

FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Explainable AI in Object Detection for Autonomous Driving

Caio Macêdo Nogueira



Mestrado em Engenharia Informática e Computação

Supervisor: João Nuno Duarte Fernandes

Co-Supervisor: Jaime dos Santos Cardoso

July 27, 2023

Explainable AI in Object Detection for Autonomous Driving

Caio Macêdo Nogueira

Mestrado em Engenharia Informática e Computação

Approved in oral examination by the committee:

Chair: Luís Filipe Teixeira

External Examiner: Wilson Silva

Supervisor: João Nuno Duarte Fernandes

July 27, 2023

Abstract

Deep learning has rapidly increased in popularity, developing perception solutions for autonomous driving. From automatic parking to fully automated vehicles, the demand for neural driving models has increased as manufacturers look to improve their users' driving experience. The increase in computational power, namely the possibility of training deep learning models faster with high-end GPUs, led to the development of several computer-vision techniques. With these techniques, deep neural networks can be developed for autonomous driving tasks (e.g., object detection, image segmentation) to provide valuable decisions while operating in a vehicle. On the other hand, it is impossible to test an autonomous driving model in all possible scenarios. As such, deploying these models into the real world motivates them to be explainable or interpretable.

This dissertation explores explainable AI techniques for the object detection task in the context of autonomous driving. The work carries out an extensive and detailed comparison between gradient-based techniques and perturbation-based methods (e.g., D-RISE). Moreover, several experimental setups are used with different backbone architectures and different datasets to observe the influence of these aspects in the explanations. All techniques explored consist of saliency methods, making their interpretation and evaluation primarily visual. Nevertheless, numerical assessment methods are also used (e.g., Pointing Game metric). Overall, D-RISE and guided backpropagation obtain more localized explanations with 0.75 and 0.757 points of game scores, respectively. However, D-RISE highlights more meaningful regions, providing more human-understandable explanations.

As far as we know, this is the first approach to obtaining explanations focusing on the regression of the bounding box coordinates.

Keywords: Explainable AI, Autonomous driving, Object Detection

ACM Classification: Computing methodologies → Machine Learning, Computing methodologies → Machine Learning → Machine Learning approaches → Neural networks

Resumo

A aprendizagem profunda tem aumentado rapidamente a sua popularidade na literatura, desenvolvendo soluções para tarefas de percepção em condução autónoma. Desde o estacionamento automático até aos veículos totalmente automatizados, a procura de modelos de condução neural tem aumentado à medida que os fabricantes procuram melhorar a experiência de condução dos seus utilizadores. O aumento do poder computacional, nomeadamente a possibilidade de treinar modelos de aprendizagem profunda mais rapidamente com GPUs topo de gama, levou ao desenvolvimento de várias técnicas de visão computacional. Com estas técnicas, as redes neuronais profundas podem ser desenvolvidas para tarefas de condução autónoma (por exemplo, deteção de objectos, segmentação de imagens) para fornecer decisões valiosas enquanto se opera num veículo. Por outro lado, é impossível testar um modelo de condução autónoma em todos os cenários possíveis. Como tal, a implementação destes modelos no mundo real motiva a que sejam explicáveis ou interpretáveis.

Esta dissertação explora técnicas de IA explicáveis para a tarefa de deteção de objectos no contexto da condução autónoma. O trabalho efectua uma comparação extensa e detalhada entre técnicas baseadas em gradientes e métodos baseados em perturbações (por exemplo, D-RISE). Além disso, são utilizadas várias configurações experimentais com diferentes arquitecturas de backbone e diferentes conjuntos de dados para observar a influência destes aspectos nas explicações. Todas as técnicas exploradas consistem em métodos de saliência, tornando a sua interpretação e avaliação essencialmente visuais. No entanto, também são utilizados métodos de avaliação numérica (por exemplo, a métrica do jogo de apontar). Em geral, o D-RISE e o backpropagation guiado obtêm explicações mais localizadas com 0,75 e 0,757 pontos de pontuação do jogo, respetivamente. No entanto, o D-RISE destaca regiões mais significativas, fornecendo explicações mais compreensíveis para os humanos.

Tanto quanto sabemos, esta é a primeira abordagem para obter explicações centradas na regressão das coordenadas da caixa delimitadora.

Palavras-chave: Explicabilidade IA, Condução autónoma, Deteção de Objetos

Classificação ACM: Metodologias Computacionais → Aprendizagem Computacional, Metodologias Computacionais → Aprendizagem Computacional → Técnicas de Aprendizagem Computacional → Redes Neuronais

Acknowledgements

I would like to express my heartfelt gratitude to my family for their continuous support and encouragement throughout this dissertation journey.

I am sincerely thankful to my supervisor, João Fernandes, and co-supervisor, Jaime Cardoso, for their invaluable guidance and support. Their expertise and mentorship have been instrumental in shaping my research.

Special thanks go to Luís Fernandes for helping me get familiar with explainable AI. His assistance and knowledge-sharing have been immensely valuable.

I am also grateful to my friends for their unwavering support and encouragement throughout this journey.

I would like to extend my sincere gratitude to the Project THEIA: Automated Perception Driving (POCI-01-0247-FEDER-047264) for providing me with the opportunity to carry out this dissertation. Their support and resources have been instrumental in facilitating my research and academic growth.

In conclusion, I extend my deep appreciation to everyone who has contributed to my academic and personal growth during this dissertation. Your support has been indispensable, and I am truly grateful for each and every one of you.

Caio Nogueira

*“I am putting myself to the fullest possible use,
which is all I think that any conscious entity can ever hope to do.”*

2001: A Space Odyssey (HAL 9000)

Contents

1	Introduction	1
1.1	Context	1
1.2	Motivation	2
1.3	Goal	2
1.4	Thesis Structure	3
2	Fundamentals	5
2.1	Deep Neural Networks	5
2.2	Feature Extraction	8
2.3	Object Detection	9
2.3.1	R-CNN	10
2.3.2	YOLO	10
2.4	Deep Learning Metrics	12
3	Related work	15
3.1	Post-hoc Methods	15
3.1.1	Counterfactual analysis	16
3.1.2	Gradient-based Methods	16
3.1.3	Perturbation-based Methods	19
3.1.4	Local Approximation Methods	20
3.2	Explainable models	21
3.2.1	ProtoPNet	21
3.2.2	Other approaches	22
3.3	xAI for regression models	23
3.4	Explainable AI Metrics	26
4	Datasets and Models	29
4.1	Datasets	29
4.1.1	Pascal VOC Dataset	29
4.1.2	The KITTI Dataset	30
4.1.3	Exploratory Data Analysis	31
4.2	Models	32
5	Gradient-based Explanations	37
5.1	Grad-CAM	37
5.1.1	Scalar Explanations	38
5.1.2	Contrastive Explanations	41
5.2	Guided Backpropagation	45

5.3	Smooth Grad	47
5.4	Comparison of Methods	49
6	Perturbation-based approach	51
6.1	Experimental Setup	51
6.2	Contrastive Explanations with D-RISE	53
6.3	Comparison with Gradient-based Methods	55
7	Conclusions	59
7.1	Further Work	60
	References	61
A	Appendix	65

List of Figures

2.1	Most popular activation functions. Extracted from [10].	6
2.2	Illustration of a deep neural network. Extracted from [43].	7
2.3	Overview of a typical CNN architecture. Extracted from [26].	9
2.4	Faster R-CNN architecture. The CNN extracts features that are used by the RPN and the classifier. Extracted from [34].	11
2.5	YOLOv3 architecture overview. Extracted from [18].	12
2.6	Confusion matrix example over classes (1 and 2).	13
2.7	Intersection-over-Union (IoU) visualization and formula.	14
3.1	Counterfactual explanations generated by the STEEX framework. The regions highlighted contain the modifications made to the original images. In the first image, the decision changed because the car in front starts moving and the traffic light is green, while the second image contains the reverse situation.	16
3.2	Taxonomy for counterfactual approaches in xAI. Proposed by Chou et al. (2021) [6]	17
3.3	Pilotnet [3] saliency maps. The most valuable pixels for the model are highlighted in green. The maps displayed were obtained using <i>VisualBackprop</i> . As expected, the algorithm highlights the lanes and other vehicles on the road.	18
3.4	Overview of the D-RISE approach. Extracted from [28].	20
3.5	LIME and SHAP methods create a new (interpretable) linear classifier that explains the predictions made by the black-box model around the vicinity of data points.	21
3.6	ProtoPNet architecture. In the original paper, the authors used ProtoPNet for bird species classification. Adapted from Singh & Yow, 2021 [39].	22
3.7	Attention map obtained from Kim & Canny. (2017) [19]. The attention map introduces interpretability to the model	24
3.8	Differences between xAI for classification and for regression. The explanation of the model's predictions is generated based on a user-specified reference value $\tilde{y}$, which can significantly impact the explanation provided [20].	25
3.9	Saliency map from grad-CAM for an image quality assessment task. Extracted from [29].	26
4.1	Class distribution for the Pascal VOC 2012 train dataset.	30
4.2	Class distribution for the Kitti train + validation dataset.	31
4.3	Example image from the Kitti dataset.	33
4.4	Example images extracted from the Pascal VOC dataset.	33
4.5	Architecture diagram of ResNet-50. Extracted from [1].	34
4.6	Train and validation loss for the regression model.	35

4.7	Loss decrease for Faster-RCNN using ResNet-50 and ResNet-101 trained with the Kitti dataset.	36
5.1	Diagram of the experimental setup used in section 5.1.1	38
5.2	Grad-CAM explanations after applying transformations to the original bounding box. (a) and (b) are generated using the ResNet-101 backbone. (c) and (d) use Grad-CAM with the distance measure to compare both architectures.	39
5.3	Grad-CAM using scalar transformations. (a), (b) and (c) are generated with the distance value. (d), (e) and (f) are generated with the slope value. (c) and (f) are generated using the model that does not perform class discrimination.	40
5.4	Overview of the Grad-CAM contrastive approach.	43
5.5	Contrastive explanations for a sample image in the VOC dataset. The red bounding boxes are predicted by the model, whereas the green rectangles represent contrastive examples.	44
5.6	Contrastive explanations in the Kitti dataset.	44
5.7	Guided Backpropagation examples using the ResNet-101* backbone.	46
5.8	Smooth Grad examples using Grad-CAM against the Kitti dataset. (a) and (c) show Grad-CAM results for two distinct examples. (b) and (d) contain the Smooth Grad heatmaps for each example.	48
5.9	Guided backpropagation with noise reduction using SmoothGrad.	49
6.1	Saliency maps using the off-the-shelf D-RISE algorithm.	52
6.2	D-RISE interpretation with probability threshold of 0.25.	53
6.3	D-RISE generated with localization information. The classification is disregarded for the heatmap generation.	54
6.4	Contrastive explanations with D-RISE. Most activated areas are in the region that exclusively belongs to the predicted bounding box.	54
6.5	Saliency maps using D-RISE and gradient-based methods. All interpretations are generated with respect to the same image.	55
6.6	Saliency maps using D-RISE and gradient-based. The red bounding boxes are predicted by the model, whereas the green rectangles represent contrastive examples.	56
A.1	Explanations using individual coordinates.	65
A.2	Contrastive analysis with guided backpropagation.	65
A.3	Guided backpropagation with slope transformation.	66
A.4	Guided backpropagation with different backbone architectures.	66

List of Tables

4.1	Exploratory Data Analysis on datasets explored in subsections 4.1.1 and 4.1.2. . .	32
4.2	Performance of all models measured with validation sets of both Pascal VOC and Kitti datasets.	34
5.1	Pointing Game scores on Pascal VOC. Higher values mean better performance. .	41
5.2	Pointing Game scores for Grad-CAM on the Kitti dataset. Higher values mean better performance.	41
5.3	Pointing Game scores for contrastive explanations on kitti. Higher values mean better performance.	45
5.4	Pointing Game scores for Guided Backpropagation. Higher values mean better performance.	47
5.5	Pointing Game scores for SmoothGrad. Higher values mean better performance. .	49
5.6	Time comparison for all gradient-based methods. Each value contains the elapsed time (in seconds) after generating a heatmap using each approach.	50
6.1	Pointing Game scores for D-RISE explanations.	56
A.1	Pointing Game scores for contrastive explanations on kitti using SmoothGrad. . .	66

Abbreviations

AI	Artificial Intelligence
AP	Average Precision
CNN	Convolutional Neural Network
D-RISE	Detector Randomized Input Sampling for Explanation
DL	Deep Learning
DNN	Deep Neural Network
FPS	Frames per Second
Grad-CAM	Gradient-weighted Class Activation Mapping
LiDAR	Light Detection and Ranging
LIME	Local Interpretable Model-agnostic Explanations
MAE	Mean Absolute Error
mAP	Mean Average Precision
MSE	Mean Squared Error
ProtoPNet	Prototypical Part Network
ResNet	Residual Neural Network
R-CNN	Region-based CNN
RMSE	Root Mean Squared Error
RoI	Region of Interest
RPN	Region Proposal Network
SHAP	SHapley Additive exPlanations
VGG	Vision Geometry Group
xAI	Explainable AI

Chapter 1

Introduction

1.1 Context

In recent years, autonomous driving has received increasing attention from the engineering industry. The demand for autonomous driving algorithms is rising as cars implement them for different high-level goals (e.g., automatic parking, driving assistance, and fully-autonomous driving). Car manufacturers deploy these models to improve the driving experience of their users. Moreover, the performance of current state-of-the-art models allows for reducing the chance of human error in traffic. However, there are challenges in developing a fully autonomous vehicle: cars should operate in different environments and potentially provide different services under distinct circumstances.

The increase in overall computational power (e.g., faster high-end GPUs) increased the popularity of the computer vision research field, as it enhanced deep neural networks, which leverage parallel calculations. Consequently, deep learning engineers started working on implementing computer-vision techniques adapted for autonomous driving tasks (e.g., object detection and image segmentation). State-of-the-art models can make valuable decisions that will help a human driver or ultimately operate the vehicle automatically. Furthermore, the medicine field is also a widely used application with regards to computer-vision architectures.

The development of the sensor industry lowered the costs of implementing these algorithms. Autonomous vehicles are equipped with 3D sensors, e.g., LiDAR, which collect a 3D point cloud of the car's surroundings. RGB cameras also extract information about the environment where the car is situated. These data acquisition methods are the base for perception algorithms. Engineers commonly use Convolutional Neural Networks (CNN) in deep learning to extract knowledge from image or spatial data.

With the development of new autonomous driving models and due to the highly critical nature of the task, the industry demands high performance and that the models are testable under all scenarios. Since it is impossible to list the complete list of situations the model may face in the

real world, the next best thing is to develop models whose decisions can be explained. However, getting these explanations is complex due to the black-box nature of neural networks. As such, a connection is created between autonomous driving and Explainable AI (xAI). In this context, explainability is a need for different reasons: engineers can better understand malfunctions in the system and prevent corner cases, while users increase their trust in the model and, in the future, decisions made by the model can teach human drivers. Legal aspects [15] should also be kept in consideration.

Explainable AI is a research topic that aims to develop AI systems whose decisions can be understood by humans. xAI establishes a difference between interpretable models and intrinsically explainable ones. Interpretability has to do with how comprehensible the model is for humans. On the other hand, a model is explainable when we can understand its decisions in all situations. We can achieve interpretability by using post-hoc methods independent from the models. Developing an intrinsically explainable model is more challenging since the network architecture needs to allow these explanations.

This work is associated with the THEIA project. The latter consists of a partnership between the University of Porto and Bosch, aiming to develop intelligent perception techniques in autonomous driving. As such, this project researches techniques in an automated driving context (e.g., object detection, semantic segmentation).

1.2 Motivation

Despite the interest in Explainable AI models, the literature regarding explainable models for autonomous driving tasks, such as object detection, is limited. Therefore, it is possible to adapt state-of-the-art xAI methods (e.g., [36], [28]) for obtaining interpretations regarding inferences made by a model. With this model-agnostic approach, the performance of black-box models is not harmed in any way. Moreover, a good technique for explaining bounding box coordinates can be incorporated in other intrinsically explainable architectures that are already able to obtain global explanations for the classification head (e.g., [5]).

1.3 Goal

This project aims to research explainable AI techniques to obtain interpretations for object detection models, allowing humans to understand the placement of bounding boxes for each object captured in an image. Some model-agnostic techniques can be used involving perturbation-based methods or other straightforward applications, e.g., D-RISE [28], Grad-CAM [36]. The end goal compare the results obtained across different techniques while assessing which factors play a crucial role in the quality of explanations. The contributions of this study are related to the results obtained, serving as a first approach to obtaining explanations for object detection in the autonomous driving context.

1.4 Thesis Structure

This document is structured into seven chapters.

Chapter 1 introduces object detection and autonomous driving. Moreover, it also explains the goals of this project and its relevance surrounding its elaboration.

Chapter 2 introduces topics well established among the deep learning literature. Moreover, it focuses on the object detection task, exploring the different phases of this process. In addition, the referred chapter explains the most used object detection architectures and their advantages and disadvantages. Moreover, the metrics used to evaluate detectors are also uncovered.

Chapter 3 does a comprehensive review regarding the related work connected to topics relevant to this project. The main subjects considered in this chapter are Explainable AI (xAI) techniques, including model-agnostic and intrinsically explainable methods. It provides an overall complement to the fundamentals chapter, expanding on the concepts thereby explored.

Chapter 4 explores the two datasets used throughout the dissertation. Exploratory data analysis is performed to outline the significant differences between the datasets. This discussion comprises aspects such as the class distributions and overall characteristics of the objects in each dataset. In addition, details regarding the black-box models are provided, including their evaluation with respect to the relevant object detection metrics.

Chapter 5 explores gradient-based methods to obtain explanations. As such, after a proper introduction to these methods, this chapter explores adaptations made to them to fit the context of object detection (e.g., scalar transformations, contrastive explanations). Moreover, the results are compared for different models and datasets, disclosing the most important aspects that influence the quality of explanations. In addition, an overall comparison regarding different gradient-based methods is also established, considering subjective and objective metrics.

Chapter 6 investigates the experimental setup and evaluates the effectiveness of D-RISE in providing interpretable explanations using the Kitti dataset. Comparative analyses are conducted to understand the strengths and limitations of D-RISE and its potential for improving the interpretability of autonomous driving systems. Additionally, the chapter compares D-RISE with gradient-based methods. The results demonstrate that D-RISE produces visually superior and more localized explanations, highlighting its potential as a valuable tool for understanding the behavior of object detection models in autonomous driving scenarios.

Chapter 7 concludes the dissertation, analyzing the critical findings resulting from the methodologies adopted. Moreover, an overview of possible future work is also delineated.

Chapter 2

Fundamentals

In this chapter, some of the most relevant aspects regarding computer-vision are described, namely for autonomous driving related tasks. With this in mind, subsection 2.1 makes an introduction on the behavior of Deep Neural Networks. Subsection 2.2 discusses feature extraction techniques for spatial inputs, highlighting deep-learning based approaches. After establishing the basic concepts of feature extraction and Convolutional Neural Networks (CNNs), subsection 2.3 explains the object detection task, as well as the most widely used techniques for it. Finally, subsection 2.4 explores the metrics for the object detection task that will be relevant in the context of this work.

2.1 Deep Neural Networks

A Deep Neural Network is a machine learning model composed of multiple layers of interconnected nodes known as neurons. Each neuron receives inputs from the previous layer, performs a dot product of inputs with its weights and applies an activation function to generate an output. These outputs are then fed as inputs to the next layer. Figure 2.1 show the most common activation functions.

Furthermore, we can summarize a deep neural network in three components (figure 2.2):

- Input layer: The first layer receives inputs from the outside world and passes them through the network.
- Hidden layers: Multiple hidden layers process the input data using weights and activation functions to generate a set of outputs.
- Output layer: The final layer generates predictions or decisions based on the processed information from the hidden layers.

In order to evaluate the performance of a model, a loss function is used, which consists of a mathematical function. The loss function acts as a feedback mechanism for the model, indicating

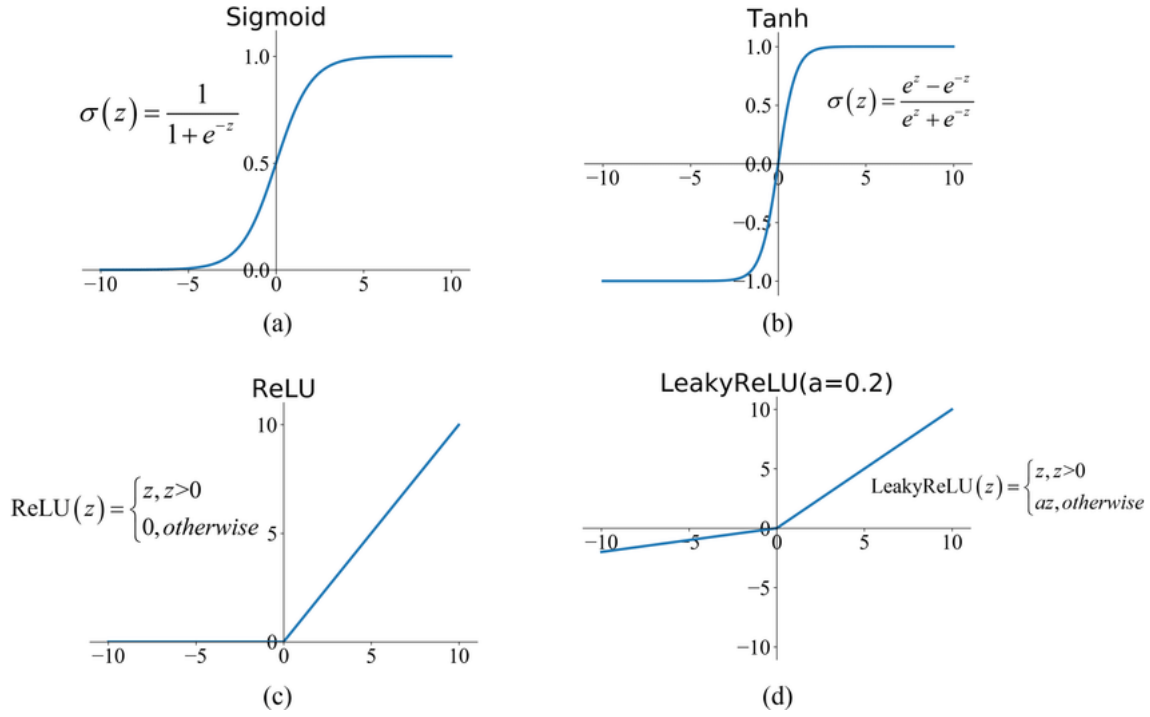


Figure 2.1: Most popular activation functions. Extracted from [10].

how well the model makes predictions during training. When the loss is high, the model is not performing well and needs to be adjusted. Moreover, the gradient of the loss function with respect to the model's parameters (weights and biases) is used to update these same parameters. This update results in the iterative decrease of the loss function across epochs. Equation 2.1 contains the update rule for each layer of a neural network: W'_x and W_x are the new and old weights matrix respectively; λ is the learning rate; L is the loss function.

$$W'_x = W_x - \lambda \left(\frac{\partial L}{\partial W_x} \right) \quad (2.1)$$

Classification tasks with deep neural networks commonly use the Cross Entropy Loss. It measures the dissimilarity between the predicted probabilities and the actual target distribution. Equation 2.2 contains the formula for Cross Entropy Loss, where Y and $\hat{Y}$ are targets and predictions, respectively.

$$\text{Loss}(Y, \hat{Y}) = - \sum Y \log(\hat{Y}) \quad (2.2)$$

Regarding loss functions for regression, the most widely used are: Root Mean Squared Error, Mean Squared Error, and Mean Absolute Error.

- Root Mean Squared Error (RMSE) outputs the difference between the targets and the predictions (equation 2.3).

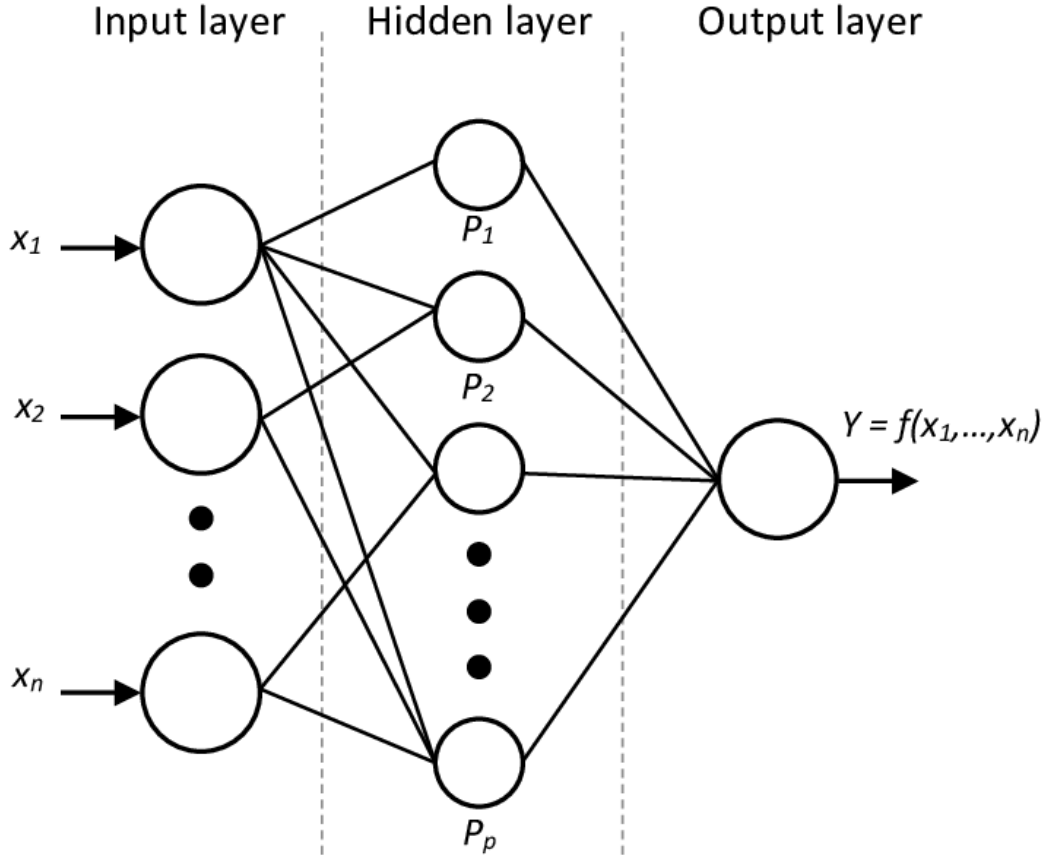


Figure 2.2: Illustration of a deep neural network. Extracted from [43].

$$RMSE(Y, \hat{Y}) = \sqrt{\frac{1}{N} \sum (Y - \hat{Y})^2} \quad (2.3)$$

- Mean Squared Error (MSE) is similar to the RMSE loss. The difference between both functions is the scale of the error. RMSE is expressed in the same units as the target variable. Equation 2.4 makes it clear that RMSE is the squared root of the MSE loss.

$$RMSE(Y, \hat{Y}) = \frac{1}{N} \sum (Y - \hat{Y})^2 \quad (2.4)$$

- Mean Absolute Error (MAE) measures the average absolute difference between the predicted values and the true values (equation 2.5).

$$MAE(Y, \hat{Y}) = \frac{1}{N} \sum |Y - \hat{Y}| \quad (2.5)$$

In object detection models, the smooth L1 loss (equation 2.6) is commonly used. This loss function penalizes larger errors more strongly than smaller errors when the absolute value of the difference is greater than 1.

$$L1 = \begin{cases} 0.5x^2 & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwise} \end{cases} \quad (2.6)$$

Choosing the loss function to use heavily impacts the performance of the model. As such, the right option depends on the nature of the problem.

Furthermore, early stopping is a common practice adopted by machine learning engineers to ensure that the model converges while preventing overfitting. This method occurs during training by tracking the performance of the model on the validation set. It is expected that the performance is optimized for the training and validation sets simultaneously during the first epochs. However, if enough epochs are performed, the loss increases on the validation set while continuing to decrease slowly on the training set, which means it starts to overfit the training data. As such, early stopping halts the training process before the model gets highly complex. The result of this method is a model that generalizes better on unseen data.

2.2 Feature Extraction

Feature extraction is the process of extracting meaningful features from images or videos representing the content of the image or video. These features are typically numerical values or vectors that can be used as input to a machine-learning model.

The most used feature extraction techniques include hand-crafted methods (e.g., HOG [7], SIFT [24]) and deep-learning based methods, such as Convolutional Neural Networks (CNNs). Each layer of the CNN is designed to extract specific types of features from the input image or video. For example, the first layers of a CNN may be intended to extract edges and simple shapes, while the last layers may be designed to extract more complex features such as patterns and textures. Once the CNN has processed the input image or video through all of its layers, the output is a set of features that represent the most important information in the image or video. These features can then be used for various tasks, such as image classification, object detection, and video analysis.

Figure 2.3 displays a visualization of a simple CNN architecture. The idea behind a convolution is to maintain the spatial characteristics of the input. Each convolutional layer uses a kernel, a matrix of weights with dimension n . As such, each layer propagates its input by computing matrix multiplications involving the kernel and all $n \times n$ windows of the input. Additionally, an activation function (e.g., sigmoid, ReLU) introduces non-linearity to the feature map produced. Furthermore, the architecture in the aforementioned model also comprises pooling layers, which reduce the dimension of the feature maps. In prediction tasks, it is also necessary to have at least one fully-connected layer which flattens the final feature map and produces logits for each class. The logits are fed to the softmax function resulting in a probabilistic distribution containing the probability of the input belonging to each class. State-of-the-art black-box models for image classification use CNN based feature extraction (e.g., ResNet [14], VGG [38]).

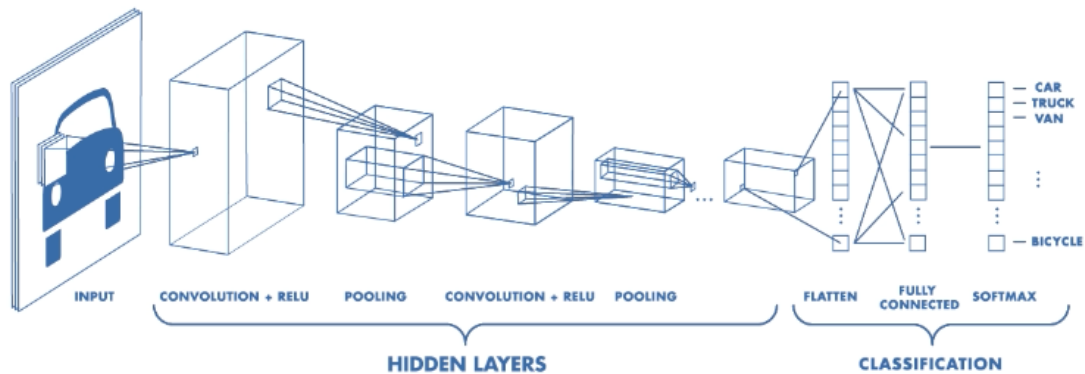


Figure 2.3: Overview of a typical CNN architecture. Extracted from [26].

2.3 Object Detection

Object detection is a computer vision task that involves identifying the presence, location, and characteristics of objects in an image or video. It is a critical component of many applications, such as autonomous vehicles, security systems, and image or video search engines. Moreover, developing an object detection model involves training a classifier that identifies the label of each object (e.g., "car", "person"). Semantic segmentation is an alternative to object detection for autonomous driving. Instead of producing coordinates that limit each object, this computer vision task assigns each pixel to a specific label.

The task of detecting objects is complex for deep learning and requires multiple steps. The first step in object detection is to preprocess the input image. This may involve resizing, cropping, and adjusting the color balance and brightness of the image to improve the quality of the input data. The preprocessing stage required depends on the quality of the dataset used.

After tidying up the data, it is ready to be passed to the feature extraction step, which will ultimately learn which parts of the image are essential to produce the output. As section 2.2 explains, learnable convolutional layers extract patterns within the images. Depending on the architecture of the detector, the information gathered in the feature extraction step flows into the network in a specific manner until it produces a respective output.

The post-processing step is also specific to object detection and valuable in removing redundant detections (i.e., bounding boxes that refer to the same object) relying on the non-max-suppression (NMS) algorithm. NMS compares the overlap between bounding boxes and selects the one with the highest confidence score. Bounding boxes with high overlap are considered duplicates and are removed from the final set of detections. However, NMS is a hard threshold and may not always produce the best results. Soft-NMS is a modification of the traditional NMS algorithm that replaces the hard threshold with a soft weighting function. This function assigns lower scores to bounding boxes with high overlap with other boxes instead of completely discarding

them, helping to retain some information from overlapping bounding boxes and resulting in better detection outputs.

Deep learning approaches to object detection often involve using convolutional neural networks. In this section, some popular architectures will be explained, as well as more recent iterations of those techniques.

2.3.1 R-CNN

Region-based Convolutional Neural Network (R-CNN) [13] is a family of methods that use features extracted from a CNN to classify objects within an image by obtaining a set of region proposals. R-CNN uses the selective search algorithm to obtain the candidate regions in a deterministic way. Each region is then fed to the CNN that assigns a label to the object included in that image.

Fast R-CNN [12] is an improvement of the original R-CNN created by the same author. The idea is to identify the regions in the convolutional feature map and reshape them into squares. This job is performed by a Region of Interest (RoI) pooling layer. Therefore, the training time is much faster because the convolutional operation is done only once. On the other hand, in the original R-CNN method, all candidate regions are fed to the CNN.

Faster R-CNN [34] is an extension of Fast R-CNN that uses a region proposal network (RPN) to generate region proposals instead of the selective search algorithm. The RPN is trained end-to-end with the rest of the Faster R-CNN model, which makes the method more efficient and accurate. The optimizer for this object detector contains regression and classification losses for both the RPN and the RoI head classification. The optimizer tries to minimize the sum of all losses, improving the overall performance of the network. Using the MS COCO dataset [23], Faster R-CNN achieves a mAP equivalent to 42.1%.

This object detection algorithm is considered a two-stage detector, as the region-based CNN and the RPN are separated from each other. This approach holds an advantage over one-stage detectors, as it is more accurate specially regarding smaller objects (i.e., objects that are farther away from the camera).

Regarding the architecture of the backbone, the authors conclude that using a ResNet-101 [14] increases the performance of the model.

2.3.2 YOLO

You Only Look Once (YOLO) [31] is a real-time object detection algorithm that is particularly fast and accurate. One of the key features of YOLO is that it processes the entire image in a single forward pass of the CNN, making it much faster than many other object detection algorithms that require multiple forward and backward passes through the network.

YOLO divides the input image into a grid of cells and predicts the bounding boxes and class probabilities for each cell. If an object is detected in a cell, the bounding box for the object is adjusted using predicted offsets relative to the cell. This allows YOLO to make more accurate

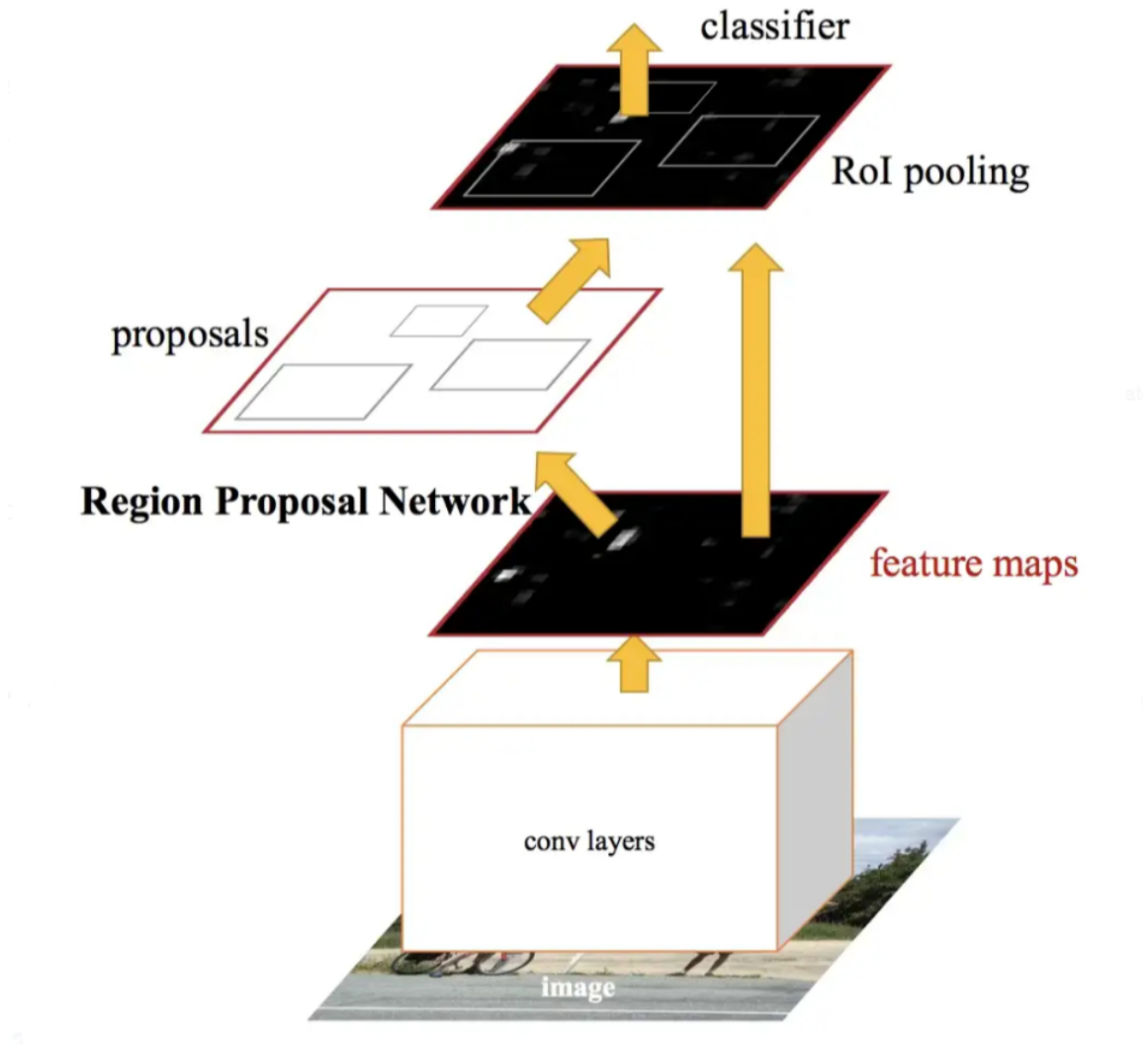


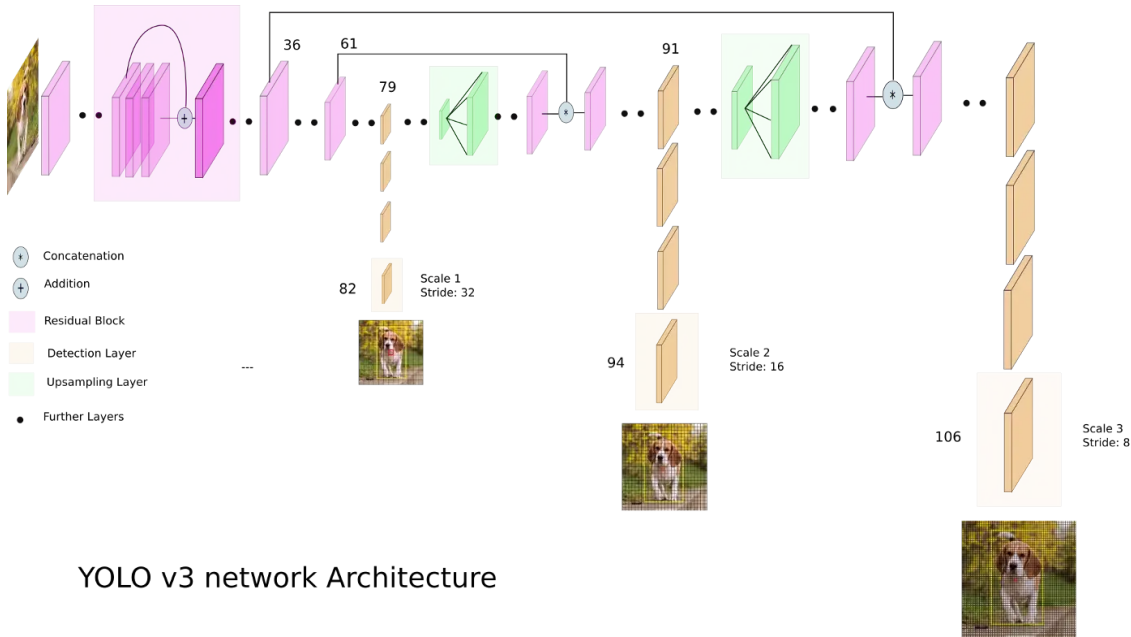
Figure 2.4: Faster R-CNN architecture. The CNN extracts features that are used by the RPN and the classifier. Extracted from [34].

predictions by taking into account the context of the entire image, rather than just looking at individual cells in isolation.

Redmon & Farhadi (2016) [32] created YoloV2, which improved the performance of the original YOLO algorithm, namely concerning the mean Average Precision (mAP) metric. The authors applied batch normalization in all convolutional layers. Moreover, the idea of anchor boxes allows each grid cell to detect more than one object. Another important aspect proposed in this paper was the integration of the Darknet architecture. Darknet-19 contains 19 convolutional layers and is used to train the model for classification and as a backbone for detection.

Subsequent iterations of YOLO improved the algorithm incrementally. YOLOv3 [33] introduced a more complex version of the backbone, namely Darknet-53. The latter uses 53 convolutional layers for classification and 106 for the detection task. The architecture of the YOLOv3 algorithm is displayed in figure 2.5.

The latest version of YOLO is YOLOv8 [17]. The goal of this iteration was to optimize the



YOLO v3 network Architecture

Figure 2.5: YOLOv3 architecture overview. Extracted from [18].

process by reducing the amount of computation necessary while improving the model's accuracy. The algorithm uses a single CNN to simultaneously predict multiple bounding boxes and class probabilities for objects in an image or video frame.

Overall, YOLO is a powerful and efficient object detection algorithm that has been widely adopted in a variety of applications. Its capacity to identify objects in real-time makes it suitable for multiple use cases (e.g., self-driving cars, surveillance systems, and robotics). Regarding results, YOLOv8 is the best object detector considering speed and accuracy. This last version of the algorithm obtains mean Average Precision up to 53.8% on the MS COCO dataset, depending on the amount of parameters used.

2.4 Deep Learning Metrics

Several metrics can be applied to assess the performance of object detection algorithms. Using different metrics allows more extensive analysis of the model. In this subsection, the most common evaluation metrics will be explained.

Precision and Recall: When we consider a prediction task where each sample is assigned a correct or incorrect label, it makes sense to use precision and recall. Moreover, in object detection, it is possible to use these values instead of IoU if a decision threshold is fixed. Therefore, we need to establish the concepts of True Positive (TP), True Negative (TN), False Positive (FP) and False Negative (FN) in object detection.

- True Positive (TP) refers to bounding boxes correctly predicted, i.e., IoU measure is greater than a specific threshold;

- True Negative (TN): represents regions of the input where no bounding boxes are predicted and no ground truth objects are present;
- False Positive (FP): represents bounding boxes that do not correspond to any ground truth or whose IoU metric is smaller than the decision threshold;
- False Negative (FN): refers to objects not identified by the model or whose predicted bounding box contains an IoU measure with the ground truth is smaller than the threshold.

Precision refers to the proportion of True Positives predicted among the total number of predictions, i.e., the percentage of objects correctly identified considering all detections 2.7. Consequently, this is a measure of efficiency: even when the model makes few predictions, Precision can be high.

Recall is a metric representing the number of predictions, considering the total number of objects in the ground truth 2.8. Therefore, Recall is considered to measure the model's effectiveness: the object detector is encouraged to make more predictions even if some result in False Positives.

$$Precision = \frac{TP}{TP + FP} \quad (2.7)$$

$$Recall = \frac{TP}{TP + FN} \quad (2.8)$$

The four components above can also be displayed in a confusion matrix 2.6.

		ACTUAL	
		Class 1 (Positive)	Class 2 (Negative)
PREDICTION	Class 1 (Positive)	TP	FP
	Class 2 (Negative)	FN	TN

Figure 2.6: Confusion matrix example over classes (1 and 2).

F1-score: This metric is a balance between precision and recall. It is calculated as the harmonic mean of precision and recall 2.9. As such, this is a good aggregated indicator because it considers both Precision and Recall.

$$\frac{2 \times Precision \times Recall}{Precision + Recall} \quad (2.9)$$

Average Precision (AP): measures the area under the Precision-Recall Curve. The recalls are segmented into N parts and AP calculates the average of the precisions obtained at each recall point (formula 2.10).

$$AP = \frac{1}{N} \sum_{r=1}^N p_{interp}(r) \quad (2.10)$$

Mean Average Precision (mAP): measures the precision of the detector across all classes. It is calculated using the Average Precision for each of the C classes. Equation 2.11 further explains the formula for this metric.

$$mAP = \frac{1}{C} \sum_i^N AP(i) \quad (2.11)$$

Intersection over Union (IoU): measures the intersection between two shapes by dividing the overlapped area by the total area occupied by both figures. In this work, IoU measures the similarity between the predicted bounding box and the ground truth.

$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$


Figure 2.7: Intersection-over-Union (IoU) visualization and formula.

Chapter 3

Related work

In the explainable AI (xAI) literature, there is a difference between explaining a neural model and developing an intrinsically explainable one. The architectures created for the latter task result in complex models. On the other hand, it is usually simpler to provide interpretations of the model using post-hoc methods. However, the latter methods often require additional computational resources and can be less interpretable compared to models that are inherently designed to provide explanations. Additionally, post-hoc explanations may not capture the full complexity of the underlying model, potentially leading to incomplete or misleading explanations.

The necessity for obtaining explanations from a deep learning model is a problem for the entire machine learning topic, as engineers never have control over the whole dataset. As such, most of the strategies involved in understanding the decisions taken by neural driving models are derived from other application domains (e.g., health industry, e-commerce).

This chapter will cover some state-of-the-art techniques regarding interpretability and explainability in neural driving models. In subsection 3.1, there is a brief review of post-hoc or model-agnostic techniques for interpreting black-box neural driving models. Subsection 3.2 describes state-of-the-art approaches for intrinsically explainable models. The topic of subsection 3.3 is related to regression and how can existing solutions for explaining classification be adapted for this task. In addition, section 3.4 explains common approaches for evaluating explanations obtained from the Explainable AI methods explored throughout this chapter.

3.1 Post-hoc Methods

Techniques known as model agnostic consist of methods applied after the model has been properly trained. These methods can be applied to any type of model, regardless of the specific algorithm or architecture used, and typically involve analyzing the input-output relationship of the model. The main benefit of being independent from the model is preserving its original performance.

In the context of this work, it is important to explore two main types of post-hoc methods: counterfactual analysis (subsection 3.1.1) and saliency methods (subsections 3.1.2, 3.1.3, 3.1.4).

3.1.1 Counterfactual analysis

The first model-agnostic technique consists of local explanations, i.e., given an input image, explaining why the model outputs a specific class. Moreover, we can distinguish between counterfactual analysis and saliency methods. In counterfactual analysis, researchers seek x' , a modified version of x . With this new input, it is possible to define which input features contributed to the outcome of the classification. There have been some contributions related to this technique [44], aiming to reduce both the distance between predictions and the distance between the inputs.

Jacob et al. (2022) [16] created the STEEX framework to find interpretable semantic modifications to the input image. Moreover, they introduced the concept of *region-targeted counterfactual explanations*, where users can specify a semantic region where the input should be modified. In figure 3.1, there are some examples of counterfactual explanations generated for a binary classifier that decides to "Move forward" or "Stop".



Figure 3.1: Counterfactual explanations generated by the STEEX framework. The regions highlighted contain the modifications made to the original images. In the first image, the decision changed because the car in front starts moving and the traffic light is green, while the second image contains the reverse situation.

Furthermore, Chou et al. (2021) [6] proposed a new taxonomy for model-agnostic counterfactual approaches. In total, the authors reviewed 26 distinct techniques that were aggregated into seven categories (figure 3.2). Some methods (e.g., SHAP, LIME) are based on techniques explored in subsection 3.1.4.

3.1.2 Gradient-based Methods

Gradient-based methods is classified as a saliency method. Saliency methods are a type of xAI technique that aim to provide a visual representation of which features or input data points are most important or influential in a model's output. There are several different saliency methods that have been proposed in the literature, but most of them can be broadly classified into three categories: gradient-based methods, perturbation-based methods and local approximations.

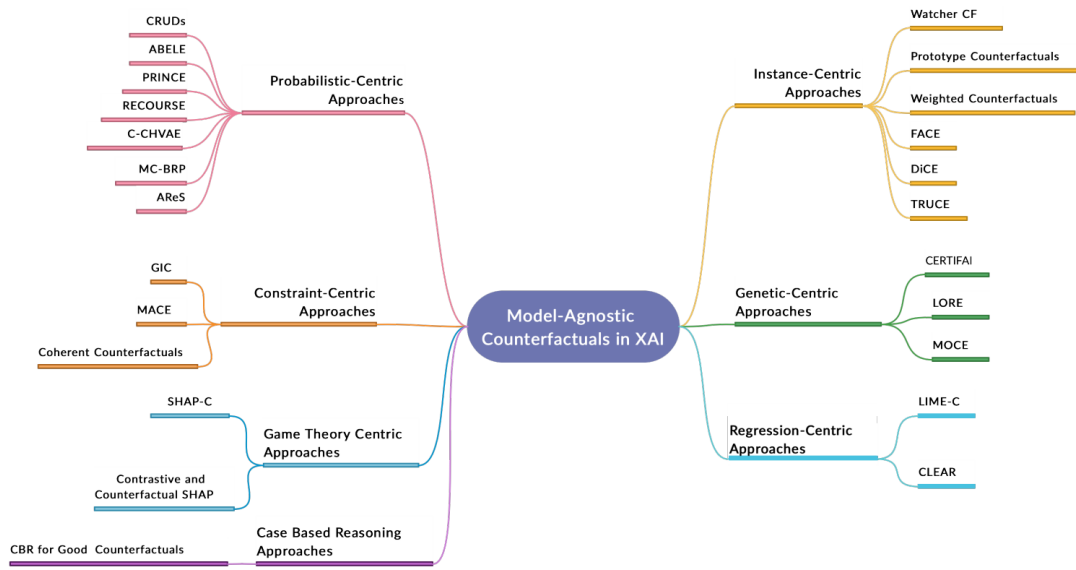


Figure 3.2: Taxonomy for counterfactual approaches in xAI. Proposed by Chou et al. (2021) [6]

Gradient-based methods perform retro-propagation of information in the network, evaluating the gradient of the output to compute a heat map consisting of the most contributing regions. Figure 3.3 shows three examples of saliency maps generated using gradient-based methods. The heat maps can be calculated at an intermediate level or with respect to the input itself. These methods are usually fast and easy to implement, but they are sensitive to noise and can sometimes provide misleading explanations. Zeiler & Fergus (2013) [50] proposed a visualization technique called deconvnet, that reproduces individual feature maps at any layer of the network. More recently, deep learning researchers developed improved versions of this technique (e.g., deepLift [37], integrated gradients [42]).

Grad-CAM [36] is a gradient-based technique that intends to produce a saliency map, highlighting the most relevant features. The main difference between grad-CAM guided backpropagation (figure 3.3) is that grad-CAM retro-propagates a specific output score with respect to the parameters of the last feature extraction convolutional layer. The other method performs a complete backward pass of the information throughout the network, thus being able to highlight information directly into the input image. In classification tasks, the mentioned score is often the output classification score. However, the context of the work at hand forces an adaptation, as interpretations must be obtained for regression instead of classification. Grad-CAM is derived from CAM (Class Activation Maps), which works by using the global average pooling layer of a convolutional neural network (CNN) to obtain a feature map, and then weighting the feature map by the final classification layer to obtain a class activation map. This class activation map highlights the regions of the input image that were most important for the model's prediction.

Gradient-based methods derived from Grad-CAM, such as SmoothGrad [40] and Guided Grad-CAM, have been developed to address some of the limitations of the original method, such



Figure 3.3: Pilotnet [3] saliency maps. The most valuable pixels for the model are highlighted in green. The maps displayed were obtained using *VisualBackprop*. As expected, the algorithm highlights the lanes and other vehicles on the road.

as its sensitivity to small changes in the input image and its failure to account for negative contributions to the prediction. These methods use techniques such as noise injection and guided backpropagation to improve the accuracy and interpretability of the heat maps.

Guided backpropagation [41] focuses on modifying the gradients during the backpropagation process to emphasize positive contributions and suppress negative contributions. By discarding negative gradients, the technique highlights the regions in the input image that positively impact the network’s decision. This approach provides a pixel-level understanding of the important features but does not explicitly localize the feature boundaries. The main of this approach, compared to Grad-CAM, is in the calculation of the gradients. Guided backpropagation utilizes the gradients of the loss (or other context-specific outputs) with respect to the original image, whereas Grad-CAM operates with the feature maps of the last convolutional layer.

The SmoothGrad algorithm is designed to improve the interpretability and robustness of neural

networks by reducing the noise and uncertainty in their predictions. It achieves this by incorporating gradient noise while obtaining model predictions. The algorithm can be summarized in the following steps:

- The algorithm takes an input image as its starting point. This image is usually the one for which we want to understand the neural network’s decision-making process.
- The algorithm applies small random perturbations to the input image to introduce gradient noise. These perturbations can be random pixel noise or other variations, such as blurring or scaling.
- The perturbed image is fed through the neural network, and the forward pass is performed. This step aims to observe how the perturbation affects the predictions.
- The gradients of the output with respect to the input image are computed. This step involves backpropagating the gradients through the network to understand the sensitivity of the predictions to changes in the input image.
- The gradients obtained from multiple perturbed versions of the input image are accumulated or averaged. This step helps reduce the noise introduced by individual perturbations and provides a smoother representation of the gradients.

3.1.3 Perturbation-based Methods

Perturbation-based methods make slight modifications to the features in a specific region of the input and observe the behavior of the network in these new circumstances. These methods are generally more robust and reliable than gradient-based methods, but they can be computationally expensive and may not scale well to large datasets. The perturbations can be produced at the input level (i.e., modifying the input data by adding noise, removing or occluding parts of the input, or changing the values of individual features, and then observing how the results produced by the model changes in response). Alternatively, engineers can rank the input features by their importance to the output of the model, based on how much this output changes when the values of the features are modified. The latter technique is known as feature importance perturbations.

Ribeiro et al. (2016) [35] proposed the super-pixel, an interpretable representation for image classification that represents the existence or lack thereof of a contiguous patch of similar pixels.

Petsiuk et al. (2021). [28] proposes the Detector Randomized Input Sampling for Explanation (D-RISE) technique. This technique generates a saliency map highlighting which parts of the image contribute to the detection. In this sense, its goals are equivalent to gradient-based methods (subsection 3.1.2). However, D-RISE accomplishes its objective with subsequent random perturbations to the input image. The perturbed images are fed to the detector, which returns a set of bounding boxes and classes corresponding to those regions. A vector is produced for each detection, compiling information regarding localization, classification, and the objectness score (for models that output this measure). In addition, each target bounding box also contains its own

detection vector, used to compute pairwise similarity scores with detections generated from perturbed input images. The saliency map is generated from the result of the weighted sum of all masks, where the weights are given by the similarity scores.

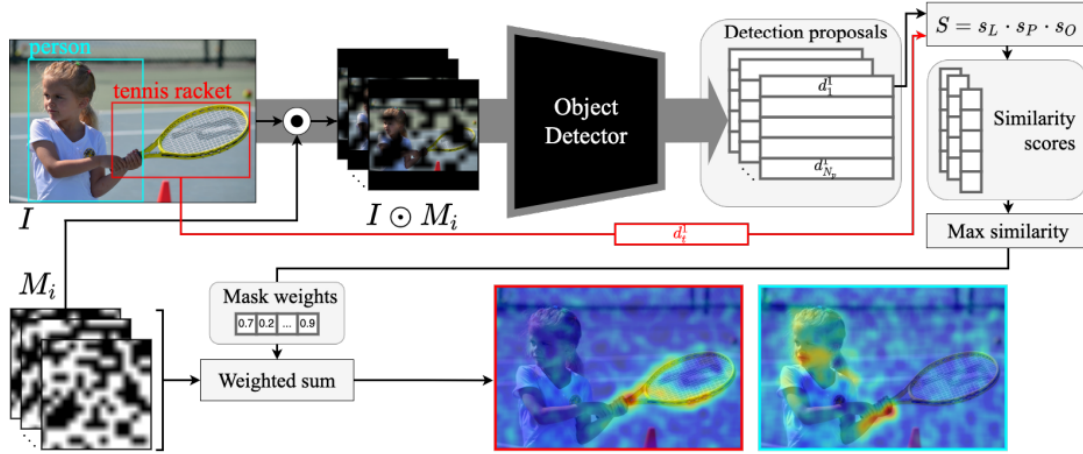


Figure 3.4: Overview of the D-RISE approach. Extracted from [28].

Figure 3.4 summarizes the D-RISE method provided by the authors of the respective paper. Each detection has its own detection vector, combined with the detection vectors computed throughout the algorithm to produce its own heatmap. Moreover, the method at hand requires a set of parameters (e.g., the number of masks generated, the mask's shape, probability of setting a pixel to 1 in the mask).

3.1.4 Local Approximation Methods

Local approximation methods approximate the behavior of a machine learning model around a specific input data point by fitting a simpler model to the model's output in the vicinity of that point (figure 3.5). One of the main advantages of local approximation methods is that they can provide straightforward explanations, even for complex machine learning models. However, they have some limitations as well. For example, they may not accurately capture the model's behavior outside of the local region they are approximating and may be sensitive to noise or outliers in the data. In Local Interpretable Model-agnostic Explanations (LIME) [35], the simple model that uses a weighting scheme based on the distance of each data point from the point of interest. SHAP (SHapley Additive exPlanations) [25] uses a game-theoretic approach based on the Shapley values from cooperative game theory. Another difference between the two methods is the type of simpler model they use to approximate the behavior of the original model: LIME can use any interpretable model, while SHAP is specifically designed to use linear models or decision trees. This means that LIME may be more flexible in terms of the types of explanations it can provide, while SHAP may be more efficient and easier to compute in some cases.

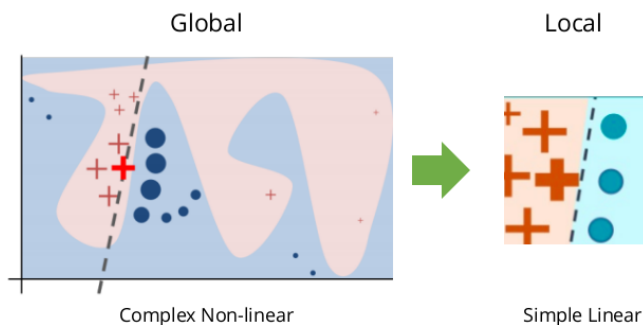


Figure 3.5: LIME and SHAP methods create a new (interpretable) linear classifier that explains the predictions made by the black-box model around the vicinity of data points.

3.2 Explainable models

Intrinsically explainable models are designed to be inherently interpretable and explainable, meaning that humans can understand the internal workings of the model and how it arrived at a particular decision. One example of an intrinsically explainable model is a decision tree, which uses a series of split points to divide data into different branches based on certain features. The resulting tree structure is easy for humans to understand and follow, as each split point represents a decision made by the model based on the input data. Post-hoc methods, on the other hand, are techniques used to explain the decisions made by an AI model after it has been trained and deployed. Overall, intrinsically explainable models offer a more transparent and straightforward approach to AI, making it easier for humans to understand and trust them. Post-hoc methods, while helpful in explaining the decisions of more complex models, may only sometimes provide a complete or accurate explanation of how the model arrived at its decisions.

Li et al. (2017) [22] proposed a deep learning architecture that incorporates case-based reasoning into a neural network. Chen et al. (2019) [5] and the authors of the aforementioned paper then made significant improvements to this model, called ProtoPNet, which allows the network to make predictions by comparing image patches with learned prototypes.

In subsection 3.2.1 there is an in-depth analysis of this architecture, whereas in subsection 3.2.2 there is a brief of newer architectures based on the ProtoPNet.

3.2.1 ProtoPNet

Chen et al. (2019) developed the *prototypical part network* (ProtoPNet) explainable architecture for image classification. Protopnet is based on the idea of prototypical learning, which involves learning from a set of prototypical examples and using these prototypes to classify new examples. Each class contains m prototypes referring to it, representing that class in the latent space. At specific epochs (iterations), the prototypes are pushed based on the points that the model has seen.

The architecture (figure 3.6) is divided into three parts: convolutional layers that extract features from the input image, a prototype layer that computes the similarity scores and a fully connected layer that outputs the logits for each class.

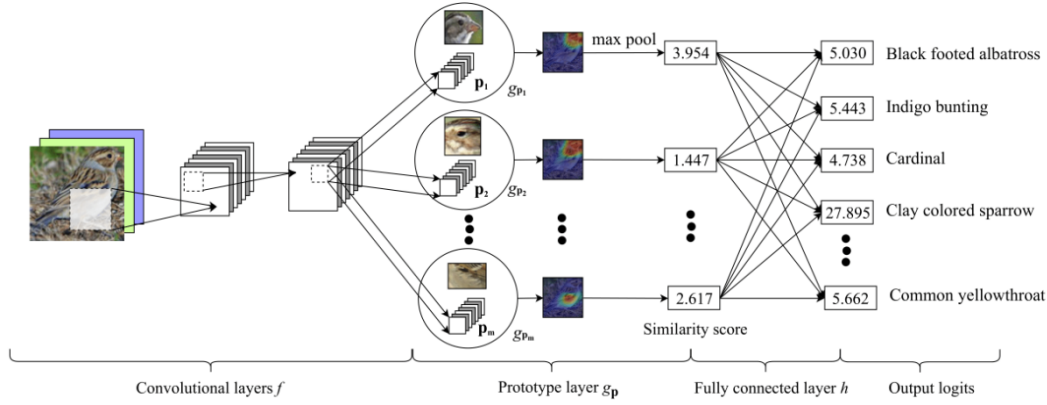


Figure 3.6: ProtoPNet architecture. In the original paper, the authors used ProtoPNet for bird species classification. Adapted from Singh & Yow, 2021 [39].

The prototype layer is a crucial component of ProtoPNet. For each input image's convolutional output, the model computes all its patches, i.e., all prototypical parts of that output. The next step is to compute the L^2 (euclidean) distance between each patch and the prototypes of every class. The similarity scores are equivalent to the inverse of the distances.

Besides penalizing misclassified samples with a Cross-Entropy optimizer, the loss function also encourages each input image to have at least one patch similar to any prototype of its class (cluster cost). Moreover, this loss also includes the separation cost, which encourages all patches of an image to be distant from prototypes of other classes.

Concerning the autonomous driving topic, the objects need to be detected before being classified. Therefore, recent work adapted the ProtoPNet with a Faster-RCNN backbone that extracts the regions with its Region Proposal Network RPN. The coordinates of the bounding boxes are obtained through a regression done by fully-connected layers.

Regarding results obtained with this architecture, ProtoPNet can achieve accuracy over 80% and sometimes improves the performance obtained by black-box models used as baseline. This means that introducing interpretability into the model may benefit its performance. On the other hand, for other models the complexity of ProtoPNet represents a tradeoff between explainability and accuracy. In the worst case, compared to ResNet152, the accuracy decreases by 3.5%.

3.2.2 Other approaches

After the creation of ProtoPNet, deep learning researchers followed this architecture style to perform image classification:

- Barnett et al. (2021) [2] created Incremental Average Interpolation for Adaptive Batch Learning (IAIA-BL), which changed the logic from max-pooling to top-k average. It is based on the idea of using an averaging operation to interpolate between the features of different examples in the training set, in order to generate new, synthesized examples that can be used to train the model. IAIA-BL is designed to be able to learn from very few examples, and it has been shown to be effective on a variety of few-shot learning tasks.
- *Deformable ProtoPNet* [8] introduces additional flexibility and adaptability into the model with deformable convolutional layers. These layers adapt the shape and size of their kernels to better fit the local features of the input data. This can allow Deformable ProtoPNet to be more robust and accurate in situations where the objects in the image do not perfectly match the prototype model.
- *ProtoPFormer* [48] is another variant of ProtoPNet, but uses a transformer-based architecture. This type of neural network is widely successful in natural language processing tasks;
- Negative-positive ProtoPNet (*NP-ProtoPNet*) [39] is very similar to the original ProtoPNet. However, it includes negative reasoning, i.e., considering the negative connection between the patches of an image and prototypes of other classes. In contrast, ProtoPNet only focuses on positive connections with the correct class.

On the other hand, other types of mechanisms have also been used to achieve explanations on a global level for image classification models. Zablocki et al. (2022) [49] compiled a list of techniques for developing explainable methods. According to the authors, explanations can refer to the input level (understanding which perceptual information the model is using) or at an intermediate level (forcing the model to provide additional semantic information).

Attention-based models are input level mechanisms that allow the model to focus on specific parts of the input data when making a prediction. Supposedly, these models give hints regarding the internal reasoning of the network (figure 3.7), being first explored in the autonomous driving context by Kim & Canny. (2017) [19]. More recently, Wang et al. (2019) [45] used visual attention to weight RoIs given by a pre-trained object detector.

3.3 xAI for regression models

Classification models are used to predict a discrete class label, while regression models, on the other hand, are used to output a continuous numerical value for a given input. The coordinates for bounding boxes are obtained through a regressor. As such, in the context of this work, it is essential to review state-of-the-art techniques for explaining regression models.

The current body of knowledge regarding xAI mainly focuses on explaining decisions in classification models. However, the specific techniques and approaches used to describe classification and regression models may differ due to the different characteristics and properties of these models. Letzgus et al.(2022) [21] outlined the challenges in explaining regression models and how

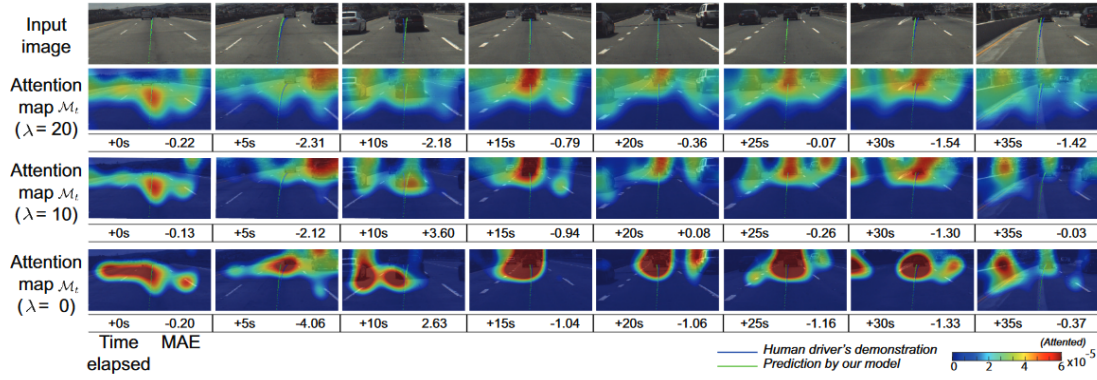


Figure 3.7: Attention map obtained from Kim & Canny. (2017) [19]. The attention map introduces interpretability to the model

engineers can use or adapt popular methods (e.g., LRP, Integrated Gradients, SHAP) explored in subsection 3.1. The aforementioned techniques are model-agnostic and were initially created for classification problems. Typically, explainability questions can be analyzed in technical terms by examining the logit (or probability) score $f(x)$ produced by the model at the output layer, and comparing it to a fixed reference value.

Furthermore, in the article cited the authors verified if gradient-based and perturbation-based explanations work as an off-the-shelf solution for regression problems. The starting point of the explanation is to define a reference point $\tilde{y}$ with a semantic meaning for the user. In classification problems, this reference value explains the decision boundary (figure 3.8).

Considering a regression model that predicts ages of people based on their portraits, one could ask why the prediction is 30 years old instead of 40, for example. As such, the authors create a new function g derived from f (the black-box model) that is defined in the equation 3.1.

$$g(x) = f(x) - \tilde{y} \quad (3.1)$$

This new function g can be seen as a new model of its own, for which explanations will be generated. It is possible to achieve g as a neural network by adjusting the top-level biases in the original network, but it is impossible to explain biases in xAI. Thus, there are two alternatives for interpreting the new model. The first approach is to retrain a surrogate network to mimic the behavior of g . In addition, it is also possible to restructure the existing model, changing its internal representation and thus avoiding the retraining step (which can take a long time).

Furthermore, depending on the chosen reference point, g can contain several root points. The correct approach is not to choose a single reference point as it may induce some bias in the explanation. Instead, they introduce a nonlinearity i.e., $g^+(x) = \max(0, g(x))$ to ensure that the original root point remains a root point.

In addition, the taxonomy proposed by Chou et al. (2021) [6] (figure 3.2) considers Regression-Centric approaches to generate explanations using counterfactual methods. The authors of the latter article identified LIME-C [30] and CLEAR [46] into this category. These techniques, similarly

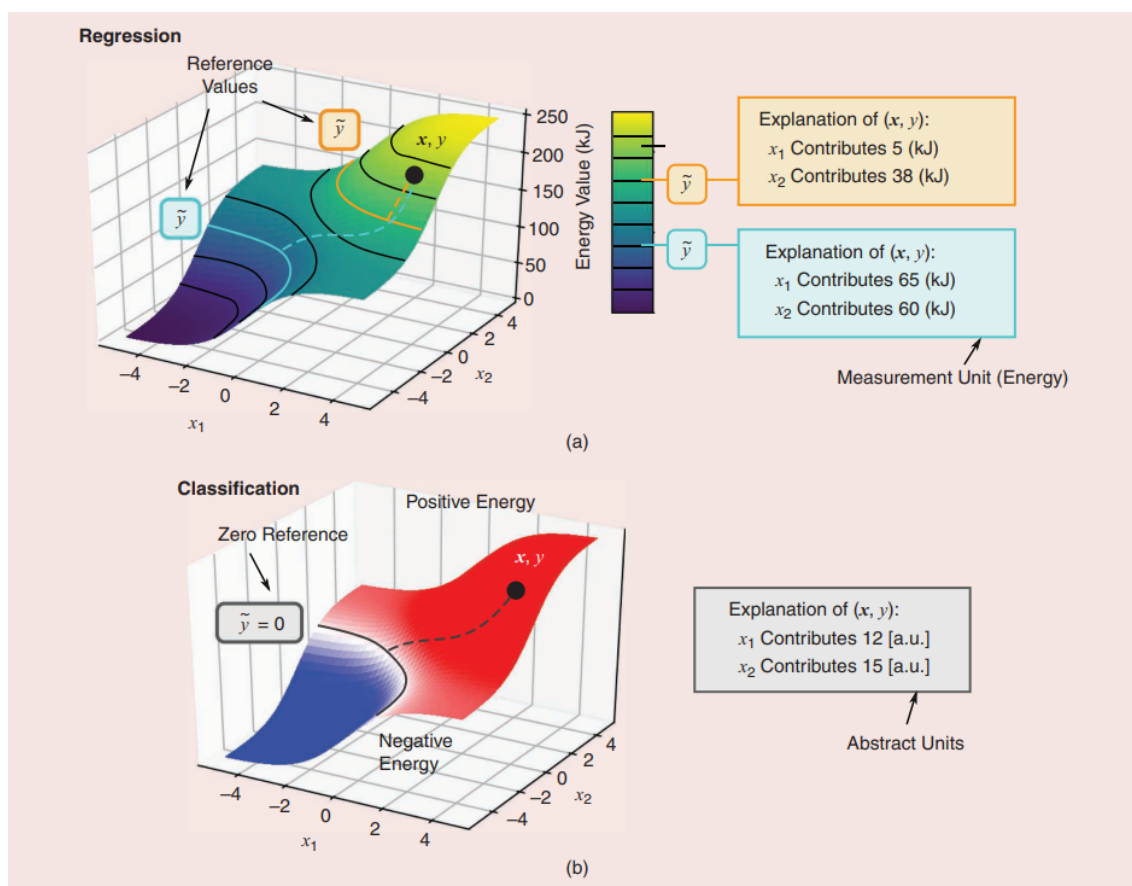


Figure 3.8: Differences between xAI for classification and for regression. The explanation of the model's predictions is generated based on a user-specified reference value $\tilde{y}$, which can significantly impact the explanation provided [20].

to LIME [35], use a linear regression to fit the permuted examples.

Prabhushankar et al. (2020). [29] explore gradient-based model agnostic methods to produce counterfactual explanations for classification and regression models. The authors use the grad-CAM technique with a relative loss in the paper.

In classification tasks, they consider the predicted class and the second most likely class (i.e., the class with the second highest score). The idea of this method is to obtain an explanation in the "Why P instead of Q style?" style, with P and Q being the predicted and contrastive classes, respectively. They use the relative loss between P and Q instead of the regular loss to obtain these contrastive explanations. This technique can also be adapted to explain a regression model. P and Q are both values in these models, and their relative loss is used like in the classification case.

Figure 3.9 is an example of the latter technique for a regression model. The saliency maps obtained explain the difference between the predicted quality returned by the model and several arbitrary qualities provided by the authors. When the difference between the predicted and contrastive quality is higher, the total area of the regions highlighted by this explanation technique naturally increases.

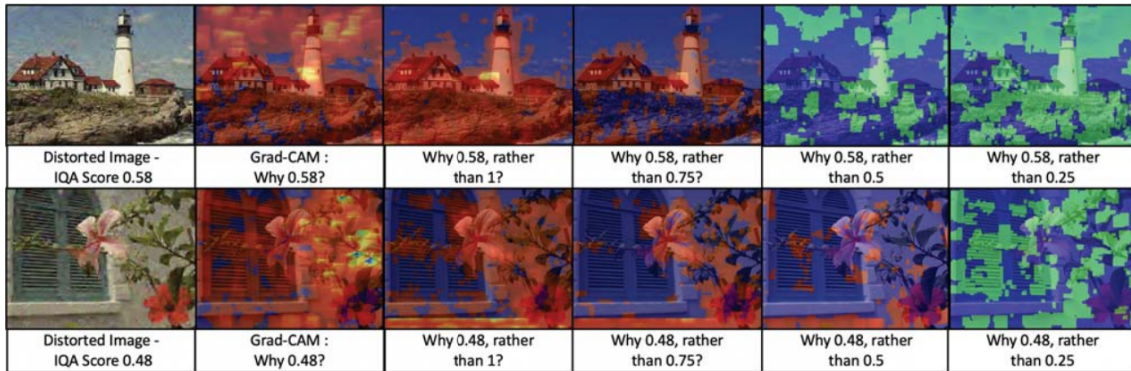


Figure 3.9: Saliency map from grad-CAM for an image quality assessment task. Extracted from [29].

3.4 Explainable AI Metrics

General machine learning models can be quickly evaluated using the metrics described in section 2.4. These measures are excellent tools for comparing different models, with complete compatibility and easy, uniform implementation for all algorithms. Consequently, the quality of deep learning models is straightforward, considering a set of metrics and a dataset.

When it comes to xAI techniques, assessing their quality and effectiveness is more challenging, as they evaluate the importance of features without any ground truth. Explanations are often subjective and open to interpretation by the users, which introduces the issue of bias in this assessment. On intrinsically explainable models such as the ProtoPNet [5], the explanation can be readily tied to a similarity score or other numerical measure used inside the model's architecture. On the other hand, model-agnostic techniques do not offer these tools for the evaluation phase, requiring customized metrics developed by their authors. The best example of this problem are techniques that produce visual explanations via heatmaps, where the explanation is embedded in the saliency map and it becomes hard to transform it into a value. Moreover, different xAI techniques do not share the same evaluation metrics (e.g., counterfactual explanations are evaluated differently from standard visual explanations).

The pointing game [51] is a metric commonly used to evaluate saliency methods, as it is simple to implement and applies to the vast majority of saliency methods. This method generates a sequence of visual explanations and considers the pixel with the highest activation for each heatmap. Therefore, when the pixel chosen for the respective heatmap is located inside the ground truth object (i.e., inside the bounding box), the algorithm considers it a hit. Instead, the heatmap counts as a miss if the pixel is located outside the bounding box. The metric is calculated by dividing the number of hits by the total heatmaps. This evaluation is valuable in determining the ability of the heatmap to generate localized explanations (i.e., highlights features close to the object being analyzed).

The Grad-CAM authors evaluate the localization aspect of their method in a weakly-supervised manner. The method consists of generating a heatmap for an image and binarizing this array by

applying a threshold. In other words, pixels whose activation is higher than the threshold value are set to 1 in the binary array, whereas all the other regions contain 0. The resulting array contains connected segments of pixels, which are then used to draw a bounding box over the biggest super-pixel. The localization error can finally be calculated between the ground truth object and the bounding box obtained by the algorithm.

Chapter 4

Datasets and Models

Before attempting to obtain explanations for object detection models, these need to be appropriately trained and evaluated. This chapter elaborates on the training procedures used in this work. As such, important aspects like the datasets used, architectures and evaluation metrics are explored.

In this chapter, some aspects regarding the training process are discussed. Section 4.1 analyses the details of the datasets, making some prospects on the results obtained for each one. Section 4.2 explores the models, explaining the reasons behind their implementation, as well as other important characteristics (e.g., results, hyperparameters).

4.1 Datasets

The datasets play a crucial role in all machine learning applications. Since this study is about object detection using RGB data, it is important to train the models with popular benchmark datasets, as it allows for easy comparison to other detectors. Furthermore, the choice of a dataset with correct annotations mitigates the risk of obtaining poor interpretations because the model is learning incorrect patterns in the training data.

The Pascal VOC 2012 (subsection 4.1.1) dataset was selected as it meets all the criteria. Moreover, because this work is focused on autonomous driving object detectors, the Kitti dataset (subsection 4.1.2) was also used to train a separate detector, for which interpretation results should be further discussed.

4.1.1 Pascal VOC Dataset

The Pascal VOC (Visual Object Classes) dataset [9] is a widely used computer vision benchmark comprising images and annotations for object recognition tasks. It is commonly used in object detection and semantic segmentation tasks. The dataset is built using standardized procedures for comparing different methods. Moreover, it was introduced in 2005 and has since been used in numerous computer vision research projects and competitions. In 2008, the dataset was considered

the benchmark for object detection. In the scope of this work, the images included in Pascal VOC represent a slightly easier challenge for the techniques tested, as each image contains fewer objects on average.

Figure 4.1 shows class distribution for the training set used. The class "person" is severely outnumbering the remaining classes, which may lead to some bias in classification. However, this problem is not a major risk in this work, as it is mainly focused on the regression that achieves the bounding box coordinates.

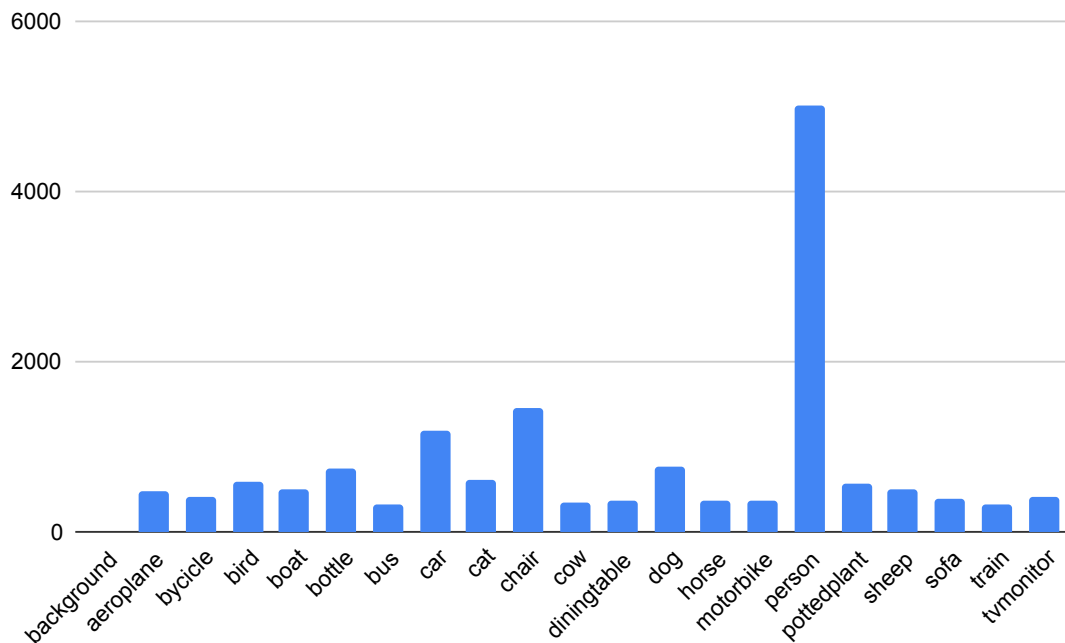


Figure 4.1: Class distribution for the Pascal VOC 2012 train dataset.

4.1.2 The KITTI Dataset

The KITTI dataset [11] is a large-scale computer vision and robotic perception research dataset. It contains data from various sensors, including cameras, LiDAR, and radar.

The dataset includes a wide range of data, including images, point clouds, and annotations. The photos are taken from various cameras, including front-facing, rear-facing, and side-facing. The point clouds are generated from LiDAR data and include information about the location and intensity of each point. The annotations include information about the location and size of objects in the images and point clouds and information about the object's class (e.g., car, pedestrian, cyclist). In total, ten classes are included in this dataset, counting with the background class.

The KITTI dataset is widely used in academic research and is also commonly used in industry for training and evaluating autonomous vehicles and robotics systems. It is considered a benchmark dataset in the field and is used to compare the performance of different algorithms. Its nature makes it an ideal benchmark to evaluate an object detector thoroughly.

Figure 4.2 highlights the class distribution on the Kitti dataset. As expected and similarly to other autonomous driving datasets, the class "Car" is the majority class with over 25.000 detections assigned to it. In addition, it is clear that this dataset contains a significantly superior number of objects detected compared to the Pascal VOC dataset.

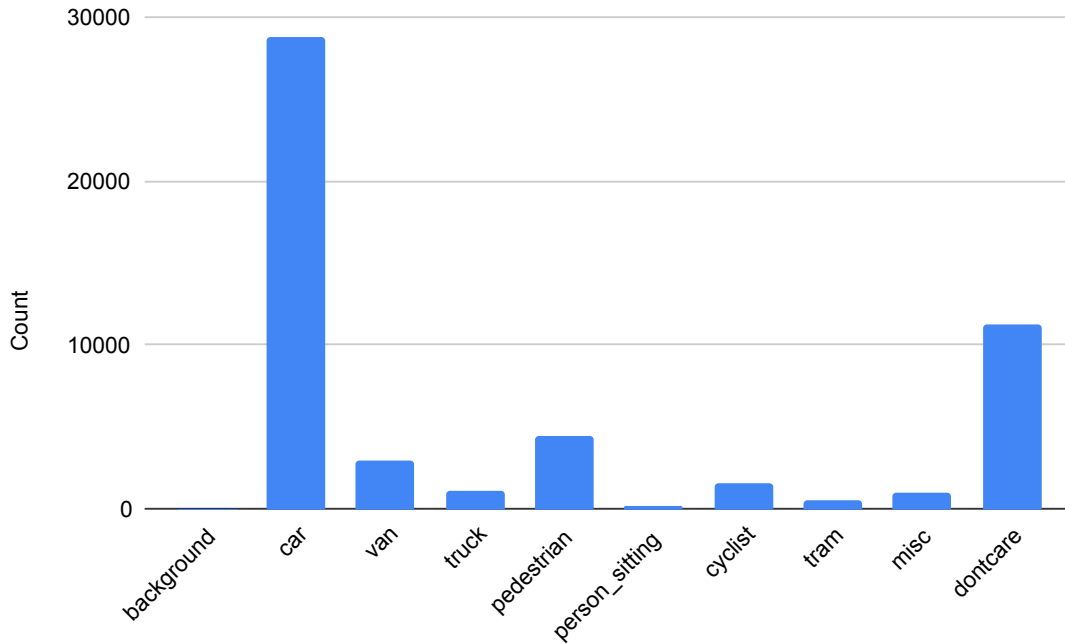


Figure 4.2: Class distribution for the Kitti train + validation dataset.

The Kitti object detection dataset does not offer a validation set. As such, for evaluation purposes, a split was made to separate it into train set (80%) and validation set (20%).

4.1.3 Exploratory Data Analysis

The nature of the datasets tested directly impacts the quality of the models, regarding their performances and interpretability results. Therefore, it is vital to assess the main characteristics of the data, as it may help justify the results obtained in future steps.

Table 4.1 provides general information regarding the images in both datasets. Regarding the amount of data, Kitti has a clear advantage. Moreover, the images in the autonomous driving dataset have more objects (mostly cars) when compared to its counterpart. Consequently, the objects contained in the Kitti dataset are smaller and generally more difficult to detect.

Furthermore, figure 4.3 is a good example of the nature of the Kitti dataset. Images usually contain a high number of objects (mostly cars) with their corresponding bounding boxes overlapping other objects. Its counterpart, the Pascal VOC dataset contains more simple images, with objects that are often emphasized in the frame, as shown in figure 4.4.

Dataset	Number of Images	Detections per image (avg)	Average object area (pixels)
Pascal VOC 2012	5717	2.76	36016.5
Kitti	7481	6.93	9096.6

Table 4.1: Exploratory Data Analysis on datasets explored in subsections 4.1.1 and 4.1.2.

4.2 Models

In the first phase of this dissertation, it is necessary to choose a model architecture to train with the datasets mentioned in subsection 4.1. A two-stage detector was chosen, as it brings a clear separation between classification and regression. Moreover, it is easier to develop a separate work for these models attempting to obtain interpretations for the classification head. Therefore, the Faster-RCNN architecture makes sense in this context, as it is the most popular object detection technique in the two-stage pool.

Regarding the backbone architecture, the model uses deep residual blocks, as they achieve better performance in faster-RCNN detectors (subsection 2.3.1). Figure 4.5 contains the architecture of the ResNet-50 backbone, which is the simplest backbone architecture used in for this thesis. In addition to the latter, experiences are explored using ResNet-101, aiming to assess the influence of the backbone's complexity in the results obtained (e.g., performance, explainability). Both neural networks are imported with weights pre-trained on the MS COCO dataset [23]. Furthermore, the figure explains the concept of residual learning block which helps mitigate the vanishing gradients problem.

Furthermore, the models use group normalization [47] instead of the usual batch normalization. The latter normalization technique introduces problems with smaller batch sizes because of inaccurate statistics estimations. Group normalization divides the channels of each feature map into groups and normalizes each group separately. This modification helps learn more robust features in the backbone, which is ultimately increases both the raw performance of the model and its potential explanations.

The work is focused on the regression part of the object detection task. However, in regular detectors, the weights of the backbone are optimized for regression and classification. In addition, xAI methods benefit from patterns learned in feature extraction layers of the network. As such, a separate model is trained, optimizing the performance for a dataset whose objects contain the same labels (i.e., the model only distinguishes between "object" and "background"). The classification head of this model only discriminates between an object and a background. Conceptually, this model should be able to produce better explanations, as the features are solely connected to the regression of the bounding box coordinates. The training process is similar to other conventional detectors, as it tries to optimize all losses, including classification related ones. However, due to the nature of the dataset, the classification losses are low and stable, with the optimization attempting to decrease the regression losses. In addition, the backbone used for this detector is the ResNet-101 aiming to maximize its performance. Furthermore, this model is only trained for



Figure 4.3: Example image from the Kitti dataset.

the Kitti dataset, as further experiences concerning the Pascal VOC are not deemed necessary and would add unnecessary complexity to this dissertation, since the work at hand is focused on the autonomous driving context.

Figure 4.6 displays information regarding the evolution of the loss function during the training epochs of the model focused on regression. It is expected that this detector outperforms the others, as it is only responsible for detecting objects, regardless of which type of object it is. According to the chart presented at the latter figure, the early stopping algorithm stops the training process at epoch 8, which is considerably earlier than other models (figure 4.7).

Regarding the performance of the detectors, the evaluation criterion requires an IoU threshold above 50%, as this is the rule adopted by both datasets. Figure 4.7 shows the evolution of the loss during training with the Kitti dataset for faster-RCNN detectors using ResNet-50 and ResNet-101 backbones, respectively. The performance boost obtained from doubling the number of convolutional layers exists but is not significant, with both models achieving very similar loss values on the validation set in the last epoch. Moreover, the number of epochs performed until the early stopping halts the training process is 18 for both models, meaning that from epoch 15, the model could not improve its performance on the validation set.

Table 4.2 establishes the numerical comparison between models and datasets using mean Average Precision at an IoU threshold of 50% (mAP_50) and the average smooth L1 loss. The latter

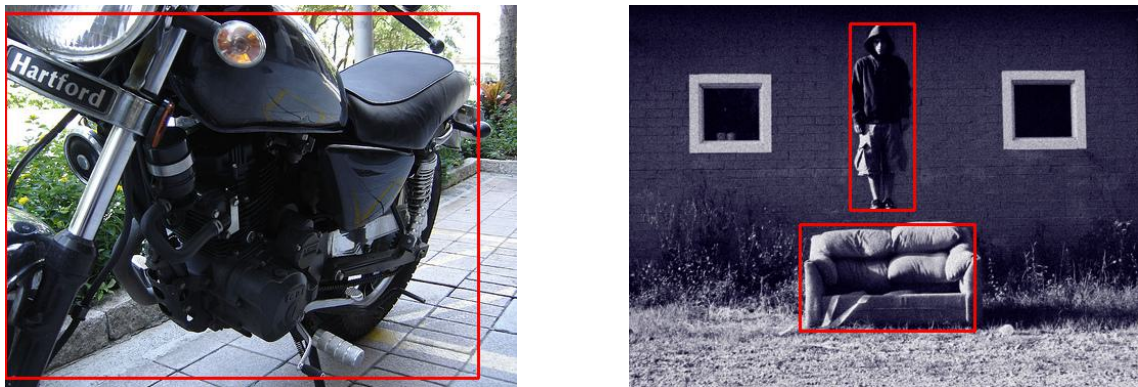


Figure 4.4: Example images extracted from the Pascal VOC dataset.

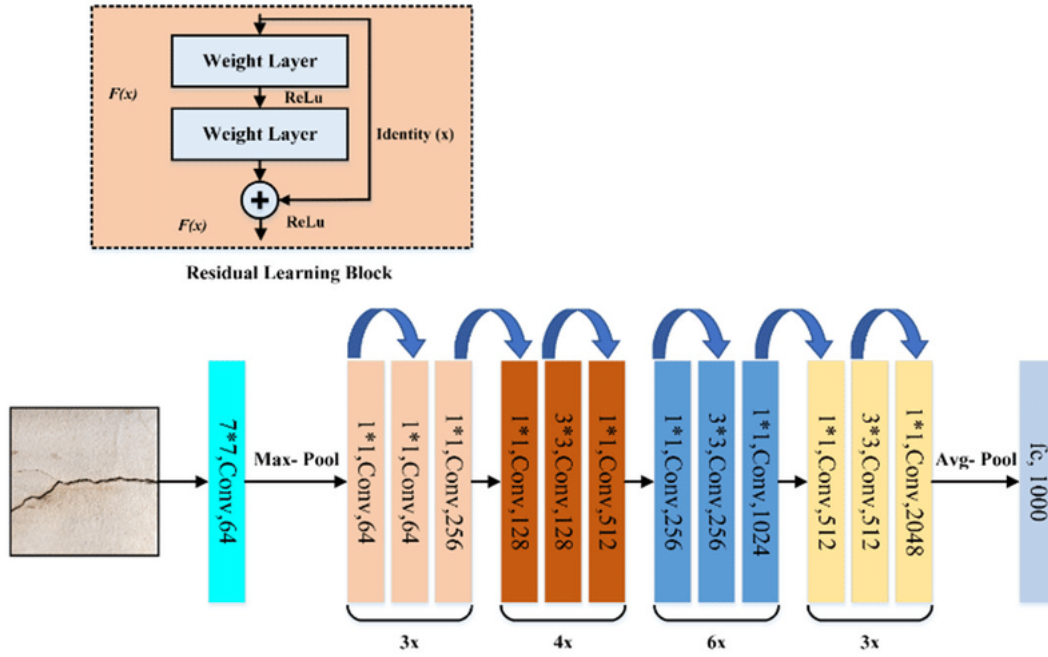


Figure 4.5: Architecture diagram of ResNet-50. Extracted from [1].

regression loss is chosen because it is also integrated in the Faster-RCNN architecture. Moreover, the train and validation split made on the Kitti dataset has determinism-related issues into account to ensure that the evaluation is always performed on the same samples. These measures guarantee an effective comparison between different models avoiding slight discrepancies caused by random factors. In addition, the faster-RCNN regression-focused model is not evaluated with mAP_50. This metric is designed to penalize mistakes made in the regression and classification phases, which is not an appropriate assessment of this model.

The empirical results obtained and used for comparison purposes are measured on the validation sets of each dataset. Despite the fact that the samples contained in these sets are not used for optimization, the results presented are optimistic. However, the comparison established is still valuable for the context of this work, validating the fact that the detectors trained are indeed capable of performing their task successfully in most cases for each dataset, achieving performances close to those of works that use the Faster-RCNN architecture for object detection in the same

Model	mAP_50	smooth L1
faster-RCNN (ResNet-50) - Kitti	0.724	10.53
faster-RCNN (ResNet-101) - Kitti	0.744	10.05
faster-RCNN (ResNet-50) - VOC	0.576	25.72
faster-RCNN (ResNet-101) - VOC	0.600	23.15
faster-RCNN Regression	-	9.60

Table 4.2: Performance of all models measured with validation sets of both Pascal VOC and Kitti datasets.

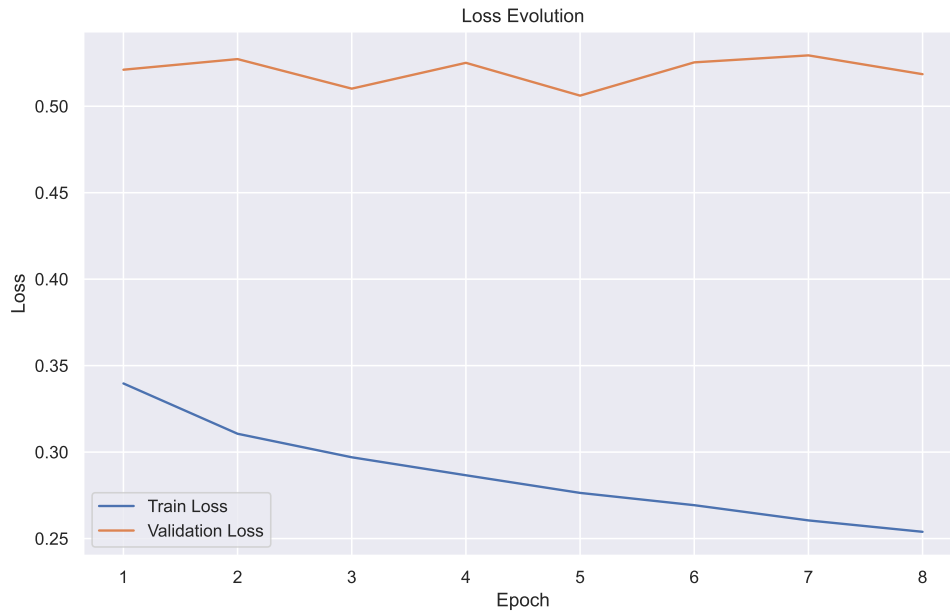


Figure 4.6: Train and validation loss for the regression model.

datasets.

Furthermore, table 4.2 confirms the theoretically expected outcome. Models with a larger backbone architecture achieve better overall performance across both metrics, although the difference between each model and its counterpart does not vary significantly on each dataset. On the other hand, there is a considerable increase in the Faster-RCNN model when trained and tested with the Kitti dataset relative to the Pascal VOC dataset. The difference is explained with the nature discrepancies between both datasets:

- Regarding mean Average Precision, VOC contains twice more classes than Kitti. Moreover, the autonomous driving dataset is heavily biased towards the "car" class while the Pascal VOC dataset is slightly more balanced;
- Regarding smooth L1 loss, the Pascal VOC contains larger objects (table 4.1) making it more vulnerable to higher regression losses during inference.

All models are trained with the Pytorch¹ framework using the same set of hyperparameters. The learning rate was set to 0.005 with the SGD optimizer. Experiments to tune the hyperparameters could boost the models' performances. However, spending this additional time and computational resources is not essential to the project's success and therefore is not performed. Overall, the object detection performance is satisfactory (figures 4.3 and 4.4), as models are able to properly locate objects in images of both datasets.

¹<https://pytorch.org/>

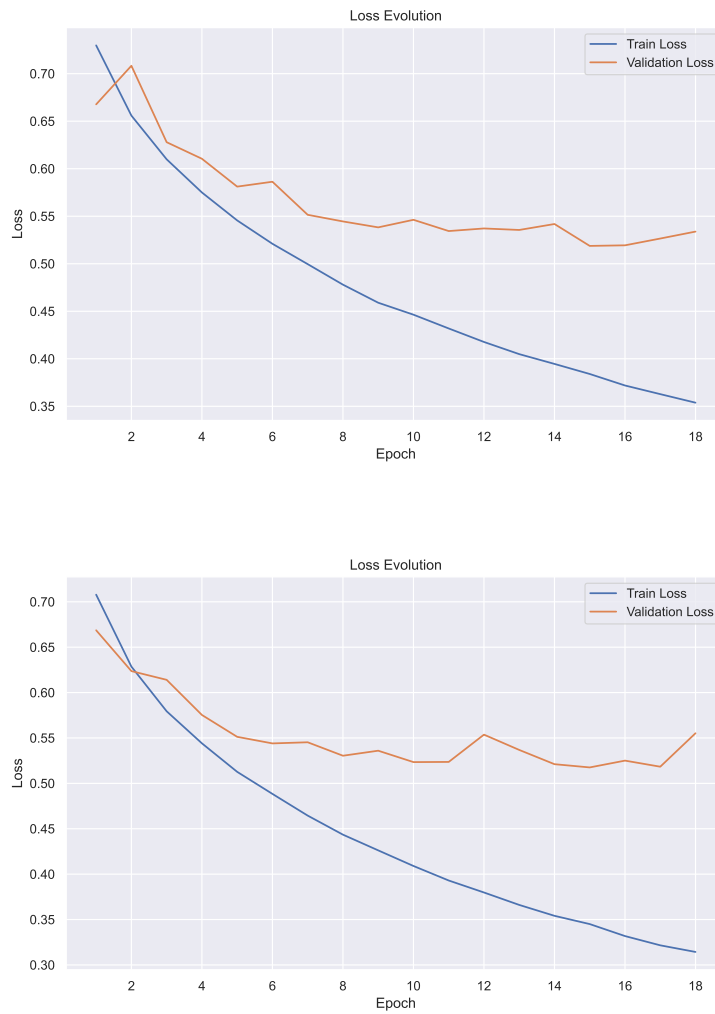


Figure 4.7: Loss decrease for Faster-RCNN using ResNet-50 and ResNet-101 trained with the Kitti dataset.

Chapter 5

Gradient-based Explanations

This chapter describes some experiences using gradient-based methods to obtain interpretations. The latter techniques are primarily used in classification tasks. As such, some adaptations are due to ensure the quality of the explanations achieved. The methods explored in this section combine both gradient-based explanations with counterfactual analysis.

Moreover, the saliency maps produced need proper discussion in terms of interpretability. Comparing results obtained for different models and datasets is also adequate, as the explanations' quality heavily depends on them. Section 5.1 introduces the Grad-CAM method adapted to the autonomous driving context. Section 5.2 explores the guided backpropagation technique to obtain explanations with the adaptations made in the Grad-CAM section. In addition, section 5.3 leverages the SmoothGrad iterative algorithm to address known issues that exist in gradient-based methods. Lastly, section 5.4 expands on the results discussion already performed throughout the chapter, making some additional considerations.

5.1 Grad-CAM

The grad-CAM method is explained in subsection 3.1.2 and produces saliency maps that highlight the most important features of an image. The technique computes the gradients of a score with respect to the parameters of the network. In classification explanations, this score is simply the logit the model produces for a specific class. The main advantage of this algorithm is its ability to be quickly applied to any model.

In this section, some techniques are explored to apply the Grad-CAM to the context of bounding box regressions. Subsection 5.1.1 establishes explanations that solely use the predicted bounding box. Subsection 5.1.2 explores the topic of contrastive explanations using gradient-based approaches.

5.1.1 Scalar Explanations

In this work, explaining bounding box coordinates makes applying the grad-CAM method directly impossible. Instead, some adaptations are necessary. The first possibility would be to obtain four saliency maps, one for each object coordinate. This approach aims to produce a heatmap whose activations are significantly higher in areas closer to the respective coordinate being analyzed. However, this approach is unfeasible with grad-CAM since it is not precise enough to produce such localized explanations (figure A.1).

The other alternative is to transform all coordinates of a specific bounding box into a single scalar value. The first technique chosen to perform the latter transformation is calculating the distance between two opposite points of the bounding box. The resulting value captures the physical characteristics of the respective rectangle and depends on the parameters of the network. Therefore, the Euclidean distance is equivalent to the length of the bounding box diagonal. In addition, the slope between the two opposite points (e.g., (x_{min}, y_{min}) and (x_{max}, y_{max})) also conveys information about the detection and can be used for the same purpose.

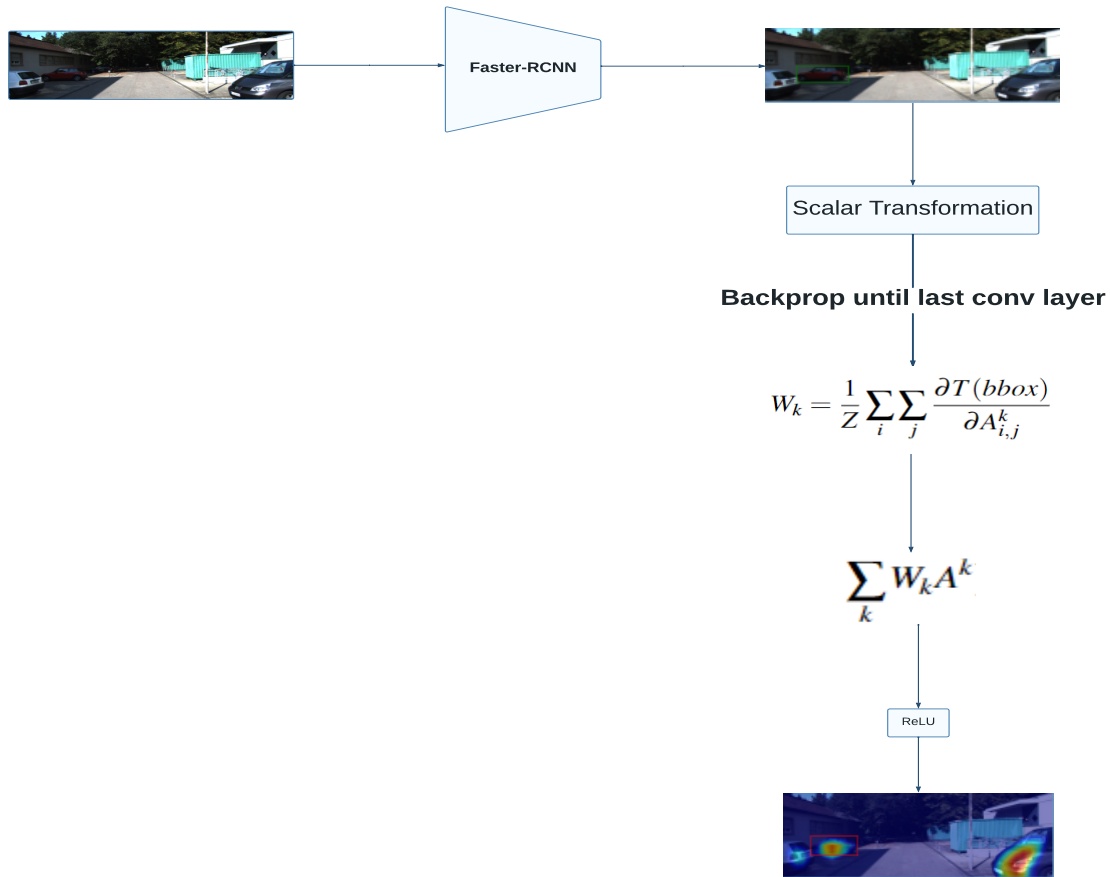


Figure 5.1: Diagram of the experimental setup used in section 5.1.1

The overview of the approach is illustrated in figure 5.1. The starting point is an example image from the Kitti dataset that is given as input to an object detector. The next step is to select a bound-

ing box for which explanations are to be generated. This bounding box selection is performed in an arbitrary manner. However, the image and the bounding box to be explained should be obtained in a deterministic way to assure the reproducibility of the results. The final step involves applying the transformations to the bounding box chosen, i.e., calculate the euclidean distance or the geometric slope with the coordinates of the bounding box. At this point, the Grad-CAM formula can be applied to generate two different saliency maps, depending on the transformation chosen in the previous step. Equation 5.1 formalizes the adaptation made with respect to the original Grad-CAM formula, replacing the classification score for a specific class with the scalar resulting from the transformation function ($T(bbox)$). In this formula, $A_{i,j}$ represents the position (i, j) of the feature map and Z is the normalization factor. Equation 5.2 contains the formula that is also applied in the original Grad-CAM paper [36]. The weights W_k multiply the feature maps A_k .

$$W_k = \frac{1}{Z} \sum_i \sum_j \frac{\partial T(bbox)}{\partial A_{i,j}^k} \quad (5.1)$$

$$M_{\text{grad-CAM}} = \text{ReLU}(\sum_k W_k A^k) \quad (5.2)$$

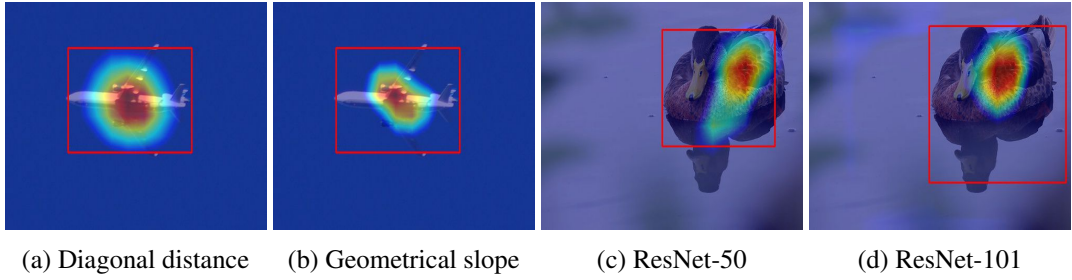


Figure 5.2: Grad-CAM explanations after applying transformations to the original bounding box. (a) and (b) are generated using the ResNet-101 backbone. (c) and (d) use Grad-CAM with the distance measure to compare both architectures.

Figures 5.2a and 5.2b show two examples of explanations using scalar transformations. Both examples use the same image, consisting of a reasonably simple case (i.e., with a single object occupying a central place). The choice is a good representative of the behavior assumed by both techniques. In theory, one would expect higher activations next to the diagonal points of the bounding box, as they are responsible for the oscillations in the slope and distance values. However, that is not the case, as in practice, Grad-CAM highlights central parts of the object. In the distance heatmap, the region is highlighted seemingly uniformly, whereas the slope heatmap attributes higher importance to areas aligned along a diagonal. The differences are marginal, as the abstract features learned at the end of the backbone do not allow it to point to the most important features for each measure (either distance or slope).

Furthermore, another comparison can be established between the behavior of this technique when applied to different model architectures. Figures 5.2c and 5.2d establish the comparison between ResNet-50 and ResNet-101. The latter task is achieved with heatmaps generated using

the distance measure with respect to the parameters of both backbone architectures. Once again, the differences are minimal though the larger backbone highlights a more compact area. Its counterpart activates pixels closer to the diagonal points, which would make sense a priori.

Given the context of this work, it is vital to test this approach against autonomous driving images to prove the explainability it provides in this situation. Figure 5.3 displays saliency maps generated using different backbone architectures and distinct transformation techniques (e.g., distance, slope). It becomes clear that Kitti images represent a more demanding challenge for the described algorithm. The higher number of objects in the image, the lower dimension and the fact that some objects are obscured by others impacts (section 4.1) the interpretations obtained.

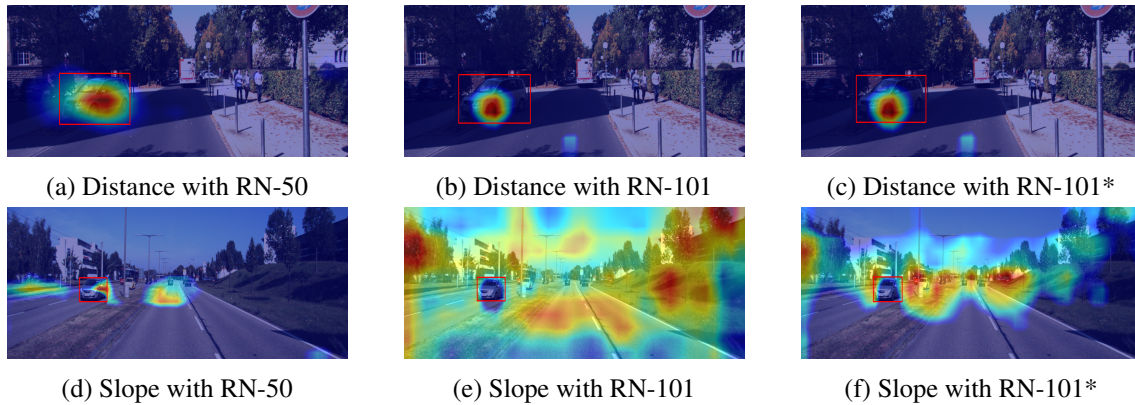


Figure 5.3: Grad-CAM using scalar transformations. (a), (b) and (c) are generated with the distance value. (d), (e) and (f) are generated with the slope value. (c) and (f) are generated using the model that does not perform class discrimination.

The cases displayed in figures 5.3a, 5.3b and 5.3c contain explanations where the most activated pixels are concentrated within the object region. In addition, ResNet-50 parameters consider many pixels important to the euclidean distance between diagonal points, as its heatmap contains activations spread out along the entire bounding box. On the other hand, the example Grad-CAM result revealed in figures 5.3d, 5.3e and 5.3f uncovers a significant problem intrinsic to gradient-based approaches, which is its vulnerability regarding noise. In this case, the other cars in the image seem to impact the interpretation concerning the localization of the object considered. Surprisingly, the simpler ResNet-50 outputs a more explicit explanation than its more complex counterparts. In other words, both models that use ResNet-101 as their backbone architecture depict imprecise heatmaps when the Grad-CAM algorithm is applied, with pixels irrelevant to the object receiving high activations. Furthermore, the behavior of Grad-CAM is similar for both ResNet-101 models, producing identical heatmaps for all examples tested.

In order to address possible bias in the evaluation of the visual explanations obtained, an objective numerical metric needs to be used. As stated in section 3.4, choosing a numerical measure that compares the effectiveness of different xAI techniques is a challenging task, with many metrics to choose from. In this type of interpretation, the Pointing Game evaluation method is selected because of its effortless implementation and high applicability (i.e., it can be readily used to test

PG	distance transformation	slope transformation	Excitation Backprop
ResNet-50	0.703	0.384	-
ResNet-101	0.733	0.574	-
VGG16	-	-	0.76

Table 5.1: Pointing Game scores on Pascal VOC. Higher values mean better performance.

other saliency methods’ results). Moreover, the main aspect that led to the latter choice is that the Pointing Game assesses the localization capability of the technique, which is the most important type of assessment in the context of this work. On the other hand, one of the drawbacks of Pointing Game-based evaluation is its high volatility regarding the setup on which it is applied (i.e., dataset and model tested).

Table 5.1 contains the results of this metric applied to Grad-CAM explanations on the Pascal VOC dataset. Furthermore, it is possible to compare results obtained in this work with those obtained using the original Grad-CAM approach on an identical dataset (Pascal VOC 2007), computed by Zhang et al. 2016 [51] using the Excitation Backprop method to highlight class-discriminative regions. According to the comparison established, both strategies previously mentioned obtain similar numerical results, with a slight advantage of approximately three percentual points to the class-discriminative CAM with respect to the Grad-CAM approach proposed in this dissertation.

Table 5.2 displays the evaluation of the Grad-CAM approach explored throughout this section with respect to the Pointing Game metric tested with images from the Kitti dataset. The results obtained in the aforementioned table expose an obvious advantage of using the distance transformation method compared to the slope. The larger number of convolutional layers brings a surprising decrease in the quality of explanations generated. In addition, the model that behaves as a pure regressor holds a slight advantage over the others, as theoretically expected.

5.1.2 Contrastive Explanations

The application of Grad-CAM using simple numerical transformations to the bounding box structures often produces vague explanations, without clear indication of which features are considered to output a rectangle with a given slope or size (subsection 5.1.1). As such, emerges the idea

PG	distance transformation	slope transformation
ResNet-50	0.388	0.217
ResNet-101	0.38	0.097
ResNet-101*	0.439	0.154

Table 5.2: Pointing Game scores for Grad-CAM on the Kitti dataset. Higher values mean better performance.

of employing visual counterfactual explanations aiming to generate heatmaps more aligned with human explanations. In this subsection, explanations target a specific difference between pairs of detections, instead of focusing on the bounding box and its intricacies.

Prabhushankar et al. (2020) [29] introduce the idea of obtaining contrastive explanations with grad-CAM. In the paper, the authors calculate the gradients of a relative loss between two classes with respect to the parameters of the model. This modification highlights the role of the gradients as a loss minimizer. The classes chosen are usually the predicted one and the second most likely class according to the model (i.e., the label with the second highest probability score). In theory, the resulting heat map should allow a clear visualization of which features led the model to opt for one class instead of another.

This method can also be used for regression tasks, namely in object detection. It is necessary to choose a loss function and calculate it for two bounding boxes. However, the detections chosen include a predicted bounding box and a contrastive example that needs to be produced arbitrarily. In this dissertation, contrastive bounding boxes are made via simple translations or scaling on the predicted detection. In addition, the algorithm responsible for producing counterfactual examples aims to produce bounding boxes with a clear but not excessive difference with respect to the predicted detection. The loss function chosen for the gradient calculation is the smooth L1 loss (equation 2.6).

The outline of the Grad-CAM approach using counterfactual analysis is presented in figure 5.4. The initial steps are similar to those described in subsection 5.1.1. The techniques diverge after choosing the bounding box for which explanations are to be generated. Counterfactual analysis approaches such as the one being employed need a contrastive example. In the scheme mentioned above and throughout the project, red rectangles represent the predicted bounding box, and green rectangles are their respective contrastive examples.

Regarding the model's parameters, grad-CAM typically uses the weights and biases of the last convolutional layer, as it corresponds to the final feature extraction step. In this work, gradients are solely calculated with respect to the weights of the respective layer. Biases are excluded for implementation reasons, as their analysis would require a separate saliency map for each detection. The gradients are then average pooled and multiplied by the feature map produced by propagating the image through the network. This multiplication is performed along the channel axis of the feature map. The last stage of this method performs upsampling in the class activation map (cam) to the dimensions of the input image after a previous ReLU activation.

$$W_k = \frac{1}{Z} \sum_i \sum_j \frac{\partial L(b_1, b_2)}{\partial A_{i,j}^k} \quad (5.3)$$

Equation 5.3 formalizes the weights formula. The calculation is similar to that of the original grad-CAM method, replacing the output score with the relative loss (smooth L1 loss) between b_1 and b_2 which substitute the predicted and contrastive bounding boxes. The gradients are calculated and global average pooled. The weights are applied as in subsection 5.1.1 and are thus formalized in equation 5.2.

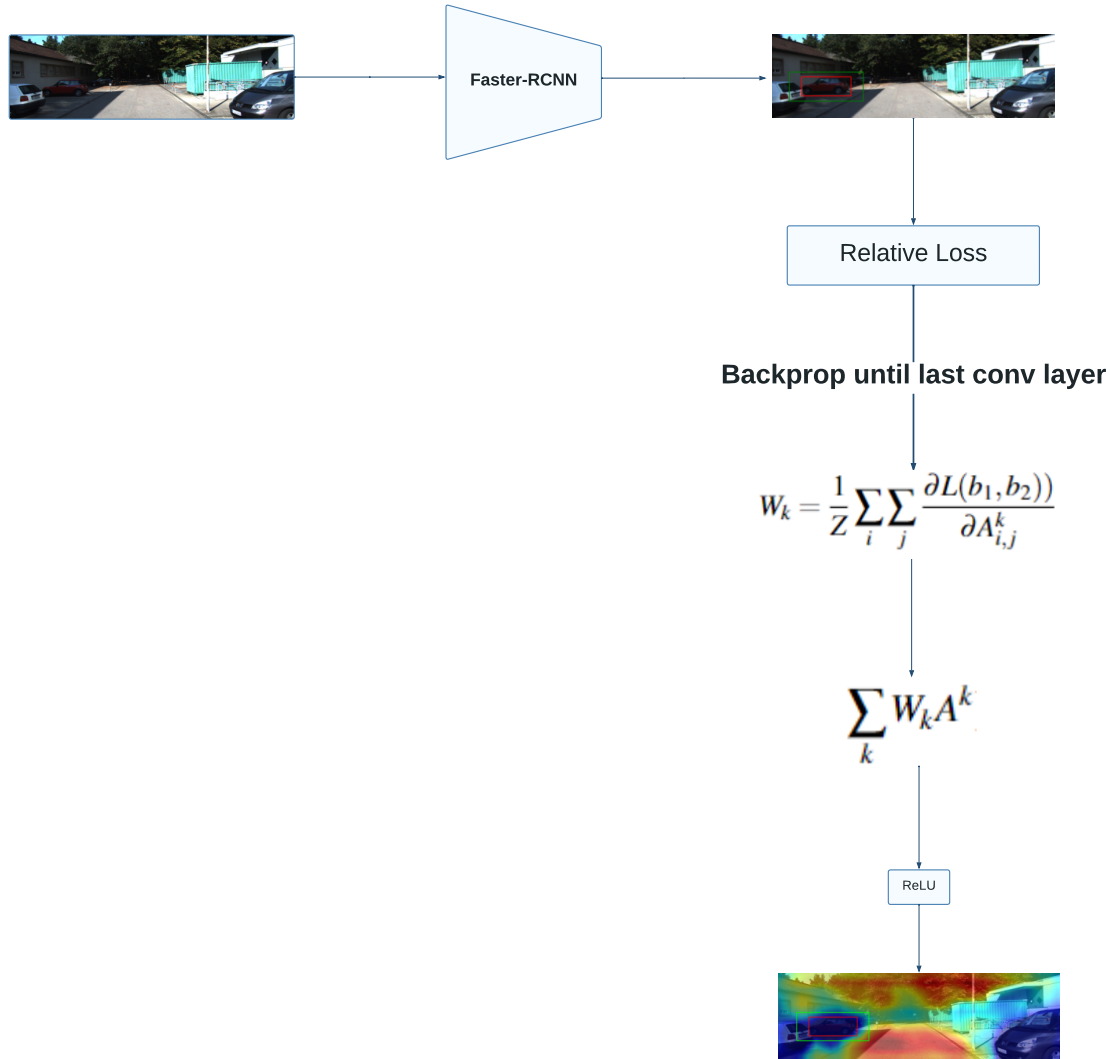
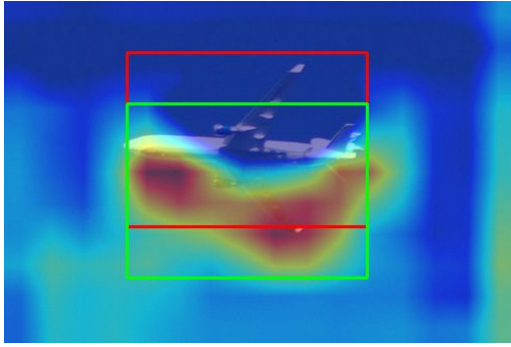


Figure 5.4: Overview of the Grad-CAM contrastive approach.

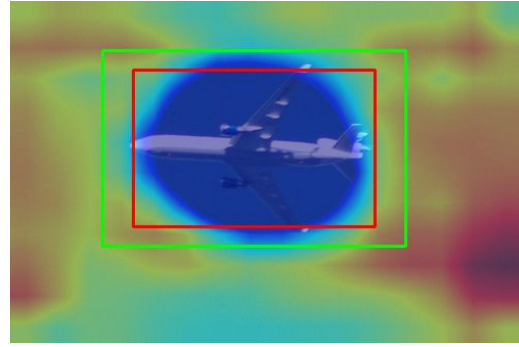
Figure 5.5 provides an example output from the grad-CAM method. The regions highlighted in red represent features positively related to the increase of the relative loss between both detections (i.e., areas that help the model decide for the predicted bounding box instead of the contrastive example). On the other hand, blue zones represent a negative (or null) correlation between the corresponding features and the relative loss.

In the image aforementioned, the heatmaps provide valuable explanations, as the red zones correspond to features that do not contain the considered object and are exclusive to the contrastive example. Moreover, producing counterfactual samples with different algorithms results in interpretations with similar quality. However, despite the promising results on the VOC dataset, grad-CAM yields inconsistent results on more complex image inputs. The complexity leap mentioned considers aspects such as the amount and dimension of the objects detected.

Figure 5.6a is an excellent example of interpretations expected with grad-CAM in the Kitti



(a) Grad-CAM with translation-guided contrastive example.



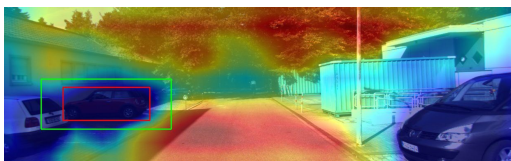
(b) Grad-CAM with scale-guided contrastive example.

Figure 5.5: Contrastive explanations for a sample image in the VOC dataset. The red bounding boxes are predicted by the model, whereas the green rectangles represent contrastive examples.

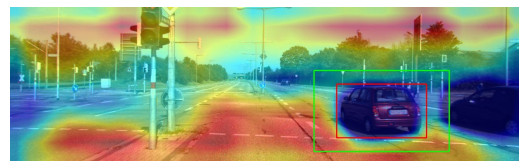
dataset. The aspects discussed in subsection 4.1.3 demonstrate their impact, as the heatmap highlights other objects as relevant for the increase of the relative loss. According to this interpretation, the model’s decision on the bounding box dimensions is highly influenced on another car placed on an apparently irrelevant position. It is unclear the causes of this behavior, although this is a common limitation of the Grad-CAM algorithm. In recent approaches [4], authors highlighted certain limitations to the latter algorithm, proposing modifications that mitigate them. The observations regarding the weaknesses of grad-CAM are target its inability to explain multiple instances of objects with the same class within an image. In addition, grad-CAM often highlights portions of the expected region of the object that it is trying to explain, instead of the complete area. According to multiple deep learning experts, the frequent upsampling and downsampling operations account for the loss of information throughout the algorithm.

On the other hand, figure 5.6b displays a slightly better explanation, which is also helped by the lower complexity of its case (i.e., it only contains two objects that are in a favorable position in the frame). However, despite these easier circumstances, the output still considers some features of the road as important in the choice between the predicted and the contrastive bounding boxes. Furthermore, explanations generated for smaller objects (far away from the camera) are useless in terms of interpretability.

The models have no significant influence on the outcome of Grad-CAM. Throughout all experiences conducted, it becomes clear that the quality of interpretations is not connected to the



(a)



(b)

Figure 5.6: Contrastive explanations in the Kitti dataset.

Pointing Game	
ResNet-50	0.095
ResNet-101	0.073
ResNet-101*	0.064

Table 5.3: Pointing Game scores for contrastive explanations on kitti. Higher values mean better performance.

training methodology. All results presented in this section are generated using the default model (i.e., performing both the regression of coordinates and classification of objects detected).

One of the possible causes of major issues in grad-CAM results is the final upsampling step. The feature maps produced by the backbone have considerably smaller dimensions than the original input. As such emerges the idea of manufacturing grad-CAM heatmaps with gradients calculated with respect to convolutional layers other than the last one. In addition to mitigating this dimensional problem, this approach also captures patterns learned by the network at intermediate backbone levels.

The evaluation of Grad-CAM is primarily visual, but due to the issues already discussed in the section 5.1.1, a complementary numerical assessment is needed. However, the nature of counterfactual analysis makes it impossible to compare with other (non-contrastive) methods in the same terms. Therefore, the Pointing Game metric is adapted to the results theoretically expected for contrastive heat maps a priori. With this in mind, a saliency map counts as a hit if its highest activated feature is located in a region that belongs exclusively to the contrastive bounding box, as these regions should be the most impactful in the increase of the relative loss function.

Table 5.3 contains the results of the pointing game experiment with the experimental setup used in this section. Narrowing the region where activated pixels are found to lesser area results in fewer hits than when the evaluation is made using the standalone predicted bounding box. Surprisingly, the ResNet-50 backbone obtains the best performance out of all models tested. Overall, the numerical assessment confirms that the results are not great. The unsatisfactory results are due to the lack of localization capability of Grad-CAM and gradient-based techniques in general due to their high susceptibility to noise.

5.2 Guided Backpropagation

Guided Backpropagation has gained considerable attention for its ability to highlight the important features and regions within an input that influence the network’s decision-making process. In this section, the results obtained using the Guided Backpropagation approach are exposed, shedding light on the interpretability and insights it provides into the decision-making process of deep neural networks. By examining the highlighted features and regions, we gain a deeper understanding of the reasoning of the network, and uncover potential avenues for improving model performance and robustness.

Similarly to Grad-CAM, this approach takes as a parameter a value for which the influence of the pixels in the input image is observed. As such, both options studied in section 5.1 are viable and, therefore, applied. However, the saliency maps generated with contrastive analysis are not satisfactory enough to be considered counterfactual explanations, as the pixels highlighted mainly focus regions of the object instead of pixels that should positively influence the loss (figure A.2).

$$W_k = \frac{1}{Z} \sum_i \sum_j \frac{\partial T(bbox)}{X} \quad (5.4)$$

Equation 5.4 formalizes the formula for the guided backpropagation approach explored in this section. Mathematically, the computation is similar to that of equation 5.1. The gradients are obtained with respect to the original image X , without any need to involve any feature maps in the calculation. Therefore, the second step needed for Grad-CAM (equation 5.2) is replaced by an average pooling of the channels to obtain a heat map with the correct dimensions.

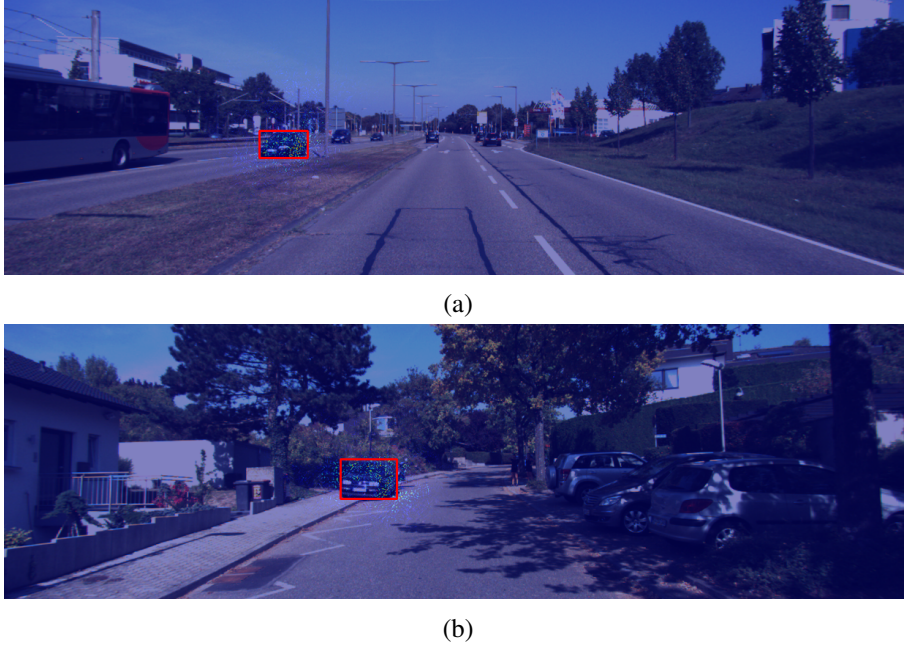


Figure 5.7: Guided Backpropagation examples using the ResNet-101* backbone.

Figure 5.7 contains saliency maps generated using guided backpropagation. The results shown are obtained using the ResNet-101 backbone, which has not been trained for class discrimination since this model behaves slightly better than the others across all explanation techniques. The visual difference between the backbone architectures is reasonably small (figure A.4). Moreover, the transformation used in the calculation of the gradient is the euclidean diagonal distance. In addition, and for visualization-related purposes, a final step is included where the gradients computed are multiplied by a constant to increase activations and establish a more evident contrast between highlighted pixels and the others.

The central aspect to observe is that the heatmap generated is much sparser than those generated with Grad-CAM. Guided backpropagation highlights regions at a pixel level, as the gradients

PG	distance transformation	slope transformation
ResNet-50	0.745	0.736
ResNet-101	0.724	0.7
ResNet-101*	0.757	0.75

Table 5.4: Pointing Game scores for Guided Backpropagation. Higher values mean better performance.

are calculated directly from the image. Consequently, the problems introduced in Grad-CAM by the upsampling step do not exist in this method. Consequently, guided backpropagation provides more intuitive explanations when the analysis focuses solely on localization arguments. In theory, the saliency map should highlight pixels close to the diagonal points of the bounding box, as they are responsible for the distance value. However, the empirical results often contain highly activated pixels in more central regions of the object without following the expected pattern (figure 5.7a). The type of transformation used does not have a significant visual impact on the results obtained (figure A.3). On the other hand, figure 5.7b depicts an example where most of the highlighted pixels are around the edge of the bounding box, serving as a viable explanation for its coordinates.

Furthermore, a numerical evaluation is performed to complement the visual analysis of the heat maps. The Pointing Game metric is also suited to this method since guided backpropagation is similar to the Grad-CAM, with both techniques producing arrays with normalized activations. Table 5.4 confirms that the localization capability of the guided backpropagation method is considerably better than that of Grad-CAM techniques explored in section 5.1 for the Kitti dataset. Moreover, comparing different backbone architectures exposes a slight advantage for the pure regression ResNet-101 model. However, the performances concerning the Pointing Game metric are close to one another. When it comes to the transformation type (e.g., euclidean distance, geometric slope), calculating the gradients with the distance seems to obtain marginally better saliency maps.

5.3 Smooth Grad

In related literature, it is common to tackle the usual problems of Grad-CAM with Smooth Grad. This technique introduces a form of regularization by leveraging the concept of gradient noise to enhance the quality and reliability of the model’s predictions. SmoothGrad has gained significant attention due to its ability to provide valuable insights into the decision-making process of neural networks and its potential applications in various domains, including computer vision, natural language processing, and medical imaging. This section of the dissertation presents an overview of SmoothGrad, its underlying principles, and the results obtained through its application in the context of this work. In addition, a comparison is established with methods explored in sections 5.1 and 5.2.

The algorithm applied takes the instructions given in the original paper [40] and follows the majority of them while making some modifications to better fit the context of this work. The noise level (σ) consists of the standard deviation used to generate random floats that should add to the original image. In the aforementioned paper, the authors conclude that a value between 10% and 20% is optimal for maintaining the image’s structural aspects and sharpening the generated heat maps. In this work, the σ value was set to 15%. Regarding the sample size (n), the authors also observed little to no smoothening for cases where more than 50 perturbed images are analyzed. Despite setting n to 50 in an initial phase, empirical results point out that in this autonomous driving context, the sample size can be lowered even further without significant differences regarding the results’ quality. As such, 25 is a comfortable sample size that balances the computational effort required per image and the quality of explanations generated.

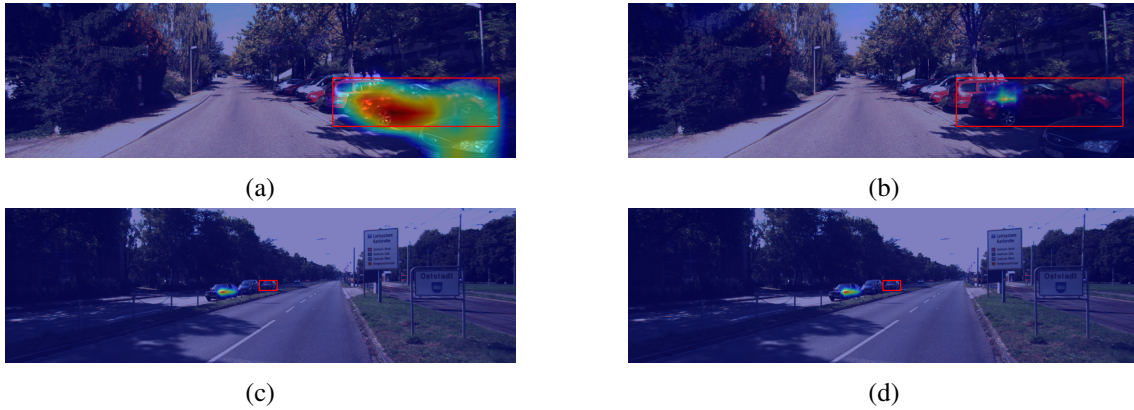


Figure 5.8: Smooth Grad examples using Grad-CAM against the Kitti dataset. (a) and (c) show Grad-CAM results for two distinct examples. (b) and (d) contain the Smooth Grad heatmaps for each example.

Figure 5.8 comprises the strengths and weaknesses of the SmoothGrad method applied to Grad-CAM activations. Figures 5.8a and 5.8b outline the capacity of SmoothGrad to reduce the number of activated pixels drastically. On the other hand, in examples where the Grad-CAM activation depicts unsatisfactory explanations due to its noise susceptibility (figures 5.8c), SmoothGrad does not solve the issue (figure 5.8d).

Furthermore, SmoothGrad is a versatile technique since it can be directly used with different saliency map techniques. Therefore, observing its behavior with guided backpropagation heat maps is also interesting. The latter method is less prone to noise interference than Grad-CAM, which, in theory, should result in a less notable quality increase when guided backpropagation is used in the context of the SmoothGrad technique. Figure 5.9 displays a saliency map similar to the ones described in section 5.2, but where pixels appear more aggregated and focus particular regions of the objects instead of being equally distributed along its area.

The numerical assessment of SmoothGrad results is also performed with the Pointing Game metric to allow a meaningful comparison with other gradient-based techniques. Table 5.5 displays the results obtained for activation maps generated using the smoothening method presented in this



Figure 5.9: Guided backpropagation with noise reduction using SmoothGrad.

section. When the euclidean diagonal distance represents the bounding boxes, the localization capability of the averaged gradients is worse than in Grad-CAM. On the other hand, explanations leveraging the slope transformation observe a consistent increase in their quality. Furthermore, contrastive explanations achieve a substantial decrease regarding the Pointing Game metric with this technique (table A.1). The noise in each iteration of the algorithm brings considerable differences between each heat map, making it hard for the averaged saliency map to capture features important in the overall picture.

Overall, SmoothGrad helps mitigate the noise-related vulnerability of the Grad-CAM method. However, in most cases where explanations could be more satisfactory, this method cannot fix the problem, as it often activates fewer pixels in the same regions. Therefore, considering the computational cost of several backpropagation operations and the numerical results obtained, it is possible to conclude that this method is the least viable amongst its competitors (e.g., Grad-CAM, guided backpropagation).

5.4 Comparison of Methods

Sections 5.1, 5.2 and 5.3 describe distinct gradient-based methods and outline what results to expect from them in an autonomous driving context. The main advantages of gradient-based techniques are related to their time efficiency (table 5.6), i.e., a heat map can be produced in a single backpropagation. The obvious exception is the SmoothGrad method, which iteratively computes n backpropagations and, thus, takes significantly longer time to run. In addition, the runtime difference between Grad-CAM and guided backpropagation is related to the fact that the

PG	distance transformation	slope transformation
ResNet-50	0.27	0.21
ResNet-101	0.29	0.27
ResNet-101*	0.4	0.14

Table 5.5: Pointing Game scores for SmoothGrad. Higher values mean better performance.

Time (seconds)	
Grad-CAM	0.161
Guided backpropagation	0.295
SmoothGrad	1.59

Table 5.6: Time comparison for all gradient-based methods. Each value contains the elapsed time (in seconds) after generating a heatmap using each approach.

latter needs more gradient chain operations to be performed, as it backpropagates information through the entire backbone. The values in the aforementioned table are obtained using a ResNet-101 backbone, which is the worst case as it has more convolutional layers and, consequently, higher inference time.

On the other hand, these methods are highly vulnerable to noise in the input data. Since gradients capture the sensitivity of the model’s output to changes in the input, noise or small perturbations can affect the gradients and, consequently, the resulting heat map. This mathematical issue impacts the overall quality of visual explanations, as the heat maps often highlight regions that are irrelevant to the bounding box regression. Moreover, it becomes clear that the Kitti dataset is more challenging for gradient-based methods, as the higher number of objects and their lower dimension in the frame make the gradients more vulnerable to noise.

Despite achieving inferior numerical results, contrastive explanations generally depict more intuitive explanations and are easier for humans to interpret. The gradients computed using scalar transformations highlight regions closer to the respective bounding box but not necessarily intrinsic to any physical attribute of the object. As such, it becomes harder to extract the reasoning that leads the model to take decisions regarding the object localization. However, as concluded in sections 5.3 and 5.2, contrastive explanations are solely viable using Grad-CAM. Given the nature of saliency methods, the visual quality of the heat map is more important than any metric concerning the interpretability of the method.

Furthermore, a meaningful comparison can be established with respect to the backbone architecture used in the model to be explained. As expected, the ResNet-101* architecture achieves better performance for all gradient-based methods. This experimental setup draws the conclusion that a model whose backbone is solely optimized for the localization part of the object detection task is slightly less vulnerable to noise in the input data when fed to a gradient-based xAI technique. In other words, the gradients computed for this model are better at determining important features than other class discriminative models. However, since class discrimination is essential in real world object detection, it is also crucial to assess the performance of ResNet-50 and ResNet-101 backbones. In this case, the largest architecture is not always related to better explainability, according to the Pointing Game metric and the visual characteristics of saliency maps produced.

Chapter 6

Perturbation-based approach

This chapter investigates a perturbation-based method known as D-RISE (Deep Recursive Instance Selection and Elimination) in the context of xAI for object detection in autonomous driving scenarios. D-RISE is an innovative approach that perturbs the input image iteratively to highlight the regions contributing to the model’s decision-making process. By systematically eliminating less relevant areas, D-RISE generates informative explanations that help understand the object detection process and the underlying behavior of the model. A more detailed overview of this algorithm is presented in section 3.1.3.

The primary objective of this chapter is to explore the experimental setup and evaluate the effectiveness of D-RISE in providing interpretable explanations for object detection in autonomous driving. Due to the context of this work, the results obtained with the technique explored in this section are focused on the Kitti dataset. By conducting rigorous experiments and comparative analyses, we aim to gain insights into the strengths and limitations of D-RISE and its potential for improving the interpretability of autonomous driving systems. In addition, it is also essential to compare this technique with gradient-based methods according to relevant aspects.

Section 6.1 explains the methodology used and adaptations made to fit the context of this work better. Section 6.2 explores further modifications to the method at hand to produce contrastive explanations. Finally, section 6.3 wraps the chapter by comparing visual and numerical results obtained with methods described in this chapter and in chapter 5.

6.1 Experimental Setup

The D-RISE algorithm includes several steps and can be tweaked according to some parameters. The noise introduction step is the same as described in the RISE paper [27]. The latter considers an h by w image and assigns its values randomly, binary using a certain probability threshold, a parameter of the method. For each cell, if the random value lies bigger than the threshold, the value is set to 1; otherwise, it is 0. This small image is then upsampled by way of linear interpolation, resulting in an image with the exact shape of the original, containing values between 0 and 1. The original image is multiplied by the noise obtained in the steps described in this paragraph, resulting

in a noisy image that is evaluated by the detector. The detection vector of the target bounding box being explained is compared to the closest bounding box obtained through the inference, and their similarity is objectified with a score. The pairwise similarities are calculated and the highest score is added to the final activation map. The similarity scores act as weights to the final saliency map. This process is repeated N times, being N another parameter.

In this work, all the parameters are set to the default values used in the D-RISE paper [28]. h and w are set to generate a grid size of (16, 16); the probability threshold is 0.5, and the number of masks N generated is 5000. The values used allow a good balance through the examples studied, both in their quality and the computational effort required for their production. Naturally, increasing the number of iterations results in sharper saliency maps. However, the differences are substantial and not worth adopting, especially when considering using this method for generating explanations on a large scale. Moreover, all D-RISE empirical results in this chapter and attached to the appendix are obtained with the ResNet-101 backbone. The backbone architecture influence in the results is not explored, as it is not intrinsic to the method's behavior, as opposed to what happens with gradient-based methods.

Furthermore, the discussion regarding detection vectors is also essential, as they are the most meaningful addition introduced by D-RISE. Petsiuk et al. (2021) [28] use detection vectors to encode bounding box information, namely localization, classification, and other aspects for individual detections. The localization element of the vector consists of an IoU score between the human-annotated bounding box and the one predicted by the model after the perturbation is applied. The classification information is the cosine similarity between the class probabilities for the target detection and the bounding box obtained through inference. In addition, the authors use the objectness score as a third element of this vector, which is omitted in this work.

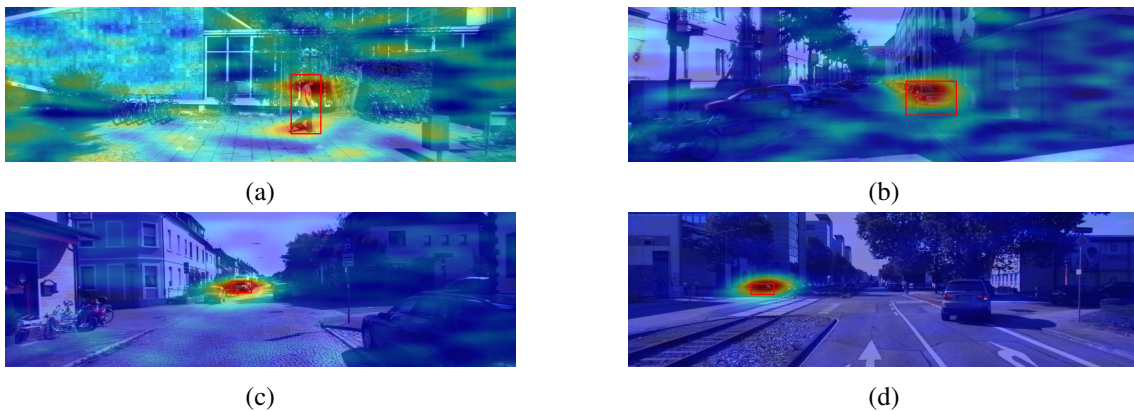


Figure 6.1: Saliency maps using the off-the-shelf D-RISE algorithm.

Figure 6.1 displays several examples of heat maps obtained through the D-RISE method. The detection vectors used for computing the final saliency maps of the latter figure comprise both localization and classification information, which is the closest approximation to the original method. The visual results are promising, as the method generally highlights areas inside the bounding box or close to it reasonably precisely. Moreover, regions irrelevant to the detection

have a low activation value, indicating that the model does not consider them important to the regression. However, the results are not totally perfect, as figure 6.1a suggests some confusion in the model’s reasoning by activating some seemingly random pixels far from the object considered.

In order to address issues related to noisy regions being captured as important to the model’s reasoning, the probability threshold parameter is adjusted to fit the characteristics of this work better. The average size of objects in the Kitti dataset is significantly smaller than that of datasets such as MS COCO or Pascal VOC (section 4.1.3). In addition, a high amount of pixels affected by the masking algorithm can result in inferences that miss entirely the object detection being explained, which harms the final interpretation quality. Therefore, lowering the probability threshold is helpful in this context, as it reduces the noise introduced in the image and assures that the object is being detected. Figure 6.2 establishes the comparison between results obtained with different thresholds, as it displays a more explicit explanation than its previous counterpart (figure 6.1a) using a probability threshold of 0.25.

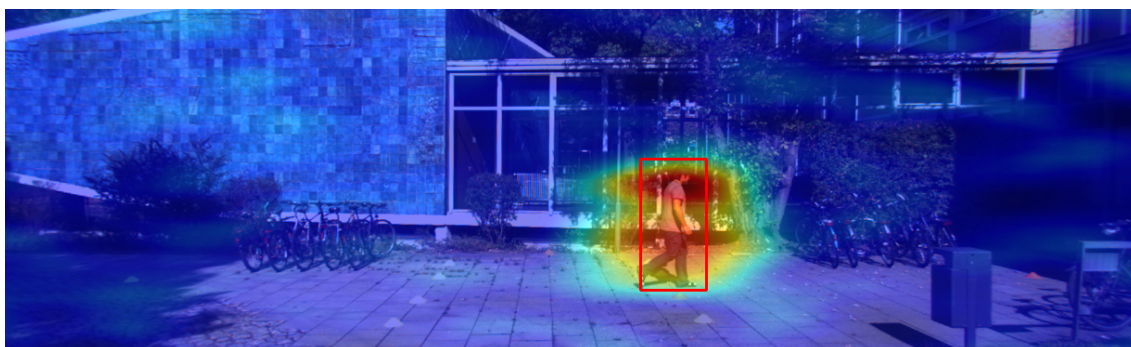


Figure 6.2: D-RISE interpretation with probability threshold of 0.25.

Furthermore, since this work focuses on the regression part of the object detection task, it makes sense to adjust the detection vector to convey solely localization information. Therefore, another setup is defined where the classification information is disregarded, and the weights are merely composed of IoU scores between inferences and annotations. Visually, the saliency maps look identical (figures 6.1d and 6.3), but the numerical assessment for these setups are divergent. Most class probability vectors are colinear with the probability distribution vector of the ground-truth detection. As such, the cosine similarity adds no useful information throughout the iterations and should not be considered in the context of regression explanations.

6.2 Contrastive Explanations with D-RISE

Petsiuk et al. (2021) [28] outline the possibility of explaining any arbitrary detection vector in their work. More specifically, they produce heat maps concerning failure cases of the model by comparing the results for a detection vector of the missed ground truth detection with the results that consider the bounding box predicted by the network. Consequently, the difference between saliency maps conveys information regarding the cause of the failure, i.e., highlights

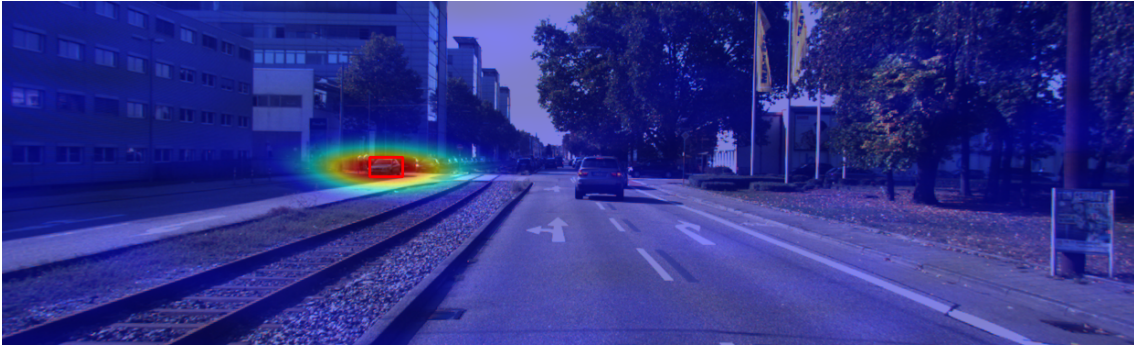


Figure 6.3: D-RISE generated with localization information. The classification is disregarded for the heatmap generation.

regions wrongly considered important for the outcome. In this work, this technique can be adapted to produce contrastive explanations similar to those obtained in section 5.1.2. Therefore, arbitrary counterfactual bounding boxes are manufactured in the same manner as in the latter section.

The counterfactual analysis method proposed in this section requires the computation of two saliency maps, one considering the target bounding box (as in section 6.1) and another with respect to the contrastive bounding box. Consequently, the first heat map salients regions that influence the prediction of bounding boxes with a high IoU score (closely located) with respect to the ground truth, while the contrastive heat map highlights zones that influence the model to predict wrong coordinates. The difference between the saliency maps brings an interesting discussion, as it is related to regions that influence correct regressions and, simultaneously, are not considered when the predictions are mislocated. Moreover, the computation of the additional saliency map does not increase the computational power considerably. Both heat maps use the same masks and, thus, can be obtained in N iterations.



Figure 6.4: Contrastive explanations with D-RISE. Most activated areas are in the region that exclusively belongs to the predicted bounding box.

Figure 6.4 displays an example of an explanation obtained with this D-RISE adaptation. Due to reasons discussed in section 6.1, the probability threshold used for the mask generation algorithm is set to 0.25 throughout the production of contrastive explanations in this work. In gradient-based methods, the smooth L1 loss function is used to compute the gradients and, thus, highlight regions

that contribute to the loss increase. On the other hand, D-RISE heat maps offer the opposite analysis, as no loss function is used. Consequently, instead of highlighting regions exclusive to the contrastive bounding box, explanations such as the one in the figure mentioned above contain high activations in regions exclusive to the target bounding box. This type of behavior in the heat maps produced is correct, as one should expect that the activated pixels shown positively influence the prediction of a correct bounding box and a negative influence on the mislocated detection.

6.3 Comparison with Gradient-based Methods

In sections 6.1 and 6.2, it is possible to observe a superiority regarding the visual quality of explanations derived from D-RISE with respect to the ones explored in chapter 5, generated with gradient-based techniques. D-RISE saliency maps output a straightforward interpretation, highlighting pixels close to the human-annotated detection and disregarding pixels far from it. Despite not being completely immune to other objects near the target to be explained, the method explored in this chapter can still produce saliency maps where their most activated pixels are within the limits of the bounding box considered. As such, the noise caused by their presence is nowhere near as disturbing as in gradient-based methods. This difference can be explained by the mathematical limitations of the gradients discussed in section 5.4, as well as the final upsampling step exclusive to techniques derived from Grad-CAM. Other gradient-based techniques like guided backpropagation provide a more simplistic approach with similar localization capability relative to D-RISE. However, the heatmaps obtained through this gradient-based technique are significantly sparser. Therefore, despite focusing on regions within the bounding box, there is no distinction between the importance of each highlighted pixel.

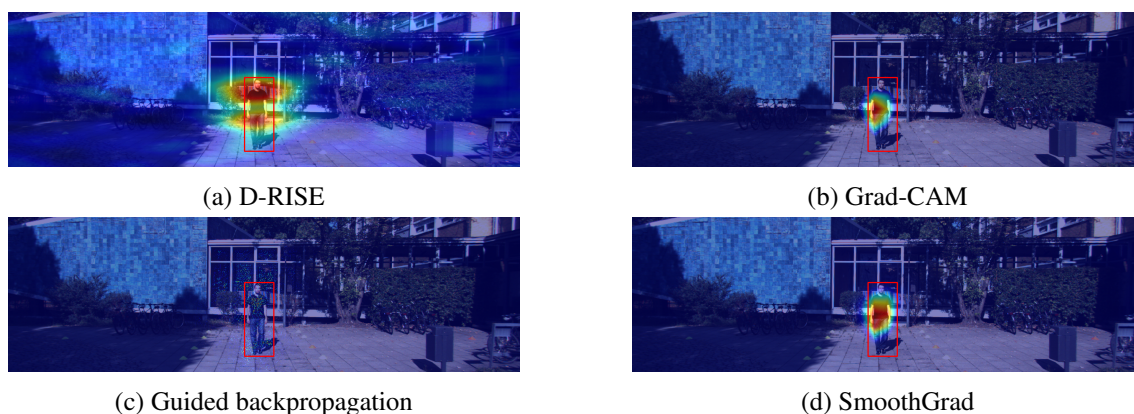


Figure 6.5: Saliency maps using D-RISE and gradient-based methods. All interpretations are generated with respect to the same image.

Figure 6.5 displays explanations using the techniques described throughout this dissertation. Gradient-based methods produce reasonably satisfactory saliency maps, highlighting features inside the human-annotated bounding box with little to no noise captured. In this case, the image

PG scores	
D-RISE (Localization)	0.75
D-RISE (Localization & Classification)	0.72
D-RISE (Contrastive)	0.21
Grad-CAM	0.44
Guided backpropagation	0.76

Table 6.1: Pointing Game scores for D-RISE explanations.

contains a single object that is emphasized in the image, making it less of a challenge for gradient-based techniques. However, from an explainability point of view, D-RISE is still slightly better than other gradient methods in this best-case scenario, as it highlights more area inside the bounding box (despite activating some pixels outside of it as well), resulting in a more intuitive interpretation. Regarding contrastive explanations for the same input image (figure 6.6), D-RISE is a much better option than Grad-CAM. It is also important to note that other gradient-based methods are not viable for contrastive explanations.

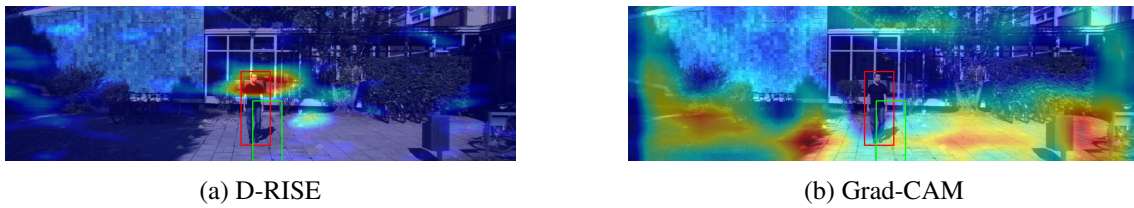


Figure 6.6: Saliency maps using D-RISE and gradient-based. The red bounding boxes are predicted by the model, whereas the green rectangles represent contrastive examples.

Furthermore, the nature of explanations generated with D-RISE is identical to that of gradient-based techniques, i.e., these methods produce activation maps with similar characteristics. Therefore, a meaningful objective comparison can be established by considering the Pointing Game metric. Table 6.1 displays results measured in the validation set of the Kitti dataset, which was the set used to evaluate the techniques described in chapter 5. The results regarding gradient-based techniques are directly extracted from tables 5.2 and 5.4. The first pair of values in the latter table exposes an expected advantage in the localization capability of the results when the classification information is disregarded in the detection vectors.

Moreover, this numerical evaluation confirms the impression of the visual observation of the results, as the D-RISE technique achieves a significantly higher score than the best Grad-CAM setup (e.g., ResNet-101* with distance transformation). In addition, results concerning contrastive explanations conclude that D-RISE performs twice as well as the best Pointing Game score obtained with Grad-CAM (table 5.3). Nevertheless, the guided backpropagation results are slightly better than the perturbation-based method explored in this chapter with respect to the localization metric considered.

On the other hand, due to the nature of perturbation-based methods, namely the issues related to their time efficiency, a proper discussion regarding this method’s runtime is in place. According to table 5.6, gradient-based methods can generally produce results in second fractions, as it only needs a single pass through the network. Nevertheless, perturbation-based methods such as D-RISE need several iterations to add noise to the original image and observe its effect on the model’s behavior. Consequently, the runtime of the latter method is bottlenecked by the inference time of a single image. In the experimental setup of this work, an Nvidia RTX A2000 takes, on average, 369 seconds to produce an interpretation for a given image.

Chapter 7

Conclusions

The increasing popularity of autonomous vehicles leads to an increase in the computer vision research body of knowledge. Consequently, the development of black-box models tackling perception tasks such as object detection pushes the state-of-the-art toward more precise and computationally efficient detectors responsible for understanding the surrounding environment of the car. Moreover, the availability of powerful computing hardware and the utilization of parallel processing techniques have accelerated the real-time processing capabilities of object detectors, making them suitable for deployment in autonomous vehicles.

On the other hand, explainable AI plays a crucial role in promoting transparency, accountability, trust, error diagnosis, user understanding, ethical considerations, human-AI collaboration, and education. Due to the highly critical nature of autonomous driving, namely the security-related concerns inherent to perception models, the need to explore xAI methods to obtain interpretations regarding their output emerges. In this dissertation, model-agnostic techniques are explored, aiming to understand and assess the reasoning of object detectors. In particular, this work focuses on explanations regarding bounding box coordinates. Therefore, and due to the choice of two-stage detectors, the techniques explored throughout this thesis can be incorporated into other complex architectures that already provide interpretations for the classification task in object detection.

This dissertation conducts intensive experimentation on different post-hoc xAI techniques tested on different datasets and object detectors. All the experimental setups used in this work consist of saliency maps, which makes their comparison more fair. As such, the results are visually displayed and discussed, while other numerical evaluation techniques are also employed to obtain an unbiased assessment of the localization capability of each technique. However, due to the context in which the work is carried out, the assessment phase of each technique assigns more value to explanations made in an autonomous driving dataset.

Gradient-based methods are fast to compute but often lead to unsatisfactory results, mainly when the image contains many objects or when the object explained is small, as these factors contribute to more noise being captured in the gradients. Exploring the latter method family comprises several experimental setups and assesses which factors of the dataset and black-box model influence the noise in the gradients. In addition, and according to the empirical results

obtained, the backbone architecture bears no significant weight in the quality of explanations. Furthermore, in autonomous driving, simplistic approaches such as guided backpropagation can provide more valuable information than more complex methods, as they are less prone to noise in the image and do not include an upsampling step.

Nevertheless, a comprehensive exploration of the D-RISE algorithm is performed. It includes a discussion of the experimental setup, presenting visual and numerical results, and comparing D-RISE with gradient-based methods, highlighting the advantages of D-RISE in terms of visual quality and interpretability. The results differ greatly from those of gradient-based methods, as the computational effort needed for this perturbation approach is significantly more extensive. However, the resulting saliency maps provide more clever interpretations, highlighting regions that should be important in the coordinates regression.

In summary, autonomous driving is a challenging context for object detectors, as the images have a particular nature with many objects, some of which are far from the camera. Therefore, according to the results obtained in the experimental setups tested, the D-RISE method is the most appropriate technique for the Kitti dataset.

7.1 Further Work

As mentioned throughout the document, all the techniques adopted in this dissertation are model-agnostic. The upside of this approach is related to its versatility to be used with different model architectures. On the other hand, post-hoc methods do not provide global explanations like those guaranteed by intrinsically explainable models. In section 3.2, explainable models are explored, and their behavior is described with respect to different contexts in computer vision. As such, a possible increment to work carried out is to adapt a specific global explanation method (e.g., ProtoPNet [5]) to the bounding box regression network.

Furthermore, additional tests and experiences with model-agnostic techniques can be performed:

- Explore sensor fusion and the influence of point cloud data in the detections;
- Explore the approaches suggested in this document to explain cases where the model predicts mislocated bounding boxes;
- Experiment with other networks, namely one-stage detectors such as YOLO.

References

- [1] Luqman Ali, Fady Alnajjar, Hamad Jassmi, Munkhjargal Gochoo, Wasif Khan, and Mohamed Serhani. Performance evaluation of deep cnn-based crack detection and localization techniques for concrete structures. *Sensors*, 21:1688, 03 2021.
- [2] Alina Jade Barnett, Fides Regina Schwartz, Chaofan Tao, Chaofan Chen, Yinhao Ren, Joseph Y. Lo, and Cynthia Rudin. IAIA-BL: A Case-based Interpretable Deep Learning Model for Classification of Mass Lesions in Digital Mammography, March 2021. arXiv:2103.12308 [cs].
- [3] Mariusz Bojarski, Chenyi Chen, Joyjit Daw, Alperen Degirmenci, Joya Deri, Bernhard Firner, Beat Flepp, Sachin Gogri, Jesse Hong, Lawrence D. Jackel, Zhenhua Jia, B. J. Lee, Bo Liu, Fei Liu, Urs Muller, Samuel Payne, Nischal Kota Nagendra Prasad, Artem Provodin, John Roach, Timur Rvachov, Neha Tadimetri, Jesper E. van Engelen, Haiguang Wen, Eric Yang, and Zongyi Yang. The NVIDIA pilotnet experiments. *CoRR*, abs/2010.08776, 2020.
- [4] Aditya Chattopadhyay, Anirban Sarkar, Prantik Howlader, and Vineeth N Balasubramanian. Grad-cam++: Generalized gradient-based visual explanations for deep convolutional networks. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 839–847, 2018.
- [5] Chaofan Chen, Oscar Li, Chaofan Tao, Alina Jade Barnett, Jonathan Su, and Cynthia Rudin. This Looks Like That: Deep Learning for Interpretable Image Recognition, December 2019. arXiv:1806.10574 [cs, stat].
- [6] Yu-Liang Chou, Catarina Moreira, Peter Bruza, Chun Ouyang, and Joaquim A. Jorge. Counterfactuals and causability in explainable artificial intelligence: Theory, algorithms, and applications. *CoRR*, abs/2103.04244, 2021.
- [7] 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)*, volume 1, pages 886–893 vol. 1, 2005.
- [8] Jon Donnelly, Alina Jade Barnett, and Chaofan Chen. Deformable ProtoPNet: An Interpretable Image Classifier Using Deformable Prototypes, July 2022. arXiv:2111.15000 [cs].
- [9] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88(2):303–338, June 2010.
- [10] Junxi Feng, Xiaohai He, Qizhi Teng, Chao Ren, Honggang Chen, and Yang Li. Reconstruction of porous media from extremely limited information using conditional generative adversarial networks. *Physical Review E*, 100, 09 2019.

- [11] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3354–3361, 2012.
- [12] Ross B. Girshick. Fast R-CNN. *CoRR*, abs/1504.08083, 2015.
- [13] Ross B. Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *CoRR*, abs/1311.2524, 2013.
- [14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.
- [15] Viktória Ilková and Adrian Ilka. Legal aspects of autonomous vehicles—an overview. In *2017 21st international conference on process control (PC)*, pages 428–433. IEEE, 2017.
- [16] Paul Jacob, Éloi Zablocki, Hedi Ben-Younes, Mickaël Chen, Patrick Pérez, and Matthieu Cord. STEEX: steering counterfactual explanations with semantics. *CoRR*, abs/2111.09094, 2021.
- [17] Glenn Jocher, Ayush Chaurasia, and Jing Qiu. YOLO by Ultralytics, January 2023.
- [18] Ayoosh Kathuria. What’s new in YOLO v3?, April 2018.
- [19] Jinkyu Kim and John F. Canny. Interpretable learning for self-driving cars by visualizing causal attention. *CoRR*, abs/1703.10631, 2017.
- [20] Simon Letzgus, Patrick Wagner, Jonas Lederer, Wojciech Samek, Klaus-Robert Müller, and Grégoire Montavon. Toward Explainable Artificial Intelligence for Regression Models: A methodological perspective. *IEEE Signal Processing Magazine*, 39(4):40–58, July 2022. Conference Name: IEEE Signal Processing Magazine.
- [21] Simon Letzgus, Patrick Wagner, Jonas Lederer, Wojciech Samek, Klaus-Robert Müller, and Grégoire Montavon. Toward explainable artificial intelligence for regression models: A methodological perspective. *IEEE Signal Processing Magazine*, 39(4):40–58, 2022.
- [22] Oscar Li, Hao Liu, Chaofan Chen, and Cynthia Rudin. Deep learning for case-based reasoning through prototypes: A neural network that explains its predictions. *CoRR*, abs/1710.04806, 2017.
- [23] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014.
- [24] David G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 60(2):91–110, November 2004.
- [25] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.
- [26] Mayank Mishra. Convolutional Neural Networks, Explained, September 2020.
- [27] Vitali Petsiuk, Abir Das, and Kate Saenko. RISE: randomized input sampling for explanation of black-box models. *CoRR*, abs/1806.07421, 2018.

- [28] Vitali Petsiuk, Rajiv Jain, Varun Manjunatha, Vlad I. Morariu, Ashutosh Mehra, Vicente Ordonez, and Kate Saenko. Black-box explanation of object detectors via saliency maps. *CoRR*, abs/2006.03204, 2020.
- [29] Mohit Prabhushankar, Gukyeong Kwon, Dogancan Temel, and Ghassan AlRegib. Contrastive explanations in neural networks. In *2020 IEEE International Conference on Image Processing (ICIP)*, pages 3289–3293, 2020.
- [30] Yanou Ramon, David Martens, Theodoros Evgeniou, and Foster Provost. A comparison of instance-level counterfactual explanation algorithms for behavioral and textual data: Sedc, lime-c and shap-c. *Advances in Data Analysis and Classification*, 09 2020.
- [31] Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. *CoRR*, abs/1506.02640, 2015.
- [32] Joseph Redmon and Ali Farhadi. YOLO9000: better, faster, stronger. *CoRR*, abs/1612.08242, 2016.
- [33] Joseph Redmon and Ali Farhadi. YoloV3: An incremental improvement. *CoRR*, abs/1804.02767, 2018.
- [34] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster R-CNN: towards real-time object detection with region proposal networks. *CoRR*, abs/1506.01497, 2015.
- [35] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. *CoRR*, February 2016.
- [36] Ramprasaath R. Selvaraju, Abhishek Das, Ramakrishna Vedantam, Michael Cogswell, Devi Parikh, and Dhruv Batra. Grad-cam: Why did you say that? visual explanations from deep networks via gradient-based localization. *CoRR*, abs/1610.02391, 2016.
- [37] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. *CoRR*, abs/1704.02685, 2017.
- [38] Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition, April 2015. arXiv:1409.1556 [cs].
- [39] Gurmail Singh and Kin-Choong Yow. These do not Look Like Those: An Interpretable Deep Learning Model for Image Recognition. *IEEE Access*, 9:41482–41493, 2021. Conference Name: IEEE Access.
- [40] Daniel Smilkov, Nikhil Thorat, Been Kim, Fernanda B. Viégas, and Martin Wattenberg. Smoothgrad: removing noise by adding noise. *CoRR*, abs/1706.03825, 2017.
- [41] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for Simplicity: The All Convolutional Net, April 2015. arXiv:1412.6806 [cs].
- [42] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. *CoRR*, abs/1703.01365, 2017.
- [43] Yugesh Verma. A Complete Understanding of Dense Layers in Neural Networks, September 2021.

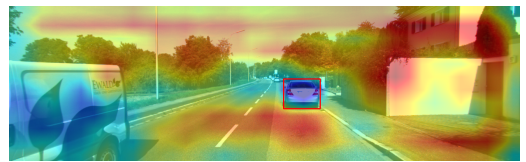
- [44] Sandra Wachter, Brent Mittelstadt, and Chris Russell. Counterfactual Explanations Without Opening the Black Box: Automated Decisions and the GDPR. *SSRN Electronic Journal*, 2017.
- [45] Dequan Wang, Coline Devin, Qi-Zhi Cai, Fisher Yu, and Trevor Darrell. Deep object centric policies for autonomous driving. *CoRR*, abs/1811.05432, 2018.
- [46] Adam White and Artur S. d’Avila Garcez. Measurable counterfactual local explanations for any classifier. *CoRR*, abs/1908.03020, 2019.
- [47] Yuxin Wu and Kaiming He. Group normalization. *CoRR*, abs/1803.08494, 2018.
- [48] Mengqi Xue, Qihan Huang, Haofei Zhang, Lechao Cheng, Jie Song, Minghui Wu, and Mingli Song. ProtoPFormer: Concentrating on Prototypical Parts in Vision Transformers for Interpretable Image Recognition, September 2022. arXiv:2208.10431 [cs].
- [49] Éloi Zablocki, Hédi Ben-Younes, Patrick Pérez, and Matthieu Cord. Explainability of deep vision-based autonomous driving systems: Review and challenges, July 2022. arXiv:2101.05307 [cs].
- [50] Matthew D. Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. *CoRR*, abs/1311.2901, 2013.
- [51] Jianming Zhang, Sarah Adel Bargal, Zhe Lin, Jonathan Brandt, Xiaohui Shen, and Stan Sclaroff. Top-down neural attention by excitation backprop. *International Journal of Computer Vision*, 126(10):1084–1102, Oct 2018.

Appendix A

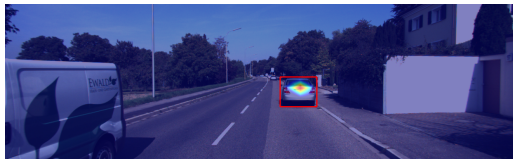
Appendix



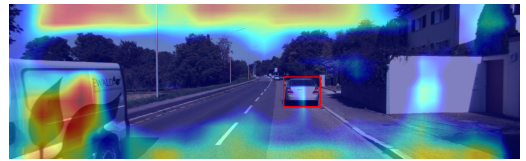
(a) xmin Grad-CAM explanation



(b) ymin Grad-CAM explanation



(c) xmax Grad-CAM explanation



(d) ymax Grad-CAM explanation

Figure A.1: Explanations using individual coordinates.



Figure A.2: Contrastive analysis with guided backpropagation.



Figure A.3: Guided backpropagation with slope transformation.

Pointing Game	
ResNet-50	0.01
ResNet-101	0.04
ResNet-101*	0.03

Table A.1: Pointing Game scores for contrastive explanations on kitti using SmoothGrad.



(a) Distance with RN-50.



(b) Distance with RN-101.



(c) Distance with RN-101*.

Figure A.4: Guided backpropagation with different backbone architectures.