semantic labels. The AI agent in this case is a Panoptic Feature Pyramid Network (PFPN)[64], a model previously discussed in Section 1.5. As detailed in that section, PFPN extends the Feature Pyramid Network (FPN) architecture by integrating instance and semantic segmentation, enabling panoptic segmentation—a unified approach to detecting both objects and amorphous background regions. While PFPN has computational constraints that may limit its real-time applicability in autonomous



(a) Summer set.



(b) Winter set.

Figure 2.10: This figure shows the timing characteristics of the developed capture software.

vehicles, its high-resolution segmentation capability makes it well-suited for generating precise semantic labels in offline annotation tasks. In this labelling workflow, the model is pretrained on the COCO dataset[50] and implemented using Detectron2[65]. The AI-generated labels are then meticulously reviewed and refined by human agents to ensure accuracy for the specific classes of interest.

Figure 2.12 illustrates the semantic labelling interface of the developed web platform. The figure highlights the typical corrections a human agent might be required make to the AI-generated initial labels. These corrections include splitting labels where multiple objects have been erroneously grouped, completing labels that are missing parts of an object, removing false positives, and identifying objects that were initially missed. Additionally, the human agent is responsible for distinguish-



Figure 2.11: This diagram outlines the process flow for labeling datasets using the developed software. It covers the stages from inputting RGB images, processing through AI and human agents, to the final output of labels which are stored in XML format.



Figure 2.12: This figure demonstrates the interface for creating the semantic labels on a given instance. The left image shows the initial guesses produced by the AI agent in white. The right image shows the refined labels by the human agent. Also indicated are some corrections that are typically required to be made by the human.

ing between drivable and non-drivable terrain, a task the AI is unable to perform. A finalised segmentation label is presented in Fig. 2.13, from which 2D bounding boxes are generated using the extreme points of the semantic regions.

Once the semantic labels are finalised, the next step is for the human agent to select the ADLs. This is done through the interface shown in Fig. 2.12. Each label has a corresponding list of options available via drop-down menus. As illustrated in the figure, not all labels apply to every object. For example, 'Static obstacle class' and 'Traffic light state' are irrelevant for an object classified as a 'Car'. After finalising the ADLs, all relevant data is stored in an XML file for future use by practitioners, with a backup in a Structured Query Language database.

Developing this proprietary software was necessary to meet the specific requirements of the proposed dataset, particularly for generating the ADLs. This approach also centralised the dataset, simplifying data management, and enabling remote access to the interface. As a result, human agents do not need to install any specialised software, and there is no need to transfer large amounts of data to the users.

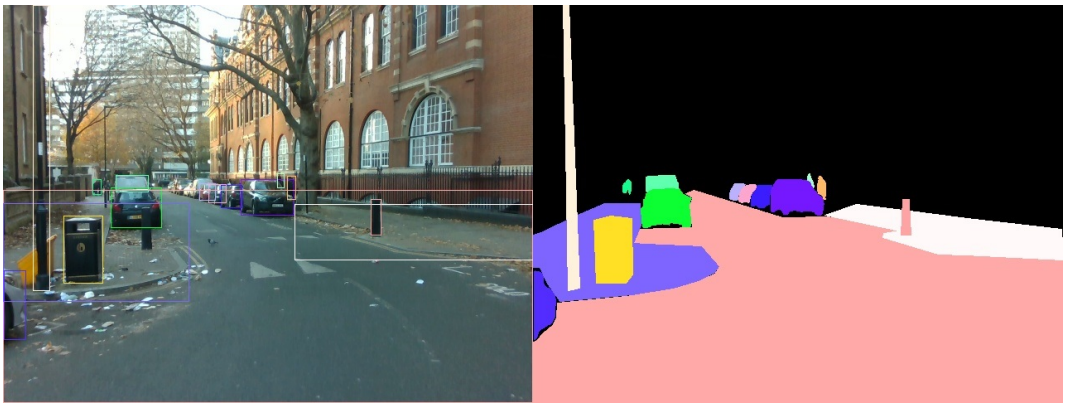


Figure 2.13: On the left portion of this figure is an image taken from our sensor rig while the right side shows the semantic label for this image.

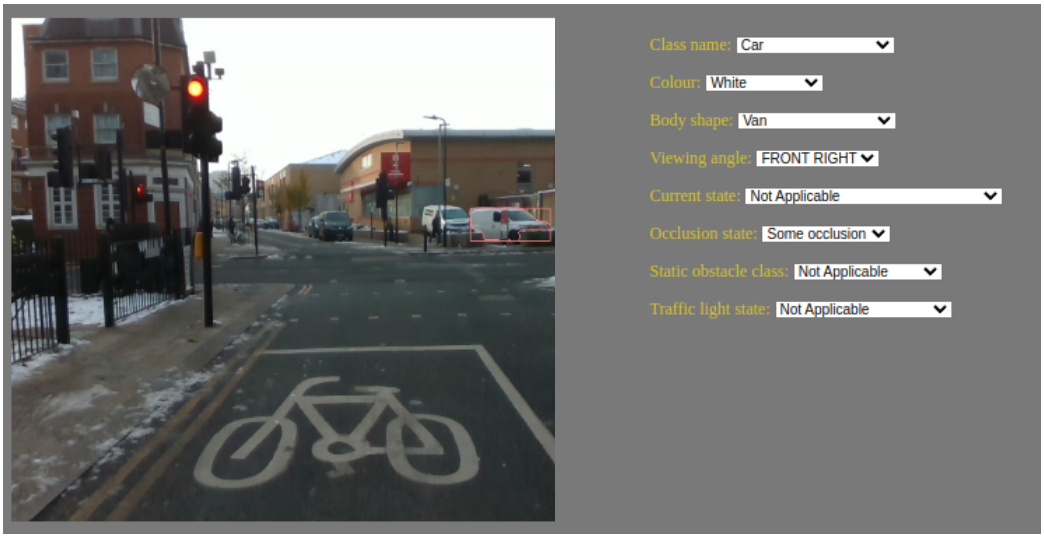
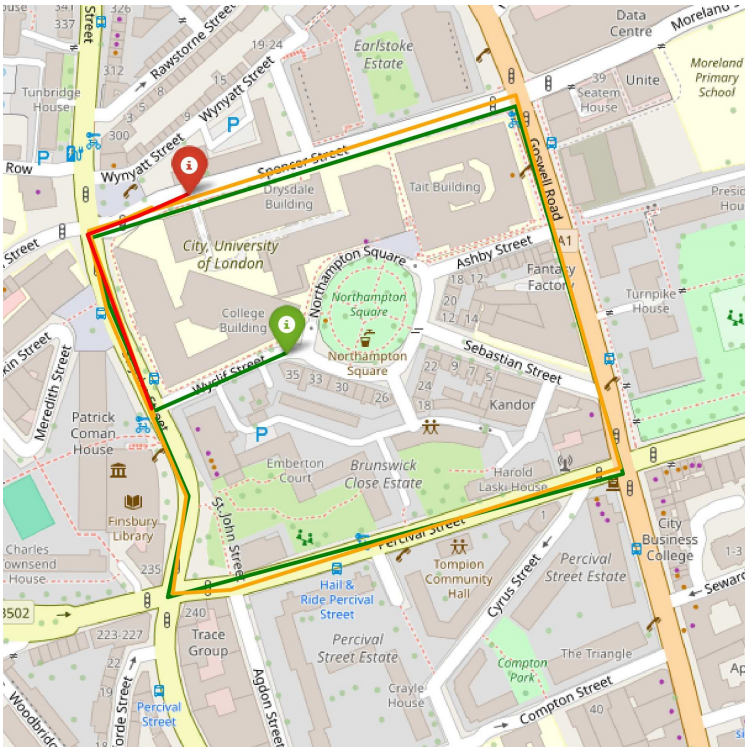


Figure 2.14: This figure shows an extract from the page that the human agent uses to set the ADLs. In this example the user is labelling the white van marked with the orange box. A full overview of the available labels are shown in Fig. 2.16.

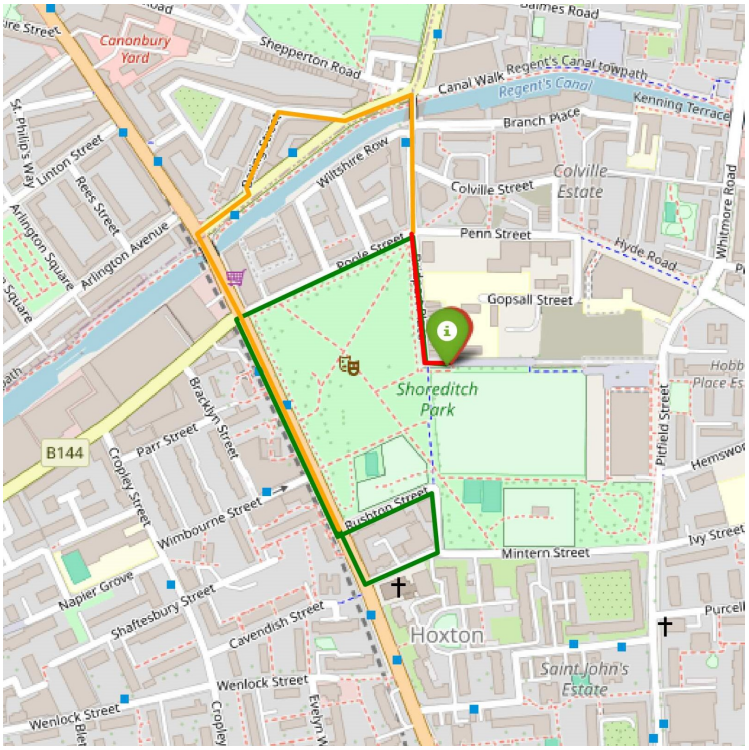
2.4 XAI dataset for Ground-based Drones

2.4.1 Environment

Two different collections were conducted, one under warm weather conditions, and one under winter conditions with snow present. The two routes taken are shown in Fig. 2.15a and Fig. 2.15b. Both depict scenes from an urban environment in central-North London. The sequence taken in warm weather conditions was collected in an area around a university and small business district. In contrast, the second sequence was captured in an area around a park in a residential district, with visible snow on most surfaces including vehicles.



(a) Summer



(b) Winter

Figure 2.15: This figure shows the routes taken during data acquisition where the green marker indicates the starting location and the red the finish. These have been produced using OpenStreetMap[153]

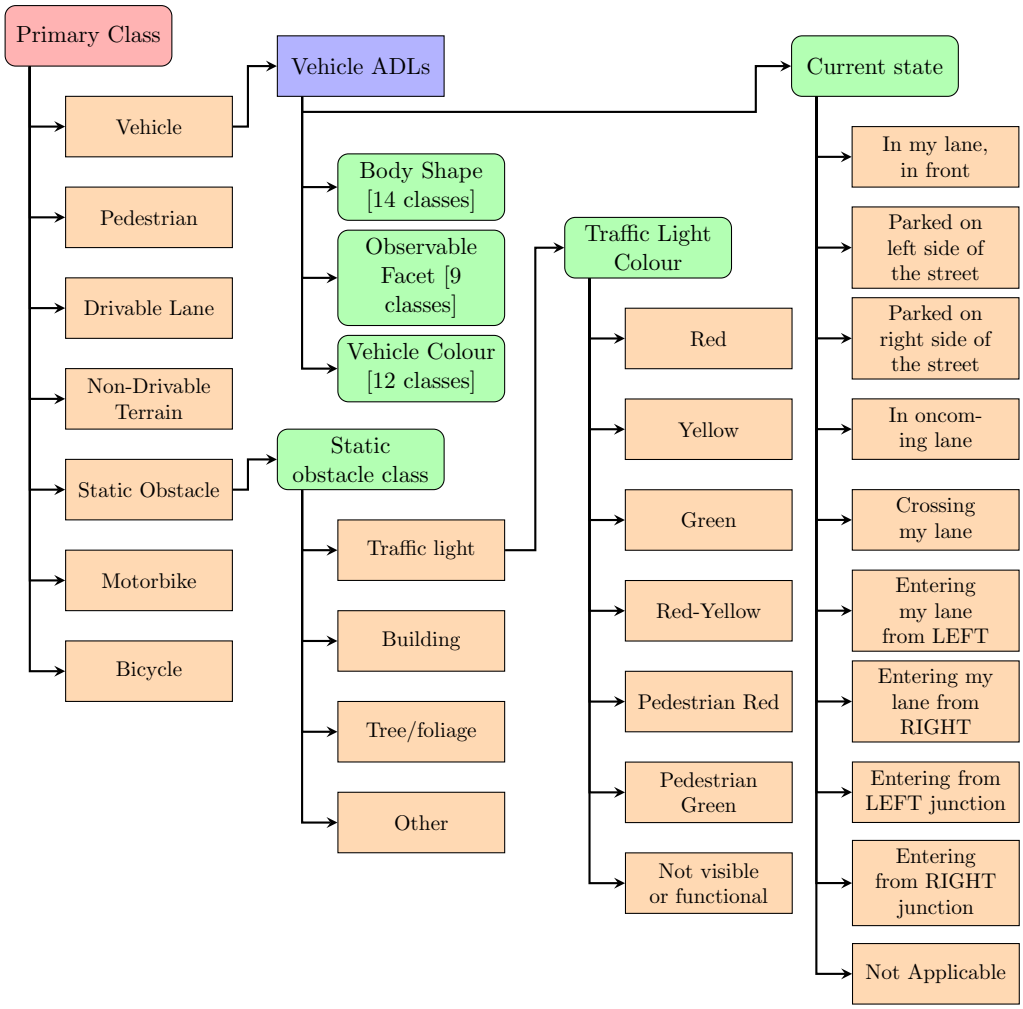


Figure 2.16: This figure shows a tree diagram which illustrates the links of the our ADLs. Each ADL is indicated in green. The class labels belonging to 'Body Shape' and 'Vehicle Colour' and their distribution can be found in Fig. 2.17. The Class labels for Observable Facet and their distribution can be found in Fig. 2.18.

2.4.2 Additional Descriptive Labels

The tree diagram in Fig. 2.16 shows the connections of the proposed Additional Descriptive Labels. While there are only seven primary classes in the proposed dataset, when considering the possible ADL amalgamations there are up to 10,762 unique label combinations. When considering this, the proposed dataset would be at the top of all the examples shown in Table 2.1 in terms of the number of categories.

Here, as in other works such as [51], [137], and [154], an occlusion label is provided to indicate how much of an object is obscured by other objects, or the background. This label is typically used to highlight challenging objects when evaluating a detector's performance. In this work occlusion is of particular interest, not only for identifying difficult cases, but also for how it affects the appearance of the object,

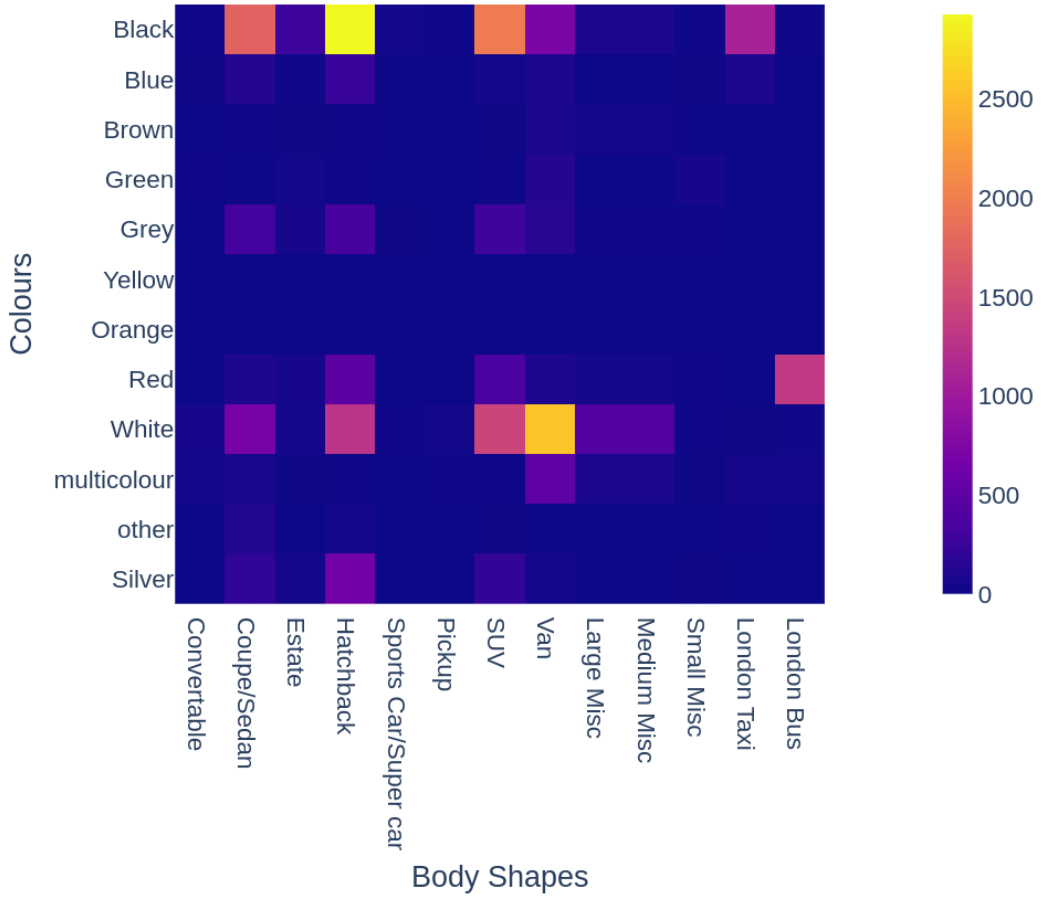


Figure 2.17: This matrix shows the distribution of the vehicle descriptions.

which is a key focus in generating the ADLs.

For the class of vehicles, there are five ADLs attached: 'Colour', 'Observed Facet', 'Body Shape', 'Current State' and 'Occlusion'. These additional criteria allow for the objects in the class to be better distinguished based on their visual characteristics. The matrix in Fig. 2.17 shows the distribution of the vehicle shape and colour in the proposed dataset. The ADL of 'Observed Facet' describes another important visual concept, which is the side of the given vehicle being observed, Fig. 2.18 shows the distribution of the nine classes of facet.

The ADL of 'Current State' is used to describe whether the current action or inaction of a given vehicle. This ADL describes whether or not said vehicle is in motion, and provides insights into that motion which could be pertinent to an AD system. In conjunction with this ADL and the 'Observed Facet' can hypothetically be used as part of a AD system in order to understand the behaviour of a given vehicle.

From examining the ADLs, it can be seen that within the vehicle class the top

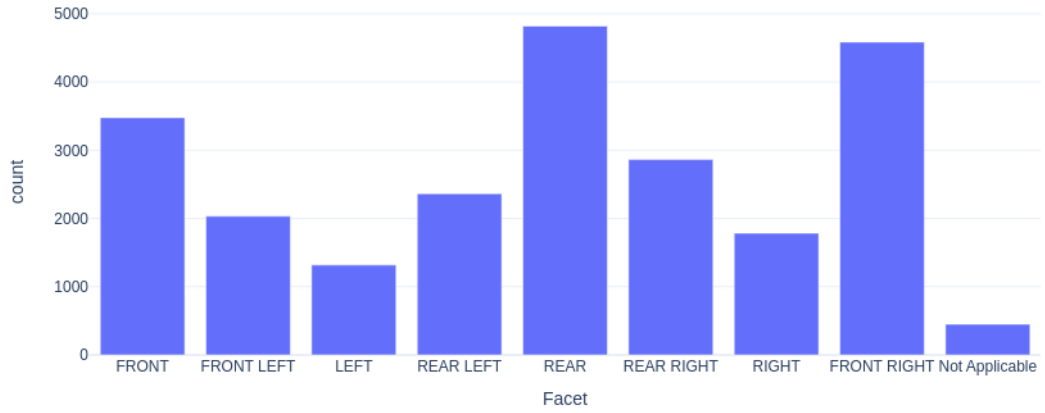


Figure 2.18: This graphic shows the distribution of observable vehicle facets in our dataset. The label of 'Not Applicable' refers to situations where there is too much occlusion or the vehicle is too far away for the observer to determine.

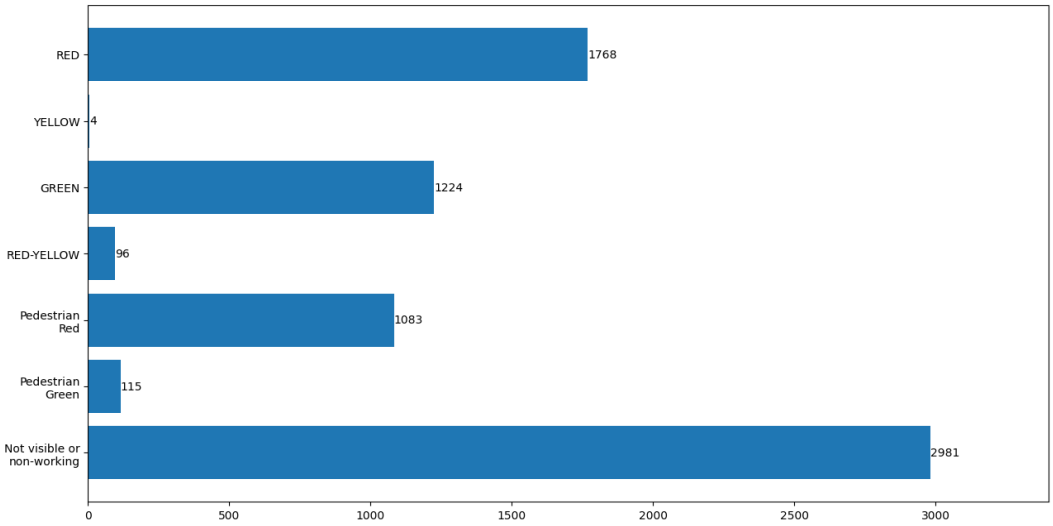
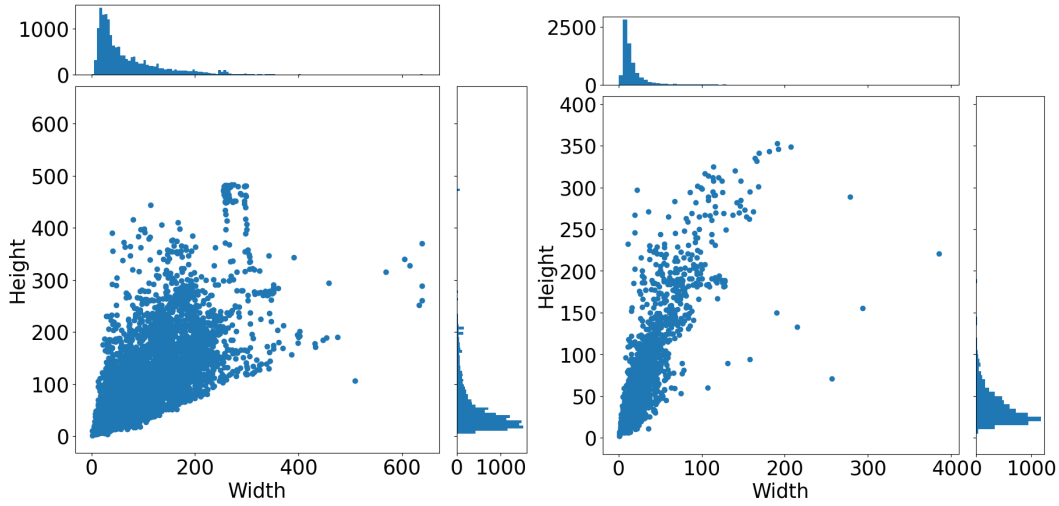


Figure 2.19: This graphic shows the states of the Traffic lights in our dataset.

appearances are 'Black Hatchback' and 'White Van'. It is also found that the most observed facets seem to be the front and rear of the vehicles. This bias would otherwise be hidden despite being critical information to a developer.

The dataset also includes the current state of the traffic lights captured during data collection, the distribution of which is found in Fig. 2.19. This information adds significant value to both autonomous vehicle navigation and advanced driver assistance systems (ADAS). Accurate detection and interpretation of traffic signal states are crucial for autonomous vehicles to navigate intersections safely and respond appropriately to changing traffic conditions. By leveraging this data, autonomous systems can make informed decisions about stopping, proceeding, or yielding to pedestrians.



(a) Bounding box sizes for 'Vehicle' class.

(b) Bounding box sizes for 'Pedestrian' class.

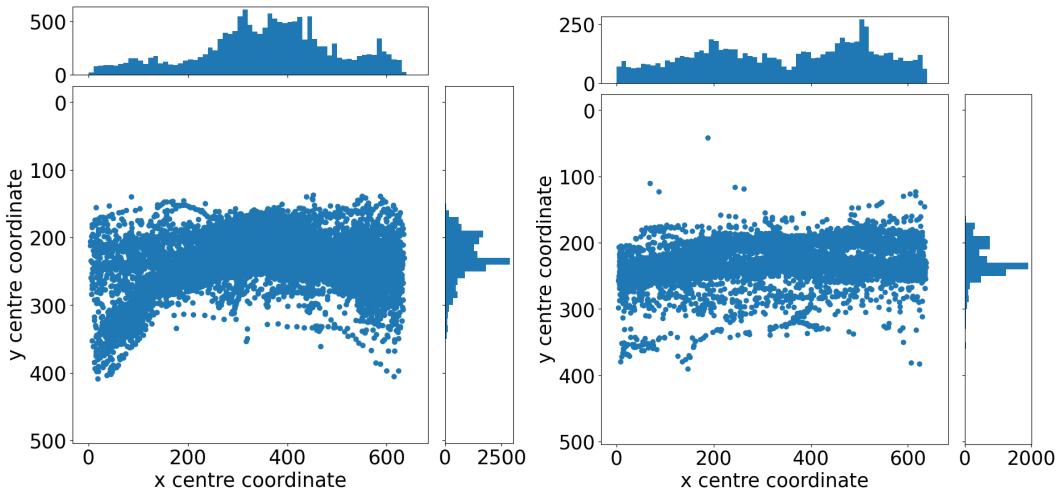
Figure 2.20: Comparison of bounding box sizes for the 'Vehicle' and 'Pedestrian' classes in the proposed dataset. The left subplot shows the width and height distributions for vehicles, while the right subplot shows the same for pedestrians. Histograms along the top and right provide further detail on the occurrences of specific width and height values.

Similarly, in ADAS, the labelled traffic light data can be used to provide drivers with real-time alerts about upcoming traffic signals, or changes in signal states, enhancing driver awareness and responsiveness. The inclusion of pedestrian signals, along with the ability to detect malfunctions (e.g., "Not working, or not visible") adds the potential for an extra layer of safety, ensuring that vehicles can handle a variety of complex traffic scenarios effectively.

2.4.3 Bounding Box Distribution

There are two design requirements that belong to this proposed dataset. Firstly, the dataset will be used to develop systems that will ultimately be deployed in the real-world. Secondly, the dataset is intended to be as transparent as possible to potential users. Given these requirements it is pertinent to provide a thorough understanding of the localisation, relative size and density of the objects of interest in the images. As such, this section will enlighten the reader by presenting the dimension and location of the bounding boxes within the images for the primary classes of interest: ["Vehicle", "Pedestrians", "Traffic Light"]. This information will also be used in later discussion when evaluating the proposed XAI methodologies.

Figure 2.20a and Fig. 2.20b illustrate the size distribution of bounding boxes for the 'Vehicles' and 'Pedestrians' classes respectively. For comparison, Fig. 2.3a and Fig. 2.4a show the bounding box sizes for 'car' and 'pedestrian' in the KITTI



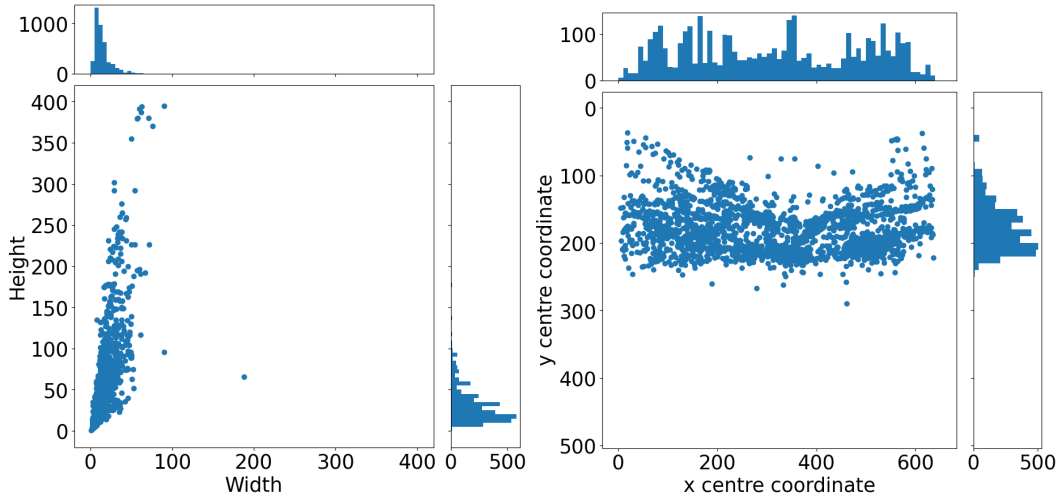
(a) Centroid locations for 'Vehicle' class.

(b) Centroid locations for 'Pedestrian' class.

Figure 2.21: Comparison of bounding box centroid locations for the 'Vehicle' and 'Pedestrian' classes in the proposed dataset. The left subplot shows the distribution of centroid locations for vehicles, while the right subplot shows the same for pedestrians. Histograms provide insight into the frequency of centroid locations along the width and height of the image.

dataset, providing a relevant autonomous driving perspective. Notably, both the data here, and the KITTI dataset, share a similar trend, characterised by relatively small object sizes compared to those in more generic object detection datasets, such as COCO. The diagonal lengths of a bounding box representing a vehicle have a mean value of 95 pixels, and a median of only 58 pixels, while pedestrian bounding boxes have a mean diagonal length of 49 pixels, and a median of 35 pixels. From the scatter plot in Fig. 2.20b, it is evident that bounding boxes for pedestrians tend to be taller than they are wide, which aligns with the data observed in the KITTI dataset. Conversely, Fig. 2.20a reveals that bounding boxes for vehicles exhibit a broader range of aspect ratios, reflecting a trend consistent with autonomous driving datasets.

Figure 2.21a and Fig. 2.21b present scatter plots showing the centroid locations of objects within the image frame. The accompanying histograms on the right side of these plots indicate that the majority of objects are situated midway down the images. This distribution aligns with expectations given that the data was captured using a front-facing camera. Most objects, such as vehicles and pedestrians, naturally appear on or alongside the roadway, rather than in the upper part of the image, which represents the sky, or the lower part, where objects would be too close to the capture vehicle to be fully visible. This pattern is consistent with typical autonomous driving data, emphasising the focus on objects within a practical field



(a) Bounding box sizes.

(b) Bounding box centroid locations.

Figure 2.22: This figure shows the bounding box statistics for instances of Traffic Light in the proposed dataset. The left subplot shows the width and height distribution of the bounding boxes in pixels. The right subplot shows the centroid location distribution of bounding boxes. Histograms in both subplots provide additional insight into the frequency of widths, heights, and locations.

of view for navigation and safety.

The top histogram in Fig. 2.21a shows that most vehicles are located between 250 and 500 pixels from the left side of the image frame, forming an 'M' shape with a dip near the centre. This pattern reflects the real-world environment: the left peak represents vehicles directly in front of the capture vehicle or parked on the left side of the street, while the right peak corresponds to vehicles in the oncoming lane or parked on the right side. This distribution is similar to that observed in Fig. 2.3b, except for a notable horizontal shift to the right in the proposed dataset. This shift is attributed to the data collection taking place in a country where driving on the left side of the road is standard, whereas the KITTI dataset was collected in a region where driving is on the right.

A similar pattern is evident in Fig. 2.21b, which shows the centroid locations of pedestrians in the proposed dataset. Although the overall distribution mirrors that of vehicles, it is broader, with large clusters of pedestrian around 200 and 475 pixels from the left side of the image frame. This wider spread can be attributed to the frequent presence of pedestrians on pavements, rather than in roadways, aligning with the typical environmental context of urban and suburban driving scenes.

In addition to the classes that have already been discussed, Fig. 2.22 presents the bounding box statistics for the 'Traffic Light' class. While there are publicly available datasets that include this class, it is notably absent from the KITTI dataset. The

	Value
Mean	9.674361
Standard Deviation	5.139993
Lower Quartile	6
Median	9
Upper Quartile	12
Max	33

Table 2.2: Descriptive statistics for number of labelled objects per image.

inclusion of 'Traffic Light' in the proposed dataset provides valuable information for tasks related to autonomous driving. The scatter plot in Fig. 2.22b shows that traffic lights are almost exclusively located in the top portion of the image frame. This distribution aligns with real-world positioning, as traffic lights are typically mounted on poles or overhead structures.

Table 2.2 summarises the object distribution statistics within the proposed dataset. Notably, the table highlights that, on average, there are 9 to 10 unique objects per image. This density of objects is significant for training object detection models, as it ensures diverse and complex scenes that mimic real-world driving environments. Furthermore, many images contain over 20 objects, presenting additional challenges and opportunities for models to detect and differentiate between multiple items within crowded scenarios. Such variety is crucial for developing robust detection algorithms that can handle real-world complexity and clutter.

While these characteristics result in a more challenging dataset, it also aligns with the operational needs of drones in urban environments, where detecting distant and densely positioned objects is essential. These challenging elements are not the primary focus of large-scale, generic image datasets, like COCO. The routes shown in Fig. 2.15 were carefully selected with the intention of capturing complex and realistic scenes. These characteristics were confirmed to be present by the investigation provided in this section. The proposed dataset offers an invaluable resource for developing and testing detection models tailored to the specific demands of autonomous vehicle and drone operations.

2.5 Deep Object Detection Evaluation

In this section an evaluation of the object detection capabilities of several state-of-the-art object detectors was conducted using the proposed dataset. This included YOLOv5[72], YOLOv8n and YOLOv8s[155], and Faster R-CNN[31]. Given that the intent is to develop deep learning for onboard intelligence, the choices were based on ease of embeddability. For the YOLO models, the same training pipelines were utilised that were proposed by the developers, including their tile data augmentation approach[72][155]. On the other hand, the training pipeline for the Faster R-CNN model used in this study is simpler - the only data augmentation that was applied was random translation and colour jitter.

It is possible, using the proposed ADLs, to reconfigure the classes to more options than just the base classes. Hence, in this section it is opted to use the following class labels for evaluating object detection: ["Vehicle", "Pedestrian", "Motorbike", "Cyclist" and "Traffic Light"]. It is important to note that there are only limited instances of "Motorbike" and "Cyclist" in this set, only 858 and 338 respectively. Whereas, the occurrences of all other classes number in the thousands.

For an added dimension, only images from the summer collection are utilised for training, while the validation set consisted of the winter collection, providing a robust basis for evaluation across varied seasonal conditions. Table 2.3 reports the typical object detection performance metrics for all the deep networks used in this study, including the top F1 score, precision, recall, mAP@50, and mAP.

Given the large class imbalances, the mAP for the individual classes is reported in Table 2.4. YOLOv5 generally performs better across most categories, especially in overall performance, car detection, pedestrian detection, and traffic light detection. Nonetheless, both of the YOLOv8 models show competitive results, achieving the highest mAP when considering all labels combined, and also the highest for classes of "Vehicle", "Pedestrian" and "Motor". The class that all networks generally perform poorest on is "cycle", this is not surprising, given the aforementioned class imbalance.

2.6 Wrapping Game Analysis

In this section, the Wrapping Game is introduced as a means to evaluate the discriminatory capabilities of the object detection explainers proposed in this thesis. Here the early usage of the Wrapping Game analysis is presented. The Wrapping

Model	$F1$	$Precision$	$Recall$	$mAP@50$	$mAP@(0.50-0.95)$
YOLOv5s	0.317	0.698	0.581	0.601	0.359
YOLOv8n	0.265	0.598	0.476	0.529	0.330
YOLOv8s	0.265	0.670	0.536	0.586	0.379
Faster R-CNN	0.297	0.623	0.567	0.617	0.395

Table 2.3: COCO Performance metrics on the Winter set, from deep object detectors trained on the summer dataset.

Model	mAP_{veh}	mAP_{ped}	mAP_{motor}	mAP_{cycle}	mAP_{TrL}
YOLOv5s	0.707	0.441	0.321	0.062	0.261
YOLOv8n	0.687	0.368	0.388	0.025	0.184
YOLOv8s	0.723	0.448	0.368	0.025	0.184
Faster R-CNN	0.752	0.511	0.326	0.000	0.387

Table 2.4: This table presents the mAP calculated for the range of IoU between $0.50-0.95$ on each class used in this evaluation. mAP_{TrL} refers to average precision on the class of 'Traffic Light'.

Game draws inspiration from the Pointing Game proposed by Zhang et al.[115]. A demonstration of this pointing analysis is available in Chapter 4, where it is used to evaluate the proposed KernelSHAP based framework.

Zhang et al. intended to utilise their proposed evaluation procedure as a tool for examining the discriminatory effectiveness of explainers in image deep network classifiers. In their framework, the most prominent pixel, identified as the highest contributor from a given attention map, is extracted. Subsequently, it is assessed whether this pixel resides within a human-annotated groundtruth represented by a 2D bounding box. A 'hit' is recorded if the pixel is contained within the groundtruth; otherwise, it is classified as a 'miss'. The overall score is calculated by dividing the number of 'hits' by the total of 'hits' and 'misses'. The Pointing Game inherently lacks the necessity for a comprehensive explanation to elucidate the complete context of the object in question. Consequently, it can only assess discrimination capabilities to a somewhat restricted extent. In the proposed assessment methodology a mask is derived from the explanation map, which can then be compared to an instance segmentation groundtruth using IoU; this will yield a more definitive evaluation of the explainer's discriminativeness.

The human-annotated groundtruth is used to guide both the Pointing Game

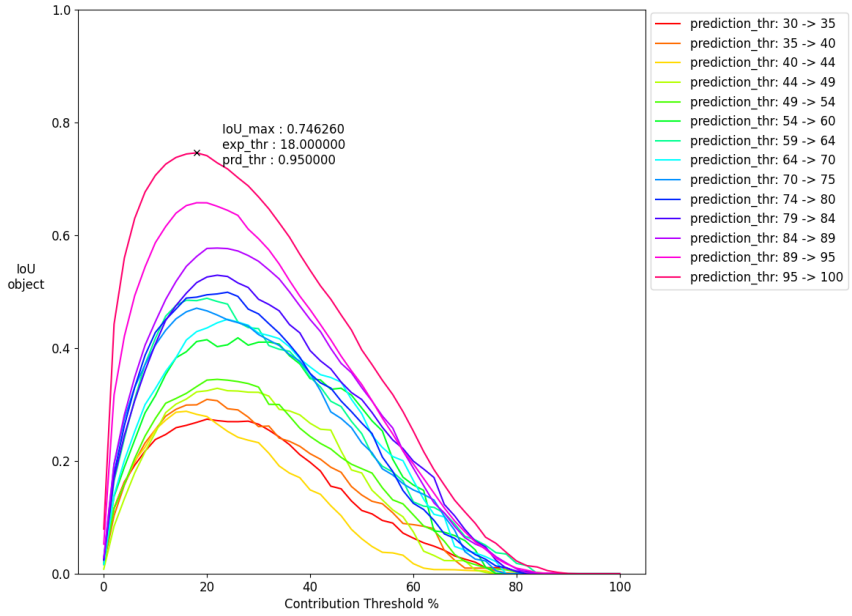


Figure 2.23: These plots show the outcome of the Wrapping Game analysis on the DetDSHAP explainer developed in this study.

and Wrapping Game analyses. Reliance on groundtruth to guide the methodology can introduce biases stemming from the performance of the deep network, which is not ideal. Although Zhang et al. [115] did not address this issue, this study explores instance segmentation performance by examining various ranges of model confidence. A prediction threshold is established by selecting a value between two specified bounds, and the Wrapping Game is applied to all instances where model confidence falls within this threshold. This approach provides a more comprehensive understanding of the explainer's discriminatory power, while also considering the impact of the deep network detector model's performance on the explanations generated by the proposed method.

The mask is created from the attention map in two steps. Firstly, the contribution of each pixel of the input image is generated. Secondly, Sklansky's Algorithm [156] - defined in Algorithm 1 - is then applied to all the pixels with contributions above a given threshold to yield a convex hull. In this algorithm, V_i is a vertex with coordinates (x_i, y_i). The algorithm identifies the extremal vertices (leftmost, rightmost, top, and bottom points) and sorts the vertices in a counter-clockwise order, starting from the leftmost vertex. It then iteratively constructs the convex hull by ensuring that the last three points on the boundary maintain convexity, and by removing any non-convex points as needed. This process results in a minimal

	DetDSHAP
Model confidence above 0.95	0.746
Model conf $\epsilon[0.30, 0.35)$	0.274
All instances	0.579

Table 2.5: This table shows the peak average IoU scores for three different scenarios in the Wrapping Game analysis. The top row shows the score for instances where the model's confidence is above 95%. The second row shows the score for instances where the model's confidences are between 30% and 35%. The final row shows the score for all instances with model confidence over 30%.

enclosing convex polygon around the relevant pixels.

The threshold used to select the pixels for Sklansky's Algorithm is given as a percentage of the most pertinent pixel, and is labelled as *exp_thr* in the plots. In the analysis shown in Fig. 2.23, the average IoU changes as this threshold value is increased. The IoU score increases, until it reaches a peak as the mask approaches the ideal shape, whereupon it will then begin decreasing. The final metric is taken as the peak IoU value.

Algorithm 1 Sklansky's Convex Hull Algorithm

Input: A set of vertices $P = \{V_1, V_2, \ldots, V_m\}$ representing a simple polygon.
Output: The convex hull $CH(P)$.
Identify the extremal vertices: leftmost, rightmost, top, and bottom points.
Sort vertices in a counter-clockwise order, starting from the leftmost vertex.
Initialise an empty stack H to store the convex hull vertices.
for each vertex V_i in the sorted list **do**
    while H contains at least 2 vertices and the last 3 points in H do not form a convex angle **do**
        Remove the second-to-last vertex from H.
    end while
    Push V_i onto H.
end for
return H as the set of vertices forming the convex hull $CH(P)$.

The evaluation procedure is intended to be utilised with the proposed dataset, although the procedure was developed prior to the proposed dataset being completed. At this point, the semantic labels were not yet available in the proprietary dataset. Hence, the instance segmentation set from the Kitti dataset[157] is utilised in the evaluation as the semantic labels. Moreover, The analysis was applied with a YOLOv5s model trained to detect the single class of 'Vehicle' on a subset of the proprietary self-driving dataset.

The Wrapping Game analysis was conducted on the proprietary DetDSHAP that will be presented in chapter 5. The outcome from this analysis is shown in Fig. 2.23. In addition to plotting the entire analysis, presented in Table 2.5, are pertinent scores for three scenarios investigated in the Wrapping Game analysis. The first two rows show the peak IoU when considering only instances with very high or very low model confidence. Also reported is the peak IoU score for all instances in the set across the possible prediction thresholds. The DetDSHAP explainer achieves a reasonable score in this latter scenario, indicating that it performs well for the vast majority of instances.

The examination detailed in Fig. 2.23 and Table 2.5 reveals a distinct relationship between the performance of the model and the IoU score. It is apparent that the explainer demonstrates enhanced scores in contexts where the model displays elevated confidence levels. This finding is consistent with the premise that a model's confidence is intrinsically linked to its ability to differentiate a particular instance from its surrounding environment. However, this pattern indicates that the explainer may struggle to clarify the causal elements influencing predictions when the model operates at lower confidence levels.

As stated, the Wrapping Game will be used to evaluate other explainers that were developed in this work, and so will be discussed further in later chapters. In these later chapters it was found that the shape of the plot also revealed qualitative information about the XAI method being evaluated. For example, a broader curve is more desirable than a narrow peak as it indicates a more detailed explanation. Narrow peaks might indicate a certain feature in the image which is the principal attribute behind a particular decision. In object detection however, this instead tends to be the object's centre, as this is important for the placement of the bounding box. This is a known characteristic of object detectors, particularly YOLO-based detectors, and thus not strictly pertinent information to provide in the explanation.

2.7 Summary

This chapter serves as a review of the datasets that currently exist in the literature which are used for developing deep object detection algorithms for autonomous and unmanned vehicles, and others that are used for evaluating XAI in other domains. Some of these are utilised throughout this thesis. However, it was determined that a

new dataset was needed to fulfill the requirements of this work. As such, this chapter also serves as an introduction to this new dataset, and the tools that were developed to collect and label the data. Finally, the Wrapping Game was introduced as a method to evaluate the discriminative ability of deep object detection explainers. The Wrapping Game analysis will be discussed further in subsequent chapters where it will be used, as intended, with the full proprietary dataset introduced earlier in this chapter.

Chapter 3

Grad-CAM Based Explainers for Object Detection from Drone Platforms

This chapter presents two key contributions aimed at enhancing object detection algorithms for drone-based platforms. The first is a Tile Dataloader designed to improve the performance of object detectors on aerial drone imagery, particularly in cluttered outdoor scenes commonly encountered by drones. The second is a Grad-CAM-based Explainer, which generates saliency maps to highlight image regions that contribute to a deep detector's class scores, thereby improving the transparency of the detection process. Both works are evaluated for their real-time capabilities, with a focus on improving detection performance and interpretability in dynamic environments.

3.1 Introduction

As first identified in section 2.2.2, Autonomous drone-based platforms encounter unique challenges in object detection. Aerial images often feature small, densely packed objects and complex, cluttered backgrounds that differ significantly from the datasets typically used for training detection models. As discussed in Chapter 1, DNNs are inherently opaque, thereby raising critical issues of trust and interpretability in safety-critical applications. Building on that background, this chapter presents two key innovations specifically tailored for drone imagery.

Firstly, a Tile Dataloader is introduced, designed to enhance the detection of small objects. By partitioning high-resolution aerial images into smaller, manageable tiles without downscaling, the proposed method preserves crucial object details—thereby overcoming the limitations of conventional scaling approaches and boosting detection performance.

Secondly, an adaptation of Gradient-weighted Class Activation Mapping (Grad-CAM) for YOLOv5 is proposed. This novel approach generates near real-time saliency maps that reveal the regions of an image most influential in the network's detection decisions. By focusing the explanation on specific detection layers and incorporating object-specific constraints, the adapted Grad-CAM not only facilitates debugging but also enhances trust in the DNN's decision-making process.

As noted in Chapter 1, in their tutorial on Gradient-Based explainers, Nielson et al. [81] found that many state-of-the-art gradient-based techniques were prone to failure due to class agnostic behaviour—a phenomenon in which the generated maps show little difference across different class labels for the same image. Grad-CAM was selected for its efficiency in generating near real-time saliency maps, as it requires only a single forward and backward pass, and notably, it was the only method in their study that did not exhibit class insensitivity.

Notably, YOLOv5 was chosen as the detection framework for this study because, at the time the work was conducted, it represented the most state-of-the-art solution in real-time object detection. Although more recent iterations have since been developed, YOLOv5 offered a competative balance between detection accuracy and computational efficiency required for UAV platforms.

Subsequent sections detail the design and implementation of both the Tile Dataloader and the Grad-CAM-based explainer. The particular challenges presented by drone-captured images are outlined, followed by a description of the methodology—including the modifications necessary to address the multiple detection layers of YOLOv5. The proposed approaches are evaluated on the VisDrone dataset, with both quantitative metrics and qualitative analyses demonstrating improvements in detection performance and interpretability.

By addressing the limitations posed by small objects in drone-captured imagery, the novel Tile Dataloader and adapted Grad-CAM framework provide a robust solution for achieving real-time, transparent object detection on UAV platforms. The methodologies and evaluations presented in this chapter set the stage for subsequent

sections, where further refinements and deeper analyses of network reasoning and detection performance will be explored.

3.2 Methodology

In this chapter two pieces of work are proposed, the first is the Tile Dataloader, and the second is the suggested use of Grad-CAM to explore and understand the behaviour of YOLOv5. As such, the section is split into three parts: The first of these will cover the Tile Dataloader. The second and third will discuss two methodologies for using Grad-CAM with YOLOv5. The first of these methodologies demonstrates how to deploy Grad-CAM in much the same way it was intended, which is to produce a map showing the class activations. The second methodology was developed later using the proprietary dataset proposed in chapter 2, and uses the predicted box to constrain the Grad-CAM to just focus on a specific detection.

3.2.1 Tile Dataloader for Small Object Detection

The Tile Dataloader was designed to enhance small object detection by splitting large aerial images into smaller, high-resolution tiles. This ensures that small objects retain their detail, avoiding detection failures caused by downscaling during processing. This method ensures that the network receives input images with improved object visibility, thereby boosting performance in scenarios involving dense or distant objects. The tiling process is automated, ensuring adaptability to different image sizes.

Here, the tile size is defined as $t_s \times t_s$ pixels, with $t_s = 512$ in this work. The subdivision calculation is performed as follows: given an input image of size $H \times W$, the number of tiles along each dimension is computed using Eq. (3.1) and Eq. (3.2), ensuring at least a 20% overlap between adjacent tiles to prevent objects from being cut at tile boundaries. In addition to tiling, a rescaled version of the entire image $(t_s \times t_s)$ is processed to preserve global context.

$$N_H = \left\lceil \frac{H}{t_s - 0.2 \times t_s} \right\rceil \tag{3.1}$$

$$N_W = \left\lceil \frac{W}{t_s - 0.2 \times t_s} \right\rceil \tag{3.2}$$

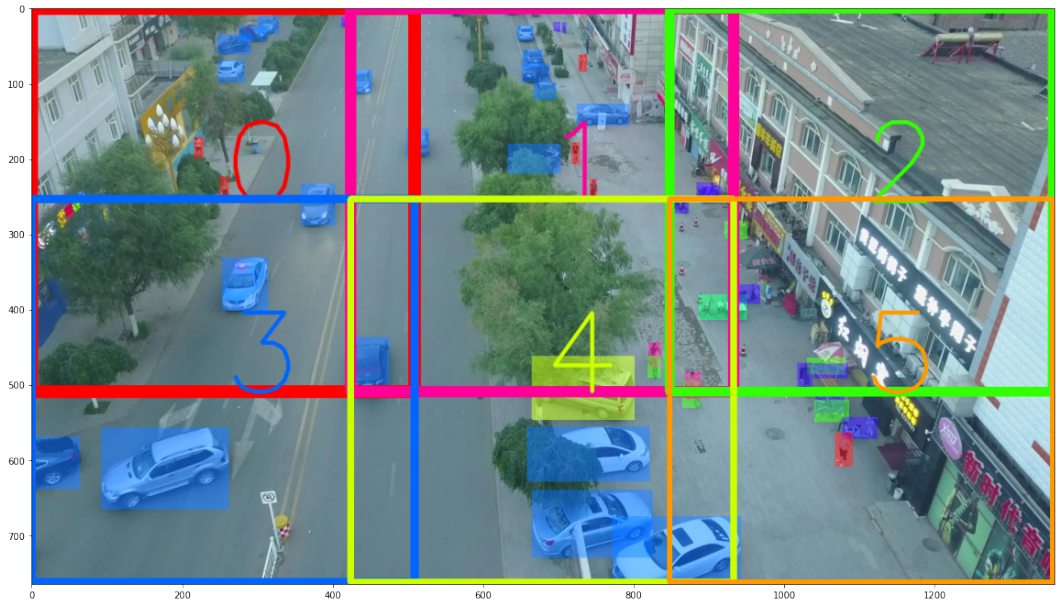


Figure 3.1: Illustration of an image with the tiles overlaid, numbered 0 to 5. The groundtruth bounding boxes are shown as solid, slightly transparent, rectangles.

An example of an image split into tiles is illustrated in Fig. 3.2, where a 1360×765 image is split into six 512×512 tiles. During inference, a batch containing all tiles and a resized copy of the full image is simultaneously passed to the DNN. Tile offsets from the top-left corner of the image are applied to reconstruct bounding box coordinates, and non-maximum suppression (NMS) is used to remove duplicates.

The Tile Dataloader was evaluated with both Yolov5 and CentreNet. Both models were retrained from scratch on images from the VisDrone 2020 training set. The base dataset was enlarged by splitting the raw images into tiles of $t_s \times t_s$ pixels using the method described above. K-means clustering was used to select 30 prior boxes from the new training set for the YOLO algorithm.

The Tile Dataloader was evaluated with both Yolov5 and CentreNet. Both models were retrained from scratch on images from the VisDrone 2020 training set. The base dataset was enlarged by splitting the raw images into tiles of $512by512$ using the method described above. K-means clustering was used to select 30 prior boxes from the new training set for the YOLO algorithm.

3.2.2 Grad-CAM with YOLOv5

An overview of the proposed explainer framework can be seen in Fig. 3.3. Normally, when using Grad-CAM in a classification network, the gradient is set to the pre-SoftMax layer as a one-hot encoding of the target class, and then backpropagated

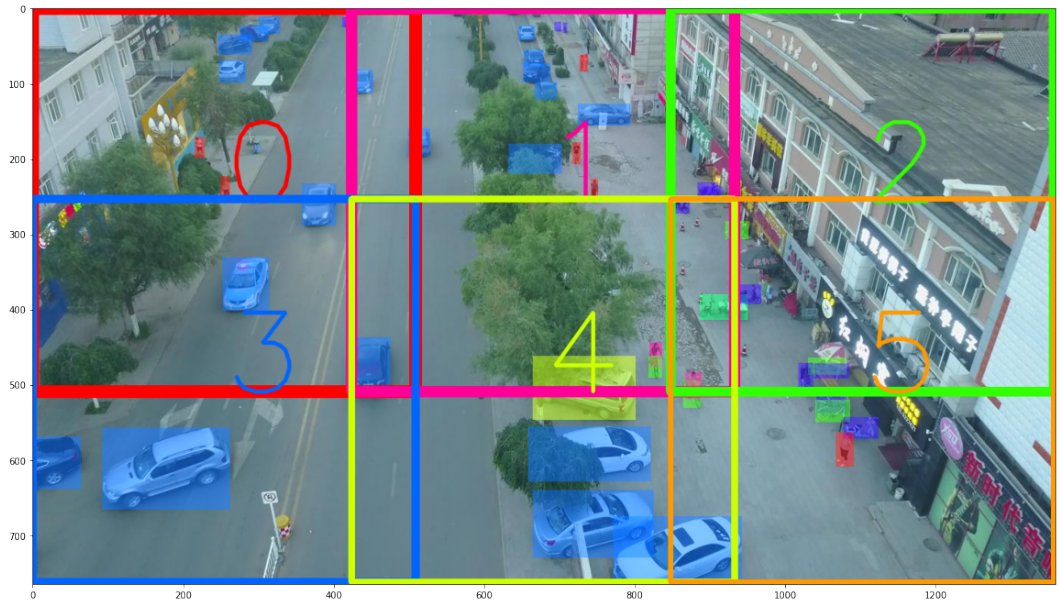


Figure 3.2: Illustration of an image with the tiles overlaid, numbered 0 to 5. The groundtruth bounding boxes are shown as solid, slightly transparent, rectangles.

until the last convolutional layer. However, more attributes are available from detection networks, and therefore a different approach is required. Firstly, the detections with low objectness scores are suppressed, followed by all the bounding box features and class scores, apart from the class for which the saliency map is being generated. As a result, what remains is a single vector containing the scores for the target class that pertain to valid objects in the image. This method was developed experimentally by suppressing different parts of the output with different rules, and observing the resulting saliency maps.

The original implementation of Grad-CAM focuses on using the final convolutional layer in the network to generate the saliency map. In their findings, the authors claimed that the best-looking visualisations are often obtained at the deepest convolutional layer, and the localisation gets progressively worse at the shallower layers. However, YOLOv5 has three detection output layers that make detections at different scales. As such, in the proposed implementation, three different convolution layers are utilised, each one proceeding a detection layer.

The generated saliency maps from the proposed method can be observed in Fig. 3.4, where the images were created when overlaying the saliency maps for 'car' over the original image. Figure 3.4a shows the car bounding boxes produced by the first detection layer, and it can be observed that at this layer the very distant cars have been detected. In contrast, it can be seen in Fig. 3.4b that some of the distant

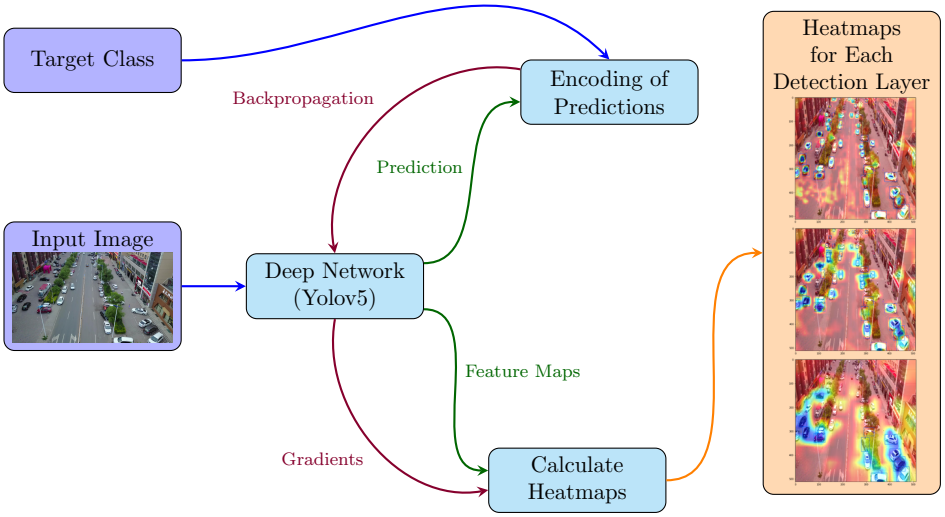


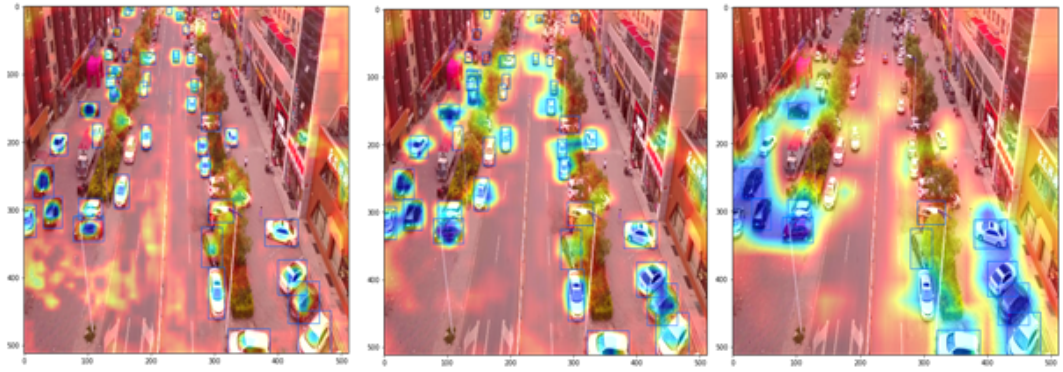
Figure 3.3: Illustration of the proposed Grad-CAM based algorithm to generate heatmaps from a Yolov5 model. Step 1: The input image and target class are selected a priori. Step 2: Network conducts a forward pass on the input image to produce predicted boxes, at the same time feature maps from each detection layer are stored. Step 3: Backpropagation occurs from the encoded predictions and Gradients are calculated. Step 4: Heatmaps are generated for each detection layer from the stored feature maps and calculated gradients.

cars are no longer detected. Furthermore, Fig. 3.4c, reveals that very few of the distant cars are detected by the final detection layer, indeed only one car is detected in the top third of the image. Each saliency map can indicate to the explainee where in the image the algorithm is paying attention, even when no bounding boxes are present.

More information can be extracted from the network by generating a saliency map that can highlight pixels pertinent to the objectness score. To do so, the bounding box features and all the class scores were suppressed, leaving a vector containing the objectness scores. Figure 3.5 shows an instance where this is useful, where it suggests that the reason the trucks along the top and bottom were not detected is due to the network believing them to be background, as indicated by Fig. 3.5b.

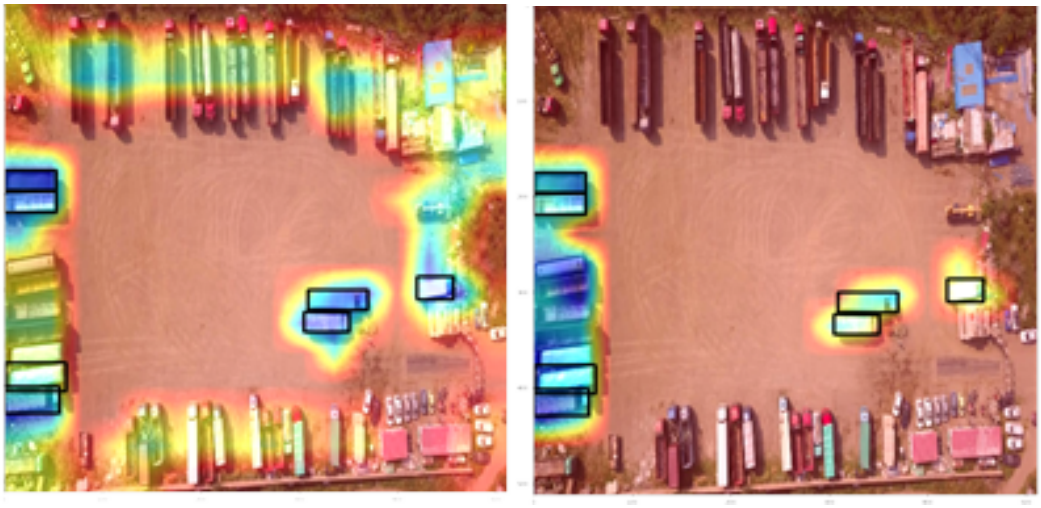
3.2.3 Detection Constrained Grad-CAM with YOLOv5

In the previous subsection, the approach taken was constrained to analysing the class and objectness scores predicted by the network. This resulted in feature maps that are not necessarily relevant to any object in particular. In this section, the motivation for this was to investigate whether Grad-CAM could be used to provide



(a) Shallowest Detection Layer **(b)** Middle Detection Layer **(c)** Deepest Detection Layer

Figure 3.4: Saliency maps generated from the convolution layers prior to each detection layer. The explicant in this case was YOLOv5l trained on the VisDrone dataset. The target class was 'car'.



(a) Class Score **(b)** Objectness Score

Figure 3.5: Saliency maps for the 'Truck' class score, and objectness score for the final detection layer. Seven trucks were detected on this layer, these have been identified by the black bounding boxes.

additional context behind predicted boxes made by a YOLOv5 model trained on the XAI-AV dataset (introduced in Chapter 2). The dataset change is necessary to enable the use of the Wrapping Game analysis, which offers a qualitative evaluation of the proposed explainer.

A modification was required to the prediction encoder in order to restrain the Grad-CAM calculation to a single predicted box. This was achieved by suppressing boxes that had a poor IoU with the target box of interest. The output from each detection layer is treated independently, this is the same as the approach outlined in the previous section. In addition, a saliency map is generated for each individual object. These can be analysed independently, such as the examples shown in Fig. 3.6.



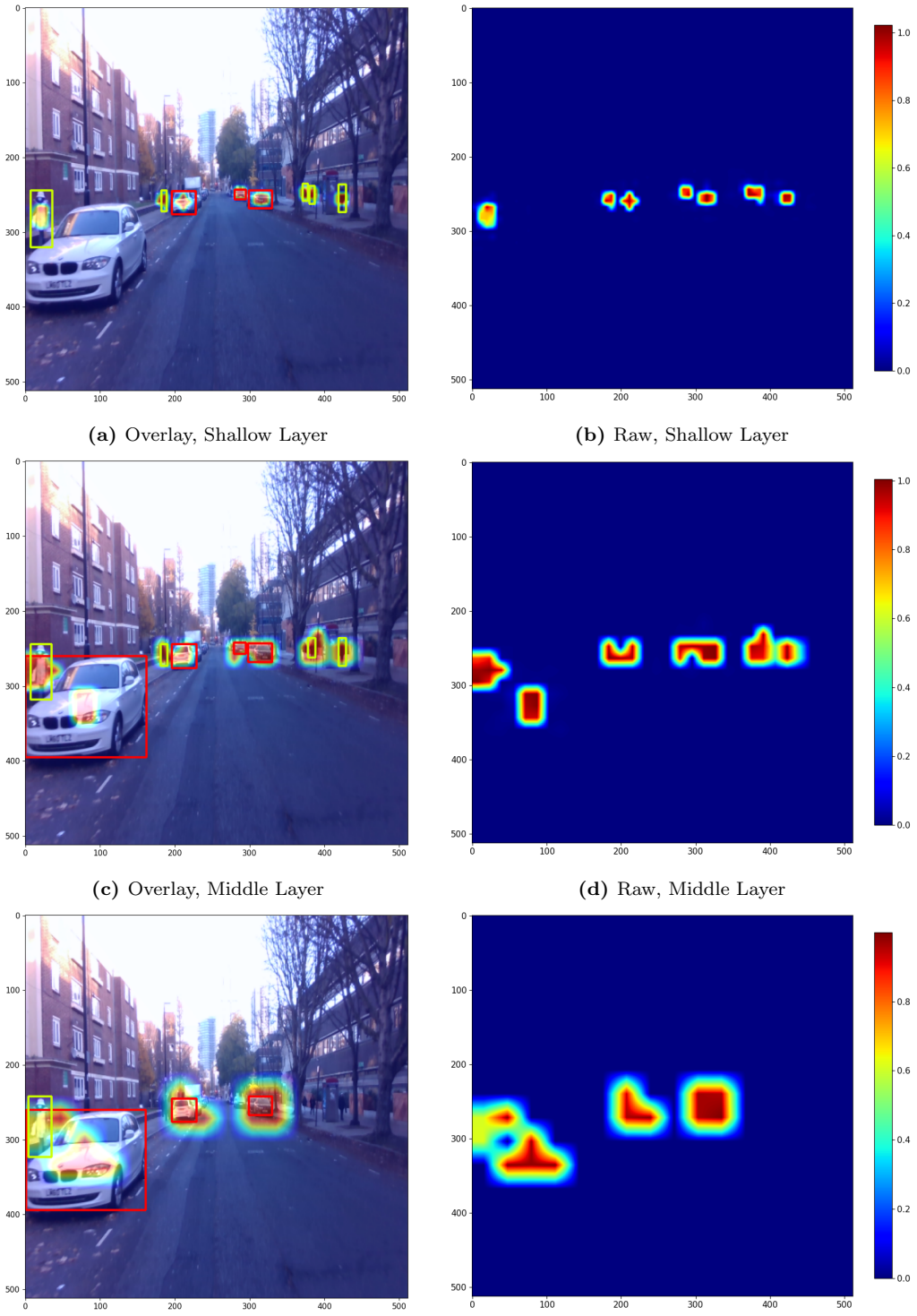
Figure 3.6: This figure illustrates the individual saliency maps that can be generated with the bounding box-constrained Grad-CAM. The maps for the vehicles are shown along the bottom, and the maps for the Pedestrians are shown along the right side.

Alternatively, the feature maps may be combined for each detection layer, as seen in Fig. 3.7. This is achieved by normalising each map and then summing them together. This creates a map that shows which pixels were pertinent to the network's reasoning specifically related to the bounding box location and dimensions.

3.3 Aerial Object Detection Results

3.3.1 VisDrone Challenge Metrics

When this work took place, the VisDrone competition had already closed, so it was not possible to receive feedback on the performance of the network on the official test set. However, the organisers released a test-dev set that can be considered



(a) Overlay, Shallow Layer

(b) Raw, Shallow Layer

(c) Overlay, Middle Layer

(d) Raw, Middle Layer

(e) Overlay, Deep Layer

(f) Raw, Deep Layer

Figure 3.7: Detection restrained Saliency maps showing the attention of the deep network. These Saliency Maps are created by summing the maps for the individual objects. The right column displays raw maps from the shallow, middle, and deep detection layers, while the left column shows these maps overlaid on the input image.

Models	AP	AP50	AP75	AR1	AR10	AR100	AR500
DroneEye2020	34.57	58.21	35.74	0.28	1.92	6.93	52.37
YOLOv5 with Tile Loader	32.36	53.92	33.59	2.01	13.47	41.59	44.53
PG-YOLO	26.05	49.63	24.15	1.45	9.2	33.65	42.63
CenterNet with Tile Loader	14.62	29.45	1.35	1.83	8.83	23.03	27.6

Table 3.1: Table showing the VisDrone competition metrics achieved by the YOLOv5 and CenterNet algorithms using the proposed Tile Dataloader. Also shown are the scores achieved by DroneEye, which was the top-scoring network in precision, and PG-YOLO, the top-scoring YOLO network entered in the competition. The Average Precision (AP) is calculated over multiple Intersection over Union values of [0.50:0.05:0.95]. The Average Recall (AR) is calculated given a maximum of 1, 10, 100, and 500 detections per image.

representative. As such, this set was used to evaluate the performance of the model developed here. The competition metrics were based on average precision and recall at different thresholds of IoU. Precision is given by Eq. (3.3) which is the network's ability to correctly classify detected objects. Recall takes into account false positives produced by the network and is given by Eq. (3.4). A false positive is when the network mistakenly detects an object when there is only background.

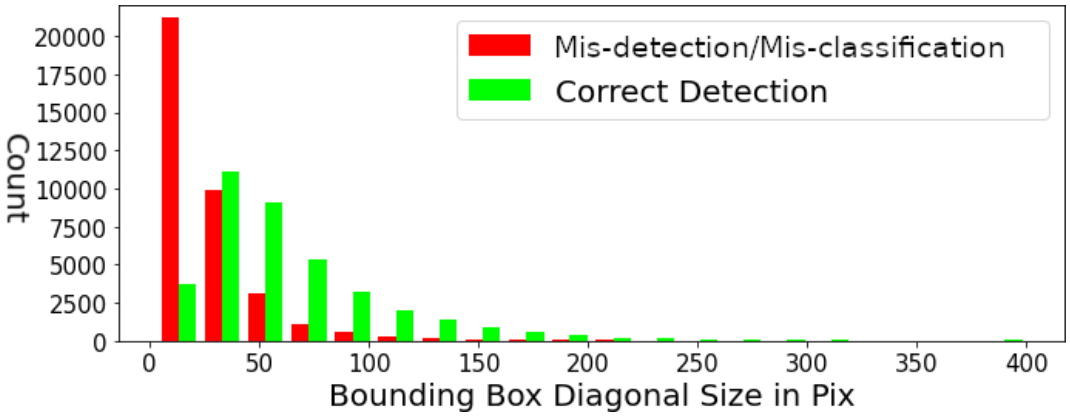
$$\text{precision} = \frac{\text{true positives}}{\text{true positives} + \text{false positives}} \tag{3.3}$$

$$\text{recall} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}} \tag{3.4}$$

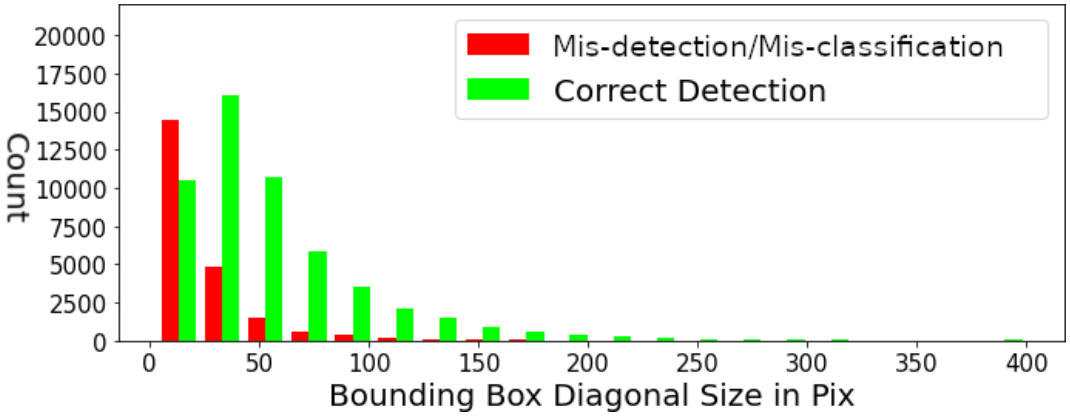
Guided by the performance of the YOLOv5 algorithm with the proposed Tile Loader on the test-dev set, it may have placed 5th, if it were possible to submit it to the VisDrone competition. Table 3.1 shows the results of the networks that were developed in this study, alongside two other networks which were entered into the competition. One can see that the YOLO network proposed here outperformed the YOLO network that was entered into the competition in all categories, and even achieved higher scores than the top scorer for precision in three of the recall metrics.

3.3.2 Tile Loader Appraisal

This section provides a comparison of the performance of YOLOv5 and CentreNet-ResNet50 with the Tile Loader, using the plain model as a control. The intention is to gauge the performance increase of each model when utilising the Tile Loader. Both algorithms were evaluated on the test-dev set from the VisDrone2020 detection



(a) Plain Model



(b) Model with Tile Loader

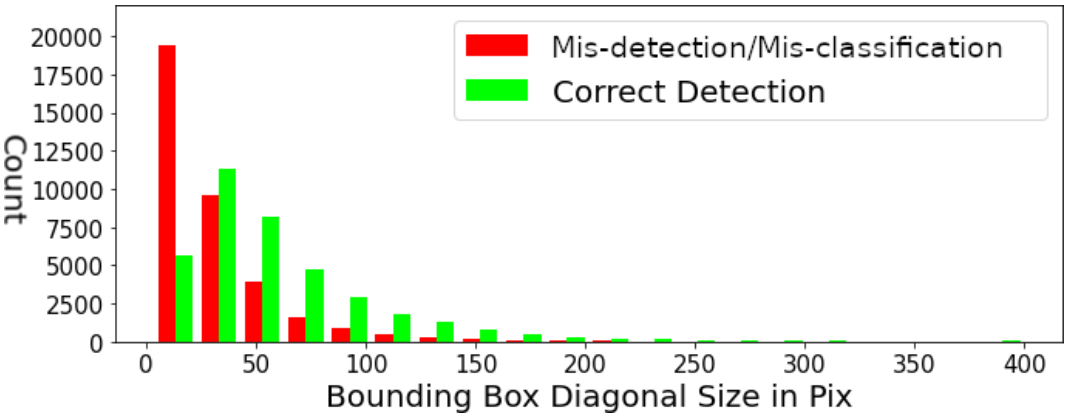
Figure 3.8: Plots to show the performance of YoloV5 at different object scales.

dataset.

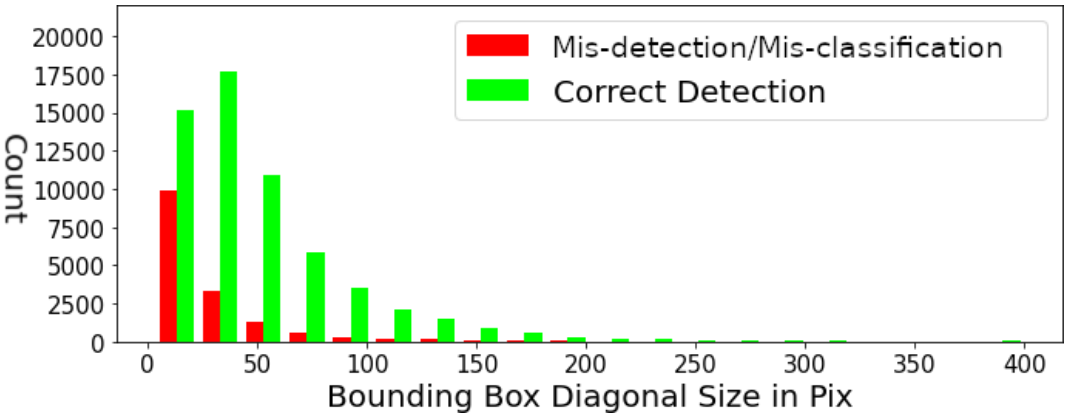
To validate the claim made here, that the Tile Loader improves the performance of small objects, the number of correct detections depending on the bounding box size is investigated. The histograms in Fig. 3.8 and Fig. 3.9 show how many boxes the model was able to correctly allocate when considering the diagonal length of the bounding box.

It was found that with the use of the Tile Loader, the networks were able to detect and correctly classify many more small objects. The plots in Fig. 3.8a and Fig. 3.9a show that when the networks were used alone, there were a substantial number of missed detections with smaller boxes. However, when using the Tile Loader, the performance on small objects is much improved, as can be seen by Fig. 3.8b and Fig. 3.9b. The most significant improvement is in the boxes smaller than 24 pixels across, where there is an improvement of 179% for YoloV5 and 168% for CenterNet.

Figure 3.10 shows the performance of YoloV5 without the Tile Loader across



(a) Plain Model



(b) Model with Tile Loader

Figure 3.9: Plots to show the performance of CenterNet-ResNet50 at different object scales.

all the classes. The classes with the highest missed detections are those that tend to contain relatively smaller objects, including 'pedestrian', 'people', bicycle' and 'motor'. Those with the least amount of missed detections include classes of four-wheeled vehicles including 'car', 'van' and 'bus'.

There is also a noted improvement in the performance across all classes when using the Tile Loader, as seen by the high scores across the diagonal in Fig. 3.11. On the other hand, these results also revealed an increased percentage of misclassifications, particularly members of the class 'van' being allocated as a member of 'car', and 'bicycle' being allocated as 'motor'. Therefore, despite the use of the Tile Loader leading to significant gains in terms of small object detection, it does not necessarily improve the classification performance of the model.

To better understand these classification errors, we analyze the confusion matrices in Fig. 3.10 and Fig. 3.11, which provide insight into the primary sources of misclassification. While the Tile Loader improves small object detection, it does not significantly enhance classification accuracy. Many misclassifications occur between

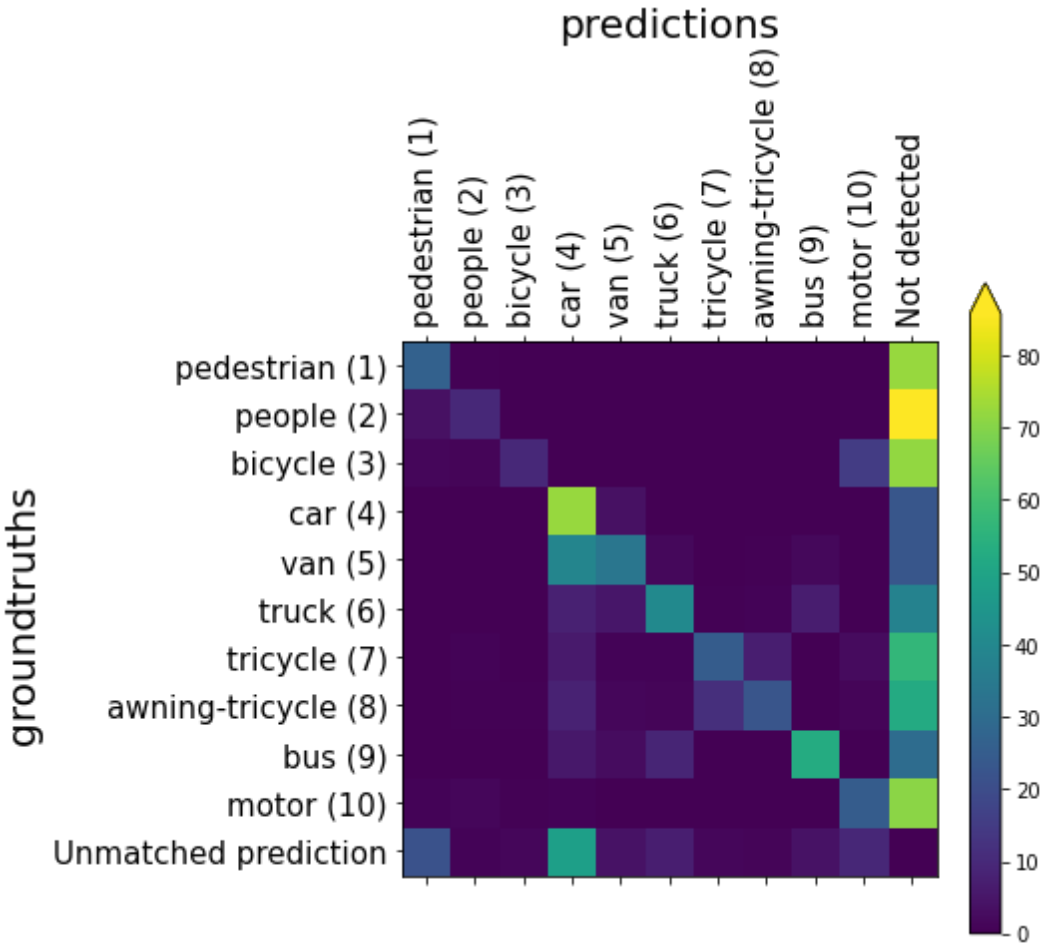


Figure 3.10: Confusion matrix illustrating the model's performance without the Tile Loader.

visually similar classes, such as 'van' being classified as 'car' and 'bicycle' as 'motor'. This suggests that the model relies heavily on global shape features—such as overall object contour and silhouette—rather than finer local details, which may lead to misclassification among visually similar categories.

Although the Tile Loader improves the detection rate of smaller objects, their misclassification rates remain high. This is likely due to the lack of detailed features in low-resolution instances, making it difficult for the model to distinguish between closely related categories. Additionally, occlusion and background clutter may introduce ambiguity, particularly in urban environments where objects frequently overlap. A potential future improvement is to incorporate the Tile Dataloader with a higher-resolution feature extractor to enhance fine-grained classification, such as a PFPN.

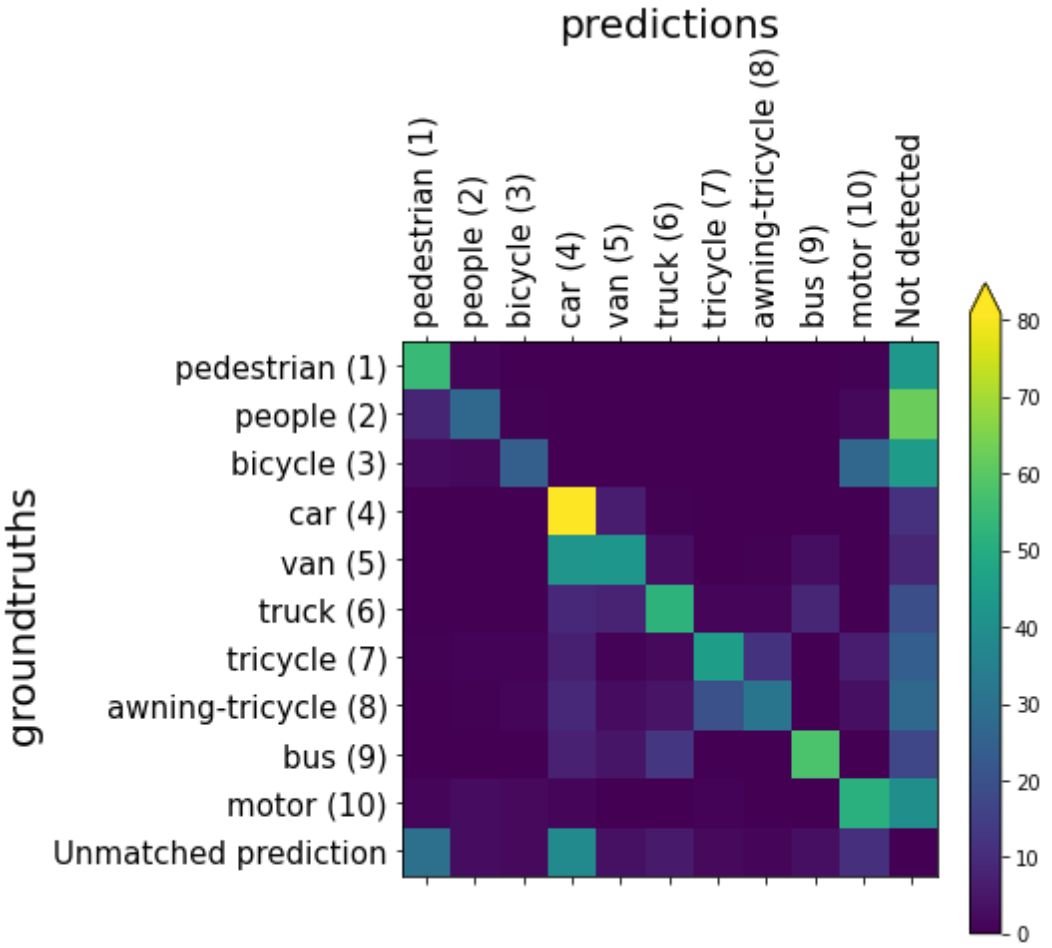


Figure 3.11: Confusion matrix illustrating the model's performance with the Tile loader

3.3.3 Timing Characteristics

The goal of this particular work was to be able to detect objects and provide explanations in real-time. As such, this section investigates the achievable Frames per Second (FPS) of the trained YOLOv5l on its own, as well as with the Tile Loader, and the achievable FPS when creating a saliency map. To do so, the time was measured for how long it took for each detection algorithm to process an image. The time was measured from immediately after the image had been opened, and existed in memory, to when the predicted bounding boxes were available. For Grad-CAM, the time measured was from when the image was loaded until the saliency map was available, this was recorded for all classes, and the objectness score. The timings were recorded for all samples in the VisDrone test-dev set.

The results are shown in Table 3.2, and it can be seen that using the Tile Loader has a significant handicap to the processing rate. The majority of the processing time is taken up by post-processing the larger number of potential bounding boxes.

	YOLOv5 with Tile Loader	YOLOv5	Grad-CAM
Mean	17.98	38.7	4.983
Median	19.83	38.8	4.972
Standard Deviation	4.05	0.3176	0.517

Table 3.2: FPS stats recorded for each process on the test-dev set.

This could also contribute to the high standard deviation, along with the fact that the images in the test-dev did not have a fixed dimension, and thus each sample may have a different number of tiles which the deep network needed to process. That being said, the FPS is still above 15 FPS, which is acceptable. This could be improved when embedded, and using a fixed image size.

The Grad-CAM was used in conjunction with the plain model. It can be seen that there is an even greater handicap to the processing rate as the average FPS is below five frames. Nonetheless, this is near real-time, and may also be improved with embedding, and further optimisation, to achieve a higher FPS which would be more user-friendly to an operator.

3.4 Validation of Grad-CAM Based Explainer

In validating the Grad-CAM-based explainer, a series of tests are employed to ensure the reliability and interpretability of the generated saliency maps. Section 3.4.1 presents the Model Dependency Test, which examines the sensitivity of the saliency method to the learned parameters of the model. This test involves comparing outputs from a trained model with those from a model whose parameters have been randomised. The goal is to confirm that the saliency outputs differ significantly, indicating that the explainer is dependent on the model's learned weights. A lack of sensitivity suggests that the explainer may be insufficient for model debugging and interpretation tasks.

Section 3.4.2 discusses the Data Dependency Test, inspired by the methodology outlined by Fong and Vedaldi[13]. This test evaluates whether the explainer accurately captures the true data-label relationship by adding noise to the input image using the saliency map as a guide. If the saliency map effectively highlights influential regions, the model's prediction should significantly diverge from the original

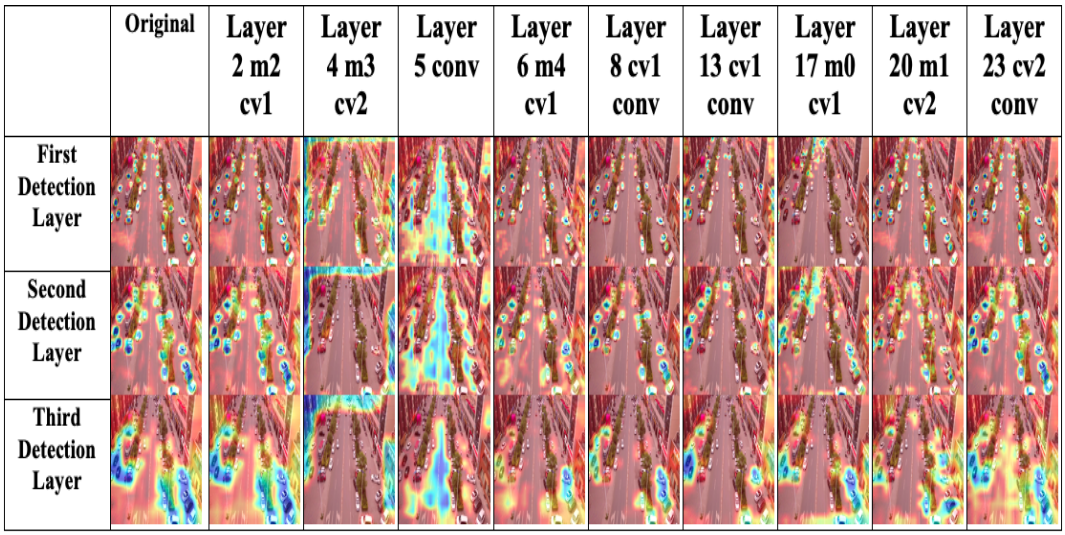


Figure 3.12: Table showing the Cascading randomisation on YOLOv5.

prediction when noise is applied to these areas. This test ensures that the explainer identifies parts of the input that are genuinely important for the model's predictions, confirming its relevance and robustness in representing data-label associations.

Section 3.4.3 utilises the Wrapping Game, previously introduced in this thesis, to evaluate the explainer's ability to discriminate the object of interest effectively. This method, while also revealing information about the sensitivity of the explainer to the model, focuses on how the model's confidence affects the explainer's discriminative power. This analysis highlights the explainer's capability to identify the most relevant image regions for the detected object, and its responsiveness to varying confidence levels.

3.4.1 Model Dependency Test

The Model Dependency Test, motivated by the work of Adebayo et al., was designed to assess the sensitivity of the Grad-CAM explainer to the learned parameters of the model and robustness to input dominance[110]. The procedure involved sequentially randomising the weights of each convolutional layer in the network, and generating corresponding saliency maps, which were then compared to those produced by the original, unmodified model using the same input data. This step-by-step approach highlighted how the explainer's output changes with parameter modifications, revealing its dependence on specific model features.

Figure 3.12 presents the results of this test, showcasing saliency maps for the 'car' class across the first, second, and third detection layers in the initial column. This

is followed by maps generated after progressively randomising the weights of deeper convolutional layers from left to right. It can be observed that the saliency maps shift noticeably when the weights of the first few layers are randomised, indicating that these layers have a significant influence on the visual explanations. However, randomising the weights of layers deeper than the first and second detection layers has a less pronounced effect on the generated saliency maps from these same layers. This outcome aligns with expectations, as the information does not propagate backwards to these layers, but only forward to detection layers further down the chain. Consequently, the explainer passes the test, confirming that it is sensitive to the model's learned weights, and capable of reflecting meaningful changes in the network's learned parameters.

3.4.2 Data Dependency Test

In order to evaluate the Grad-CAM explainer's ability to select important pixels, a perturbation-based approach was devised - motivated by [13]. The first step was to use the generated saliency map to perturb the image. This was done by overlaying the image with Gaussian noise, where the magnitude of the noise was related to the magnitude of importance from the saliency map. An example of this is shown in Fig. 3.13.

The next step was to run an inference on the perturbed image, and then attempt to match predicted boxes made on both the original and perturbed images. If the boxes had an IoU higher than 0.5 it was considered a match. For the matched boxes the change in the confidence score was recorded. The number of boxes that could no longer be matched was also recorded as this meant that the object was no longer considered valid by the detector, and can be considered a false negative.

This was done for all object classes on every image in the VisDrone test set. In the findings, it was discovered that around 38% of the boxes could not be matched. This is likely because the perturbation caused the network to predict the new boxes in different locations, or with different dimensions. In either case, this could result in a low IoU which would result in a mismatch. Therefore, key features that had allowed the network to make the original prediction were obscured by the perturbation.

Of the boxes that could be matched, the average confidence score by YOLOv5 dropped by 77.52% across the entire set. Fig. 3.15 shows the distribution of the drop in confidence across all the matched objects in the test set. The drop in confidence

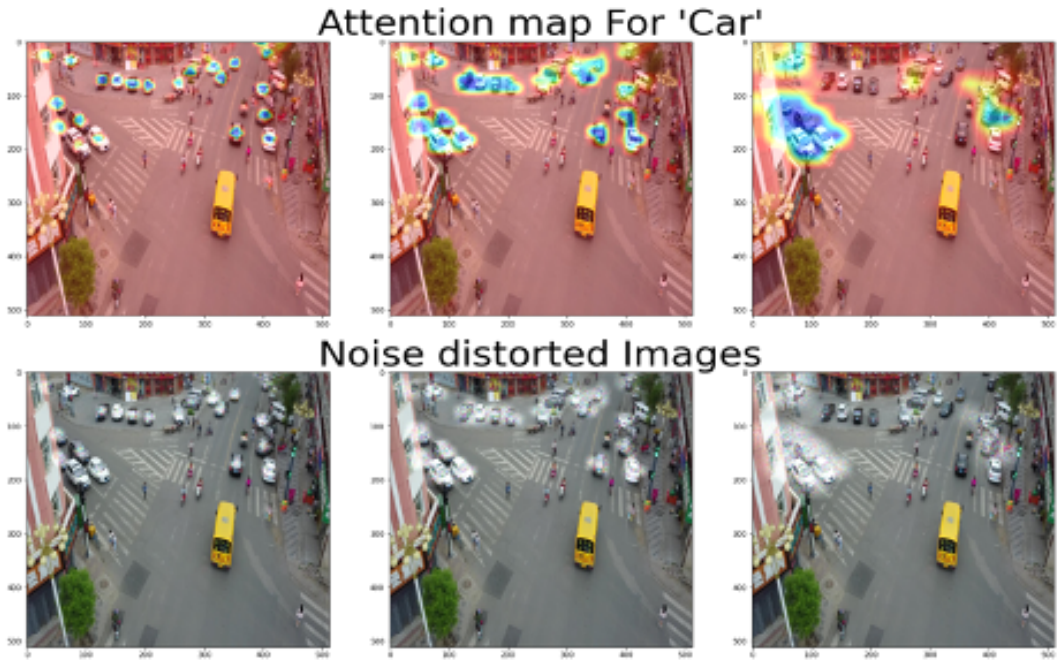


Figure 3.13: Top row: Grad-CAM feature maps for 'car' on the three detection layers. Bottom row: Images that were masked using the Grad-CAM maps to obscure important features.

also suggests that the perturbation which was applied obscured the features that assisted the model in its prediction.

3.4.3 Wrapping Game Analysis

In this section, the results of the Wrapping Game analysis on the bounding box-constrained Grad-CAM will be discussed. The Wrapping Game was previously introduced in Section 2.6. This is an evaluation methodology intended to validate XAI techniques that provide a saliency map for individual detections. Hence, in this section the analysis was not conducted on the unconstrained Grad-CAM. This is because the saliency maps produced by that methodology will show the contribution to the class scores, or objectness scores, of all the detected objects in the image, rather than one detection individually.

Comparisons will be drawn between the results in this section and the plots in section 2.6. It is important to note, that these other results were obtained with slightly different data, so these are not strictly methodologically appropriate. However, further discussion is made in subsequent chapters where the Wrapping Game is deployed with the same deep network and training data.

The entire Wrapping Game analysis was applied using both the summer set,

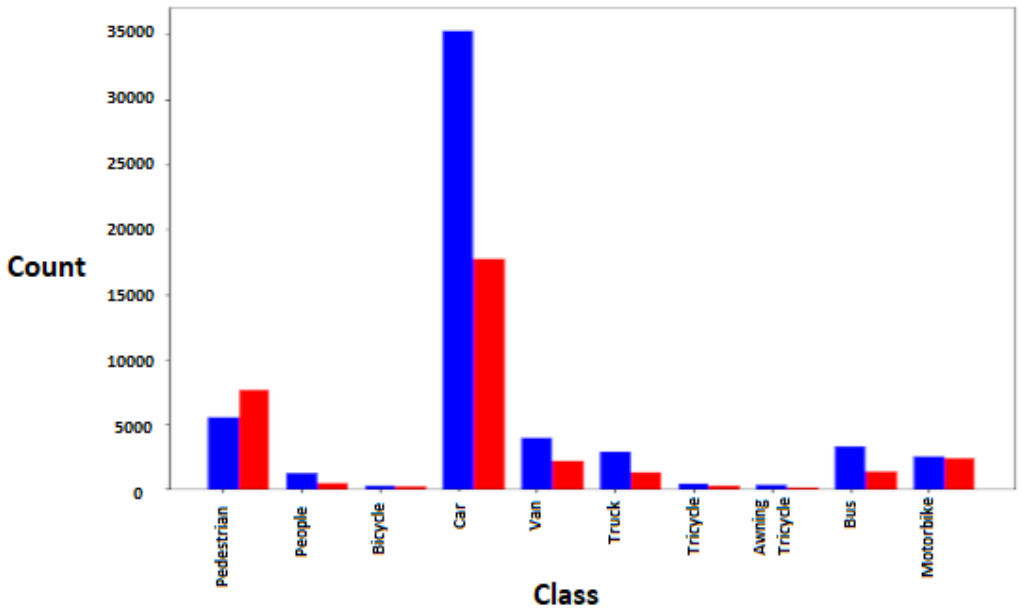


Figure 3.14: The Graph shows the number of matched boxes in blue, and the number of unmatched boxes in red, for each class after salience-weighted perturbation is applied.

shown in Fig. 3.16, and the winter set, shown in Fig. 3.17. The analysis is split into two settings. For the first setting, the Wrapping Game analysis is applied to all objects, regardless of the network's confidence score in the detected objects. Whereas, for the second setting, the Wrapping Game analysis is applied while also considering the network's confidence. The second setting is shown in Fig. 3.16b and Fig. 3.17b, in these plots the different coloured lines represent a different threshold of the prediction confidence.

In all the figures, apart from Fig. 3.16a, there is an abnormal spike early in the analysis. This is not observed when applying the same analysis with the other explainers presented in this thesis. From observing the plots in Fig. 3.16b and Fig. 3.17b, it can be seen that the spikes in the IoU scores are attributed to the instances where the model has high confidence in the detection. When investigating this further, it is found that when the model had high confidence, the saliency maps were more focused near the centre of the object. Given that YOLO-based detectors predict the centre coordinate of the bounding box, the centre region would be the most important. This phenomenon can be seen in Fig. 3.7, and unfortunately, does result in less attractive and uninformative saliency maps for the user.

When evaluating all instances, regardless of the model's confidence, the maximum IoU achieved in the Wrapping Game on the summer set is 0.355. This IoU

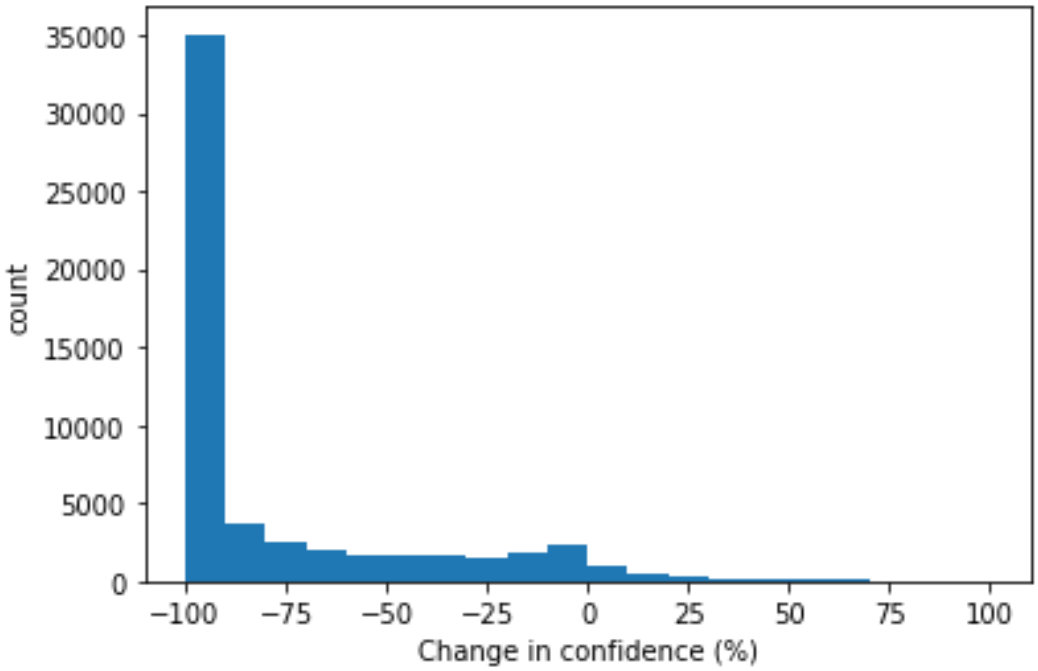
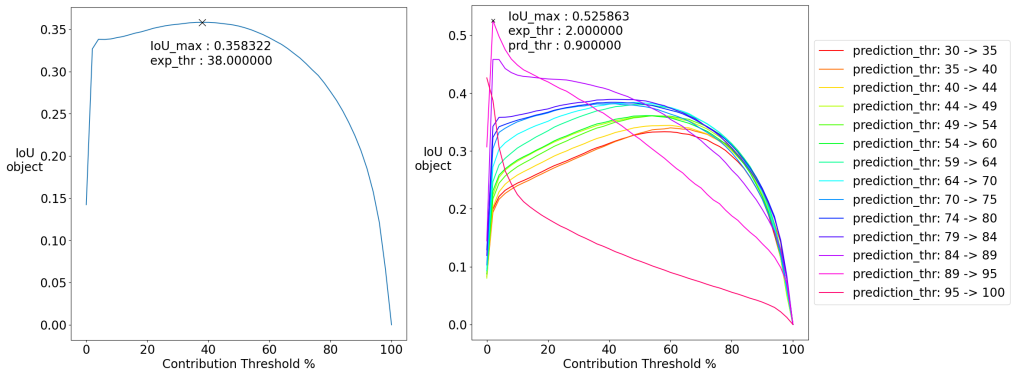


Figure 3.15: Change in confidence score due to salience-weighted perturbation.

score is obtained when the mask is generated using pixels from the saliency map that exceed 42% of the maximum value. Using this same threshold, and disregarding the peak in Fig. 3.17a, a similar IoU score is achieved on the winter set. However, if the peak generated by the spike is taken into account, the IoU score for the winter set slightly improves to 0.37. Therefore, the spike has ultimately a negligible effect on the final metric.

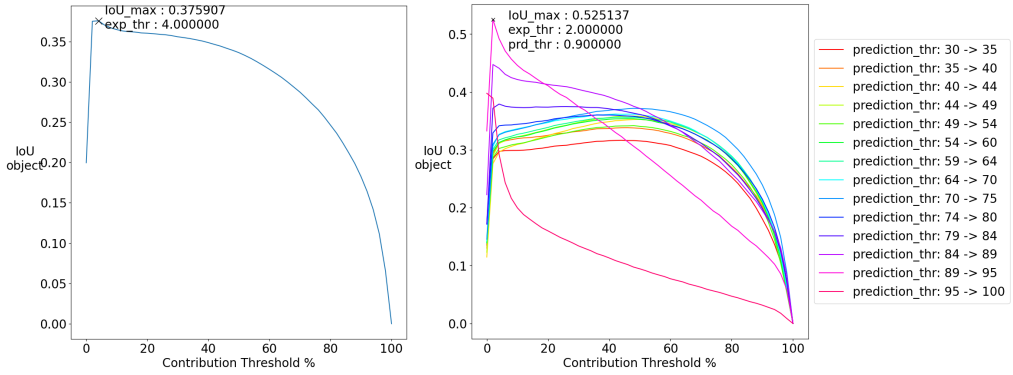
Changing the discussion to the second setting, it can be seen that the lines representing model confidence $\in (30, 85)$ have less variance when compared to the plot in Fig. 2.23. This observation, coupled with the fact that the IoU does not reach zero until nearly 100% of the pixels have been removed, indicates that the explainer is less susceptible to the model's performance, except for the aforementioned spike.

Based on the outcomes of the Wrapping Game, it can be inferred that the proposed Grad-CAM explainer demonstrates reasonable discriminativeness. However, when comparing the outcome of this analysis to the performance of DetDSHAP in Fig. 2.23, Grad-CAM comes off much poorer. Firstly, the maximum IoU score for Grad-CAM achieved in any scenario is 0.528, whereas in previous experiments DetDSHAP achieved a top score of 0.746. Secondly, the aforementioned spikes are a result of much of the attention being wasted on the centre mass of the detected objects, making Grad-CAM less descriptive. Further comparisons with this Grad-



(a) **(b)**

Figure 3.16: Plots showing the Wrapping Game analysis of the proposed Grad-CAM framework with YOLOv5 on the summer set. (a) analysis across all instances in the dataset. (b) analysis considering the model's confidence in a given instance.



(a) **(b)**

Figure 3.17: Plots showing the Wrapping Game analysis of the proposed Grad-CAM framework with YOLOv5 on the winter set. (a) analysis across all instances in the dataset. (b) analysis considering the model's confidence in a given instance.

CAM based explainer, and other novel explainers that are proposed as part of this thesis, will be discussed in the subsequent chapters.

3.4.4 Diagnosing Deep Detector Reasoning with Grad-CAM

The analysis in this section is conducted on a YOLOv5 model that was trained on the XAI-AV dataset that has previously been introduced in Chapter 2. Two samples from the summer set are selected, and saliency maps are generated for all objects detected by the network in these samples, and displayed. These two samples are shown in Fig. 3.18. The first sample was taken as the collection vehicle was driving down a straight road, with vehicles parked along the side of the road, and pedestrians walking along the pavement. The second sample taken from when the capture vehicle was stopped at a traffic light, contains more classes of objects, including a motorbike, and some instances of traffic light.



(a) Sample 1. **(b)** Sample 2.

Figure 3.18: This figure shows the two samples used in the diagnosis of the deep detector's reasoning which was conducted using Grad-CAM. The bounding boxes predicted by the deep network have been plotted.

In each figure where the saliency maps have been plotted, both the raw saliency map, and the input image with the saliency map superimposed over the top, are available. The images with the map superimposed over the top are referred to as the 'overlay'. The first row of each figure displays the overlaid maps from the shallow, middle, and deep detection layers, while the second row shows the corresponding raw maps. In each overlay, there are the detections that were made by a given detection layer. In this section, the dark red parts are the most salient, whereas, the dark blue parts are the least salient.

The first class discussed is 'vehicle'. Saliency maps for this class were generated for both samples and are displayed in Fig. 3.19 and Fig. 3.20. In Sample 2, the model's lower detection layer did not detect any vehicles; thus, no saliency maps are shown for that layer. Analysing both sets of saliency maps, it is evident that the model's attention is primarily focused around the central regions of the bounding boxes. Additionally, the Shallow and Middle Layers in both figures allocate some attention to areas along the bottom left and right edges of the image, outside of the roadway.

However, an exception is noted in Fig. 3.20a, where the attention appears more diffusely spread throughout the image. This could indicate that the network's attention has 'wandered'. In reality, this dispersion is due to the deep network predicting a significantly lower class score for this instance, resulting in a reduced backpropa-

gated value, and a smaller class activation. Given that this bounding box is detected by the top layer with higher certainty, it is likely that it would have been rejected during the NMS post-processing step.

As previously discussed for the class of 'vehicle', the shallow and middle layers allocate some attention to the roadway, while the deep layer also focuses on the road surface, particularly towards the centre, rather than the edges. This pattern could suggest that the network leverages contextual information from the road surface to aid in the detection of these vehicles.

Juxtaposed, Fig. 3.21c and Fig. 3.21f illustrate an example of where the network's attention appears misplaced, potentially indicating confusion. In these saliency maps, a significant amount of attention is directed toward the sky and bright reflections at the top of a building, areas where no relevant objects are present. This could suggest that the network is misinterpreting bright reflections, or high-contrast areas, as cues for object presence, which could lead to false positives. Such behaviour underscores a limitation, where the model may struggle to differentiate between contextual cues that support correct detections, and those that introduce ambiguity.

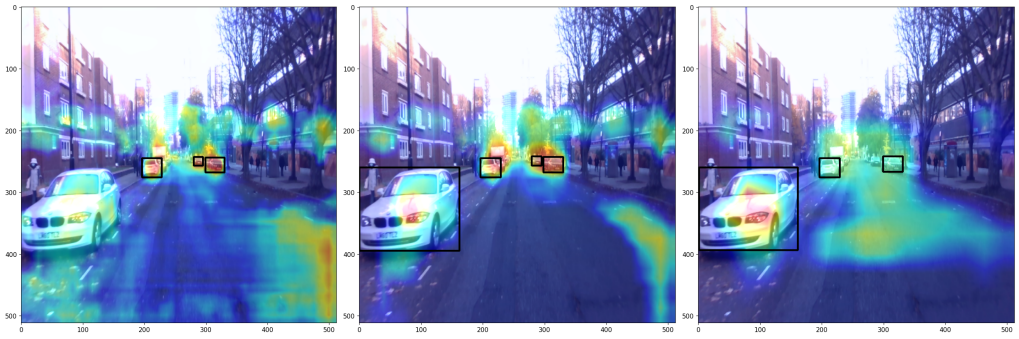
Figure 3.23 illustrates how the proposed Grad-CAM framework can be valuable for investigators who lack access to the original dataset. The saliency maps for the class 'traffic light' show that the model's attention is predominantly concentrated in the top portion of the image. Notably, Fig. 3.23a shows a distinct boundary across the image around 220 pixels down, suggesting that the model's focus is biased towards this upper area. This observation aligns with the distribution of traffic lights, as shown in Fig. 2.22b. Indeed, for many of the saliency maps shown in figures 3.19, 3.20, 3.21, 3.22, and 3.23, the regions highlighted by Grad-CAM closely align with the object distributions presented in Section 2.4.3.

3.5 Summary

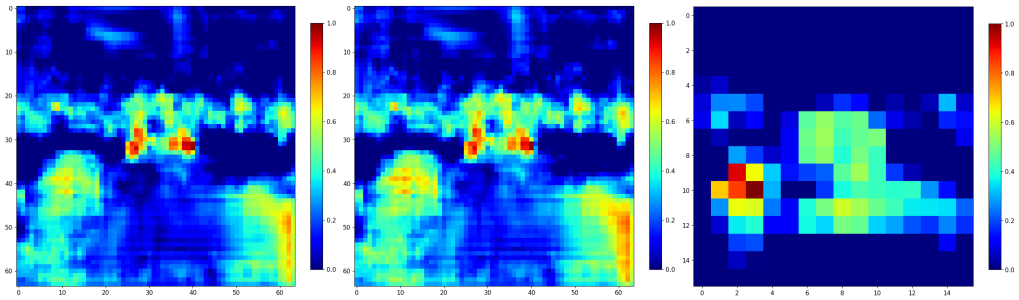
On-board object detection is a crucial requirement for UAVs when performing tasks such as obstacle avoidance, search and rescue, and automatic target recognition. One of the main challenges of conducting object detection for a UAV is that objects of interest appear very small in images captured from higher altitudes, making detection more difficult. This chapter presents a solution to address this challenge, while also meeting two additional criteria essential for the deployment of artificial

intelligence on UAVs: real-time operational capability; and user trust in the explicant's predictions. To tackle the challenge of small object detection, an image Tile Loader was introduced to enhance the capabilities of DNN-style detectors while minimising processing time.

In addition, a Grad-CAM-based framework was employed to provide near real-time visual explanations, enhancing the transparency of the deep object detectors analysed in this chapter. This framework highlights which parts of the input image contribute most significantly to the model's decisions, facilitating clearer insight into model behaviour during operation. This capability is vital for applications that require immediate insights, such as autonomous navigation, and real-time decision-making.

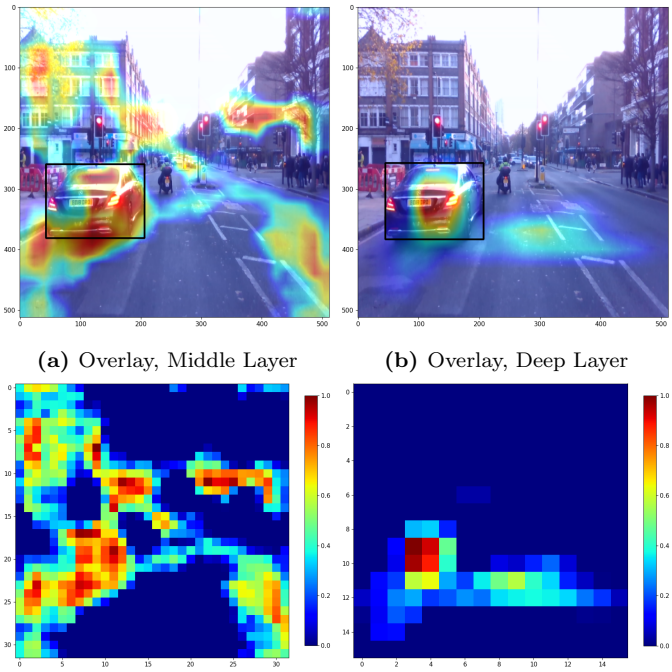


(a) Overlay, Shallow Layer **(b)** Overlay, Middle Layer **(c)** Overlay, Deep Layer



(d) Raw, Shallow Layer **(e)** Raw, Middle Layer **(f)** Raw, Deep Layer

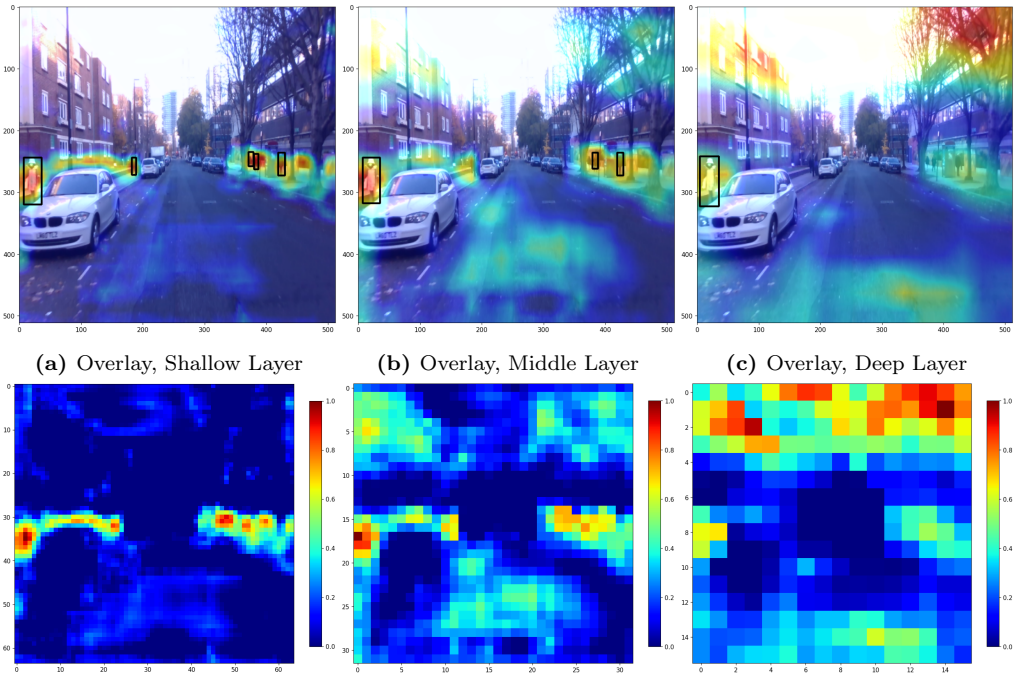
Figure 3.19: Saliency maps showing the attention of the deep network on sample 1 highlighting the network's attention for the class 'Vehicle'.



(a) Overlay, Middle Layer **(b)** Overlay, Deep Layer

(c) Raw, Middle Layer **(d)** Raw, Deep Layer

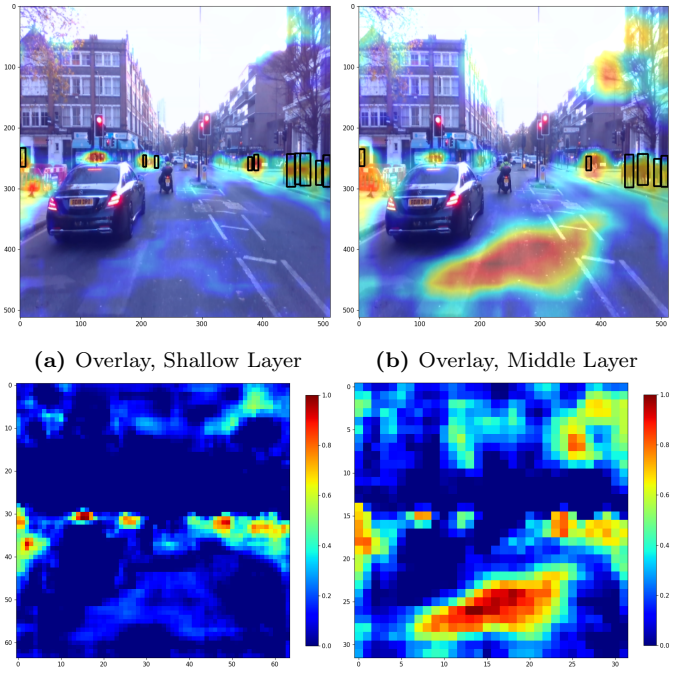
Figure 3.20: Saliency maps showing the attention of the deep network on sample 2 highlighting the network's attention for the class 'Vehicle'.



(a) Overlay, Shallow Layer **(b)** Overlay, Middle Layer **(c)** Overlay, Deep Layer

(d) Raw, Shallow Layer **(e)** Raw, Middle Layer **(f)** Raw, Deep Layer

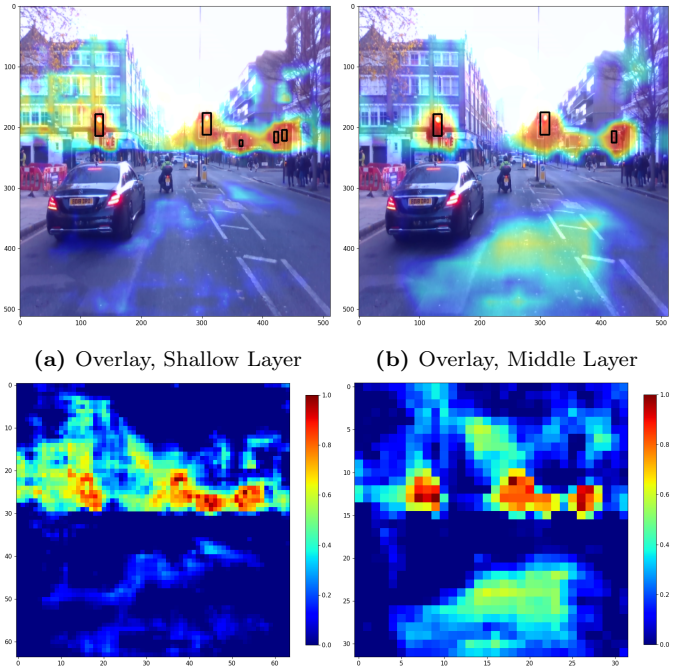
Figure 3.21: Saliency maps showing the attention of the deep network on sample 1 highlighting the network's attention for the class 'Pedestrian'.



(a) Overlay, Shallow Layer **(b)** Overlay, Middle Layer

(c) Raw, Shallow Layer **(d)** Raw, Middle Layer

Figure 3.22: Saliency maps showing the attention of the deep network on sample 2 highlighting the network's attention for the class 'Pedestrian'.



(a) Overlay, Shallow Layer **(b)** Overlay, Middle Layer

(c) Raw, Shallow Layer **(d)** Raw, Middle Layer

Figure 3.23: Saliency maps showing the attention of the deep network trained on sample 2. The first row displays the overlaid maps from the shallow, middle, and deep detection layers, while the second row shows the corresponding raw maps, highlighting the attention for the class 'Traffic Light'.

Chapter 4

Explainable Object Detection for Autonomous Vehicles using KernelSHAP

This section will outline how a permutation style explainer, KernelSHAP was adapted so that it could be used with detection style architectures. When given an image I, a detection network f, and a target bounding box T, the explainer produces a saliency map that indicates how the f relates I to T. Images from both aerial and ground-based robotic platforms were used to showcase the effectiveness of the explainers.

4.1 Introduction and Motivation

Previously, Chapter 3 presented a gradient-based explainer for interpreting YOLOv5, which generated saliency maps by propagating gradients to the final convolutional layer. Although effective for certain tasks, this approach requires direct access to the network's internals and is thus less portable when comparing different architectures. To ensure consistency with that earlier work and to provide a coherent baseline across this thesis, YOLOv5 is maintained as the main detection model of interest. In contrast, perturbation-based methods treat the network as a closed box, making them applicable to a wide range of models without necessitating architectural modifications. Furthermore, KernelSHAP computes Shapley values to fairly distribute feature contributions, ensuring that both positive and negative influences

are accurately captured.

In this chapter, KernelSHAP has been adapted for detection-style networks. The proposed framework extracts image features using superpixel segmentation via Simple Linear Iterative Clustering (SLIC), thereby generating interpretable segments from the input image. These superpixels serve as the basis for evaluating feature contributions: by computing Shapley values for each segment, the method produces "SHAP maps" that visually represent the influence of different regions on the predicted bounding boxes. Figure 4.1 shows an example of a saliency map that the method can produce.

A key motivation for the approach in this chapter is to improve upon methods like D-RISE. Petsiuk et al.[84] propose D-RISE, which aims to generate saliency maps for detection networks by perturbing the entire image. However, when dealing with aerial imagery, where the majority of objects are small, focusing on the entire image may dilute the explanation. In contrast, the approach presented here concentrates on a subsection of the image that contains the object of interest. Such conditions demand an explainer that can accurately capture low-resolution features and disentangle overlapping contributions—a need that is addressed by the proposed KernelSHAP framework.

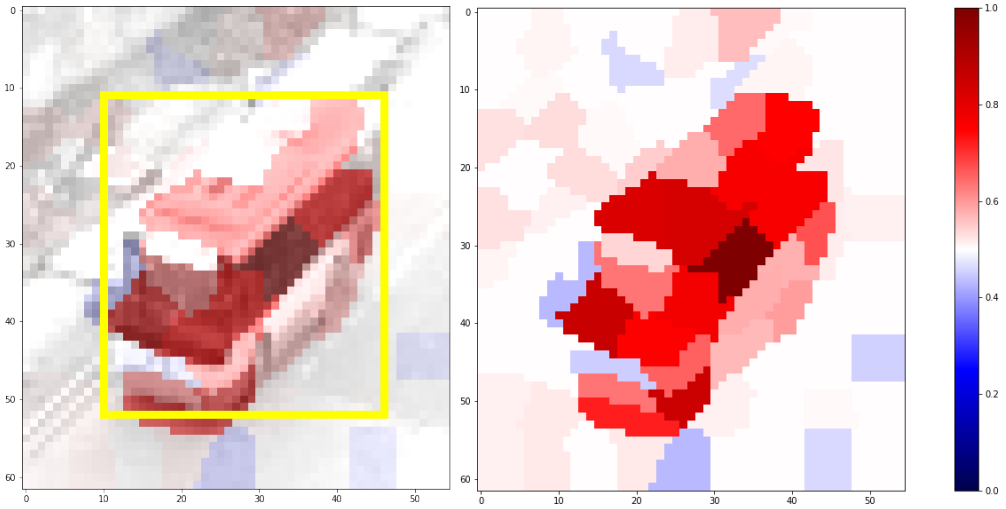
Both D-RISE and RISE[12] rely on randomly generated masks to perturb the image and then produce a weighted average of these masks as the final saliency map. For RISE, the weights are derived from the target class score, while D-RISE employs a similarity metric that evaluates how different a bounding box predicted from a masked image is from that of the original image. A notable limitation of these methods is that they do not guarantee a fair distribution of contributions among features. By leveraging Shapley values, which possess the property of efficiency [8], the proposed KernelSHAP adaptation guarantees that the total score from a given instance is fairly distributed. An additional advantage is its ability to identify image regions that have a negative contribution, thereby providing enhanced investigatory power.

D-RISE typically treats each pixel as the fundamental unit of explanation, a granularity that does not align well with human perception [83]. Here, the aim is to extract features that could be more easily understood by a human agent. This drove the choice of extractor that is outlined in section 4.2.1.

The main contributions of this chapter are as follows:



(a) Input image with the target bounding box



(b) (L) Saliency Map over the original image.
(R) The saliency map alone with the scales shown.

Figure 4.1: An example of the KernelSHAP explanation. (a) Shows the image with the target bounding box in blue. (b) Shows the saliency map that indicates the important parts of the object that caused the network to predict this box.

- A novel adaptation of KernelSHAP that integrates SLIC-based segmentation with detection outputs to generate SHAP maps, offering a robust and architecture-independent explanation framework.

- A qualitative demonstration of the explainer's effectiveness, including its ability to identify artificially introduced bias, thereby providing deeper insights into the network's decision-making.

- A comprehensive quantitative evaluation employing metrics such as the Pointing Game, Deletion, and Insertion protocols, which collectively assess the accuracy of the explainer in capturing feature contributions across both aerial and ground-based images.

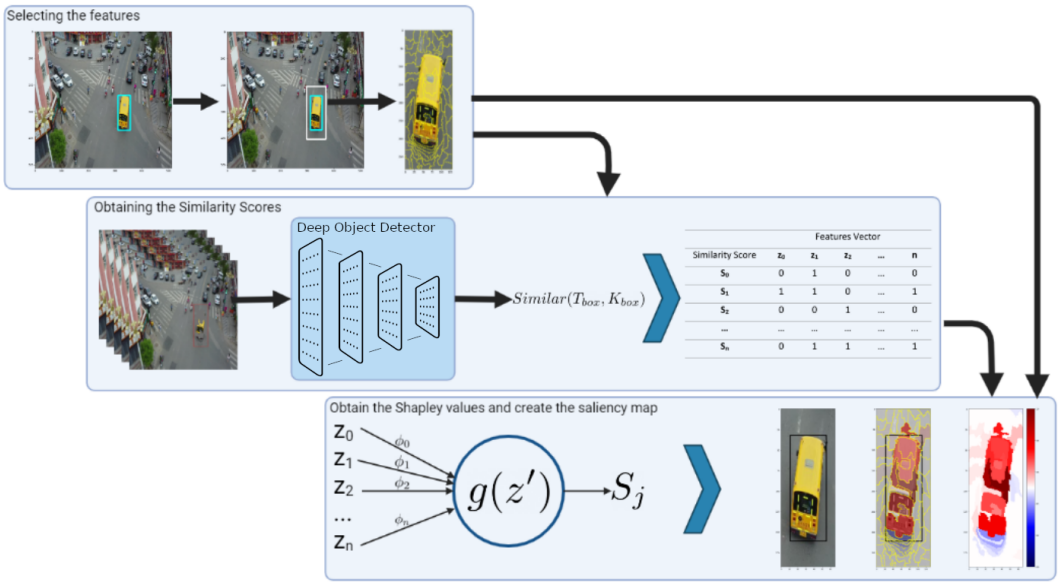


Figure 4.2: Overview of the proposed KernelSHAP framework.

The following sections detail the methodology behind this adaptation—including feature extraction, Shapley value computation, and the mapping process for creating SHAP maps—and present experimental results that underscore its effectiveness. This work demonstrates that the adapted KernelSHAP framework offers a significant improvement over existing methods such as D-RISE, particularly in the challenging context of aerial object detection.

4.2 Methodology

The objective of this piece of work is to adapt the vanilla KernelSHAP such that, when given an image I, a detection network f, and a target bounding box T, the explainer produces a saliency map that indicates how the f relates I to T. As stated, KernelSHAP estimates the Shapley values of each feature value of an instance. KernelSHAP represents Shapley value explanation as a linear model g, given by Eq. (4.1). Where M is the maximum coalition size and the number of features, and ϕ_j is the Shapley value for a given feature. z' is the coalition vector and is described by Eq. (4.2), where 0 indicates an absent feature, and 1 indicates the feature is present.

$$g(z') = \phi_0 + \sum_{j=1}^{M} \phi_j z'_j \tag{4.1}$$



(a) Feature Boundaries

(b) (L) Feature selection (R) example of masking

Figure 4.3: (a) shows the feature boundary in white for the bounding box for the bus in yellow. (b) shows the features that are extracted using SLIC on the left, and an example of a perturbed image on the right.

$$z'_k \in \{0,1\}^M, k \in \{1,...,M\} \tag{4.2}$$

KernelSHAP will sample coalitions of the features using the coalition vector. Each sample is converted to the original feature space, and then given to the network to compute the weight of a given z'_k. This new data is used to fit the linear model and return the Shapley values. Therefore, to adapt KernelSHAP, this method needed to extract the features from the image I, remove features based on z'_k – which will be achieved by masking - before passing the masked image to the detection network. Lastly, it needed to be able to calculate the scores for a given masked image which will be used to fit the linear model. The full framework is shown in Fig. 4.2.

4.2.1 Extracting Features

The first step is to define the features of the instance. The feature extraction process involves segmenting the input into interpretable parts, such as object patches or super-pixels, which are analysed to determine their contribution to the overall prediction. While increasing the number of features allows more detailed information to be extracted from an instance, it also increases the computational burden and variance in the estimated Shapley Values. To manage this, the feature selec-

tion is limited to only the parts of the image contained within the bounding box and some areas immediately surrounding it. An example of this boundary is shown in Fig. 4.3a, where the original bounding box is enlarged by 50%. Analysing the entire image would be inefficient, as distant regions are expected to have minimal contribution and would require a significantly higher number of features.

Super-pixels are extracted from within the boundary using Simple Linear Iterative Clustering (SLIC) [158] to perform pseudo-clustering. Super-pixels group perceptually similar pixels, effectively preserving object boundaries and reducing image complexity. Unlike fixed grid-based methods, super-pixels adapt to the underlying image content, ensuring that segments align with natural structures within the image. In the proposed framework, SLIC was chosen for its balance between performance and efficiency [159]. Additionally, the flexibility in controlling the compactness of super-pixels allows the explainer to be fine-tuned, yielding more interpretable and visually coherent explanations.

Other region-based clustering approaches, such as watershed segmentation and k-means clustering, were considered but found to be less suitable for this task. Watershed segmentation tends to over-segment images, producing highly fragmented regions that do not align with object boundaries, while k-means clustering lacks spatial awareness, leading to irregular segment shapes. In contrast, SLIC efficiently balances spatial compactness and colour similarity, making it more effective for generating meaningful feature groupings in the context of explainable object detection.

The aim is to group similar pixels based on their colour and location, while also keeping the number of pixels in each group fairly balanced. Removing a larger group of pixels would cause the image to change drastically, which could mislead the explainer into attributing the group with a bigger contribution than it deserved. Figure 4.3b shows the segments that were extracted from the bus.

With the features selected a mask can be generated using a given coalition vector to remove them from the image. An absent feature is replaced with the average pixel value of the image. Figure 4.3b shows an example of such masking applied to the image of the bus, where a random selection of features have been removed.

Further examples of masked images are available in Fig. 4.4. In clockwise order, and starting in the top left, the first image shows when the full mask is applied and, in this case, a completely different object is detected. The second image shows when no masking is applied with the original box. In the third image, the first half of the



Figure 4.4: Examples of masks applied to the image with the nearest matched bounding box.

feature vector is obscured which has resulted in the top portion of the object being masked. This leads the network to make two mistakes: the object is misclassified as 'awning-tricycle', and the bounding box is misplaced. The final image was created when a random sample of the features was turned off, much like it would be during evaluation. In this scenario, despite the network placing the box very similarly to the original box, the object was misclassified as 'truck'.

4.2.2 Similarity Metric

KernelSHAP requires a method to score how well a given bounding box, K, from a masked image matches the target bounding box, T. This scoring is essential for calculating the weights used to fit the linear model, g(Eq. (4.1)). The metrics outlined in [84] are adopted to assess the similarity between bounding box dimensions and class scores.

The bounding box similarity score S_{box}, is computed using Intersection over

Union (IoU), which measures the area of overlap divided by the total area covered by both bounding boxes:

$$S_{box} = IoU(T_{box}, K_{box}) = \frac{|T_{box} \cap K_{box}|}{|T_{box} \cup K_{box}|} \tag{4.3}$$

where T_{box} and K_{box} represent the bounding box coordinates for the target and predicted boxes, respectively.

The class probability similarity, S_{cls}, is computed using cosine similarity, which measures the alignment between two class probability vectors:

$$S_{cls} = \frac{T_{cls} \cdot K_{cls}}{\|T_{cls}\|\|K_{cls}\|} \tag{4.4}$$

where T_{cls} and K_{cls} denote the probability distributions over the class labels for the target and predicted bounding boxes, respectively.

For networks such as YOLOv5, which also produce an objectness score, we introduce an additional term, S_{obj}, which evaluates the relative objectness confidence between the target and predicted bounding boxes:

$$S_{obj} = \min\left(\frac{K_{obj}}{T_{obj}}, 1\right) \tag{4.5}$$

where T_{obj} and K_{obj} represent the objectness scores of the target and predicted bounding boxes, respectively. The score is capped at 1 to prevent excessive scaling.

The final similarity score, S, is computed as the product of the three metrics:

$$S = S_{box} \times S_{cls} \times S_{obj} \tag{4.6}$$

This combined metric ensures that the generated saliency maps incorporate both classification and localisation information, providing a comprehensive assessment of feature importance.

4.2.3 Creating the Saliency Map

After the Shapley values have been obtained, the saliency map is created by mapping these values to the image space using the segmentation map. Figure 4.5 shows the final saliency map of the bus from Fig. 4.3a. The network pays particular attention to the roof of the bus. This makes sense since the images of the buses in the dataset will be taken from an overhead perspective. Thus, this is the component that the

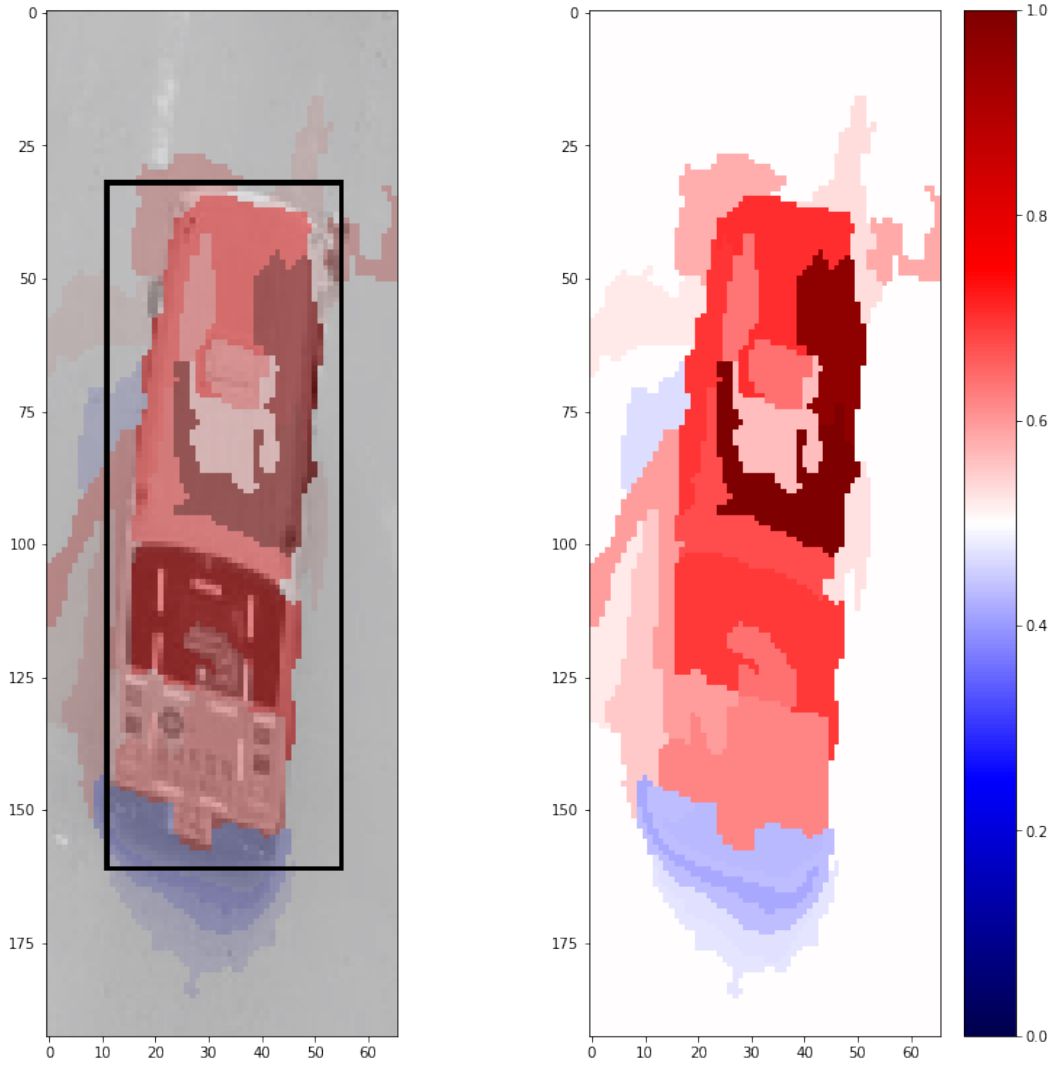


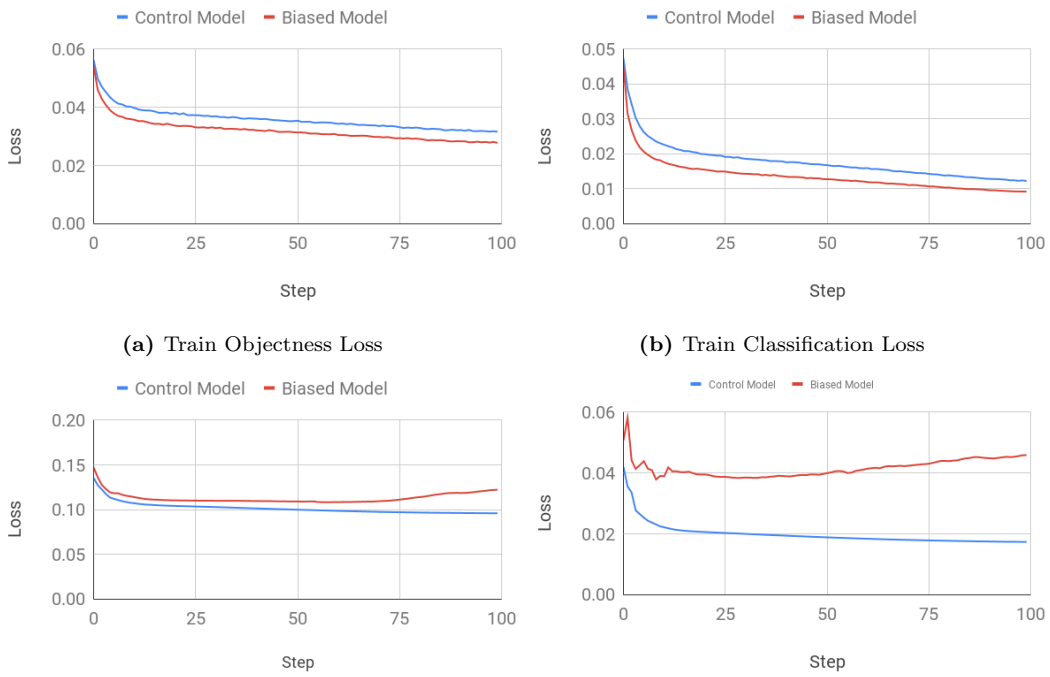
Figure 4.5: Saliency Map created using the proposed method showing the regions of the bus that are important to the DNN.

network would distinguish as separate from the background.

4.3 Experimental Setup

The experimental results presented in this chapter aim to thoroughly evaluate the proposed explainability framework across multiple settings, using a variety of models and datasets. These experiments are structured to assess both the qualitative and quantitative aspects of the explainer's performance, ensuring a comprehensive understanding of its strengths and limitations.

Most of the experiments presented here utilise a YOLOv5l model trained on the VisDrone dataset. In these tests, a deliberate bias was introduced to evaluate the explainer's ability to detect unnatural correlations in the dataset, following a similar



(a) Train Objectness Loss

(b) Train Classification Loss

(c) Valid Objectness Loss

(d) Valid Classification Loss

Figure 4.6: Loss plots for the training and validation sets during training of the biased and control model. Only the training set for the biased model contained the artificial bias, although both models were evaluated on the same validation set.

approach to [84]. A marker was artificially placed at the centre of every bounding box for the 'car' class, as shown in Fig. 4.7. A new model with the same architecture was trained on the biased dataset, defined as the biased network. A control network was re-trained on the default dataset with the same hyper-parameters in order to compare the saliency maps from the two. The goal was to simulate a real-world scenario where a dataset might unintentionally contain biased features.

Two types of quantitative analyses were conducted on the models trained on the Visdrone dataset. First, the Pointing Game from [115] was employed to evaluate the discriminative power of the explainer. This method measures how accurately the explainer identifies important pixels related to the network's predictions. Second, the Deletion and Insertion metrics from [12] were used to assess the explainer's ability to capture the true cause of the network's decisions by incrementally removing or inserting information in the saliency maps. Given the computational cost of applying this explainer, a random subset of 10% of the VisDrone test-dev set was used in these analyses. Although this limited number of data can restrict us from being fully conclusive, the full dataset was used in the rest of the thesis.

Figure 4.6 shows the loss plots from training the biased and control models.

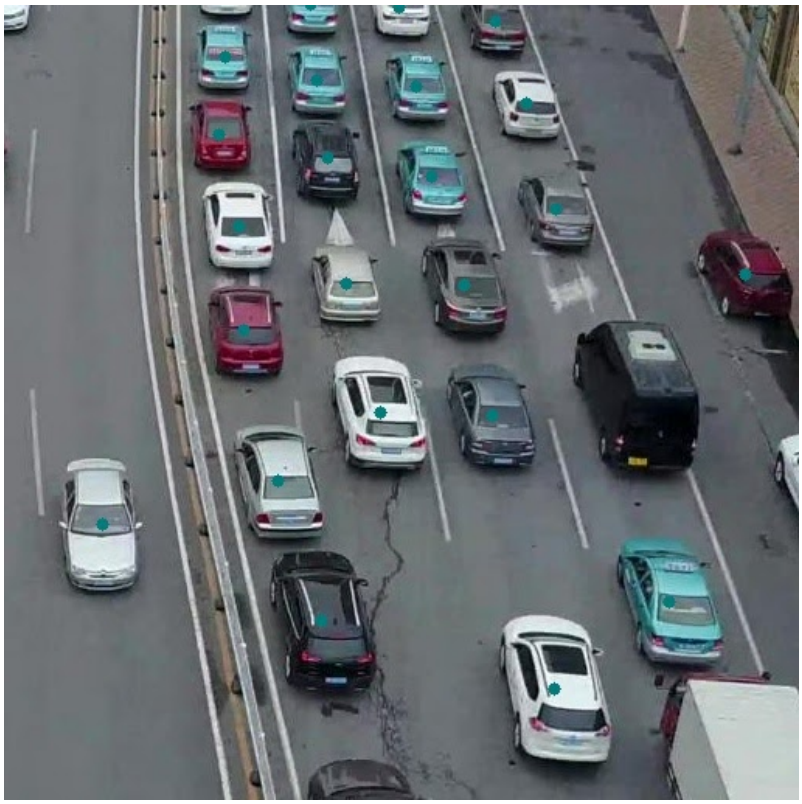


Figure 4.7: An example from the biased dataset showing the marker over the top of cars. Note the black van on the right side of the image does not have a marker.

At each epoch, both models were evaluated on the default Visdrone validation set, which was left unbiased and remained the same for both models. Figure 4.6a and Fig. 4.6b show the loss curve for the two networks on their respective datasets. Based on these plots alone, one might incorrectly assume the biased model is superior, as it reaches a lower loss value. However, the plots in Fig. 4.6c and Fig. 4.6d, reveal that the loss on the validation set diverged for the biased model. This behaviour is expected when a natural bias exists in the training set, but not in the validation set. The intended outcome was achieved, indicating that the biased network has overfitted to the biased marker. With this insight, the explainer will be applied to determine if it appropriately identifies the bias.

In addition to models trained on the Visdrone dataset, models that were trained on the proprietary dataset introduced in Chapter 2 are also utilised to validate the explanation framework introduced in this chapter. SHAP maps were generated from detections made by the Faster R-CNN and YOLOv5 models that were introduced in the aforementioned chapter for comparison. Moreover, from the same chapter, the Wrapping Game analysis was applied to yield a deeper examination of the explainer's effectiveness across different model architectures and confidence levels.

The semantic labels are not available in the current VisDrone dataset and, as such, the Wrapping Game could not be applied to this dataset. However, since the bias was introduced programatically, a similar analysis was still possible. It is important to confirm that the bias is detectable in aggregate data, not just in the examples used in the qualitative analysis. Therefore, an analysis was designed to leverage prior knowledge of the bias by measuring the IoU between the bias and the highest contributing feature in each instance. The IoU was calculated for all the samples within the utilised test-dev subset, providing a more comprehensive measure of the explainer's ability to detect the bias across a broader set of data.

4.4 Qualitative Analysis

This section evaluates the quality of the SHAP maps generated by the proposed methodology, with a focus on interpretability and effectiveness in highlighting key features. SHAP maps were visualised for both the biased and control models to assess how the introduced bias impacts model decision-making. Additionally, SHAP maps were generated for different model architectures, specifically YOLOv5 and Faster R-CNN, to compare how each architecture highlights important image regions.

4.4.1 Identifying the Deliberate Bias in SHAP Maps

Figure 4.8 shows the input image used to investigate how the bias impacted both the biased and control models. When the bias was present, both networks produced a bounding box similar to the groundtruth shown. Figure 4.9 shows the saliency map which explains the box predicted by the biased network. It can be seen that the bias shows up in dark red, which is the expected response. Figure 4.10 shows the saliency map for the control network. Unlike the biased network, much more of the car is indicated as having a positive contribution, particularly the parts around the front bumper and roof. The bias appears light blue, which indicates that it is causing the network some confusion - an appropriate response since it was trained on a dataset where the bias was never present.

When the bias was not included in the image, the biased network no longer labelled the object as a car, instead detecting it as a 'van'. By applying the method, it is possible to ask the network 'why' this distinction was made, as shown in Fig. 4.11.



Figure 4.8: Input image with the groundtruth bounding box in orange, with the marker present.

Similarly, when provided with the groundtruth, the network's response to 'why not' resulted in the saliency map in Fig. 4.12, highlighting the absence of the bias as the key reason for predicting 'van' instead of 'car'. The SHAP maps consistently pointed to the region where the bias marker would have been, confirming that the network relied on the marker for its prediction.

4.4.2 Identifying Failure Modes

This experiment is to place the bias in a location where the biased model does not expect it, causing the model to be misled. The method will then be applied to diagnose the failure - this will be done by generating the saliency maps from the groundtruth. As was shown previously, using the groundtruth, instead of a predicted box, can reveal the parts of the image that are causing the network to be misled. Figure 4.13 shows the saliency maps that were generated from the biased model.

When the bias is absent, the biased network can correctly predict the object as a 'bus'. From Fig. 4.13b it can be seen that the network was able to find a strong relationship between the front and roof of the bus and the groundtruth. This is

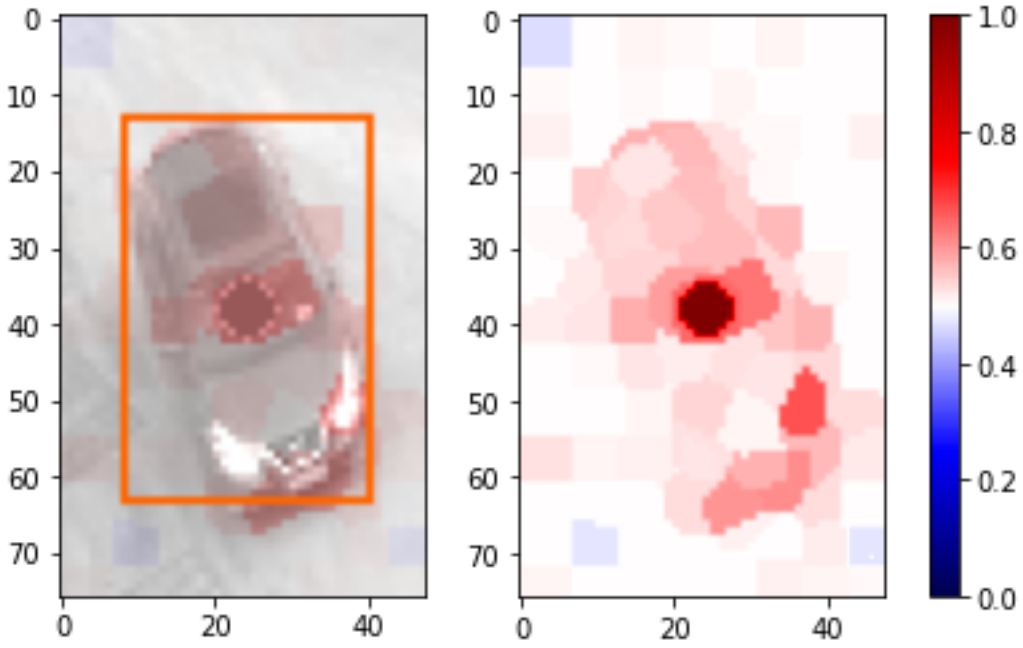


Figure 4.9: SHAP map generated for the biased model of a car with the bias present.

the same when the bias is present in Fig. 4.13a, even though the network misclassified the bus as a car. Nevertheless, the saliency map has picked out the bias as problematic, as indicated by the darker blue colour. This would indicate to a developer that this was the reason behind the failure.

4.4.3 Comparing Model Behaviours

Next, the analysis is extended to examine how YOLOv5 and Faster R-CNN highlight important regions in the input image. Both models were trained on the proprietary dataset introduced in Chapter 2, and SHAP maps were generated to identify the regions each model considers critical for object detection. These visualisations provide a comparison of how the two architectures prioritise key features, offering further insight into their decision-making processes.

The image used in this analysis is shown in Fig. 4.14, featuring a black sedan in front of the capture vehicle. The SHAP maps for both models are presented in Fig. 4.15. While the models predicted similar, though slightly different, bounding boxes for the object, resulting in marginally different SLIC segmentation, several key regions of the scene are highlighted by both models. Notably, both models agree on the importance of the area near the top right of the rear window, parts of the roadway beneath the right side of the rear bumper, and sections of the right facets

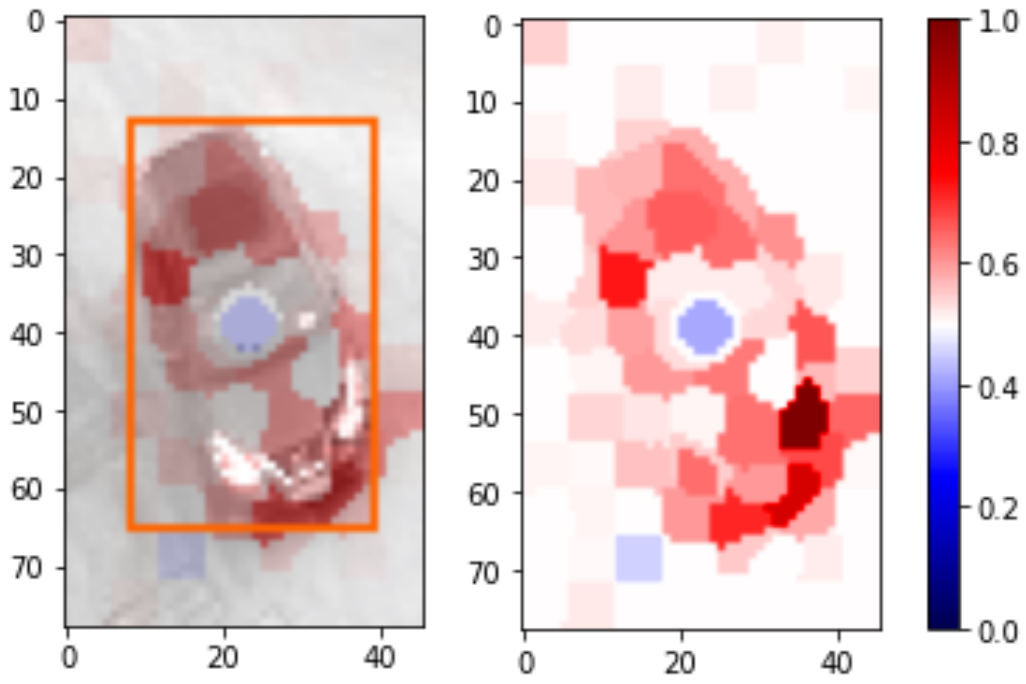


Figure 4.10: SHAP map generated for the control model of a car with the bias present.

of the vehicle partially exposed to the camera.

However, the SHAP maps also reveal differences in how the models process the scene. YOLOv5 appears to rely more heavily on the left side of the vehicle, as indicated by the darker red patches along the left part of the rear window, the left side of the trunk, and especially the left brake light. There is also glare on the rear window behind the driver, and interestingly, the SHAP map suggests that YOLOv5 is negatively affected by this glare.

4.5 Quantitative Evaluation

While the qualitative analysis provided valuable insights by focusing on individual samples, a broader quantitative analysis is required to validate the robustness of the explainer on a larger scale, and ensure that the insights gained are not isolated to specific examples. Moreover, qualitative metrics are necessary in order to objectively compare different explainers. This section will outline the experiments conducted and present the results of the quantitative analysis.

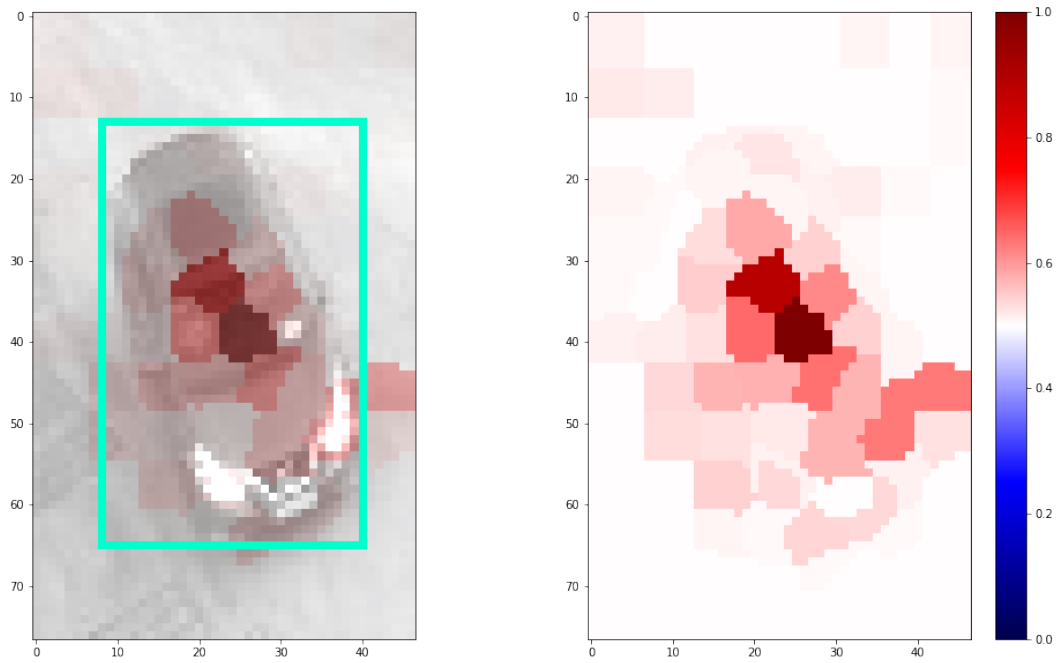


Figure 4.11: Saliency maps generated from the biased model on the predicted bounding box without the bias present in the input.

4.5.1 The Pointing Game

The Pointing Game was briefly discussed in Chapter 2. This protocol compares the explanation to a human-annotated bounding box - the groundtruth bounding box. A 'hit' is defined as an instance where the maximum point of the saliency map lies within the groundtruth bounding box. A 'miss' is an instance where the maximum point lies outside the groundtruth bounding box. The localisation accuracy is then defined using Eq. (1.13). The investigation conducted here led to the development of the Wrapping Game that was proposed in Chapter 2.

This protocol has been applied in [4] and [12] to examine saliency maps created for classification algorithms. In these scenarios, the performance of the model may have an impact on the outcomes of the analysis. On the other hand, given that the explicant in this case is a detection network, the model's performance - as it was found to be the case - is almost certainly to influence the outcome of the vanilla Pointing Game. In order to mitigate this influence in the evaluation here, in addition to the groundtruth, the Pointing Game metric is also determined for the predicted box and the bias marker. The bias marker was utilised since it is known to have a significant influence on the biased model.

Table 4.1 shows the results from the Pointing Game study. When applying the protocol to the groundtruth and predicted boxes, the saliency maps generated

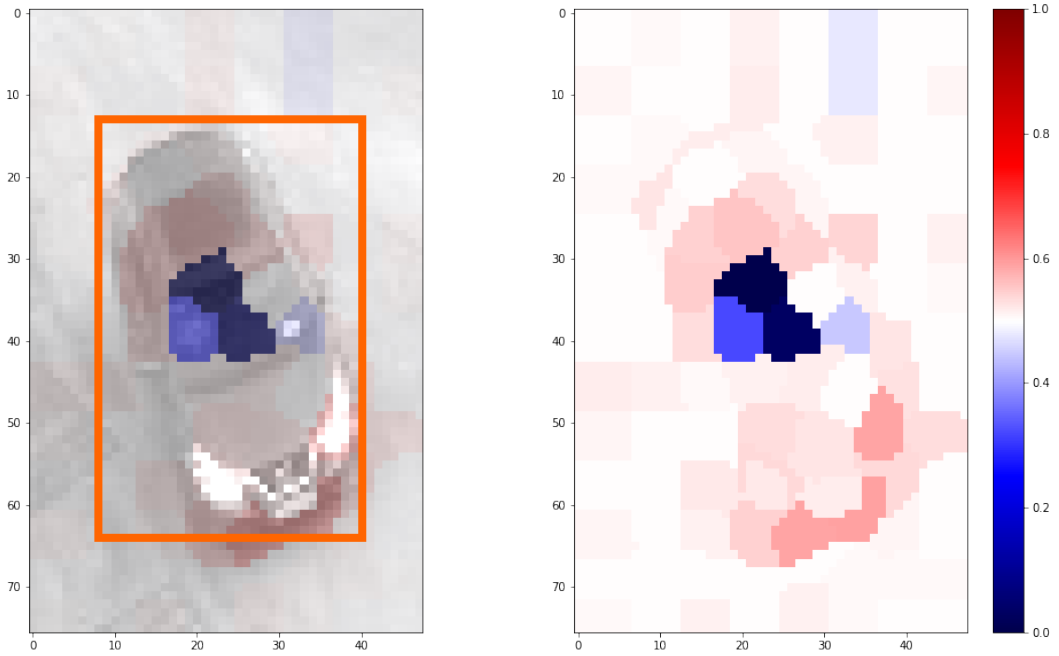
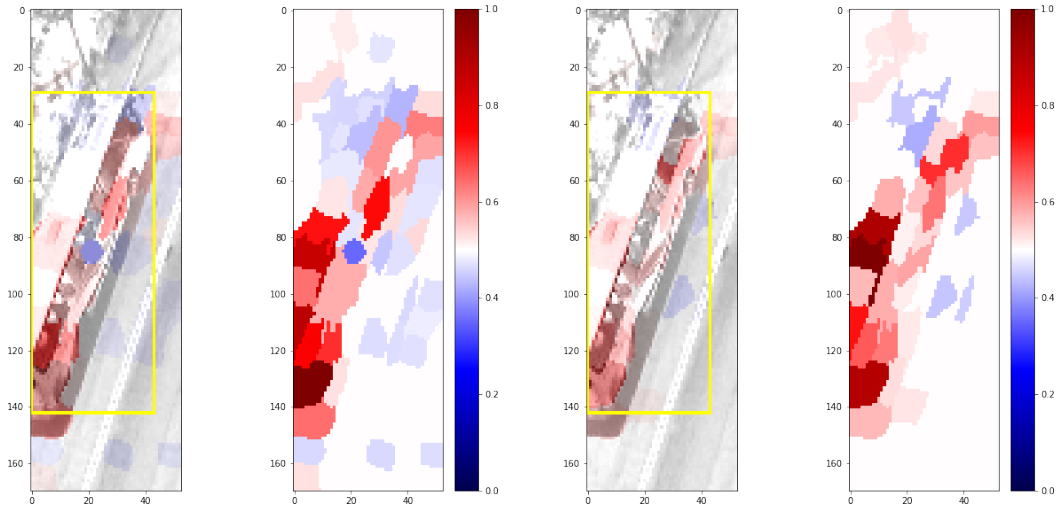


Figure 4.12: Saliency maps generated from the biased model on the groundtruth bounding box without the bias present in the input.

from the control network were used to gauge the accuracy on natural images. On the other hand, the bias network was used when applying the protocol to the bias maker, which can only be applied to the class 'car'.

As can be seen, the explainer achieved an accuracy of 75% on the groundtruth boxes, which is much lower than what is achieved on the other two targets. This can be attributed to three situations: Firstly, the network picked up on some objects that the human agent did not, which meant there would be no box to hit. Secondly, there were situations where the human and the network placed the bounds slightly differently. Thirdly, there may be instances where contextual information outside of the box was more important to some predictions - this may not be the preferred behaviour from the network, nonetheless, it would be inappropriate to attribute this negatively to the explainer.

On the other hand, the high Pointing Game score on the predicted box indicates the proposed explainer can discriminate the part of the image where one might expect the network's attention to be. In addition, the high score on the bias marker suggests the explainer can also discriminate the feature that is known to have a significant impact on the biased network.



(a) Bias is present **(b)** Bias is absent

Figure 4.13: The saliency maps generated of an object of class 'bus' using the groundtruth bounding box

	Groundtruth Box	Predicted Box	Bias Marker
Accuracy	75%	98%	92%

Table 4.1: Pointing Game accuracy under various different targets.

4.5.2 Evaluation of Bias Detection

Here, the ability of DetDSHAP to discriminate the deliberate bias in groups of data is assessed. This assessment is conducted in four steps using the subset from the test-dev set. In the first step, the saliency map was generated for the biased model with the bias present in the data. Secondly, the segment of the SHAP map with the highest contribution is extracted from each sample. In the third step, the IoU of this feature and the bias marker is calculated. With the IoU calculated for all instances of the bias, the fourth and final step is to calculate the accuracy using Eq. (1.13). Unlike the vanilla Pointing Game, a 'hit' is now considered if a feature has an IoU value over a given threshold, instead of just a single point. This was performed using IoU thresholds from 0.05 to 0.95 at intervals of 0.05. This is the same methodology used for calculating the precision of deep object detectors.

One of the ambitions of this work is to develop explainers aimed at explaining object detection algorithms to be deployed on aerial drone platforms. Given that objects in aerial images appear very small, since the images are often captured at



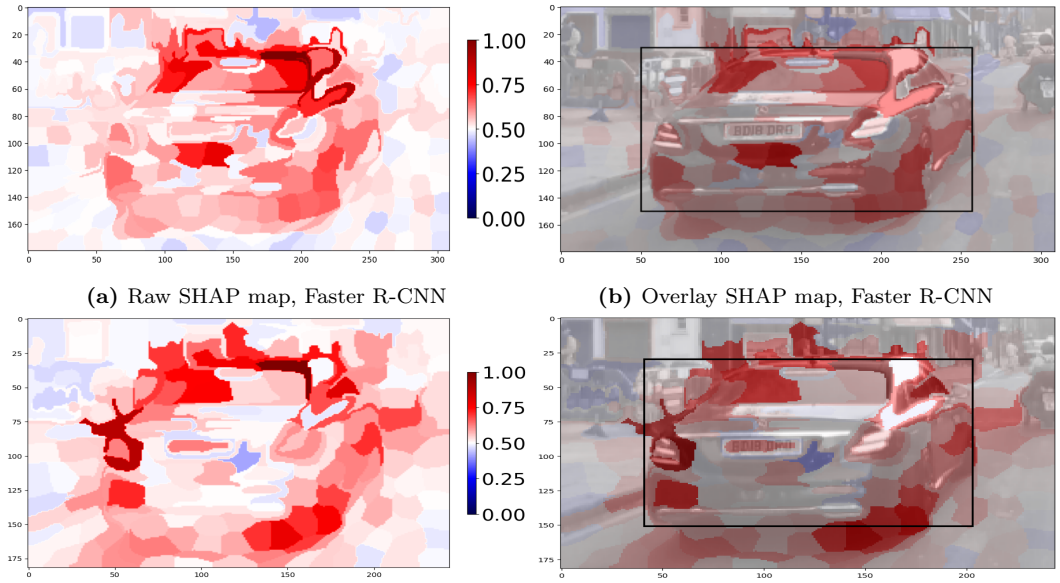
Figure 4.14: Image with the groundtruth bounding box used to compare the behaviour of different architectures.

higher altitudes, it is pertinent to include this consideration in evaluation. Hence, the evaluation was performed on boxes split into different sizes, and shown in Fig. 4.16. In this plot, boxes are considered 'small' when they are under 27 pixels across, 'medium' if they are between 27 to 45 pixels, and 'large' if they are above 45 pixels.

What this plot revealed is that, while the proposed framework is effective on the vast majority of instances, it is approaching the limit of its effectiveness on the small boxes. This is evident when considering the IoU threshold of 0.5, for which the accuracy on medium and large boxes is 70%, whereas for smaller boxes, this same threshold yields an accuracy of only 52%. This is a significant drop, prompting the need for further investigation of the framework's performance on small objects during the Deletion and Insertion analyses.

4.5.3 Deletion and Insertion

Deletion sequentially removes pixels, starting from the maximum value of the saliency map, while measuring how quickly the network's output deviates from the original prediction using 4.6. It is expected that, if the explainer does correctly highlight the contribution, the prediction will diverge quickly and then plateau when pixels become less important. Contrarily, Insertion measures how much the prediction



(a) Raw SHAP map, Faster R-CNN

(b) Overlay SHAP map, Faster R-CNN

(c) Raw SHAP map, YOLOv5

(d) Overlay SHAP map, YOLOv5

Figure 4.15: SHAP map comparison for Faster R-CNN and YOLOv5. The first row displays raw and overlay SHAP maps for Faster R-CNN, and the second row shows the corresponding maps for Yolov5.

converges as pixels are added back into the image. The opposite is expected here - the prediction will approach the original prediction quickly, as more pixels are added back and then plateau. The results are plotted and the final metrics are obtained from the AUC. A low AUC for Deletion and a high value for Insertion is desired.

Figure 4.17a shows the average trend for Deletion on all the objects in the subset of the Test-dev set. As can be seen, when the first 5% of pixels are removed, there is a sharp drop in the similarity of about 0.3. The similarity continues to drop until around 40% of the pixels have been removed, at which point the trend begins to level off. Figure 4.17b shows the trend for the Insertion protocol. This shows that there is a sharp improvement in the similarity as the first 20% of pixels are added back in, this then levels off at between 20% to 60%, and continues to improve, albeit at a lower rate.

Since one of the more challenging aspects of aerial imagery is that objects appear small in the image, we calculated the Deletion and Insertion metrics for 'small', 'medium' and 'large' boxes. This is shown in Table 4.2, from this, we can see that the explainer achieves better scores for larger boxes. Fewer pixels make up small objects, hence, they are represented by low-resolution features which are challenging to discern.

The sharp drop in similarity during Deletion, and the sharp rise during Insertion,

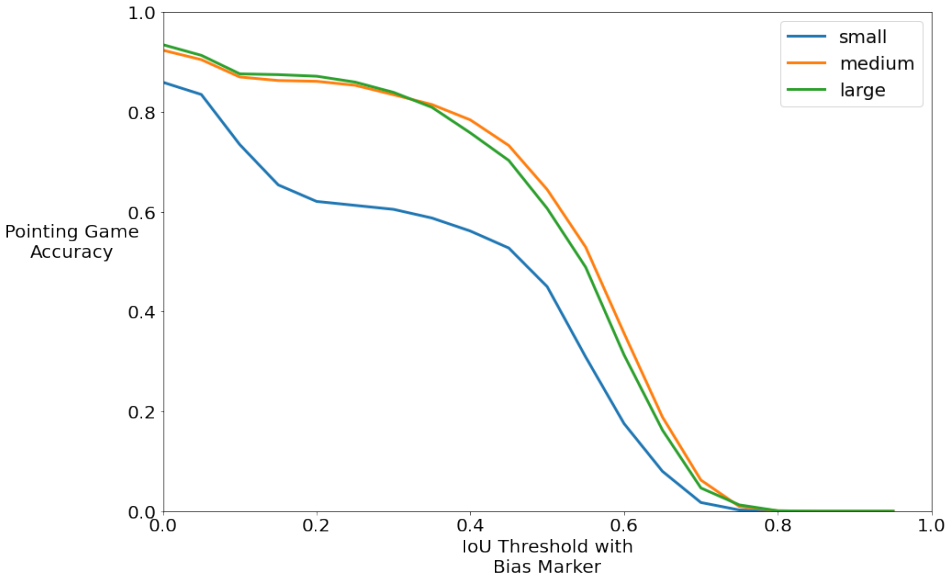
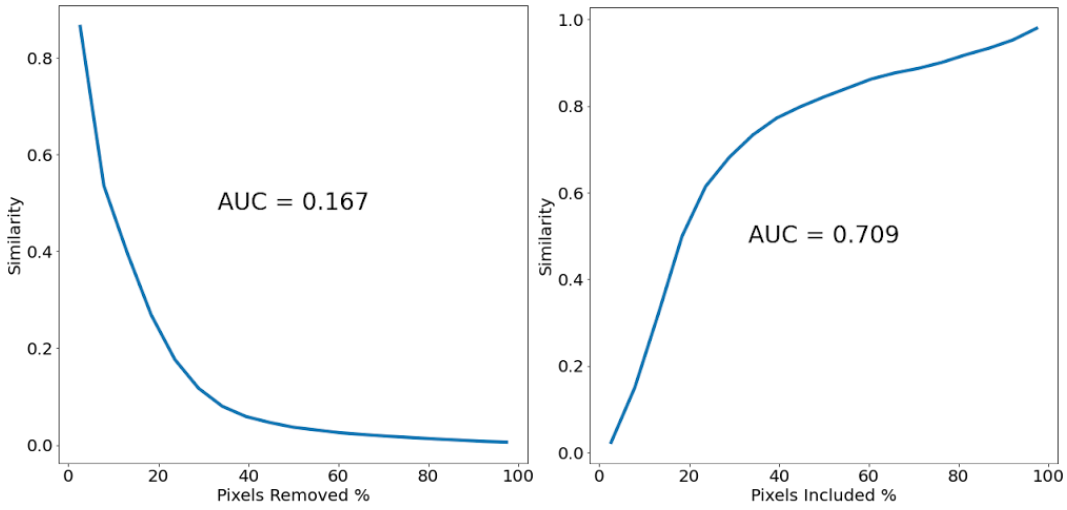


Figure 4.16: Plot showing the accuracy of the KernelSHAP framework at identifying the bias when considering different IoU thresholds, and three scales of bounding boxes



(a) Deletion plot **(b)** Insertion Plot

Figure 4.17: Plots showing the similarity in relation to (a) the amount of pixels removed from the image, and (b) the amount of pixels added back to an image.

strongly supports the assertion that the explainer can correctly allocate the contribution of the image pixels. In addition, the AUC was 0.709, which is a good score if compared to D-RISE, which achieved 0.562 reported in [84]. The AUC for deletion was 0.167, which is near the scores reported by [12] for classification explainers, but not as close to D-RISE which achieved 0.044. However, it is hard to make an avid comparison, since the investigation here is using a different, more challenging dataset, with much smaller objects than the datasets used by these other works.

Bounding box Size	Deletion	Insertion
Small	0.181	0.659
Medium	0.165	0.720
Large	0.153	0.742

Table 4.2: This table shows the Deletion and Insertion scores for different sized bounding boxes. Boxes are considered 'small' when they are under 27 pixels across, 'medium' if they are between 27 to 45 pixels, and 'large' if they are above 45 pixels.

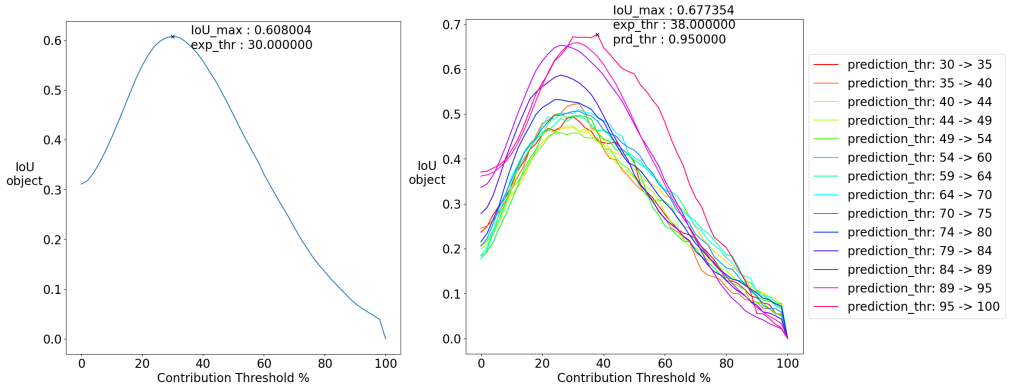
4.5.4 Wrapping Game Analysis

This final analysis in the chapter applies the Wrapping Game, a novel evaluation method introduced in Chapter 2. The Wrapping Game is designed to assess the effectiveness of the explainer in distinguishing the object of interest from background clutter. The primary purpose of this analysis is to evaluate how discriminative the explainer is when isolating relevant objects, as discriminative explanations are more valuable to human users. Additionally, the analysis investigates how the model's confidence levels affect the descriptiveness of the explainer, providing insight into how well the explainer performs under different predictive conditions.

For this evaluation, two deep networks that were trained on the proprietary XAI-AV dataset introduced in Chapter 2, are analysed - YOLOv5 and Faster R-CNN. The outcomes from YOLOv5 and Faster R-CNN are shown in Fig. 4.18 and Fig. 4.19. From the shape of the plots and the maximum IoU scores in Fig. 4.18a and Fig. 4.19, it can be seen that the explainer performs very similarly on both networks. This consistent performance across different network architectures suggests that the explainer is well-suited for comparative evaluations of various deep networks, providing reliable insights regardless of architectural differences.

Notably in Fig. 4.18b, the plots reveal considerable volatility, with frequent and irregular oscillations compared to similar analyses conducted on other explainers in Fig. 2.23, Fig. 3.16b and Fig. 3.17b. This volatility likely stems from KernelSHAP's approach of treating groups of pixels as a single feature rather than as individual features, which differs from other methods discussed in previous chapters.

Despite this variability, the KernelSHAP framework demonstrates strong performance in the Wrapping Game, achieving significantly higher scores than Grad-CAM in both evaluation settings. Comparing the performance of explainer on the same YOLOv5, even the lowest score observed in the KernelSHAP evaluation in



(a) **(b)**

Figure 4.18: Plots showing the Wrapping Game analysis of the proposed KernelSHAP framework with YoloV5. (a) analysis across all instances in the dataset. (b) analysis considering the model's confidence in a given instance.

Fig. 4.18b surpassed the scores obtained from the Grad-CAM analysis shown in Fig. 3.16b. This result underscores the effectiveness of KernelSHAP in providing discriminative explanations that align well with the object of interest, reinforcing its value for applications requiring detailed, interpretable insights.

4.6 Summary

This chapter introduced a novel adaptation of KernelSHAP, enabling its use for generating local explanations in detection-style algorithms, independent of architecture. The effectiveness of this approach has been demonstrated in identifying visual biases within detection algorithms, as well as in accurately assigning feature contributions through Deletion and Insertion protocols. This explainer provides developers with a reliable tool for understanding the limitations of their algorithms, aiding in the planning of necessary safeguards.

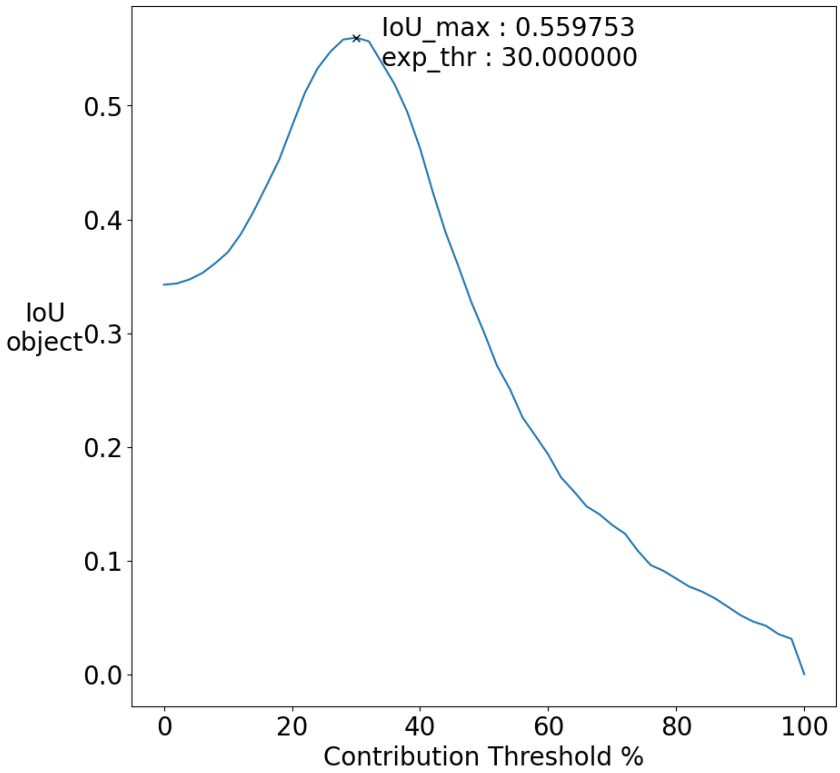


Figure 4.19: Plot showing the Wrapping Game analysis of the proposed KernelSHAP framework, with Faster R-CNN for all instances in the dataset.

Chapter 5

DetDSHAP: Explainable Object Detection for Uncrewed and Autonomous Platforms with Shapley Values

This chapter presents DetDSHAP, a method for quickly approximating the SHAP values that elucidate the casual factors behind a deep object detector's decision making. DetDSHAP can calculate the contribution of every unit of a deep detector to its detection via backpropagation. In addition, the contribution scores produced by DetDSHAP are used in a novel pruning technique to achieve high performance with reduced computational complexity. Finally, the performance results of the DetDSHAP as an explainer and effective pruning criterion is demonstrated on multiple dataset including real data collected from drone-based cameras and publicly benchmark datasets.

5.1 Introduction and Motivation

This chapter introduces DetDSHAP, a novel method for rapidly approximating SHAP values to elucidate the causal factors behind a deep object detector's decisions. DetDSHAP generates detailed "SHAP maps" that explain both the classification and localisation of detected objects by efficiently propagating contribution scores via a DeepSHAP-inspired backpropagation procedure. Notably, the approach

not only provides high-fidelity explanations but also serves as an effective pruning criterion. By quantifying the contribution of individual network units, DetDSHAP enables the systematic removal of redundant components, thereby enhancing computational efficiency without significant performance loss.

The explanation methods introduced in Chapters 3 and 4 provided different levels of fidelity. DetDSHAP offers distinct advantages over the previously introduced Grad-CAM and KernelSHAP approaches. While Grad-CAM provided useful insights, its explanations tended to be somewhat blurry and lacked the fine-grained detail necessary for interpreting complex object detection tasks. KernelSHAP, though more precise, was computationally expensive and relied on SLIC segmentation as a preprocessing step, grouping pixels in a way that resulted in a loss of detail. Furthermore, neither approach evaluates the entire deep network to assess the contribution of individual units, rendering them unsuitable for use as pruning criteria. In contrast, DetDSHAP not only generates more detailed explanations but also measures the contribution of individual units across the network, thereby serving as an effective pruning criterion. As in Chapter 4, the term "SHAP maps" is employed to describe the visual representations of feature importance produced by these explainability frameworks.

Furthermore, the decision to continue using YOLOv5 throughout this thesis is intentional. Despite the advent of more modern YOLO versions, YOLOv5 remains a robust and widely adopted detection framework with well-documented performance. Maintaining consistency by using YOLOv5 across Chapters 3, 4, and this chapter enables a coherent comparison of explainers. Moreover, the ensemble of explainers proposed in this thesis—combining gradient-based, perturbation-based, and DetDSHAP methodologies—offers enhanced resilience against adversarial attacks, as the complementary strengths of these methods can help mitigate the vulnerabilities of any single approach [9].

A major obstacle to the advancement of XAI is the absence of established benchmarks and a lack of consensus within the academic literature regarding the evaluation of novel explanation techniques. Numerous studies assess the effectiveness of their methods by comparing the generated explanations with human-annotated bounding boxes [12, 16, 84, 115]. Here, the following axiom is surmised: a propagation-based XAI method that offers a robust criterion for pruning must also allocate causal information correctly. Specifically, the contribution scores de-

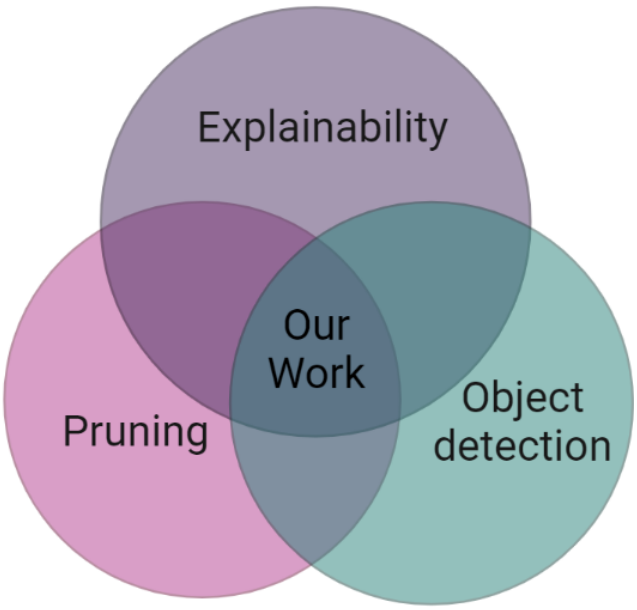


Figure 5.1: The domain of the present work. While literature exists on one or the intersection of two domains represented by each circle, this study introduces a connection between all three.

rived from the DetDSHAP framework not only serve as an effective pruning criterion but also validate that DetDSHAP accurately allocates causal information.

The intention is to propose a solution for object detection that emphasises both explainability and pruning, addressing the diverse requirements of drones functioning in urban settings. This initiative is motivated by the current deficiency of explainable algorithms capable of achieving state-of-the-art performance across a wide array of datasets [160]. To that end, the explainability and pruning techniques are evaluated using two datasets derived from different drone platforms. Additionally, an evaluation on the COCO2017 [161] dataset is included to facilitate comparisons with future research endeavours.

Previous studies have introduced explainers specifically designed for deep object detectors, and other literature has suggested the application of XAI as a criterion for pruning in a broader context. However, there is a notable absence of research that has explored the convergence of these two areas. Consequently, this chapter aims to contribute to the field by examining this intersection (Fig. 5.1). By addressing the dual challenges of interpretability and resource constraints, DetDSHAP offers a robust framework that bridges the gap between explainability and model optimisation. The following sections detail the methodology behind DetDSHAP—including the computation and aggregation of SHAP values—and present experimental results on various datasets, demonstrating its efficacy in both explaining deep object

detections and guiding effective pruning of the underlying network.

The main contributions of this chapter are as follows:

- A new explainable artificial intelligence (XAI) framework for object detection has been developed, aimed at querying bounding boxes that were either predicted or overlooked by the chosen YOLO-based detector architectures.

- A presentation of the DetDSHAP framework for accurately attributing the contributions of individual neurons within the DNN and the corresponding input image. This presentation includes DNNs trained on the proprietary self-driving car dataset, as well as the VisDrone and COCO2017 public datasets.

- A novel framework for pruning DNN detectors utilising the contribution metrics provided by DetDSHAP as a criterion. This approach significantly enhances the efficiency of the YOLOv5 detector while maintaining minimal performance degradation.

- A comprehensive quantitative examination of the impact of the proposed methodology on the efficiency and quality of the deep object detection system.

5.2 Related Work

This section explores the current landscape of explainability and pruning in deep object detection, focusing on methods that aim to generate interpretable saliency maps and utilise explainability as a pruning criterion. This discussion expands on the works introduced in Chapter 1 by providing a deeper understanding of the typical deep network pruning framework and reviewing studies that have attempted pruning guided via XAI, which are particularly relevant to the work conducted in this chapter. The framework developed herein is compared closely to Layer-wise Relevance Propagation (LRP), a propagation-based explainer that has been applied to detection-style architectures. As discussed in Chapter 1, LRP has been applied to detection-style architectures in [73] and [106]. In [73], the authors proposed Contrastive Relevance Propagation (CRP) to provide explanations for detections made by SSD models. Because SSD models are generally weaker than more modern detectors [71, 106], Karasmanoglou et al. [106] expanded on this work to develop an

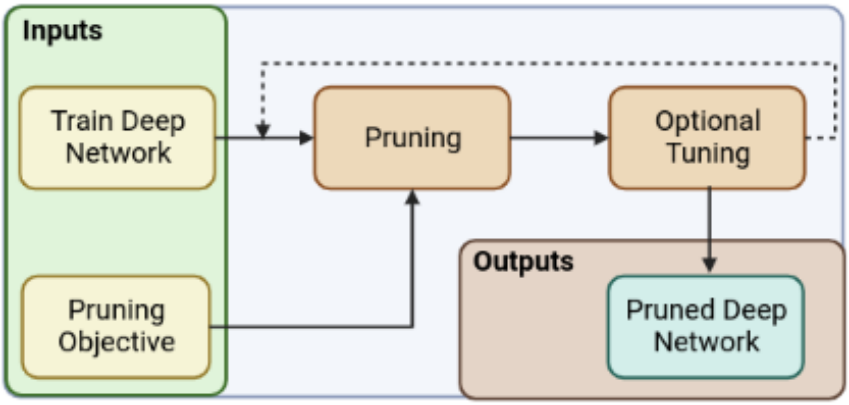


Figure 5.2: Generalised pruning framework [162]

LRP framework for YOLOv5. Although both methods produce visually appealing saliency maps, neither formally quantifies the fidelity of the explanations, as emphasised in [106]. Moreover, as a gradient-based method, LRP is susceptible to model saturation [96], whereby the importance of certain features is underestimated, leading to uncertainty in the explanations.

In [162], a comprehensive review of existing pruning techniques was conducted, culminating in the proposal of a generalised pruning framework, as illustrated in Fig. 5.2. The authors identify four essential components within such a framework: model training, objective selection, the pruning process, and subsequent tuning. During the training phase, the foundational network architecture f is optimised by adjusting its weights W until convergence is achieved. Objective selection then involves determining the criteria for halting the training process—such as achieving a desired level of model compression, decreasing inference time, or maintaining an acceptable threshold for performance loss. Within the set of weights W, a subset of redundant components, denoted as w, is identified to fulfil these objectives. In the pruning phase, these redundant components are eliminated to meet the specified pruning goals. The methodologies for pruning differ mainly in their strategies for selecting components to prune, the criteria used, the scheduling of the pruning process, and the decision to implement fine-tuning afterward.

Yeom et al.[163] introduced the use of LRP as a criterion for model pruning. Their research validates this approach across multiple scenarios, offering valuable insights into the pruning of deep networks, albeit with a focus primarily on classification tasks. The authors demonstrated that the LRP criterion can achieve enhanced compression performance across various datasets in comparison to conven-

tional methods. However, it is important to note that LRP may exhibit tendencies toward over-saturation, which is not ideal for a pruning criterion. Furthermore, this work presents evidence suggesting that LRP may not be appropriate for deep object detection tasks, indicating that DeepSHAP serves as a more effective criterion.

An alternative approach to LRP is proposed by Sabih et al. [160] , who advocate for the utilisation of DeepLIFT (Deep Learning Important FeaTures) as a criterion for model pruning. DeepLIFT, developed by Shrikumar et al.[96], addresses the challenge of model saturation that affects LRP and other gradient-based techniques. However, DeepLIFT relies on comparing each neuron's activation to a reference activation. Establishing a reference point for DeepLIFT poses challenges due to its significant impact on the quality of the generated explanations. This particular trait of DeepLIFT somewhat undermines the necessity for the explainer to maintain consistency, potentially leading to failures when it encounters unfamiliar data.

Both, Sabih et al.[160] and Yeom et al. [163] implement their criteria within a global pruning framework. In contrast, Lundstrom et al.[164] restricts pruning to the dense layers of their network, where the majority of parameters are concentrated. Lundstrom et al. utilise Integrated Gradients as a criterion for pruning. The target network employed in their study is AlexNet [30], which has been trained on a dataset derived from the Mars Rover [165].

Unlike global pruning, local pruning involves applying pruning techniques to specific sections of the model, such as individual layers or uniformly across each layer, which generally results in less effective pruning outcomes [166], [167]. The findings of Lundstrom et al. [164] indicate that critical neurons tend to specialise in particular classes. This specialisation likely explains why Yeom et al. [163] observed that employing positive relevance to determine their criterion yielded the most favourable results.

The primary objective of pruning is typically to facilitate the compression of deep networks for deployment on hardware systems. Nevertheless, the studies discussed in this section do not adequately assess their criteria within a practical context involving portable robotic platforms. Only the work by [164] utilises a dataset obtained from a machine vision platform, yet it employs an outdated architecture. The framework proposed in this chapter is destinguished from the established work from demonstrating prune deep object detectors using realistic data collected from both ground-based and aerial drones.

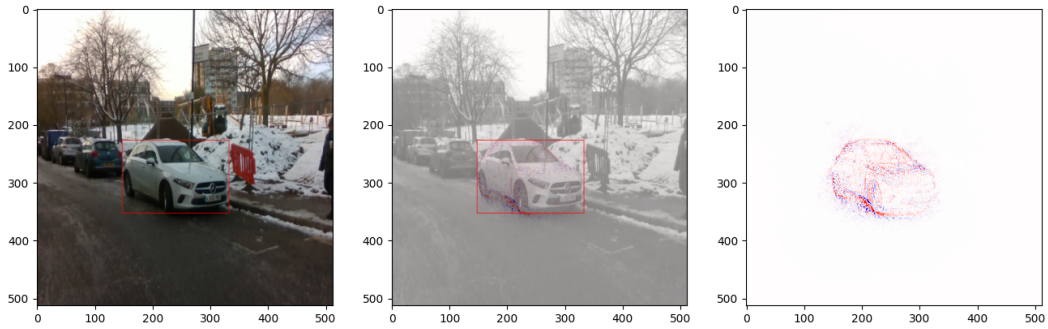


Figure 5.3: Example of SHAP map generated using the approach introduced in this work with an image in the proprietary dataset. (L) The input image with the predicted box. (C) The SHAP map superimposed over the input image. (R) The SHAP map and the colour bar.

5.3 Methodology

This section outlines the primary principles of this chapter: Firstly, DetDSHAP is proposed, an explanatory model designed to generate SHAP values, which elucidate the causal information associated with individual bounding boxes generated by architectures based on the YOLOv5 detector. Secondly, this explanatory model is utilised to develop an innovative pruning framework aimed at optimising our trained models.

5.3.1 DeepSHAP for Object Detection (DetDSHAP)

This section presents DetDSHAP as an explanation method for Deep Object Detectors. When provided with an image I, a deep detection network f, and a target bounding box T, the explainer generates a SHAP map that illustrates the relationship between I and T as interpreted by f. Notably, the bounding box T is not restricted to those predicted by the model; it can also be the groundtruth bounding box or any hypothetical box specified by the developer. This flexibility significantly enhances the developer's ability to investigate and analyse their network.

Algorithm 2 provides a comprehensive overview of the DetDSHAP algorithm, which is structured into three primary phases. The initial phase involves executing a forward pass through the detector deep network, during which the forward activations are recorded. The second phase entails the initialisation of the forward pass output to eliminate information that is irrelevant to T, a process that is elaborated upon in this section. The concluding phase, which results in the generation of the SHAP map, consists of a backward pass that utilises the backpropagation rule set of DeepSHAP. Additionally, this framework is enhanced with a proprietary

backpropagation rule, which will be detailed later in this section..

Algorithm 2 DetDSHAP Framework

Input: Fully trained deep network F, Target bounding box T, Image I.
Output: SHAP Map ϕ
  Step 1: Forward Pass
    $x_i = I$
    for layer in F **do**
      $x_i = layer(x_{i-1})$
      if layer is type activation **then**
        $save(x_{i-1}, x_i)$
      end if
    end for
  Step 2: Initialise prediction based on T
    $\phi^i = Initialise(x_i, T)$
  Step 3: Backwards pass
    for layer in $reversed(F)$ **do**
      $\phi^i = apply_rule(layer, \phi^i)$
    end for
  $\phi = \phi^i$
  return ϕ

Object detectors differ from image classifiers in that they must identify and localise one or more objects within an image, rather than merely assigning a class score. In the context of YOLOv5, the task T encompasses a class score c, a bounding box represented as $b = [x, y, w, h]$, and an objectness score o. To refine the prediction P generated by the function f, it is necessary to preprocess this information to eliminate any irrelevant data, which is categorised as attribution noise. This preprocessing phase is referred to as the initialisation step, resulting in the initial contribution denoted as ϕ^i. Here, two approaches are applied depending if the task is a single-class detection or multi-class detection problem.

In the context of a single-class detector, the primary objective is to ascertain the elements that influence the predicted bounding box. To achieve this, we begin by initialising P through the selection of boxes that exhibit a high degree of similarity to the target box. The similarity between the target box T_{box} and the predicted box P_{box} is quantified through a specific calculation of IoU (Eq. (5.1)).

The initial contribution of each box, denoted as ϕ^i_j, is assigned using Eq. (5.2),with the threshold parameter thr ranging from 0 to 1. In this work, $thr = 0.7$ was found to be effective in the majority of scenarios. The initial contribution for the P is then calculated by Eq. (5.3), where k is the number of predictions. Subsequently, the initial contribution for P is determined through Eq. (5.3), where k represents

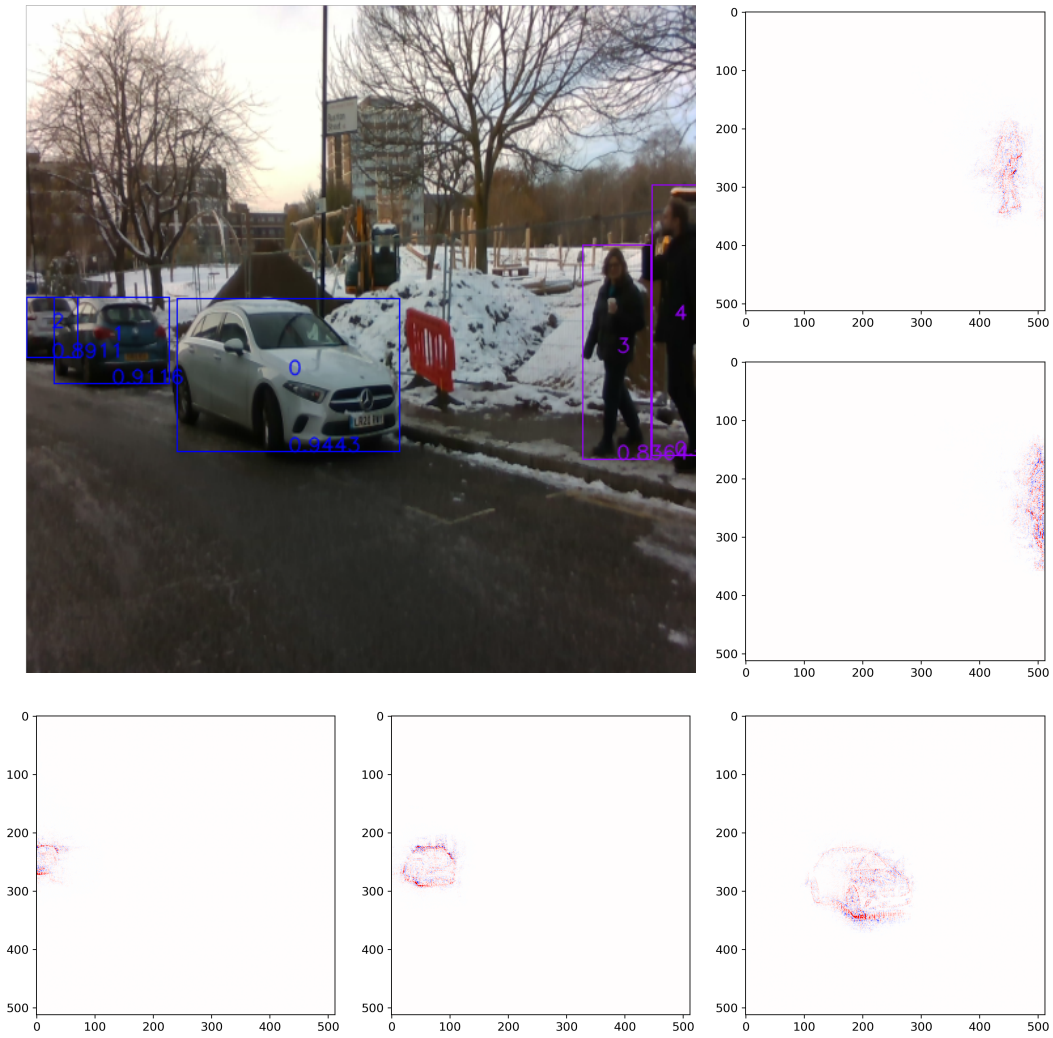
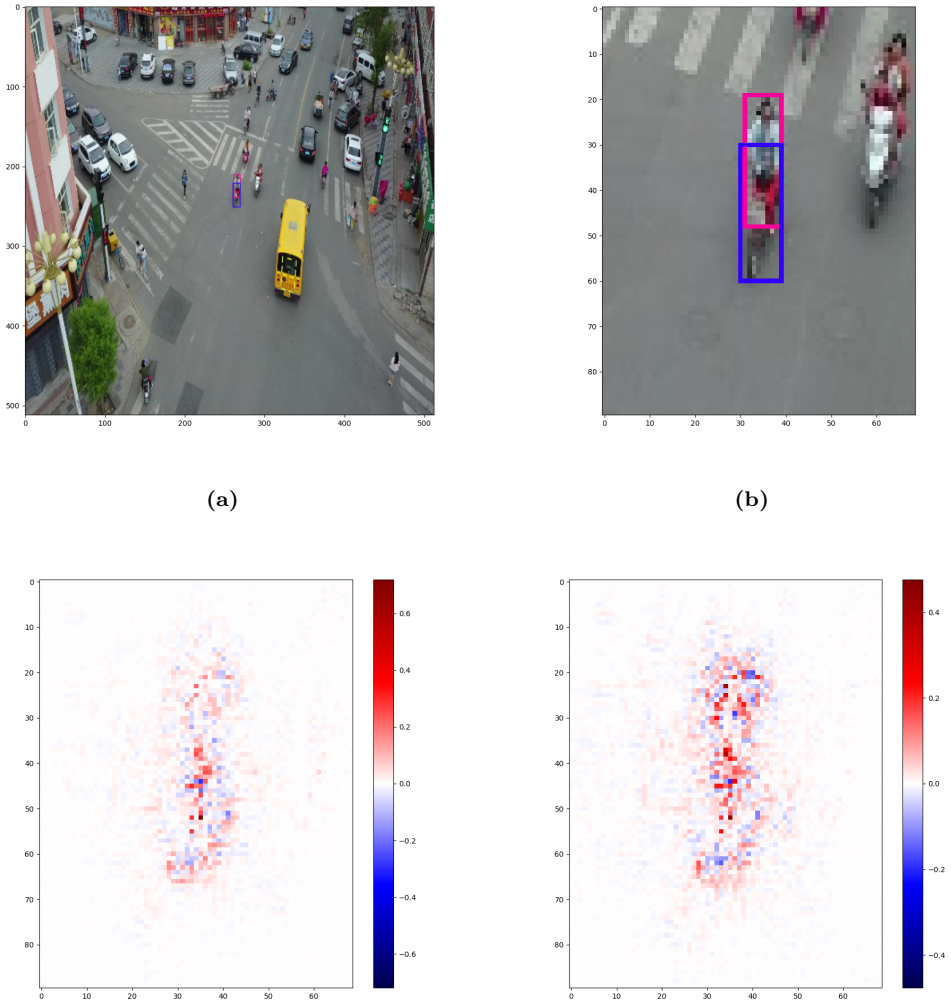


Figure 5.4: This figure illustrates the SHAP maps that can be generated for individual objects in the input image. The maps for the vehicles are shown along the bottom and the maps for the Pedestrians are shown along the right side.

the total number of predictions. The resulting SHAP values illustrate the units that significantly impact the localisation of T, as exemplified in Fig. 5.3. There is also provided an example of multiple SHAP maps generated by the proposed methodology in Fig. 5.4.

$$IoU(T_{box}, B_{box}) = \frac{Area\ of\ Overlap\ of\ (T_{box}, B_{box})}{Area\ of\ Union\ of\ (T_{box}, B_{box})} \tag{5.1}$$

$$\phi_j^i = \begin{cases} P_j, & IoU(T_{box}, P_{(j,box)}) \geq thr \\ 0, & otherwise \end{cases} \tag{5.2}$$



(a) **(b)**

(c) **(d)**

Figure 5.5: An example of multi-class explanation. (a) Shows the image with the target bounding boxes of a person and a motorcycle. (b) Shows the area of interest around the bounding box, (c) The SHAP map generated for the motorcycle, (d) map for the rider.

$$\phi^i = \sum_{j=0}^{k} \phi_j^i \tag{5.3}$$

For multi-class applications the initialisation needs to also account for the predicted class scores. The same steps that are applied in a single class method to localise the SHAP calculation are applied for multi-class applications. Following this, the objectness score and the score for the target class to one. The resulting SHAP values represent units that are not only heavily influencing the localisation of T, but also of the class that T belongs to.

Figure 5.5 illustrates an instance of the SHAP maps produced through this

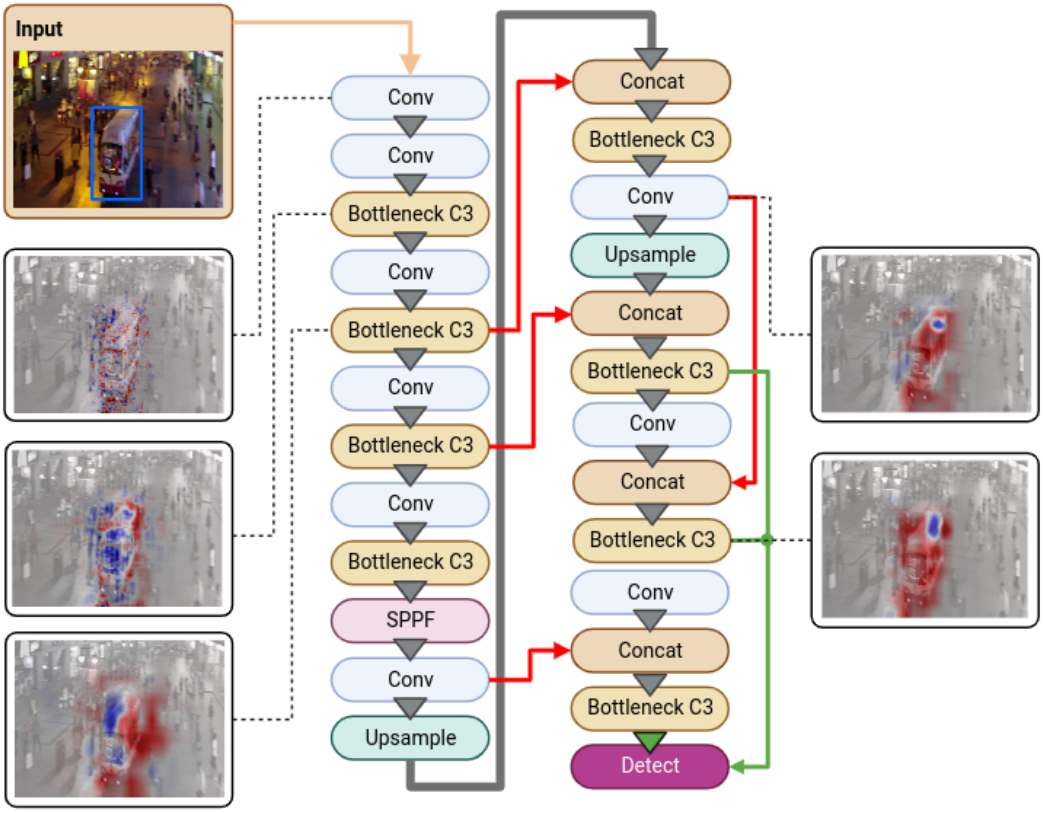


Figure 5.6: This figure shows an overview of the YOLOv5 architecture. The figure also shows various SHAP maps extracted at different layers.

methodology. This particular example demonstrates the explainer's ability to differentiate between a rider and their motorcycle. In Figure Figure 5.5c, the map pertains to the "motorcycle" class, highlighting that the focus of attention is primarily on the vehicle. Conversely, in Figure Figure 5.5d, which corresponds to the "person" class, it is evident that the silhouette of the rider plays a more significant role in the detection process.

After the initialisation phase, it is possible to perform backpropagation utilising the rules established by DeepSHAP. In the course of backpropagation, the influence of each unit in the network is assessed. Should the user focus solely on the input's contribution to the prediction, they have the option to omit this data to save memory. Nevertheless, this information can be leveraged to illustrate the attention of specific layers, as demonstrated in Fig. 5.6.

YOLO-based detection networks generate predictions of offsets relative to predefined anchor boxes, which may be either positive or negative. In instances where an offset is negative, a higher value of ϕ should be assigned to the weights that influence the negativity of that offset. To address this requirement, a specific rule is proposed

for implementation in the final convolutional layers preceding the detection layer, as illustrated by the green arrows in Fig. 5.6.

YOLO-based detection networks predict offsets from predefined anchor boxes, which can be positive or negative. Hence, in a situation where one of the offsets is negative it would be prudent to apply a greater contribution is applied to weights that contribute to the negativity of that offset. Here, a unique rule is proposed which is applied to the final convolutional layers before the detection layer, as indicated by the green arrows in Fig. 5.6.

The rule under consideration is articulated in Eq. (5.4), drawing inspiration from the LRP-*ab* rule as outlined by [102]. In this equation, the term $a_j \cdot w_{jk}$ represents the degree to which filter j influences the relevance of filter k. The expression $a_j \cdot w_{jk}^{+}$ corresponds to the positive contribution, while $a_j \cdot w_{jk}^{-}$ indicates the negative contribution. The denominator normalizes the relevance assignment, ensuring that the total amount of relevance remains conserved according to the LRP framework.

The variable ϕ_k denotes the relevance score assigned to neuron k in the next layer, which is propagated backward. The rule distinguishes between positive and negative activations of neuron k (denoted by a_k). When $a_k \geq 0$, relevance is redistributed according to positive contributions, while for $a_k < 0$, relevance is redistributed according to negative contributions.

The summation term $\sum_{0,j}$ ensures proper normalization across all neurons contributing to k, thereby preserving the conservation principle of LRP. Finally, $T\phi_j$ represents the transformed relevance assigned to neuron j after redistribution.

$$T\phi_j = \begin{cases} \sum_k \frac{a_j \cdot w_{jk}^{+}}{\sum_{0,j} a_j \cdot w_{jk}^{+}} \cdot \phi_k, & a_k \geq 0 \\ \sum_k \frac{a_j \cdot w_{jk}^{-}}{\sum_{0,j} a_j \cdot w_{jk}^{-}} \cdot \phi_k, & a_k < 0 \end{cases} \tag{5.4}$$

5.3.2 DetDSHAP Pruning Framework

In this section, an innovative framework is proposed, designed for the pruning of the YOLOv5 detector architecture. This pruning framework is intended to prune the architecture of the detector rather than pruning the explainer that will be applied to the pruned network. Here, the emphasis is on the global pruning of individual filters within the network. The generalisation of the framework is shown in Algorithm 3. Throughout the pruning procedure, the model is systematically reduce by a

specified quantity until a designated pruning objective, denoted as O, is achieved. Each iteration in this process is termed a pruning step, during which filters are evaluated and ranked based on our established criterion, leading to the removal of those with the lowest rankings. Following each pruning step, the model undergoes fine-tuning to recover any potential decline in performance.

Algorithm 3 Pruning Framework

Input: Fully trained deep network F, training data x, pruning objective O , magnitude of pruning step r.
Output: Optimised deep network F
  for batch b in x **do**
    Step 1: generate SHAP values for this sample
      for x_i in b **do**
        compute SHAP ϕ_i *values w.r.t* x_i *for* F
      end for
    Step 2: Identify redundant units and remove them:
      Compute filter rank $\phi_b = \sum_{i=0}^{k} |\phi_i|$, $k = len(b)$.
      Remove r *lowest contributing units from* F
    Step 3: fine-tuning F on x *fine tune* F *on* x
    Step 4: Evaluate model - determine if more pruning is desired.
      if O is True **then**
        break
      end if
  end for
  return F

In accordance with the pruning criterion, the contribution scores generated are by the DetDSHAP explainer, as detailed in Section 5.3.1. Consequently, at the initiation of the pruning process, a batch of samples are selected from the dataset and their SHAP values of a batch of samples (b) are calculated. Each sample consists of an image paired with its corresponding label, which encompasses the 2D coordinates and class labels for all objects present in the image. Although this label typically reflects the model's prediction, the benefit of the approach used here is that the groundtruth label can be used to calculate the SHAP values. This capability enhances the proposed framework, enabling the pruning process to be based on the groundtruth rather than the model's predictions, which is a common limitation in conventional pruning methods.

After generating the SHAP values for each image within a batch, the subsequent task involves determining the rankings of the filters. Research indicates that neurons exhibit a tendency to specialise, suggesting that the negative SHAP values associated with one instance may also be relevant to a nearby instance. Consequently, filters

are ranked according to the magnitude of their contributions, with lower rankings assigned to those filters whose SHAP scores are nearest to zero. The ultimate filter rank, denoted as ϕ_b, is computed by aggregating the absolute SHAP values $|\phi_i|$ from each individual sample i.

After calculating the ranks of the filters, the filters that can be eliminated have been identified. In this study, structural pruning is applied to the individual filters. In the PyTorch framework, the process of pruning a single filter entails modifying a specific convolutional module by decreasing the number of output channels by one and transferring the remaining weights accordingly. This adjustment is then mirrored in the subsequent layer, where the number of input channels is similarly reduced. This task is complex, particularly because information in contemporary deep network architectures often flows in a non-linear manner, as illustrated by the red connections in Fig. 5.6. Consequently, we have developed a custom framework to identify these types of connections, enabling us to effectively restructure the network.

5.4 Experimental Results

This section presents the results of the performance analysis of our DetDSHAP-based explainer and pruning frameworks. The analysis begins by evaluating the discriminative ability of the proposed explainer using the Wrapping Game analysis and the dataset introduced in Chapter 2. After assessing DetDSHAP's effectiveness as an explainer, the focus shifts to evaluating the pruning framework. The primary goal was to assess the feasibility of a DeepSHAP-based pruning method in comparison to a previously developed LRP-based pruning approach. Given the need for computational efficiency, the experiment was designed to be low-cost. To that end, we pruned simple neural networks trained to cluster 2D points, following an experimental setup similar to that of [163]. Three distinct 2D datasets were generated for the experiment.

After demonstrating the feasibility of DeepSHAP as a pruning criterion using simple experiments on 2D datasets, the evaluation moves to the primary focus of this work: object detection. Table 5.1 illustrates the models employed in each experiment, along with the corresponding datasets on which they were trained. For the Visdrone dataset, which presents significant challenges, YOLOv5l was utilised. In contrast, the self-driving dataset required only YOLOv5s. Additionally, for the

Coco2007 dataset, a pretrained YOLOv5m sourced from [72] was implemented to ensure reproducibility. was implemented to ensure reproducibility. Fig. 5.7 presents samples of each dataset employed in this study.

The recently developed Wrapping Game is initially employed to assess the discriminatory effectiveness of the DetDSHAP explainer in comparison to the LRP-based method proposed by Karasmanoglou et al. [106]. For this specific experiment, YOLOv5s, trained on the self-driving car dataset, is exclusively utilised to reduce the computational expenses associated with preliminary analysis.

Experiments	**Model**		
	YOLOv5s	**YOLOv5l**	**YOLOv5m**
Dataset	Simple AD dataset	Visdrone	Coco2017
Wrapping Game*	✓		
Pruning Trends	✓	✓	✓
Post Pruning Performance	✓	✓	✓
Pruning Vs Design			✓

Table 5.1: This table shows the models and datasets used in our experimental evaluation. *The Wrapping Game utilises the full and final AD dataset introduced in Chapter 2.

The subsequent sub-sections examine the efficacy of the DetDSHAP-based pruning framework. Initially, trends in pruning performance are explored, illustrating the impact of increasing pruning magnitude on the quality and efficiency of the pruned deep model. For comparative analysis, the effectiveness of pruning is also assessed using the LRP-based explainer as described in [106]. Three pruned networks are selected and subjected to additional training, with their detection performance subsequently compared to that of their unpruned equivalents.

The study concludes with an examination of the deep learning-based detector networks following the pruning process. All networks utilised in this study are based on the YOLOv5 architecture and have been trained on a proprietary self-driving car dataset, as well as two publicly accessible datasets: Visdrone [62] and COCO2017 [161]. The efficiency of the deep network is assessed through two distinct metrics. The first metric involves the count of network parameters, which evaluates the extent of compression achieved by the pruning framework. The second metric pertains to the number of Floating point OPerations per Second (FLOPs) required for conducting an inference. Enhancements in both metrics are illustrated, and the



Figure 5.7: Examples from the three datasets used in the experiments: Simple AD (left), Visdrone (center), and COCO2017 (right). The COCO2017 images show objects of interest for this study.

final results are presented as detailed in Eq. (5.5).

$$improvement = 1 - \frac{Score\ on\ pruned\ Model}{Score\ on\ original\ Model} \tag{5.5}$$

In this research, the efficacy of the deep networks is assessed based on their detection capabilities. To quantitatively evaluate the performance of the pruned networks, four widely recognised metrics are employed: MAP@0.5, MAP@0.5-0.95, Average Recall (AR), and F1 Score. Initially, precision is determined using the formula presented in Eq. (3.3), where a detection is classified as a true positive if the classification is accurate and the IoU between the predicted bounding box and the groundtruth exceeds a specified threshold. Specifically, a true positive is identified when there is an IoU of 50% between the predicted box and the actual

groundtruth. Precision is subsequently averaged across all classes to derive the mean average precision (MAP). The second metric, MAP@0.5-0.95, follows a similar methodology but considers a range of IoU values between 50% and 95%. The IoU has been previously defined in Eq. (5.1). . Additionally, the average recall is reported, calculated as the ratio of true positives to the total number of groundtruth boxes. The final metric, the F1 score, is computed as outlined in Eq. (5.6), incorporating both recall and precision into its evaluation.

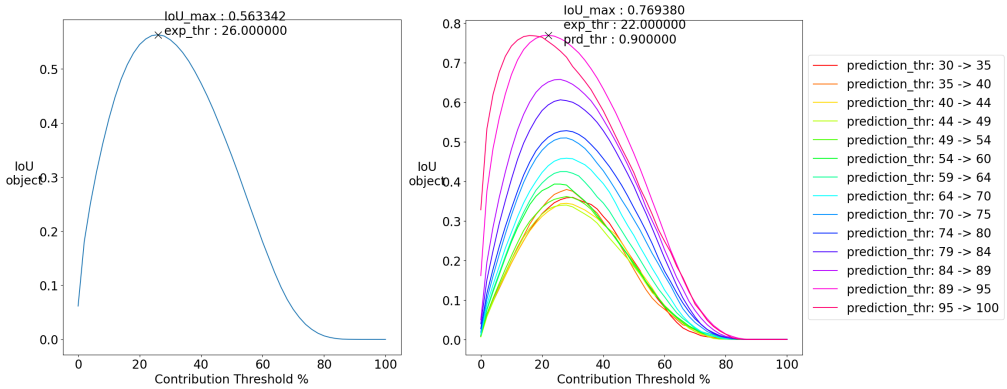
$$F1 = 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{5.6}$$

5.4.1 Wrapping Game Analysis

The DetDSHAP explainer proposed in this chapter was initially applied in Section 2.6 to demonstrate the Wrapping Game for a single-class detection setting on a subset of the XAI AD dataset. In this section, results are presented from assessing the discriminatory performance of the DetDSHAP-based explainer alongside the LRP-based explainer developed by Karasmanoglou et al. [106], designated as YOLO-LRP. Moreover, the evaluation is conducted in a multi-class detection setting using the full dataset. The conditions for this analysis are the same as those used in Sections 3.4.3 and 4.5.4, ensuring consistency, and enabling direct comparison.

For clarity, a brief reminder is available here:

- Points in the SHAP map are selected based on a contribution threshold that ranges from zero to the maximum SHAP value.
- Using the selected points, a mask is created, which is then compared to the human-annotated groundtruth to calculate the Intersection over Union (IoU) score.
- The analysis is performed using two distinct approaches:
  - Confidence-Agnostic Analysis: All detected objects are considered, regardless of model confidence, to assess the explainer's performance across all predictions.
  - Confidence-Based Analysis: This setting filters objects based on varying confidence thresholds, allowing the analysis to account for the influence of model confidence on detection quality.



(a) **(b)**

Figure 5.8: Plots showing the Wrapping Game analysis of the proposed DetDSHAP framework with YoloV5 for the Summer Set. (a) analysis across all instances in the set. (b) analysis considering the model's confidence in a given instance.

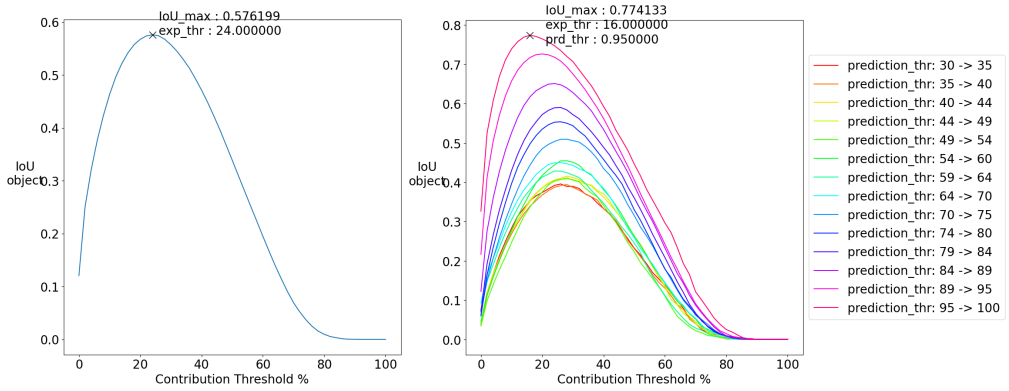
The Wrapping Game analysis of DetDSHAP reveals that the explainer's performance across the two sets was nearly indistinguishable for YOLOv5 (Fig. 5.8 and Fig. 5.9), with only minimal differences observed as a first key observation. From the Confidence-Agnostic Analysis, shown in Fig. 5.8a and Fig. 5.9a, the plots appear virtually identical, with the maximum average IoU achieved at a contribution threshold of approximately 25%. This maximum average IoU was similarly close for both sets, reaching approximately 0.585.

When comparing to the previous explainers that have been proposed in this thesis, DetDSHAP performed similarly to that of KernelSHAP, and much more superior than Grad-CAM. Indeed, the highest average IoU scored in these scenarios is almost twice that achieved by the Grad-CAM framework in the same scenarios reported in Chapter 3.

Figure 5.10 illustrates the Wrapping Game analysis for the LRP-based framework. Initially, this explainer appears superior due to its performance in the Confidence-Agnostic Analysis Fig. 5.10a. However, Fig. 5.10b, reveals that the explainer's discriminative ability diminishes significantly on instances where the model's confidence is low. This limitation is critical, as understanding the model's decision-making process becomes even more essential in low-confidence scenarios, where explanations are especially valuable.

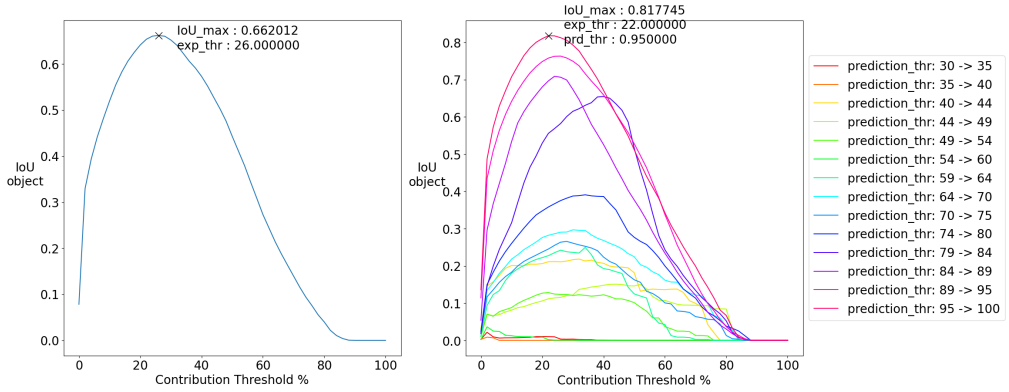
5.4.2 Pruning Toy Models

This section will discuss the findings from applying the chosen pruning criterion on toy models. The motivation here is to investigate the properties and effectiveness



(a) **(b)**

Figure 5.9: Plots showing the Wrapping Game analysis of the proposed DetDSHAP framework with YoloV5 for the Winter Set. (a) analysis across all instances in the set. (b) analysis considering the model's confidence in a given instance.



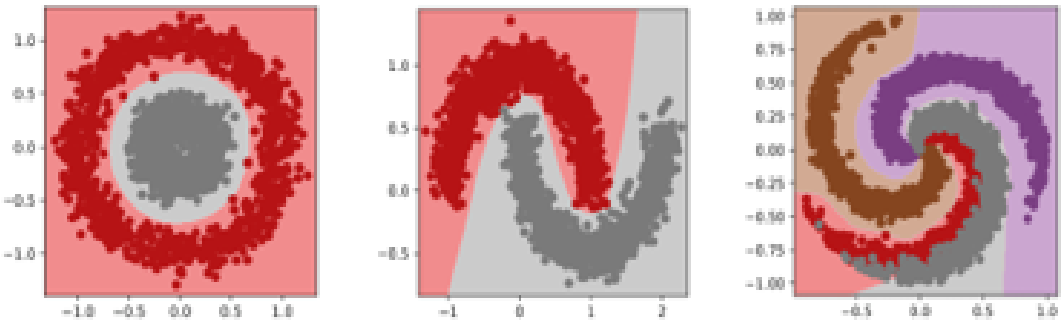
(a) **(b)**

Figure 5.10: Plots showing the Wrapping Game analysis of the proposed YOLO LRP framework with YoloV5 for the Summer Set. (a) analysis across all instances in the set. (b) analysis considering the model's confidence in a given instance.

of SHAP values as a criterion in a computationally inexpensive manner. Yeom et al[163] use this same procedure to demonstrate the superiority of their chosen criterion, LRP, over traditional pruning criterion. Here, the proposed criterion is evaluated, and the effectiveness is gauged compared to LRP as established by Yeom et al.

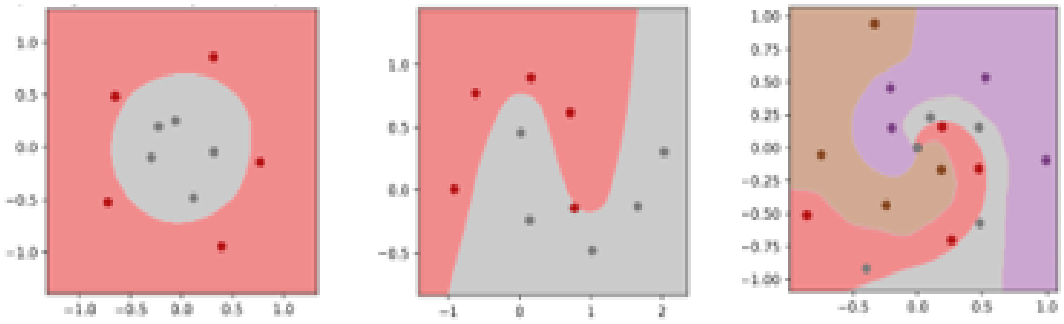
The experiment is set up similar to [163], where a simple model is trained to cluster 2D points. Three simple 2D datasets are generated, and the training and test sets for each dataset are shown in Fig. 5.11 and Fig. 5.12 respectively. The simple model is a fully connected neural network consisting of three layers with 1,000 neurons in each. The first row of Table 5.2 shows the pre-pruning class accuracy of the models on each dataset.

There is no fine tuning applied, and all the pruning is conducted in a single step.



(a) Circle **(b)** Moon **(c)** Multi

Figure 5.11: Toy training sets used to fit the simple model. The decision boundaries are superimposed over each plot.



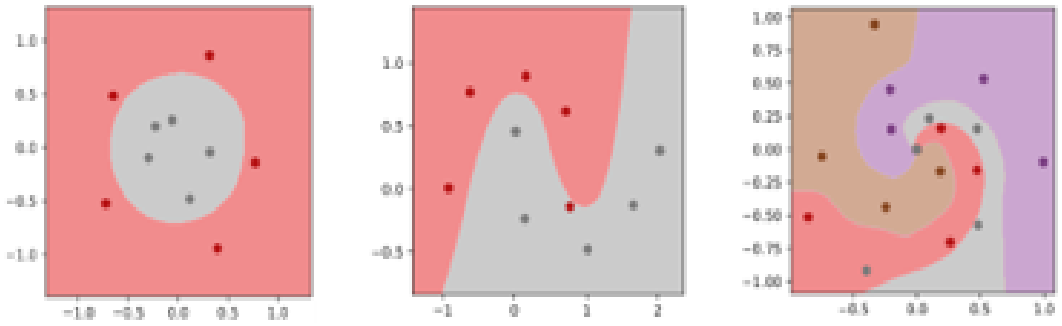
(a) Circle **(b)** Moon **(c)** Multi

Figure 5.12: Toy validation sets used to evaluate the toy models after pruning.

In [163], the author's experiments involved pruning a third of the total neurons and then inspecting the change of decision boundaries, and the classification accuracy, on both the test and validation sets. To extend the analysis in this work, a second experiment is also conducted where half of the neurons are pruned.

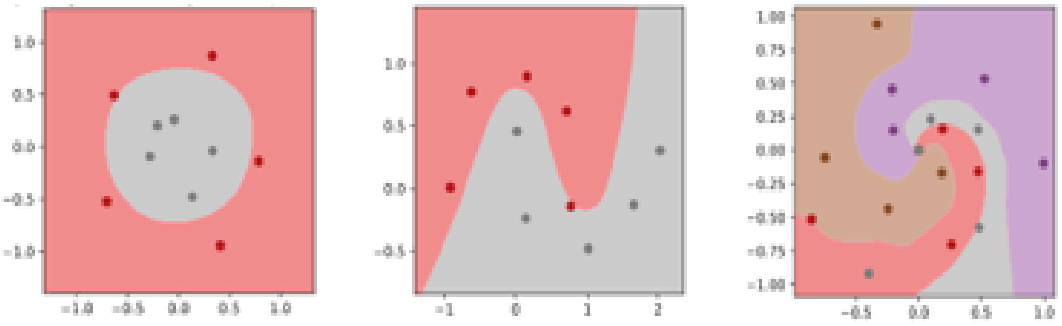
At 33% pruning, there was no significant difference in the decision boundaries of the network pruned with the SHAP criterion (Fig. 5.13) compared to LRP(Fig. 5.14). However, at 50% pruning, it was observed that the network's performance was more effectively maintained when using SHAP values as the pruning criterion. This can be seen in Fig. 5.15. On the other hand, the LRP criterion failed to maintain the network's decision boundaries, as can be seen in Fig. 5.16. The final scores are recorded in Table 5.2. The SHAP criterion performed better in all scenarios, apart from on the 33% pruning on the moon dataset. This is shown by a lower drop in the class accuracy after pruning when using the SHAP criterion.

The results suggest that SHAP values are an effective pruning criterion, preserving network performance better than LRP, particularly at higher pruning magnitudes. These findings support the viability of further developing a SHAP-based



(a) Circle **(b)** Moon **(c)** Multi

Figure 5.13: DeepSHAP pruning at 33%.



(a) Circle **(b)** Moon **(c)** Multi

Figure 5.14: LRP pruning at 33%.

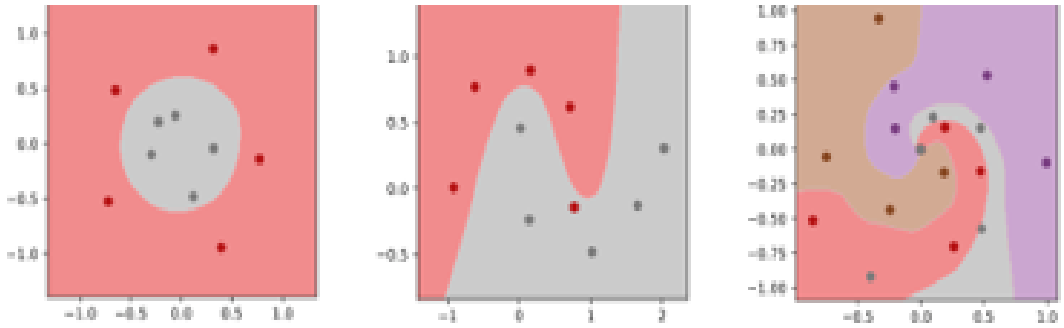
pruning framework, as it maintains accuracy with substantial model compression, offering a promising approach for efficient deep networks.

5.4.3 Pruning Performance

In this section the performance of the proposed pruning framework when using DetDSHAP as criterion is presented and compared to with using LRP as criterion. The proposed framework is used to prune various deep detector networks and the

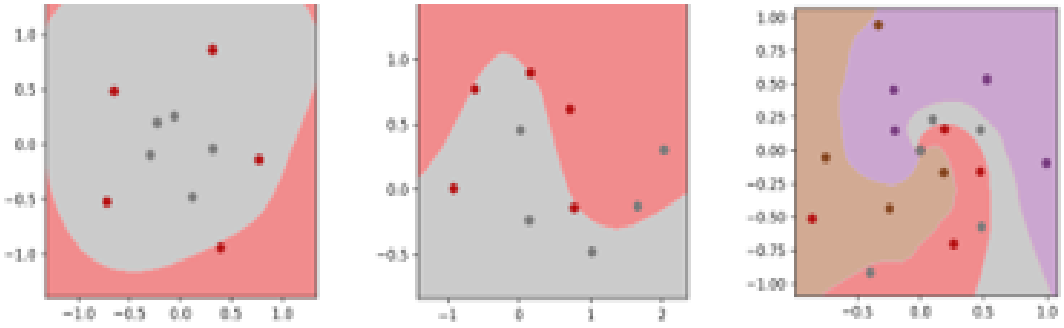
	Circle	Moon	Multi
Pre-pruning Accuracy	100%	90%	80%
33% pruning using LRP	100%	90%	75%
33% pruning using DeepSHAP	100%	90%	80%
50% pruning using LRP	60%	50%	65%
50% pruning using DeepSHAP	100%	90%	80%

Table 5.2: This table contains the accuracy achieved by the simple fully connected models trained to cluster 2D points, both before and after pruning was applied.



(a) Circle **(b)** Moon **(c)** Multi

Figure 5.15: DeepSHAP pruning at 50%.



(a) Circle **(b)** Moon **(c)** Multi

Figure 5.16: LRP pruning at 50%.

change in the performance quality against the structural efficiency are plotted. These plots are shown in Fig. 5.17 to Fig. 5.22. Each figure contains two plots, with the left plot shows the network's performance against the reduction in GFLOPS, and the right shows the network performance against the percentage of parameters pruned. When pruning deep network classifiers, the authors of [163] prune 5% of their deep network's parameters and fine-tune for 10 epochs each pruning step. Here, different parameters are chosen for each dataset to achieve the best performance for each scenario.

In the initial scenario, pruning is implemented on the YOLOv5s model, which has been trained using data collected from the self-driving car platform. During each pruning iteration, a batch of 10 samples is selected to establish the filter rankings, and 2.5% of the total filters are subsequently pruned, followed by 10 epochs of fine-tuning after each pruning step. This process is executed utilising the DetDSHAP method to derive the filter rankings. For comparative purposes, the same methodology is also applied using the LRP explainer as referenced in [106]. The results of this analysis are illustrated in Fig. 5.17 and Fig. 5.18, respectively.

The analysis indicates that the application of DetDSHAP enables the pruning

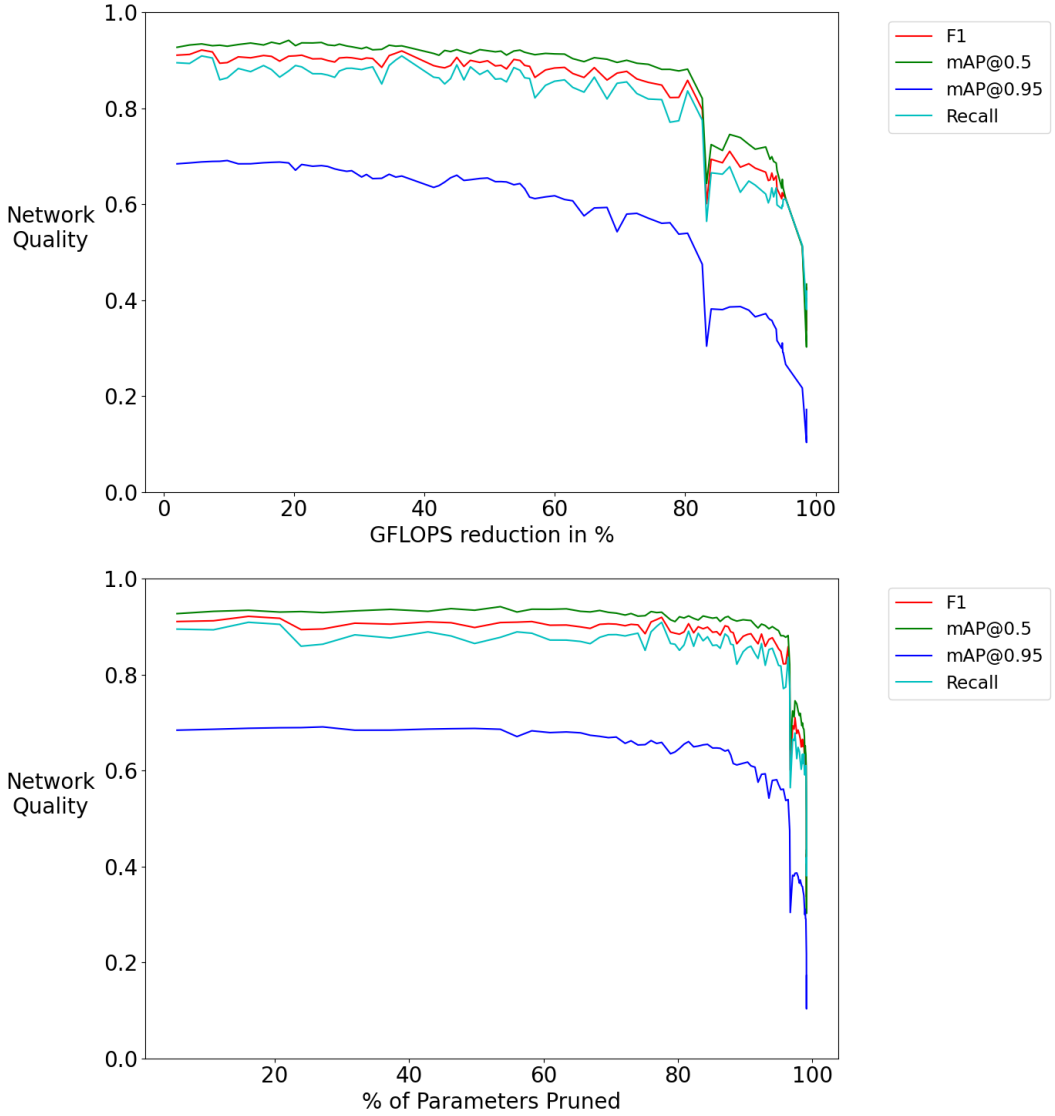


Figure 5.17: This figure displays the pruning trend when using DetDSHAP as a criterion to prune YOLOv5s trained on proprietary self-driving car dataset.

of approximately 95% of the parameters while maintaining relatively stable performance in the deep detection model. This results in a reduction of around 80% in floating point operations per second (FLOPs). In contrast, the implementation of layer-wise relevance propagation (LRP) leads to an immediate decline in performance. Although instances of performance improvement are observed—attributable to adjustments made during the pruning process—this approach ultimately fails to exceed 50% of the initial performance level. Notably, when 80% of the parameters are pruned, the model's performance is entirely compromised.

Figure 5.19 and Fig. 5.20 illustrate the results of pruning the YOLOv51 model, trained on the VisDrone dataset. Due to the substantial size of the detector network

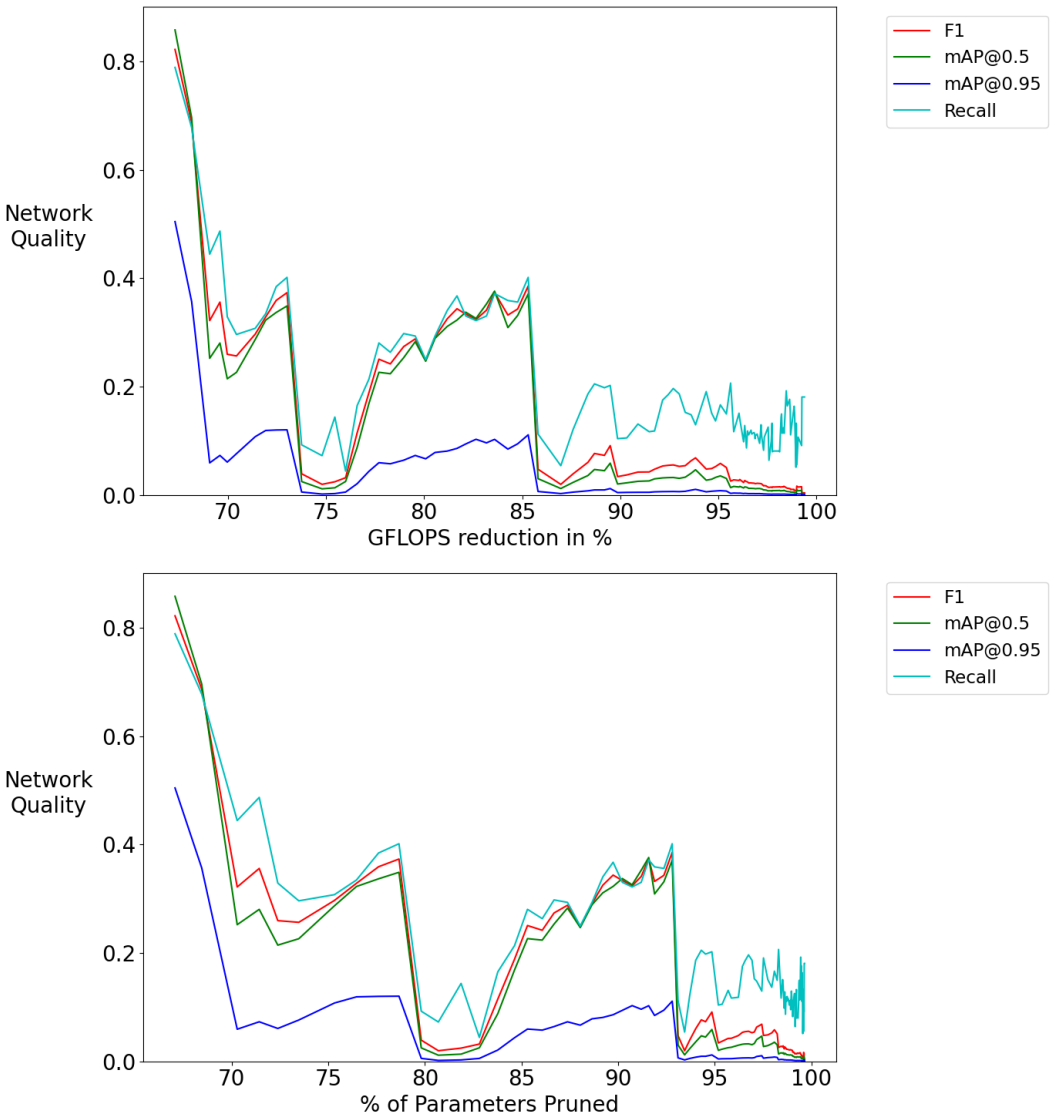


Figure 5.18: This figure displays the pruning trend when using LRP as a criterion to prune YOLOv5s trained on the proprietary self-driving car dataset.

and the variety of objects present in each image, the pruning rate was elevated to 10% for each iteration, while maintaining a consistent batch size relative to the samples. The VisDrone dataset presents a greater challenge compared to the dataset utilised in the initial scenario, as it encompasses multiple object classes and exhibits a wider range of bounding box dimensions.

It is evident from Fig. 5.19, that with DetDSHAP as a pruning criterion, more than 60% of the parameters in the deep detector can be pruned without significantly affecting performance. This results in a 22% decrease in Giga Floating Point Operations Per Second (GFLOPS). Furthermore, performance degradation remains minimal until the proportion of pruned parameters exceeds 93%, corresponding to

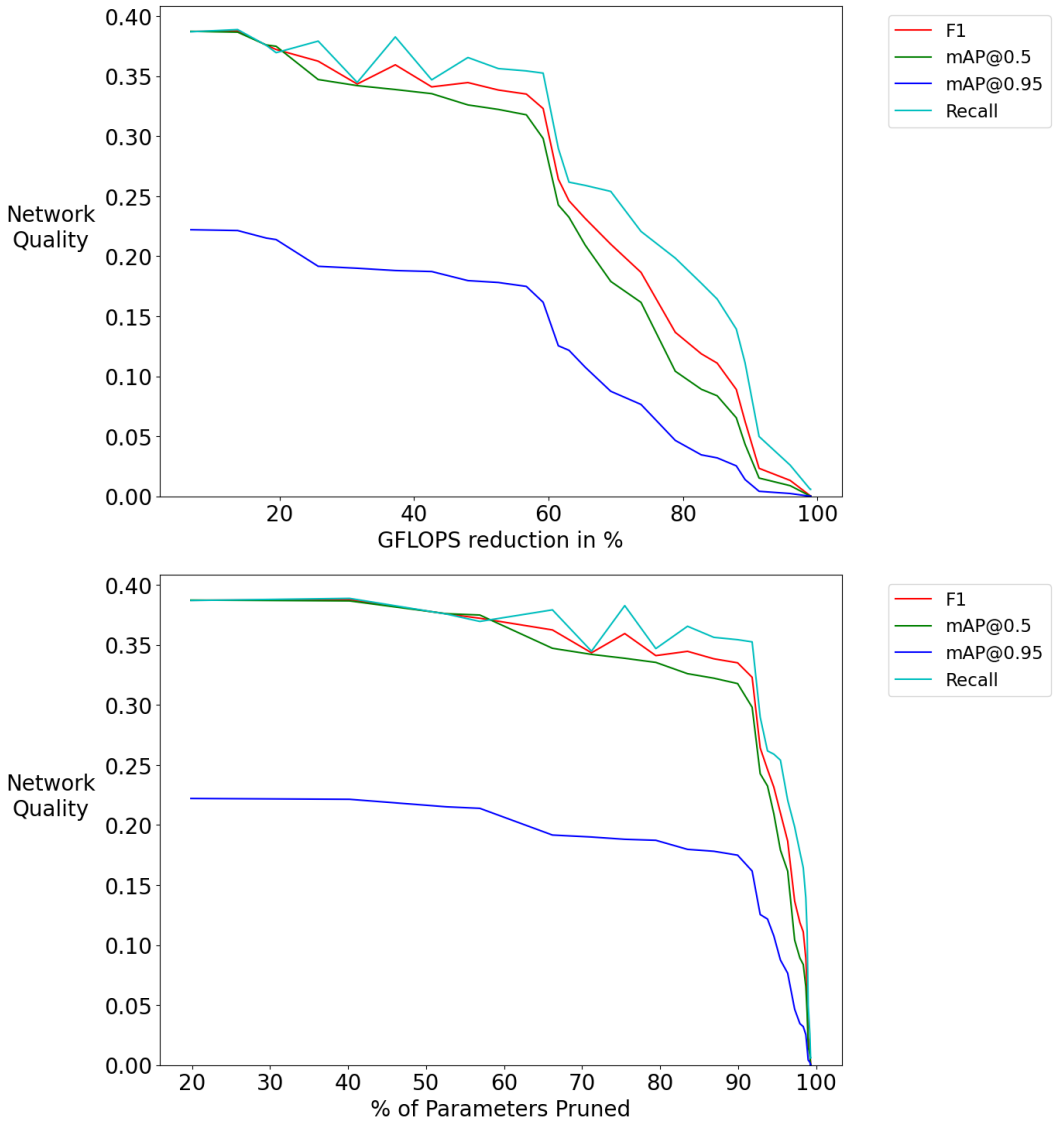


Figure 5.19: This figure displays the pruning trend when using DetDSHAP as a criterion to prune YOLOv5l trained on the Visdrone dataset.

a 60% reduction in GFLOPS.

Adopting LRP as a criterion may yield greater success in multi-class scenarios. This assertion is supported by a comparison of the trends observed in the single-class dataset (refer to Fig. 5.18) with those from the Visdrone 10-class dataset (Fig. 5.20). The findings suggest that the underlying explainer exhibits a bias towards the class score. Although LRP demonstrates enhanced performance, an initial decline in the network's quality is notable. Consequently, the overall quality throughout the subsequent pruning process remains inferior compared to when employing DetDSHAP as the criterion.

In the concluding scenario, the pruning efficacy of the DetDSHAP method is as-

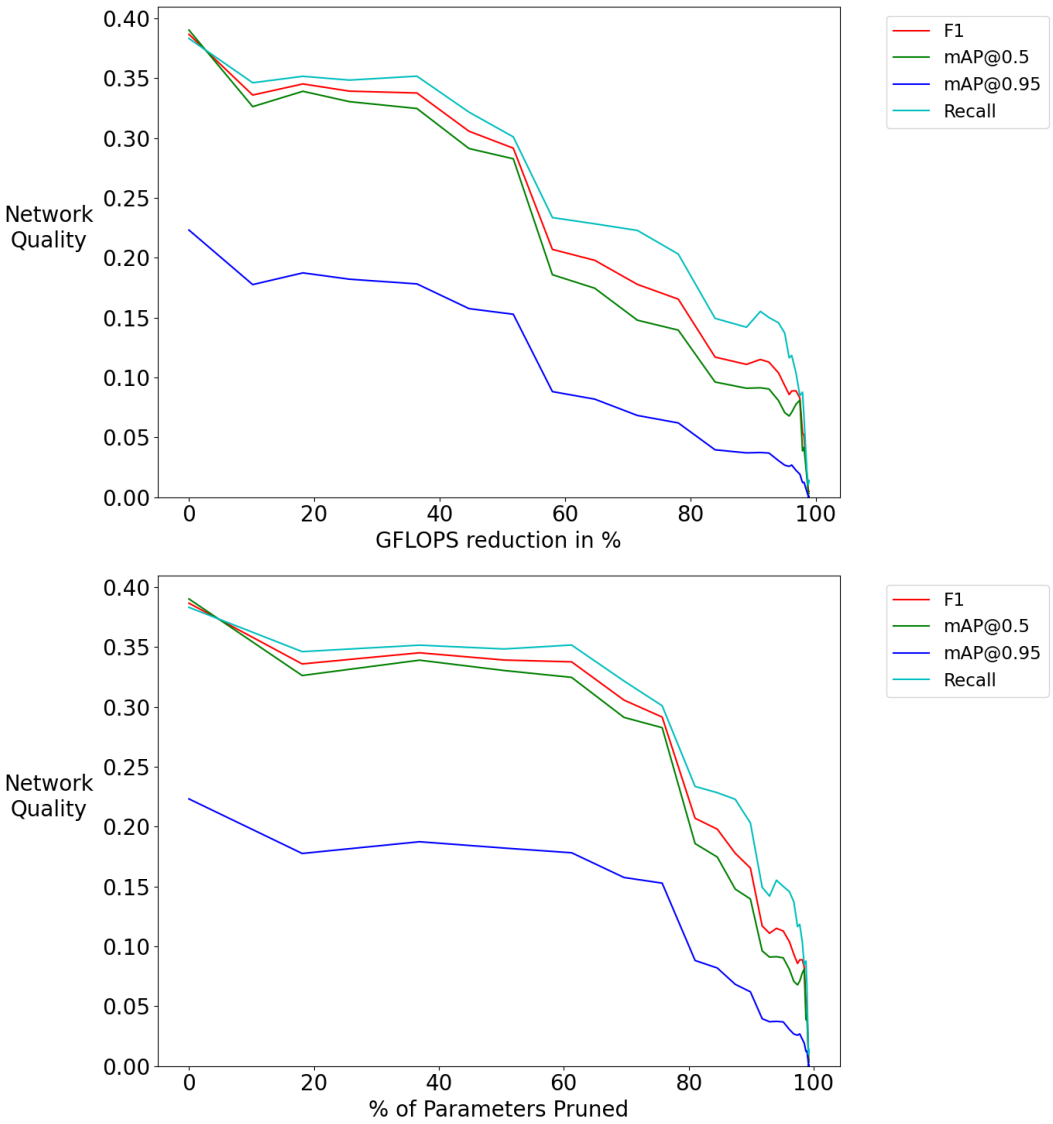


Figure 5.20: This figure displays the pruning trend when using LRP as a criterion to prune YOLOv5l trained on the Visdrone dataset.

sessed using the Coco2017 dataset. A pruning rate of 5% is applied to the YOLOv5m model, accompanied by 10 tuning epochs. To achieve a more equitable distribution of pruning across the various classes, the number of samples per pruning iteration is increased to 100. This adjustment is necessary due to the dataset comprising 80 distinct object classes, with each sample averaging 7.2 objects, many of which belong to the same class.

The trend observed in the final scenario is illustrated in the plots presented in Fig. 5.21. In this scenario, the decline in performance is more pronounced. Specifically, when 50% of the parameters have been pruned, each quality metric experiences a reduction of approximately 40%. However, it remains feasible to prune roughly

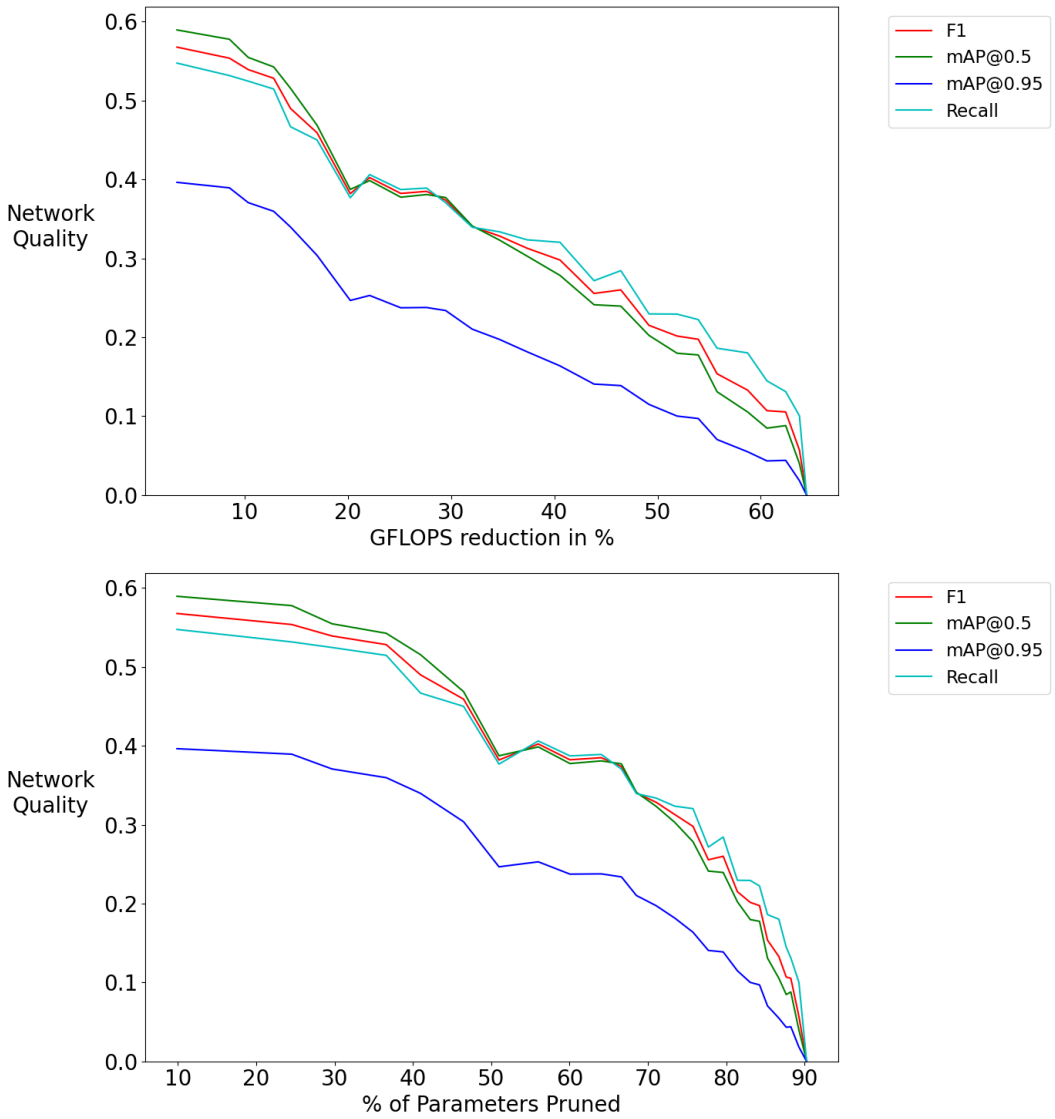


Figure 5.21: This figure displays the pruning trend when using DetDSHAP as a criterion to prune YOLOv5s trained the Coco2017 dataset.

25% of the parameters, which corresponds to a 10% enhancement in GFLOPs, while exerting minimal impact on the performance quality of the deep network detector.

Using LRP as a criterion may yield more favourable outcomes in multi-class scenarios. This is particularly evident when contrasting the trends observed in the single-class dataset (as illustrated in Fig. 5.18) with those from the Visdrone 10-class dataset. These observations suggest that the underlying explainer exhibits a bias towards the class score. Although LRP demonstrates enhanced performance, a significant initial decline in the network's quality is noted. Consequently, the quality throughout the subsequent pruning process remains inferior compared to when DetDSHAP is employed as the criterion. This pattern is similarly reflected

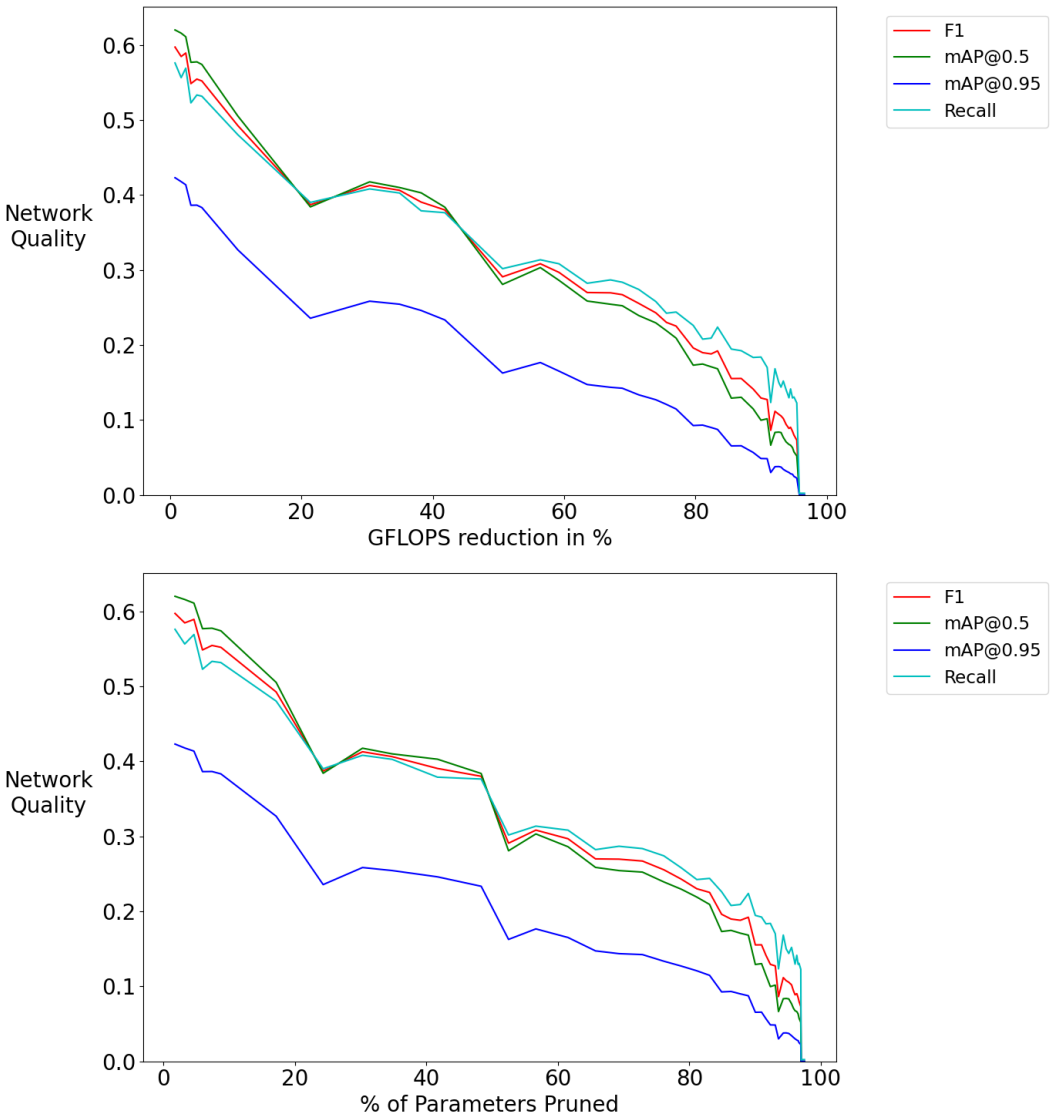


Figure 5.22: This figure displays the pruning trend when using LRP as a criterion to prune YOLOv5s trained the Coco2017 dataset.

in Fig. 5.22. Furthermore, this latter scenario indicates that pruning can extend beyond DetDSHAP levels when using LRP as the criterion; however, this extension is limited to approximately 10% beyond DetDSHAP, at which point the original performance of the network is severely compromised, likely rendering it unusable.

5.4.4 Post-Pruning Network Performance Quality

In this sub-section, a thorough examination is conducted on the impact of the pruning framework on the performance quality of the YOLOv5 detection models used in this research. Three deep detection models were selected for further development during the pruning process, with each pruned model undergoing an additional fine-

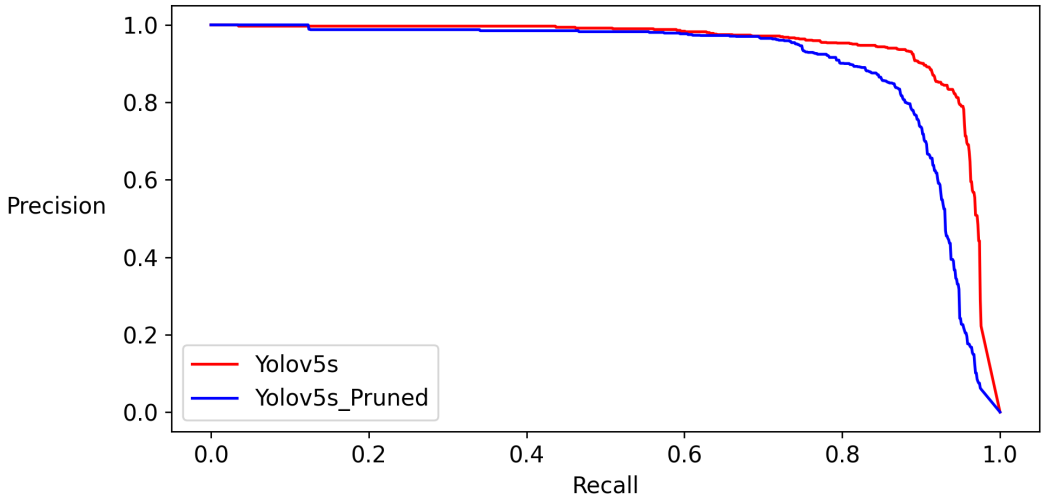


Figure 5.23: This figure depicts the precision-recall curves before and after pruning for the YOLOv5s that was trained on the self driving dataset.

	Self-driving car data		
	YOLOv5s	YOLOv5s pruned	% Change
mAP_{50-95}	70.7	68.8	-2.69
mAP_{50}	94.7	94.8	+0.11
Params (M)	7.01	1.30	-81.46
GFLOPs_{512}	10.08	5.91	-41.36
Infer Time(ms)	1.62	1.22	-25.0%

Table 5.3: The performance comparison between the unpruned and pruned YOLOv5s model trained on data gathered from the proprietary Self-Driving car dataset.

tuning phase for 200 epochs. The quality of the final networks, along with mean average precision (mAP) values and efficiency scores, within Table 5.3, Table 5.4 and Table 5.5. These efficiency scores encompass not only the number of parameters and floating-point operations per second (FLOPs) but also inference times measured on an NVIDIA GeForce RTX 2060S. To enhance the analysis, Precision-Recall (PR) curves are presented for the models trained on the three datasets used in this study. This graphical representation of precision against recall serves as a critical evaluation of the performance quality of the object detection models. A deep detection network is deemed effective if it maintains high precision as recall increases.

In Table 5.3, the final results are reported of the pruned YOLOv5s model which has been developed for the proprietary self-driving car dataset. It can be seen that

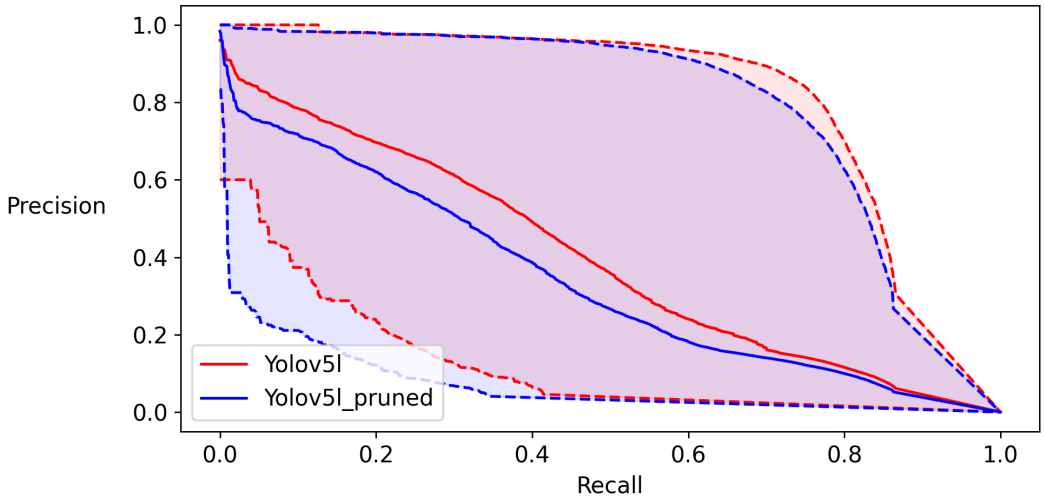


Figure 5.24: This figure depicts the precision-recall curves before and after pruning for the YOLOv5m that was trained on the Coco2017 dataset. The solid lines denote the average trend for all classes in their respective dataset, and the dotted lines represent the best and worst cases across all classes.

	Visdrone Dataset		
	YOLOv5l	YOLOv5l pruned	% Change
mAP_{50-95}	23.1	22.1	-3.90
mAP_{50}	39.6	38.6	-2.52
Params (M)	46.3	23.9	-48.38
$GFLOPs_{512}$	69.4	58.2	-16.14
Infer Time(ms)	6.40	5.67	-11.4%

Table 5.4: The performance comparison between the unpruned and pruned YOLOv5l model trained on the Visdrone dataset.

following the additional fine tuning there is only a minor drop in the mAP@0.5-0.95 score and even a slight improvement in the mAP@0.5. This is despite a reduction of over 80% of the parameters and 40% the GFLOPS. Moreover, Figure 5.23 shows the PR curves for the models trained on the proprietary self-driving car dataset. It can seen that there is very little change in the PR curve between the unpruned and pruned models. This is despite the large reduction in parameters and FLOPs that were reported in Section 5.4.3. As previously addressed, the self driving car dataset used here is relatively simple when compared to the other datasets used in this study making pruning more challenging in the others, and the results shown here reflect that.

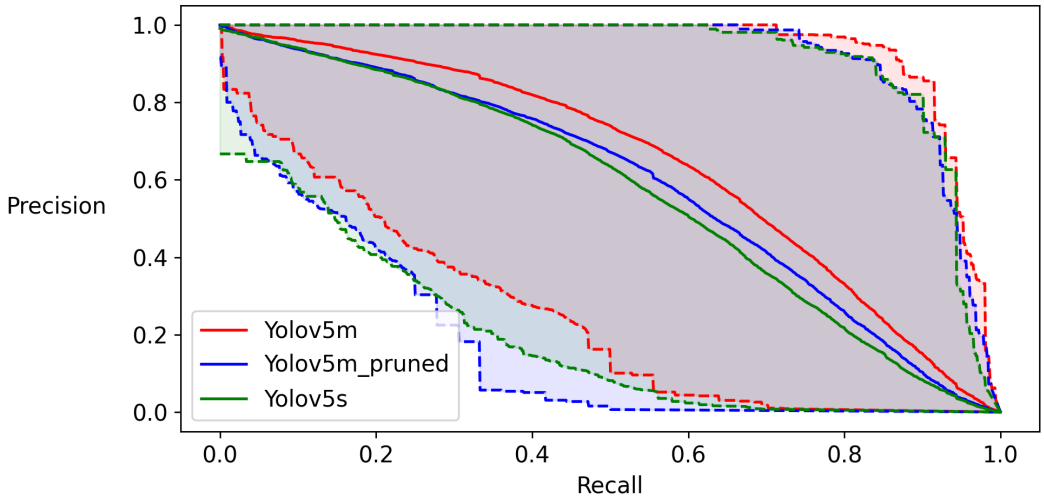


Figure 5.25: This figure depicts the precision-recall curves before and after pruning for the YOLOv5l that was trained on the Visdrone dataset. The solid lines denote the average trend for all classes in their respective dataset, and the dotted lines represent the best and worst cases across all classes.

The results seen Table 5.4 that show that it is possible to produce a similar outcome for a network trained on the more challenging dataset of Visdrone. It can be seen that the number of parameters of the YOLOv5l network was reduced by almost 50% and the GFLOPs by 16% without significantly reducing the performance. Figure 5.24 shows the PR curve of the model developed for the Visdrone dataset.

In the context of Visdrone, confusion matrices obtained from the unpruned models is illustrated in Fig. 5.26, and after pruning has been applied to the same model in Figure 5.27. The predominant class detected in this dataset for both models is 'car,' which exhibits the least decline in the true positive rate and is the most frequently occurring class within the dataset. However, a significant drawback of the pruned model is the increased proportion of false negatives, as depicted in Fig. 5.27. This indicates a decline in recall performance; nevertheless, an evaluation of the PR curve in Fig. 5.24 suggests that this decline is not substantial.

Table 5.5 presents the outcomes of pruning the YOLOv5m model, trained on the COCO2017 dataset. Additionally, Fig. 5.25illustrates the precision-recall curves for both the pruned YOLOv5m model and the unpruned version. This analysis also includes the PR curve for a YOLOv5s model trained on the same dataset. The data indicates that, unlike other scenarios discussed in this section, there is a notable decline in the network's performance, with a reduction exceeding 10% in mAP at the 0.5-0.95 threshold. Nevertheless, the PR curves depicted in Fig. 5.25 reveal that the pruned YOLOv5m model outperforms the YOLOv5s model on average across

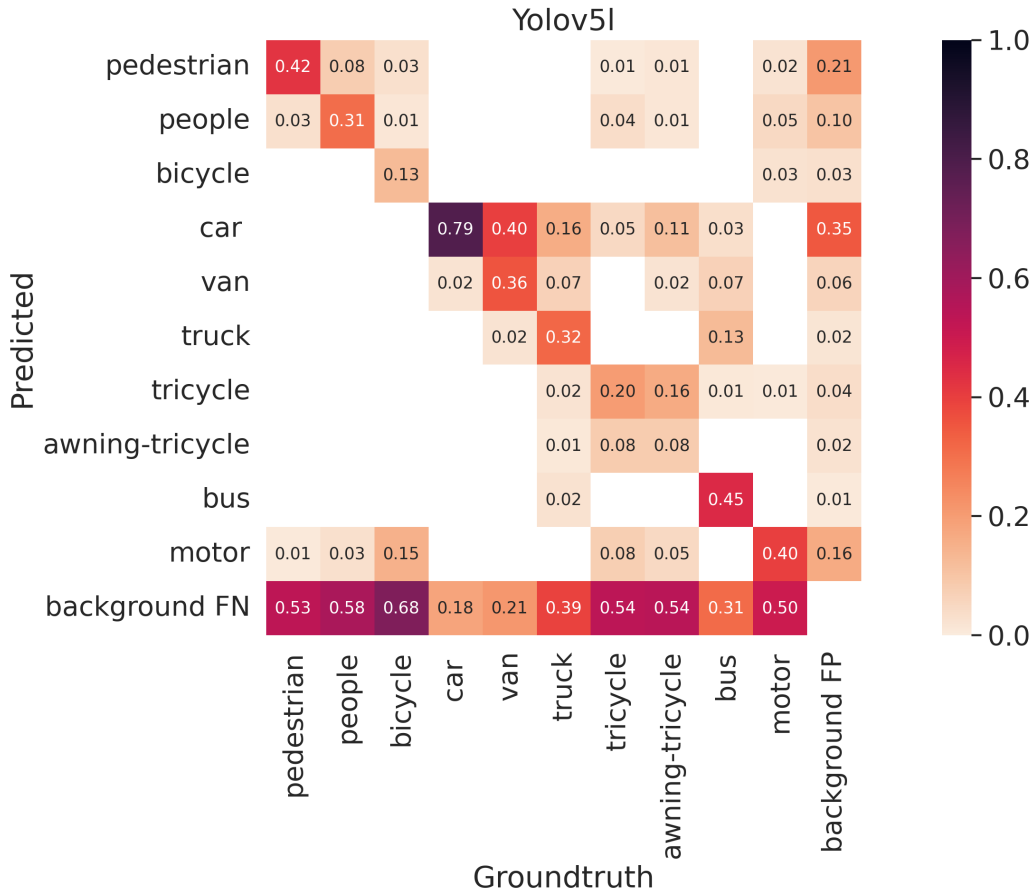


Figure 5.26: Confusion matrix of the unpruned YOLOv5l network trained on the Visdrone Dataset.

all classes. However, it is important to note that performance diminishes in the class with the lowest performance. This trend is also evident in Fig. 5.24 and is likely attributable to class imbalance, which affects the filter rankings. These rankings are crucial as they determine which components of the network should be pruned, and in the proposed methodology, they are derived from samples taken from the dataset. Future research could explore more advanced sampling techniques to mitigate the bias in filter rankings towards any specific class.

That being said, from PR curves in Fig. 5.25, it can be seen that generally the PR curve for the pruned YOLOv5m is equivalent to the curve for YOLOv5s. This indicates that one can produce a high performing object detector by starting with a larger one and pruning rather than developing a smaller one from scratch. This will be discussed in further detail in the following section.

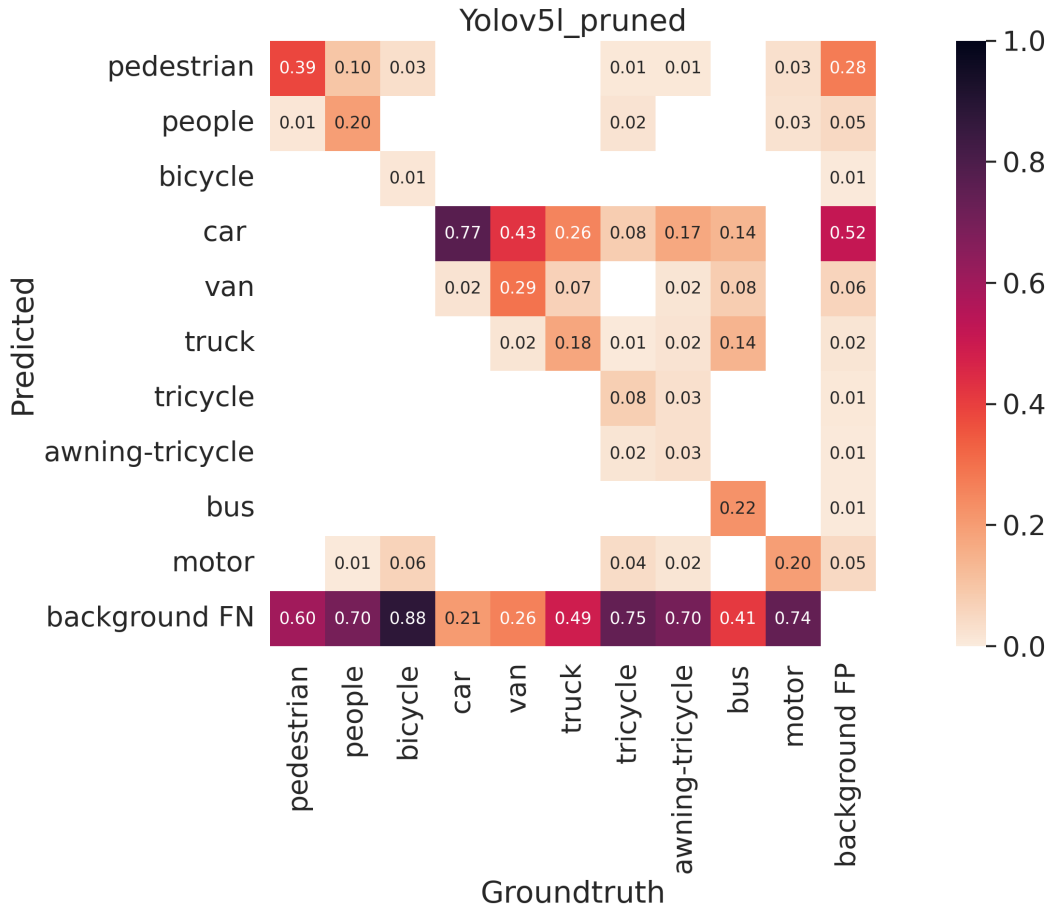


Figure 5.27: Confusion matrix of the YOLOv5l network trained on the Visdrone Dataset after pruning has been applied.

5.4.5 Pruning Verses Design

A frequently neglected aspect in the existing literature is the comparative effectiveness of pruning versus beginning with a more straightforward architecture. This section explores this issue by contrasting the pruned YOLOv5m, trained on the COCO2007 dataset, with YOLOv5s trained on the same dataset. For this analysis, the focus is restricted to publicly available pretrained models from [72] to eliminate any biases that might arise from custom implementations.

The bar graph in Fig. 5.28 illustrates the comparative performance of the pruned YOLOv5m against its unpruned counterpart and YOLOv5s. Performance metrics for the unpruned models are sourced from [72]. Initially, YOLOv5m comprised 21.2 million parameters, which were subsequently reduced to 6.5 million through pruning. This parameter count is significantly lower than that of YOLOv5s. Furthermore, the pruned YOLOv5m demonstrates superior performance in both mAP metrics.

YOLOv5s demonstrates superior performance in GFLOPS compared to the

	Coco2017 dataset		
	YOLOv5m	YOLOv5m pruned	% Change
mAP_{50-95}	44.1	39.1	-11.90
mAP_{50}	63.5	58.6	-7.71
Params (M)	21.2	6.5	-69.34
GFLOPs_{512}	48.9	33	-32.52
Infer Time(ms)	5.38	4.38	-18.59%

Table 5.5: The performance comparison between the unpruned and pruned YOLOv5m model trained on the Coco2017 dataset.

pruned YOLOv5m. This discrepancy can be attributed to the pruning criterion, which predominantly emphasised the upper layers nearer to the model's output. However, if the objective of pruning is to minimise the FLOPs required during inference, it would be more advantageous to concentrate on pruning the lower layers closer to the model's input. Future research will aim to modify the pruning framework to prioritise these layers, thereby enabling a more aggressive reduction of FLOPs.

5.5 Summary

This chapter illustrated the intersection of explainability and pruning within the YOLOv5 framework. The DetDSHAP methodology was proposed as a means to investigate a neural network, thereby enhancing comprehension of its decision-making processes and enabling designers to identify the constraints of their deep learning models. This concept was further developed by showcasing DetDSHAP as a viable criterion for pruning, resulting in the removal of a substantial portion of the parameters from deep networks with negligible impact on performance—nearly 50% on the Visdrone dataset and over 80% on the single-class self-driving car dataset. Additionally, the pruning framework presented in this chapter demonstrated that a large network could achieve greater memory efficiency compared to a similarly performing smaller network. Ultimately, through the analysis of the Wrapping Game and the demonstrated efficacy as a pruning criterion, it can be confidently asserted that DetDSHAP accurately allocates causal information.

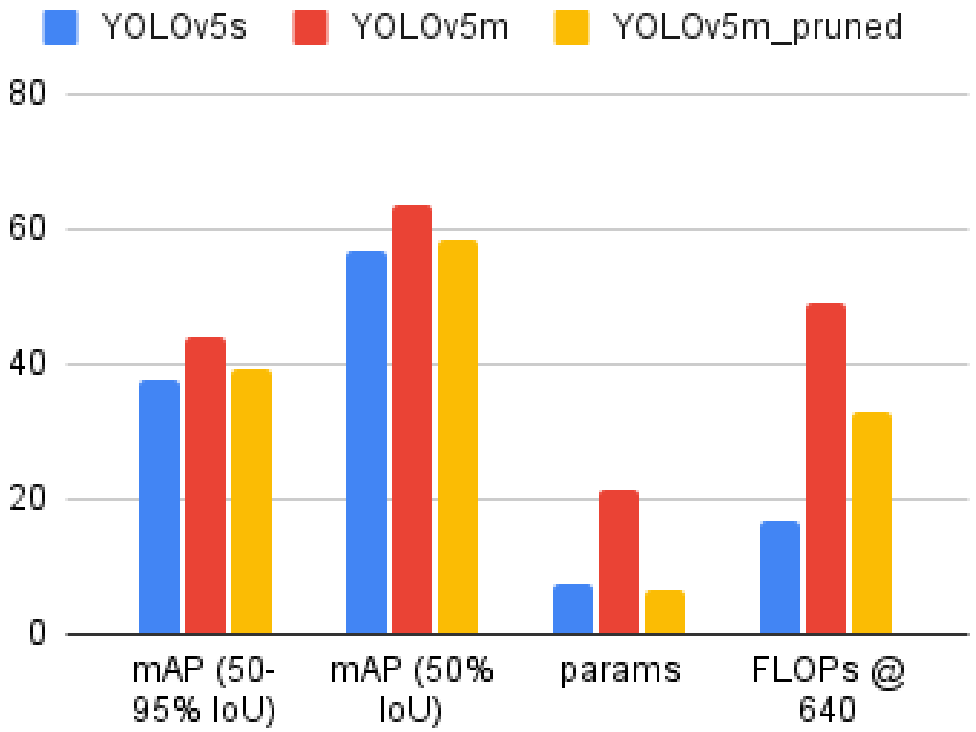


Figure 5.28: This plot shows the performance quality and structure efficacy of the pruned YOLOv5m model compared to two other models released by [72]. Training and evaluation are conducted on the COCO2017 dataset.

Chapter 6

Conclusion

6.1 Overview

This study is focused on exploring the application of explainable AI for the perception capabilities of drones, a domain that pushes the challenge of object detection to its very limits. By leveraging explainability methods, the research addresses critical concerns regarding the transparency and reliability of deep learning models deployed on autonomous platforms. In scenarios where confidence is paramount, such as in autonomous navigation, automatic target recognition, and obstacle avoidance, it is essential to understand the factors influencing model predictions. This study not only proposes several novel frameworks to improve explainability for detection systems, but provides extensive testing and verification of the proposed frameworks using established approaches, and proposes new testing methodologies for new explainers. Fundamentally, the work aims to bridge the gap between complex model outputs and human understanding, ensuring that autonomous vehicles operate in a safe, predictable, and accountable manner.

6.2 Summary and Discussion

In Chapter 1, the goal of this thesis is formulated, and this is revisited here. The key contributions have been to develop several novel explainer frameworks for the task of object detection. It was established that the effectiveness of explanations in AI relies on several key properties that enhance interpretability and trust. Explanations are most valuable when they are 'contrastive', addressing why one outcome occurred over another, and 'selective', focusing on the most relevant causes. Explanations also

hold social and adaptive aspects: they must convey information from the explainer to the explainee in a way that aligns with the explainee's prior knowledge and context. Tailoring explanations to an audience's expertise ensures the level of complexity is appropriate. Finally, faithfulness and robustness are essential: 'faithful' explanations accurately reflect the model's decision-making, and 'robustness' ensures stability against minor data variations, thereby, enhancing reliability in practical applications. Taken together, these properties shape effective, transparent AI explanations.

Chapter 2 extended the review of currently available literature, particularly on the subject of datasets. Here, the existing datasets that are proposed for the use of object detection and XAI validation are thoroughly investigated. Following the review, a new dataset was proposed that would exploit the lack of overlap between these two applications. The proposed dataset is intended to complement other works to accelerate the deployment of explainable deep learning based ADL systems.

The work in Chapter 3 aimed at contributing to the area of object detection from a UAV platform. The focus is on improving performance for small object detection, and also improving model explainability. Results have been provided that justify the use of the Tile Loader to improve the detection of small objects while still operating in the real-time domain. This component was combined with YOLOv5 in order to create an algorithm which can achieve reasonable results on the VisDrone test-dev set.

In this work several different explainers using contrasting approaches were investigated for the task of object detection, and these were covered in detail in Chapters 3,4 and 5. The primary aim, across all methods, was to produce heatmaps showing how the input image influenced predictions made by a deep detection model. Each method was designed with unique goals or specific requirements in mind, addressing various challenges associated with explaining complex detection models.

The first explainer that was investigated was Grad-CAM in chapter 3. The proposed methodology was introduced as a fast and efficient means to extract additional information from YOLO-style algorithms. This information could be helpful to a developer, or an operator, in understanding failure modes and was capable of running in near real time. The heatmaps produced by Grad-CAM proved reasonably discriminative of the target object and passed Model Dependency and Data Dependency Tests.

The drawback of Grad-CAM was that it was not well suited to the architectures

that were investigated. The heatmaps did not reveal a lot of information about specific objects, and in this approach the output features are treated independently. As such, further progression was centred on two core developments. Firstly, developing methodologies that produced more concise explanations for individual objects. The second development, was to produce explainers yielding explanations that were dependent on both the localisation and classification predictions.

Chapter 4 introduced a novel adaptation to KernelSHAP - a perturbation-based explainer designed to operate effectively across diverse model architectures. This approach offers a distinct advantage over other explainers explored in this work, such as DetDSHAP and Layer-wise Relevance Propagation (LRP), which often require custom implementations tailored to specific model architectures. By contrast, the KernelSHAP explainer can be quickly and efficiently deployed to investigate various architectures, even those not based on neural networks. This flexibility enables robust comparative analyses across models, providing a valuable tool for evaluating architectures that might otherwise lack support for explainers.

The proposed KernelSHAP framework was evaluated with multiple methodologies. Firstly, it demonstrated effectiveness in identifying visual biases within images. Following this, the method demonstrated its ability to allocate causal information correctly in the Deletion and Insertion protocols which were applied. Finally, its discriminative ability was analysed with the Wrapping Game. The outcomes of these experiments provide strong evidence that the KernelSHAP framework is faithful to the explicant.

Beyond this, the KernelSHAP framework provides enhanced social and adaptive aspects to the explainee. Its visual maps are more descriptive than those produced by the Grad-CAM framework, and by allowing the explainee to input hypothetical predictions, it creates an interactive 'dialogue' between the explainee and the explainer.

The final explainer that was proposed in this thesis is DetDSHAP in Chapter 4. Building on insights from previous chapters, DetDSHAP provides detailed, instance-level explanations, allowing developers to better understand how various image regions contribute to the model's predictions in object detection scenarios. Additionally, this chapter explores the intersection between model interpretability and network compression, demonstrating how explainability insights can inform network pruning without sacrificing performance. This contribution expands the

toolkit available for deploying and refining explainable AI on autonomous platforms, ensuring robust performance and accountability in real-world, safety-critical environments.

In each chapter where an explainer was proposed, the Wrapping Game was used to evaluate it. This methodology was proposed in Chapter 2 and intended to evaluate the discriminativeness of the explainer on the target object. Unlike contemporary metrics, like the Pointing Game, the proposed approach allows for a more comprehensive evaluation measure, especially for images with multiple objects or complex scenes, where simply noting the highest point may misrepresent the saliency map's quality.

Discriminativeness is valued by the human for interpretability of the provided explanation. While the Wrapping Game does yield some information about the explainer's faithfulness to the explicant, faithfulness is not the main focus of the the Wrapping Game analysis. As such, the Wrapping Game should not be used exclusively to analyse new explainers. The Wrapping Game will also take into account how well the explainer's discriminativeness aligned with the model's performance. This is important, given that when a model predicts with low confidence, understanding its decision-making process can be more challenging - yet this is precisely when explanations are crucial. Therefore, the limitations of the explainer on these edge cases needs to be understood. In such cases, explanations can help detect potential model uncertainties and errors, offering insights into when and why a model's predictions may be less reliable.

In Chapter 5 DetDSHAP was compared to the LRP framework proposed in [106]. In these investigations, LRP proved more discriminative on instances where the model had high confidence in its predictions, but struggled heavily on the instances of low confidence, where the DetDSHAP explainer did not. Moreover, in this same chapter, the LRP framework was less capable as a pruning criterion when compared to DetDSHAP. This indicates a lower faithfulness to the model's processes.

Ultimately, the proposed explainers equip developers with advanced tools to identify model limitations, inform model improvements, and implement appropriate safeguards in autonomous system designs. This work provides a foundation for future advancements in XAI for autonomous vehicles, where both transparency and reliability are essential. Moving forward, the potential exists to combine local explanations to achieve global insights, and further expand the applicability of explainers

in multi-object scenes, thereby broadening the understanding and utility of XAI in the autonomous systems domain.

While the Grad-CAM-based framework proposed in Chapter 3 offers a computationally lightweight solution, enabling near real-time explanations, the SHAP-based frameworks introduced in Chapters 4 and 5 require significantly higher computational resources. This is due to the need for perturbation-based sampling, where numerous background samples are evaluated to determine the model's reliance on different input features. Despite this computational overhead, these methods offer superior fidelity and faithfulness, ensuring that explanations accurately reflect the true decision-making process of the model.

6.3 Future Work

This research proposed several saliency map based approaches to disentangle predictions made by 2D object detectors. The ultimate goal of XAI is to improve the performance and capabilities of the network being investigated. So far, only the work with DetDSHAP has yielded new networks with better capabilities. Currently for Grad-CAM and KernelSHAP, this thesis only presented examples of their usage in small, controlled experiments, so their usefulness in realistic settings is as yet unproven. Beyond this there are several other open challenges and opportunities that are presented below.

6.3.1 Extending the XAI-AV Dataset

This work will continue beyond this thesis to adapt with future needs over time. This includes expanding the volume of collected data, and the supported applications beyond object detection. As has been established, the current dataset contains a bias of 'White Van' and 'Black Hatchback'; while this bias may be representative of what can be found on the streets that were surveyed, this may not always be the case. Hence, progress will involve collecting sufficient samples to balance the distribution of type and colour of vehicles.

6.3.2 Local to Global Explanations

Saliency maps explain the behaviour of a model for a single test point, such an explanation may be too fragile and could lead one to a false conclusion about the

model's strategy [168]. To get a holistic view of the model's strategies, one would need to review many points in the dataset. However, screening through a large number of individual explanations can be too intensive, especially on larger datasets.

An open research question is: can local explanations be joined together to build a globally interpretable model? Lapuschkin et al.[103] also took the approach of joining together multiple local explanations - this time from LRP - to create a global perspective of the DNN's behaviour. They propose Spectral Relevance Analysis (SpRAy), which detects typical and atypical behaviour as frequently reoccurring patterns in the saliency maps via eigenvalue-based spectral clustering. This can then be presented to a human in a concise and interpretable manner using t-Stochastic Neighbourhood Embedding (t-SNE).

The essential foundations have already been put in place for providing Global Explanations in Chapter 5. Individual explanations were joined together to calculate the weight of individual units in the network for pruning. They were joined simply by averaging all the scores for a particular unit. Like the work in [103], what is needed is to employ an unsupervised learning approach to isolate key patterns present in the saliency maps, or from within the network itself.

6.3.3 Advancing Pruning Framework

Improvements can be made to the existing pruning framework to enhance the quality of models following the pruning process. In a manner akin to the findings of Yeom et al. [163], the DetDSHAP pruning criterion proposed here emphasises layers that are situated closer to the network's output. This is because, in both approaches, the contribution is distributed evenly backwards through the network and thus becomes more diffused in the layers closer to the input. This distribution diminishes the criterion's effectiveness in minimising the number of Floating-Point Operations Per Second (FlOPS) which are more concentrated in those layers. Future work could adjust the framework to specifically target the layers nearer to the input, thereby enhancing the improvements in inference speed achieved through pruning. Moreover, further modifying the selection process for samples used in determining filter ranks, could lead to a reduced decline in performance after pruning. This could involve strategies, such as ensuring a balance in class labels among the samples, as well as considering the positioning and dimensions of the bounding boxes in the samples.

Bibliography

[1] B. Goodman and S. Flaxman, "European union regulations on algorithmic decision-making and a "right to explanation"," *AI Magazine*, vol. 38, no. 3, pp. 50–57, Oct. 2017, ISSN: 0738-4602.

[2] F. Wang and C. Rudin, "Causal falling rule lists," *arXiv preprint arXiv:1510.05189*, 2017.

[3] P. W. Koh and P. Liang, "Understanding black-box predictions via influence functions," *arXiv preprint arXiv:1703.04730*, 2020.

[4] R. R. Selvaraju *et al.*, "Grad-cam: Visual explanations from deep networks via gradient-based localization," *International Journal of Computer Vision*, vol. 128, no. 2, pp. 336–359, Oct. 2019, ISSN: 1573-1405.

[5] UK Civil Aviation Authority, *Airspace policy concept: Airspace requirements for the integration of beyond visual line of sight (bvlos) unmanned aircraft*, Accessed: October 23, 2024, 2024.

[6] UK Department for Transport, *Code of practice: Vehicle authorisations and exemptions for more complex cav trials*, Accessed: October 23, 2024, 2024.

[7] UK Department for Transport, *Code of practice: Automated vehicle trialling*, Accessed: October 23, 2024, 2024.

[8] C. Molnar, *Interpretable Machine Learning, A Guide for Making Black Box Models Explainable*. 2019, https://christophm.github.io/interpretable-ml-book/.

[9] H. Baniecki and P. Biecek, "Adversarial attacks and defenses in explainable artificial intelligence: A survey," *Information Fusion*, vol. 107, p. 102 303, 2024, ISSN: 1566-2535.

[10] P. Zhu, L. Wen, X. Bian, H. Ling, and Q. Hu, "Vision meets drones: A challenge," *arXiv preprint*, vol. abs/1804.07437, 2018.

[11] A. Balasubramaniam and S. Pasricha, "Object detection in autonomous cyber-physical vehicle platforms: Status and open challenges," in *Machine Learning and Optimization Techniques for Automotive Cyber-Physical Systems*, V. K. Kukkala and S. Pasricha, Eds. Cham: Springer International Publishing, 2023, pp. 509–523, ISBN: 978-3-031-28016-0.

[12] V. Petsiuk, A. Das, and K. Saenko, "Rise: Randomized input sampling for explanation of black-box models," *ArXiv*, vol. abs/1806.07421, 2018.

[13] R. C. Fong and A. Vedaldi, "Interpretable explanations of black boxes by meaningful perturbation," *2017 IEEE International Conference on Computer Vision (ICCV)*, Oct. 2017.

[14] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, "Learning deep features for discriminative localization," *arXiv preprint arXiv:1512.04150*, 2015.

[15] M. Hogan and P. N. Aouf, "Towards real time interpretable object detection for uav platform by saliency maps," in *2021 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 2021, pp. 1178–1183.

[16] M. Hogan, N. Aouf, P. Spencer, and J. Almond, "Explainable object detection for uncrewed aerial vehicles using kernelshap," in *2022 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, 2022, pp. 136–141.

[17] M. Hogan and P. N. Aouf, "Explainable dataset for ground based drones in urban environments," in *2024 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, Accepted for publication, 2024.

[18] M. Hogan and P. N. Aouf, "Detdshap: Explainable object detection for uncrewed and autonomous drones with shapley values," *Trust through eXplAInability (XAI), Robustness and Verification of Autonomous Systems*, 2025, Manuscript in submission.

[19] Sapience Project, *Sapience project: Exploring consciousness and ai*, Accessed: October 23, 2024, 2024.

[20] Velodyne Lidar, Inc., *VLP-16 User Manual*, Rev F, San Jose, CA, USA, 2022.

[21] Intel Corporation, *Intel RealSense Depth Camera D455f*, Accessed: October 23, 2024, 2024.

[22] NVIDIA Corporation, *Nvidia jetson orin nano series modules datasheet*, DS-11105-001_v1.3, Rev F, August 2024, Santa Clara, CA, USA, 2024.

[23] V. K. Kukkala and S. Pasricha, Eds., *Machine Learning and Optimization Techniques for Automotive Cyber-Physical Systems*, English, 1st 2023. Cham: Springer International Publishing, 2023.

[24] T. Mouats, N. Aouf, L. Chermak, and M. A. Richardson, "Thermal stereo odometry for uavs," *IEEE Sensors Journal*, vol. 15, no. 11, pp. 6335–6347, 2015.

[25] O. Kechagias-Stamatis, N. Aouf, and D. Nam, "3d automatic target recognition for uav platforms," in *2017 Sensor Signal Processing for Defence Conference (SSPD)*, IEEE, 2017, pp. 1–5.

[26] M. Turk and A. Pentland, "Face recognition using eigenfaces," in *Proceedings. 1991 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1991, pp. 586–591.

[27] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, vol. 1, 2001, pp. I–I.

[28] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[29] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1, 2005, 886–893 vol. 1.

[30] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, F. Pereira, C. Burges, L. Bottou, and K. Weinberger, Eds., vol. 25, Curran Associates, Inc., 2012.

[31] 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*, C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, Eds., vol. 28, Curran Associates, Inc., 2015.

[32] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 779–788.

[33] W. Liu *et al.*, "Ssd: Single shot multibox detector," *Lecture Notes in Computer Science*, pp. 21–37, 2016, ISSN: 1611-3349.

[34] M. M. Chong, H. N. Tan, L. Jun, and R. K. Gay, "Geometric framework for fingerprint image classification," *Pattern Recognition*, vol. 30, no. 9, pp. 1475–1488, 1997.

[35] C. Samson, L. Blanc-Feraud, G. Aubert, and J. Zerubia, "A variational model for image classification and restoration," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 5, pp. 460–472, 2000.

[36] J. L. Mundy, "Object recognition in the geometric era: A retrospective," in *Toward Category-Level Object Recognition*, J. Ponce, M. Hebert, C. Schmid, and A. Zisserman, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 3–28, ISBN: 978-3-540-68795-5.

[37] A. Zisserman *et al.*, "Class-based grouping in perspective images," in *Proceedings of IEEE International Conference on Computer Vision*, 1995, pp. 183–188.

[38] 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.

[39] G. J. Gray *et al.*, "Feature-based target recognition in infrared images for future unmanned aerial vehicles," *Journal of Battlefield Technology*, vol. 14, no. 2, pp. 27–36, 2011.

[40] S. Hwang, J. Park, N. Kim, Y. N. Wu, and B. Han, "Multispectral pedestrian detection: Benchmark dataset and baseline," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 1037–1045.

[41] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 9, pp. 1627–1645, 2010.

[42] C. Wojek and B. Schiele, "A performance evaluation of single and multi-feature people detection," in *DAGM Symposium on Pattern Recognition*, Springer, 2008, pp. 82–91.

[43] Q. Zhu, S. Avidan, M.-C. Yeh, and K.-T. Cheng, "Fast human detection using a cascade of histograms of oriented gradients," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2006, pp. 1491–1498.

[44] M. Enzweiler and D. M. Gavrila, "Monocular pedestrian detection: Survey and experiments," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 12, pp. 2179–2195, 2009.

[45] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[46] A. Vaswani *et al.*, "Attention is all you need," in *Advances in neural information processing systems*, vol. 30, 2017, pp. 5998–6008.

[47] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015, ISSN: 1476-4687.

[48] T. Mitchell, *Machine learning*, English. McGraw-Hill, 1997.

[49] K. P. Murphy, *Machine learning: a probabilistic perspective*, English. London;Cambridge, Mass; MIT Press, 2012.

[50] T. Lin *et al.*, "Microsoft COCO: common objects in context," *CoRR*, 2014.

[51] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, 2012, pp. 3354–3361.

[52] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[53] X. Chen *et al.*, "Safety-assured design and adaptation of connected and autonomous vehicles," in *Machine Learning and Optimization Techniques for Automotive Cyber-Physical Systems*, V. K. Kukkala and S. Pasricha, Eds. Cham: Springer International Publishing, 2023, pp. 735–757, ISBN: 978-3-031-28016-0.

[54] L. Bottou and O. Bousquet, *Optimization for Machine Learning*. MIT Press, 2012, ch. Stochastic Gradient Descent Tricks, pp. 421–436.

[55] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[56] G. Hinton, *Neural Networks for Machine Learning*, https://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf, Coursera Lecture 6e, 2012.

[57] S. Ruder, "An overview of gradient descent optimization algorithms," *arXiv preprint arXiv:1609.04747*, 2016.

[58] M. Zhou, J. Xiong, C. Ma, *et al.*, "Multi-sensor fusion for intelligent autonomous driving," *Sensors*, vol. 24, no. 11, p. 3573, 2024.

[59] M. I. Haque and Z. Zhang, "Multimodal object detection and sensor fusion for autonomous driving: A review," *arXiv preprint arXiv:2201.07706*, 2022.

[60] M. Abdulsalam and N. Aouf, "Deep weed detector/classifier network for precision agriculture," in *2020 28th Mediterranean Conference on Control and Automation (MED)*, IEEE, 2020, pp. 1087–1092.

[61] P. Pyrrö, H. Naseri, and A. Jung, "Rethinking drone-based search and rescue with aerial person detection," *ArXiv*, vol. abs/2111.09406, 2021.

[62] P. Zhu, L. Wen, X. Bian, H. Ling, and Q. Hu, "Vision meets drones: A challenge," *arXiv preprint*, vol. abs/1804.07437, 2018.

[63] T.-Y. Lin *et al.*, *Feature pyramid networks for object detection*, 2017.

[64] A. Kirillov, R. B. Girshick, K. He, and P. Dollár, "Panoptic feature pyramid networks," *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6392–6401, 2019.

[65] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, and R. Girshick, *Detectron2*, https://github.com/facebookresearch/detectron2, 2019.

[66] J. Šarić, M. Oršić, and S. Šegvić, "Panoptic swiftnet: Pyramidal fusion for real-time panoptic segmentation," *Remote Sensing*, vol. 15, no. 8, 2023, ISSN: 2072-4292.

[67] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 9, pp. 1627–1645, 2010.

[68] R. Girshick, J. Donahue, T. Darrell, and J. Malik, *Rich feature hierarchies for accurate object detection and semantic segmentation*, 2014.

[69] J. Redmon and A. Farhadi, *Yolo9000: Better, faster, stronger*, 2016.

[70] J. Redmon and A. Farhadi, *Yolov3: An incremental improvement*, 2018.

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

[72] G. Jocher, A. Stoken, and J. Borovec, *Ultralytics/YOLOv5*, version v5.0, Apr. 2021.

[73] H. Tsunakawa, Y. Kameya, H. Lee, Y. Shinya, and N. Mitsumoto, "Contrastive relevance propagation for interpreting predictions by a single-shot object detector," in *2019 International Joint Conference on Neural Networks (IJCNN)*, 2019, pp. 1–9.

[74] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *International Conference on Learning Representations*, vol. abs/1409.1556, 2014.

[75] R. Girshick, "Fast r-cnn," in *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 1440–1448.

[76] J. Huang *et al.*, "Speed/accuracy trade-offs for modern convolutional object detectors," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 3296–3297.

[77] F. Doshi-Velez and B. Kim, "Towards a rigorous science of interpretable machine learning," *arXiv: Machine Learning*, 2017.

[78] T. Hickling, N. Aouf, and P. Spencer, "Robust adversarial attacks detection based on explainable deep reinforcement learning for uav guidance and planning," *IEEE Transactions on Intelligent Vehicles*, vol. 8, no. 10, pp. 4381–4394, 2023.

[79] K. Aas, M. Jullum, and A. Løland, "Explaining individual predictions when features are dependent: More accurate approximations to shapley values," *ArXiv*, vol. abs/1903.10464, 2019.

[80] L. H. Gilpin *et al.*, "Explaining explanations: An overview of interpretability of machine learning," in *2018 IEEE 5th International Conference on Data Science and Advanced Analytics (DSAA)*, 2018, pp. 80–89.

[81] I. E. Nielsen, D. Dera, G. Rasool, R. P. Ramachandran, and N. C. Bouaynaya, *Robust explainability: A tutorial on gradient-based attribution methods for deep neural networks*, 2022.

[82] T. Miller, "Explanation in artificial intelligence: Insights from the social sciences," *Artificial Intelligence*, vol. 267, pp. 1–38, 2019.

[83] E. Chu, D. K. Roy, and J. Andreas, "Are visual explanations useful? a case study in model-in-the-loop prediction," *ArXiv*, vol. abs/2007.12248, 2020.

[84] V. Petsiuk *et al.*, "Black-box explanation of object detectors via saliency maps," *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11 438–11 447, 2020.

[85] C. V. Dolph, L. Tran, and B. D. Allen, "Towards explainability of uav-based convolutional neural networks for object classification," in *2018 Aviation Technology, Integration, and Operations Conference*. 2018.

[86] J. Borowski *et al.*, "Exemplary natural images explain cnn activations better than feature visualizations," *ArXiv*, vol. abs/2010.12606, 2020.

[87] B. Lai and X. Gong, "Saliency guided end-to-end learning for weakly supervised object detection," in *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, ser. IJCAI'17, Melbourne, Australia: AAAI Press, 2017, pp. 2053–2059, ISBN: 9780999241103.

[88] M. Zhang *et al.*, "Context-aware saliency for explainable object detection," in *Proceedings of the 30th ACM International Conference on Multimedia*, ACM, 2022, pp. 1–9.

[89] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep inside convolutional networks: Visualising image classification models and saliency maps," *CoRR*, vol. abs/1312.6034, 2013.

[90] C. F. Flores, A. Gonzalez-Garcia, J. van de Weijer, and B. Raducanu, "Saliency for fine-grained object recognition in domains with scarce training data," *CoRR*, vol. abs/1808.00262, 2018.

[91] P.-J. Kindermans *et al.*, "Learning how to explain neural networks: Patternnet and patternattribution," in *Proceedings of the 6th International Conference on Learning Representations (ICLR)*, 2018.

[92] M. T. Ribeiro, S. Singh, and C. Guestrin, ""why should i trust you?": Explaining the predictions of any classifier," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '16, San Francisco, California, USA: Association for Computing Machinery, 2016, pp. 1135–1144, ISBN: 9781450342322.

[93] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, ser. NIPS'17, Long Beach, California, USA: Curran Associates Inc., 2017, pp. 4768–4777, ISBN: 9781510860964.

[94] L. S. Shapley, "17. a value for n-person games," in *Contributions to the Theory of Games (AM-28), Volume II*, H. W. Kuhn and A. W. Tucker, Eds. Princeton University Press, 2016, pp. 307–318.

[95] D. Smilkov, N. Thorat, B. Kim, F. B. Viégas, and M. Wattenberg, "Smoothgrad: Removing noise by adding noise," *CoRR*, vol. abs/1706.03825, 2017.

[96] A. Shrikumar, P. Greenside, and A. Kundaje, "Learning important features through propagating activation differences," in *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ser. ICML'17, Sydney, NSW, Australia: JMLR.org, 2017, pp. 3145–3153.

[97] M. Sundararajan, A. Taly, and Q. Yan, "Axiomatic attribution for deep networks," in *International Conference on Machine Learning*, 2017.

[98] A. Shrikumar, P. Greenside, A. Shcherbina, and A. Kundaje, "Not just a black box: Learning important features through propagating activation differences," *CoRR*, vol. abs/1605.01713, 2016.

[99] J. Adebayo *et al.*, "Sanity checks for saliency maps," in *NIPS'18: Proceedings of the 32nd International Conference on Neural Information Processing Systems*, ser. NIPS'18, Montréal, Canada: Curran Associates Inc., 2018, pp. 9525–9536.

[100] Y. Bengio, A. C. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, pp. 1798–1828, 2012.

[101] A. Mahendran and A. Vedaldi, "Visualizing deep convolutional neural networks using natural pre-images," *International Journal of Computer Vision*, vol. 120, no. 3, pp. 233–255, May 2016, ISSN: 1573-1405.

[102] S. Bach *et al.*, "On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation," *PloS one*, vol. 10, no. 7, e0130140–e0130140, Jul. 2015.

[103] S. Lapuschkin *et al.*, "Unmasking clever hans predictors and assessing what machines really learn," *Nature Communications*, vol. 10, no. 1, Mar. 2019, ISSN: 2041-1723.

[104] G. Montavon, S. Lapuschkin, A. Binder, W. Samek, and K.-R. Müller, "Explaining nonlinear classification decisions with deep taylor decomposition," *Pattern Recognition*, vol. 65, pp. 211–222, May 2017, ISSN: 0031-3203.

[105] G. Montavon, A. Binder, S. Lapuschkin, W. Samek, and K.-R. Müller, "Layer-wise relevance propagation: An overview," in *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*, W. Samek, G. Montavon, A. Vedaldi, L. K. Hansen, and K.-R. Müller, Eds. Springer International Publishing, 2019, pp. 193–209, ISBN: 978-3-030-28954-6.

[106] A. Karasmanoglou, M. Antonakakis, and M. Zervakis, "Heatmap-based explanation of yolov5 object detection with layer-wise relevance propagation," in *2022 IEEE International Conference on Imaging Systems and Techniques (IST)*, 2022, pp. 1–6.

[107] L. Deng, "The mnist database of handwritten digit images for machine learning research [best of the web]," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141–142, 2012.

[108] A. Krizhevsky, "Learning multiple layers of features from tiny images," Tech. Rep., 2009.

[109] L. He, N. Aouf, and B. Song, "Explainable deep reinforcement learning for uav autonomous path planning," *Aerospace Science and Technology*, vol. 118, p. 107 052, 2021, ISSN: 1270-9638.

[110] R. J. Tomsett, D. Harborne, S. Chakraborty, P. K. Gurram, and A. D. Preece, "Sanity checks for saliency metrics," *ArXiv*, vol. abs/1912.01451, 2019.

[111] KimYoung-Jin and KimEun-Gyung, "Real-time fire detection based on cnn and grad-cam," *Journal of the Korea Institute of Information and Communication Engineering*, vol. 22, no. 12, pp. 1596–1603, Dec. 2018.

[112] H. Chen, S. M. Lundberg, and S.-I. Lee, "Explaining models by propagating shapley values of local components," *ArXiv*, vol. abs/1911.11888, 2019.

[113] A. Binder, G. Montavon, S. Lapuschkin, K.-R. Müller, and W. Samek, "Layer-wise relevance propagation for neural networks with local renormalization layers," in *Artificial Neural Networks and Machine Learning – ICANN 2016*, A. E. Villa, P. Masulli, and A. J. Pons Rivero, Eds., Cham: Springer International Publishing, 2016, pp. 63–71, ISBN: 978-3-319-44781-0.

[114] W. Samek, A. Binder, G. Montavon, S. Lapuschkin, and K.-R. Müller, "Evaluating the visualization of what a deep neural network has learned," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, pp. 2660–2673, 2015.

[115] J. Zhang, Z. Lin, J. Brandt, X. Shen, and S. Sclaroff, *Top-down neural attention by excitation backprop*, 2016.

[116] R. Mottaghi *et al.*, "The role of context for object detection and semantic segmentation in the wild," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 891–898.

[117] A. Torralba, "Contextual priming for object detection," *International Journal of Computer Vision*, vol. 53, no. 2, pp. 169–191, 2003, ISSN: 1573-1405.

[118] C. Wang and N. Aouf, "Fusion attention network for autonomous cars semantic segmentation," in *2022 IEEE Intelligent Vehicles Symposium (IV)*, 2022, pp. 1525–1530.

[119] C. Wang and N. Aouf, "Explainable deep adversarial reinforcement learning approach for robust autonomous driving," *IEEE Transactions on Intelligent Vehicles*, pp. 1–13, 2024.

[120] C. Chen, B. Wang, C. X. Lu, N. Trigoni, and A. Markham, "Deep learning for visual localization and mapping: A survey," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–21, 2023.

[121] A. Vellido, J. D. Martín-Guerrero, and P. J. G. Lisboa, "Making machine learning models interpretable," in *The European Symposium on Artificial Neural Networks*, 2012.

[122] F. Doshi-Velez *et al.*, "Accountability of ai under the law: The role of explanation," *SSRN Electronic Journal*, Nov. 2017.

[123] J. Tritscher, A. Krause, and A. Hotho, "Feature relevance XAI in anomaly detection: Reviewing approaches and challenges," *Front. Artif. Intell.*, vol. 6, Feb. 2023.

[124] I. D. Apostolopoulos and P. P. Groumpos, "Fuzzy cognitive maps: Their role in explainable artificial intelligence," *Applied Sciences*, vol. 13, no. 6, 2023, ISSN: 2076-3417.

[125] G. Cadamuro, R. Gilad-Bachrach, and J. Zhu, "Debugging machine learning models," in *ICML Workshop on Reliable Machine Learning in the Wild.*, 2016.

[126] H. Lakkaraju, E. Kamar, R. Caruana, and J. Leskovec, "Interpretable & explorable approximations of black box models," *ArXiv*, vol. abs/1707.01154, 2017.

[127] C. Agarwal *et al.*, "Openxai: Towards a transparent evaluation of post hoc model explanations," in *Proceedings of the 36th International Conference on Neural Information Processing Systems*, ser. NIPS '22, New Orleans, LA, USA: Curran Associates Inc., 2024, ISBN: 9781713871088.

[128] M. Miró-Nicolau, A. Jaume-i-Capó, and G. Moyà-Alcover, "A novel approach to generate datasets with xai ground truth to evaluate image models," *arXiv preprint arXiv:2302.05624*, 2023.

[129] P. Cortez and M. J. Embrechts, "Using sensitivity analysis and visualization techniques to open black box data mining models," *Information Sciences*, vol. 225, pp. 1–17, 2013, ISSN: 0020-0255.

[130] L. Arras, A. Osman, and W. Samek, "Clevr-xai: A benchmark dataset for the ground truth evaluation of neural network explanations," *Information Fusion*, vol. 81, pp. 14–40, 2022, ISSN: 1566-2535.

[131] Transport for London, *Connected and autonomous vehicles statement*, Online, Local Government Statement, Available: https://content.tfl.gov.uk/connected-and-autonomous-vehicle-statement.pdf, 2019.

[132] S. Krishna *et al.*, "The disagreement problem in explainable machine learning: A practitioner's perspective," *Transactions on Machine Learning Research*, 2024, ISSN: 2835-8856.

[133] J. Kim, A. Rohrbach, T. Darrell, J. Canny, and Z. Akata, "Textual explanations for self-driving vehicles," in *Computer Vision – ECCV 2018*, V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, Eds., Cham: Springer International Publishing, 2018, pp. 577–593.

[134] L. Faber, A. K. Moghaddam, and R. Wattenhofer, "When comparing to ground truth is wrong: On evaluating gnn explanation methods," in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, ser. KDD '21, Virtual Event, Singapore: Association for Computing Machinery, 2021, pp. 332–341, ISBN: 9781450383325.

[135] B. Russell, A. Torralba, K. Murphy, and W. Freeman, "Labelme: A database and web-based tool for image annotation," *International Journal of Computer Vision*, vol. 77, May 2008.

[136] A. Ess, B. Leibe, and L. Van Gool, "Depth and appearance for mobile scene analysis," in *2007 IEEE 11th International Conference on Computer Vision*, 2007, pp. 1–8.

[137] C. G. Keller, M. Enzweiler, and D. M. Gavrila, "A new benchmark for stereo-based pedestrian detection," in *2011 IEEE Intelligent Vehicles Symposium (IV)*, 2011, pp. 691–696.

[138] P. Dollar, C. Wojek, B. Schiele, and P. Perona, "Pedestrian detection: An evaluation of the state of the art," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 4, pp. 743–761, 2012.

[139] C. Sakaridis, D. Dai, and L. Van Gool, "ACDC: The adverse conditions dataset with correspondences for semantic driving scene understanding," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, Oct. 2021.

[140] M. Ozuysal, V. Lepetit, and P. Fua, "Pose estimation for category specific multiview object localization," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 778–785.

[141] M. Cordts *et al.*, "The cityscapes dataset for semantic urban scene understanding," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[142] F. Yu *et al.*, "Bdd100k: A diverse driving dataset for heterogeneous multitask learning," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2020*, 2020.

[143] M. Everingham *et al.*, "The pascal visual object classes challenge: A retrospective," *International Journal of Computer Vision*, vol. 111, pp. 98–136, 2015.

[144] M. Everingham, L. V. Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *International Journal of Computer Vision*, vol. 88, pp. 303–338, 2010.

[145] J. Deng *et al.*, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 248–255.

[146] G. Pandey, J. R. McBride, and R. M. Eustice, "Ford campus vision and lidar data set," *International Journal of Robotics Research*, vol. 30, no. 13, pp. 1543–1552, 2011.

[147] P. Dollár, C. Wojek, B. Schiele, and P. Perona, "Pedestrian detection: An evaluation of the state of the art," *IEEE transactions on pattern analysis and machine intelligence*, vol. 34, pp. 743–61, Jul. 2011.

[148] D. Du *et al.*, "Visdrone-det2020: The vision meets drone object detection in image challenge results," in *Computer Vision – ECCV 2020 Workshops*, A. Bartoli and A. Fusiello, Eds., Cham: Springer International Publishing, Jan. 2020, pp. 692–712, ISBN: 978-3-030-66822-8.

[149] D. Du *et al.*, "Visdrone-det2019: The vision meets drone object detection in image challenge results," in *ICCV visdrone workshop*, Oct. 2019.

[150] D. Du *et al.*, "The unmanned aerial vehicle benchmark: Object detection and tracking," in *European Conference on Computer Vision*, 2018.

[151] Ouster, Inc., *Software user manual*, Firmware v2.0.0 for all Ouster sensors, Rev F, San Francisco, CA, USA, 2021.

[152] Django Software Foundation, *Django: The web framework for perfectionists with deadlines*, Version 4.2.7, accessed September 16, 2024, 2024.

[153] OpenStreetMap contributors, *Planet dump retrieved from https://planet.osm.org*, https://www.openstreetmap.org, 2017.

[154] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, *The PASCAL Visual Object Classes Challenge 2011 (VOC2011) Results*, http://www.pascal-network.org/challenges/VOC/voc2011/workshop/index.html.

[155] G. Jocher, A. Chaurasia, and J. Qiu, *Ultralytics yolov8*, version 8.0.0, Available: https://github.com/ultralytics/ultralytics, 2023.

[156] J. Sklansky, "Finding the convex hull of a simple polygon," *Pattern Recognition Letters*, vol. 1, no. 2, pp. 79–83, 1982, ISSN: 0167-8655.

[157] H. Alhaija, S. Mustikovela, L. Mescheder, A. Geiger, and C. Rother, "Augmented reality meets computer vision: Efficient data generation for urban driving scenes," *International Journal of Computer Vision (IJCV)*, 2018.

[158] R. Achanta *et al.*, "Slic superpixels compared to state-of-the-art superpixel methods," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 11, pp. 2274–2282, 2012.

[159] R. Achanta *et al.*, "Slic superpixels compared to state-of-the-art superpixel methods," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 11, pp. 2274–2282, 2012.

[160] M. Sabih, F. Hannig, and J. Teich, "Utilizing explainable ai for quantization and pruning of deep neural networks," *ArXiv*, vol. abs/2008.09072, 2020.

[161] T.-Y. Lin *et al.*, "Microsoft coco: Common objects in context," in *Computer Vision – ECCV 2014*, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds., Cham: Springer International Publishing, 2014, pp. 740–755, ISBN: 978-3-319-10602-1.

[162] M. M. Pasandi, M. Hajabdollahi, N. Karimi, and S. Samavi, "Modeling of pruning techniques for deep neural networks simplification," *ArXiv*, vol. abs/2001.04062, 2020.

[163] S.-K. Yeom *et al.*, "Pruning by explaining: A novel criterion for deep neural network pruning," *Pattern Recognition*, vol. 115, p. 107 899, 2021, ISSN: 0031-3203.

[164] D. Lundstrom, A. Huyen, A. Mevada, K. Yun, and T. Lu, "Explainability tools enabling deep learning in future in-situ real-time planetary explorations," in *2022 IEEE Aerospace Conference (AERO)*, 2022, pp. 1–8.

[165] K. Wagstaff *et al.*, *Mars image content classification: Three years of nasa deployment and recent advances*, Presented at the AAAI Conference on Artificial Intelligence, 2021.

[166] D. W. Blalock, J. J. G. Ortiz, J. Frankle, and J. V. Guttag, "What is the state of neural network pruning?" *ArXiv*, vol. abs/2003.03033, 2020.

[167] N. Lee, T. Ajanthan, and P. Torr, "SNIP: SINGLE-SHOT NETWORK PRUNING BASED ON CONNECTION SENSITIVITY," in *International Conference on Learning Representations*, 2019.

[168] D. Alvarez-Melis and T. Jaakkola, "On the robustness of interpretability methods," *ArXiv*, vol. abs/1806.08049, 2018.