Instituto Superior de Engenharia de Lisboa
Área Departamental de Matemática

Comparison between interpretability methods in object detection

Diogo Camilo de Carvalho

Mestrado em Matemática Aplicada para a Indústria

Júri:

Presidente: Professor Doutor Luís Silva
Vogais: Professor Doutor Gonçalo Morais
Professor Doutor Ricardo Enguiça

Orientadores:

Professor Doutor Carlos Geraldes
Professor Doutor Ricardo Enguiça

2023

Resumo

As redes neuronais profundas (DNN) são ferramentas muito poderosas e flexíveis. Com tais capacidades, estas têm sido adotadas em diversos tipos de problemas: desde problemas clássicos com dados tabulares até problemas de texto e imagem. Na base dos problemas de imagem, estão as redes neuronais convolucionais (CNN). A CNN é um tipo de DNN especializado em classificação de imagens, segmentação de imagens e deteção de objetos.

No entanto, devido à forma como as DNN são construídas, estes constituem modelos de caixa preta. Ou seja, não é possível compreender as razões subjacentes que levam o modelo a tomar uma decisão específica. Esta limitação do algoritmo leva à resistência em adotar estes métodos em áreas mais sensíveis, como medicina, condução autónoma, finanças, etc. Portanto, a capacidade de interpretar e explicar as razões subjacente de cada decisão do modelo é fundamental para diversas aplicações e será o principal foco desta tese.

Atualmente, existem diversas técnicas que tentam apresentar uma explicação para qualquer modelo de DNN, no entanto, dada a diversidade de técnicas, torna-se difícil identificar quais são as melhores ou mais adequadas para um determinado cenário. Assim, neste estudo, é apresentada uma comparação entre várias técnicas populares de explicabilidade.

Os resultados mostram que técnicas como *D-RISE* e *LIME* conseguem produzir explicações claras sobre as razões por trás de diversos modelo de visão computacional, proporcionando confiança ao utilizador. No entanto, estes métodos revelão-se computacionalmente dispendiosos. Porém, outros algoritmos menos dispendiosos em termos computacionais, como por exemplo o *Grad-CAM* e o *Eigen-CAM* tornam-se boas opções, fornecendo explicações rápidas, mesmo com um decréscimo na qualidade. É também possível concluir que estes métodos são excelentes para detetar problemas não identificados nos dados considerados, evitando dificuldades na fase de produção do sistema.

Palavras Chave

inteligência artificial explicável, técnicas de interpretabilidade, visão computacional, redes neuronais profundas

Abstract

Deep neural networks (DNN) are very powerful and flexible tools that have taken over the entire field. With such capabilities, they have seen a huge adoption rate in wide range of problems: from classical tabular data to text and image problems. On the basis of image problems are Convolutional neural networks (CNN). CNN is a type of DNN specialized for certain tasks like image classification, image segmentation, and object detection, among others.

However, by their design nature, DNNs are a black-box model. That is, it is not possible to understand the underlying reasons that lead the model to make a certain decision. Such inability leads to resistance to adopting these methods in more sensitive areas like medicine, autonomous driving, finance, etc. Hence, the ability to interpret and explain the underlying reason for a model decision is fundamental for a multitude of applications. These are the focus of this study.

Currently, there exists a large variety of techniques that try to provide an explanation for any computer vision DNN model. However, with such a diversity in techniques, it becomes hard to identify which are the best or which are the most adequate for a given scenario. In this study, a comparison between several popular explainability techniques is presented.

The results show that techniques such as D-RISE and LIME can provide clear and insightful knowledge about the reasons behind any computer vision model, hence, providing trust to the user. However, these methods are computationally expensive. Nonetheless, less computationally expensive algorithms, like Grad-CAM and Eigen-CAM, are good variants, providing fast explanations even if with a drop in quality. It was also possible to conclude that these methods are efficient at finding unidentified problems with the data considered, thus saving difficulties in the production phase of the system.

Keywords

explainable artificial intelligence, interpretability techniques, computer vision, deep neural networks

Contents

List of Figures

List of Tables

Acronyms

CAM Class Activation Mapping.

CNN Convolution Neural Network.

DNN Deep Neural Network.

D-RISE Detector Randomized Input Sampling for Explanation.

Grad-CAM Gradient Class Activation Mapping.

IoU Intersection over Union.

LIME Local Interpretable Model-agnostic Explanations.

RNN Recurrent Neural Network.

ROI Region of Interest.

XAI Explainable artificial intelligence.

Chapter 1

Introduction

In this chapter, we explain the main reasons that led to the accomplishment of this work. Furthermore, an initial overview of the developed work is shown.

1.1 Motivation

In the past decades, machine learning techniques have been a very researched topic, and with this research, many breakthroughs have been made in the field. One of the biggest advances made was in the topic of deep neural networks (DNN).

DNNs have seen an increase in popularity in the research domain due to its great predictive capacity. This is especially true for non-structured problems, such as, those with images or text, after the creation of Convolution Neural Networks (CNN) and Recurrent Neural Networks (RNN), respectively [9, 10]. Hence, DNNs are currently one of the main tools used by researchers in a diverse range of areas, such as medicine, in automated diagnostic techniques, for instance through image processing [11]; in finance, in areas such as the evaluation of mortgage risk [12]; in autonomous driving [13]; in judicial systems, in the assessment of risk of recidivism in crime; in image recognition [14] and speech recognition [15]; in machine translation [16]; etc.

However, this adoption has not been replicated in industry. One of the main reasons for that is the fact that DNNs remain opaque tools with a complex and unintuitive process of decision-making, resulting in them being hard to understand, debug and improve. In fact, the impossibility of knowing why a DNN model has made a decision is the key factor to why certain domains where a decision can have serious consequences (e.g., medical diagnosis, criminal justice, etc.) have not adopted these models.

To solve this issue, an emerging new research area has been gaining a lot of attention: Explainable Artificial Intelligence (XAI). As the name indicates, the goal of this area is the development of methods that can give an interpretation of why a specific decision was taken by any model. Improving the interpretability of DNN is not only important for the industry, but also for researchers, as an increased understanding of its learning process could better guide the development of networks, ensuring the models are not learning artifacts or are vulnerable to adversarial attacks. A telling example of such attacks was the discovery that by making certain changes in an image, imperceptible to the human viewer, the correct results of an image classification network would change to an incorrect prediction [17].

Therefore, this work will focus on the study of explainability methods for image classification problems with DNN in order to be able to identify which pixels on a given image had the biggest impact on the prediction. More precisely, the work will focus on explainability methods of DNN models for object detection problems, where the model not only gives a classification of the image but also a localization bounding box of the target.

1.2 Goals

Despite the long history and a significant amount of work in the computer vision field, creating good model explanations is not a trivial task. The motivation for this thesis follows the need to guarantee the reliability of explanation methods, a crucial aspect in tasks where visual inspection of results is not easy, or the costs of incorrect attribution are high. Hence, the purpose of this thesis is to analyze at least one of each sub-category of model explainability algorithms for DNN predictions in object detection problems.

More in detail, the aim of this thesis is:

- Understanding explainable Artificial Intelligence (XAI) and its importance;
- Proposing an overview of the theory and the explanation methods that are more suitable for deep neural networks analysis in object detection problems;
- Comparing the results of the studied interpretability methods, indicating advantages and disadvantages of each. To accomplish this, it will be necessary to:
  - Studying the main deep neural network frameworks, such as PyTorch and TensorFlow;
  - Developing an object detection model to study each XAI methodology.

1.3 Work overview

To accomplish the goals proposed for the current work, this thesis will be presented with the following structure:

- First, the introduction, in which motivation and goals for this thesis are presented.
- Afterwards, in the second chapter a state of the art of object detection is presented. In the same chapter, it is also presented the reason why having an explanation of any model decision is so crucial to any practical task. Lastly, a review of the most common explainability techniques is also made.
- In the third chapter, a thorough analysis of the theoretical background of the explainability algorithms considered is made. Besides it, the metrics used to evaluate each explanation are also shown.
- The fourth chapter contains all the tests and analyses made in order to accomplish the goals of this work. As such, the data and code used to obtain the results are also presented.
- Finally, in the fifth and final chapter, the main conclusions of the project will be stated, also the limitations of the work carried out, and some suggestions for future work.

Chapter 2

State-of-the-art

In this chapter, we review the state-of-the-art for all the relevant subjects relevant to this thesis. Namely, computer vision and object detection techniques, the importance of good model explainability, and the main techniques to do so.

In section 2.1, it is shown the difference between some computer vision problems and the most essential model used in DNN for computer vision problems, the Convolution Neural Network, is described. Finally, specific object detection algorithms are detailed, as well as, the sub-category used to distinguish them: single-stage and two-stage detectors.

Section 2.2 describes when and why it is necessary to have a good model explainability. It is also explained that to obtain a good model explanation, five different characteristics can be assessed, namely: assessing correctness, assessing fairness, knowledge discovery, justification, and user acceptance.

Finally, in Section 2.3, it is given an overview of model explainabilities for all types of models. Then a deeper view of DNN model interpretability techniques is presented. Within this review, three subcategories for model explanations algorithm is analysed: Gradient-based methods, Permutation-based methods, and Meta-model methods.

2.1 Object detection

With the developments achieved in DNNs, several classification areas also advanced, more precisely, areas where the data is not structured, like computer vision and text classification. For these type of problems, traditional classification methods were ineffective, due to the absence of structure in the data.

In these problems, it is not possible to anticipate the structure of the input, hence making

them harder to solve when compared to the traditional classification problems. However, due to the high flexibility of the DNNs, it is now possible to solve these problems. So a big development has been made in the computer vision field, which lead to the creation of three main subcategories of computer vision problems [18–20]:

- Image Classification - This is the first and the main task of computer vision problems. The goal of image classification is to identify the object present in the image, within a certain range of available classes. However, sometimes it is important to know where the class of interest is located in the image. For that reason, two other subcategories were created: Object Detection and Segmentation.
- Object Detection - Object detection is a direct extension of an image classification problem. While in image classification, the goal is only to identify the object present in the image, in object detection, the goal is not just to predict the class of the object but also its localization in the picture. For this, an object detection algorithm will predict the object class and the corresponding bounding box. Due to the extra task of identifying the target's position, object detection is a harder task than image classification. Lastly, object detection will be the focus of this work.
- Segmentation - Segmentation takes the idea of object detection to an extreme. That is, instead of only putting a bounding box on the object of interest, segmentation highlights every single pixel of the object. Since segmentation implies the necessity to classify every pixel in the image, it is considered the most challenging task in computer vision.

Figure 2.1: Representation of the various computer vision tasks. Image from [1].

2.1.1 Algorithms

All the DNN models used to solve a computer vision problem, object detection included, have as base a Convolution Neural Network (CNN). The CNN can be divided into two types of layers: convolution layers and pooling layers.

Convolution layers use an idea from traditional computer vision techniques, the kernel. That is, CNN applies filters to the input in order to extract important properties from the image. The extracted features from the kernels could be vertical/horizontal edges, circles, background/foreground information, etc. For such, a smaller matrix called kernel (or filter) slides over the image and does a dot product with the overlapped pixel values [21, 22], as shown in Figure 2.2.

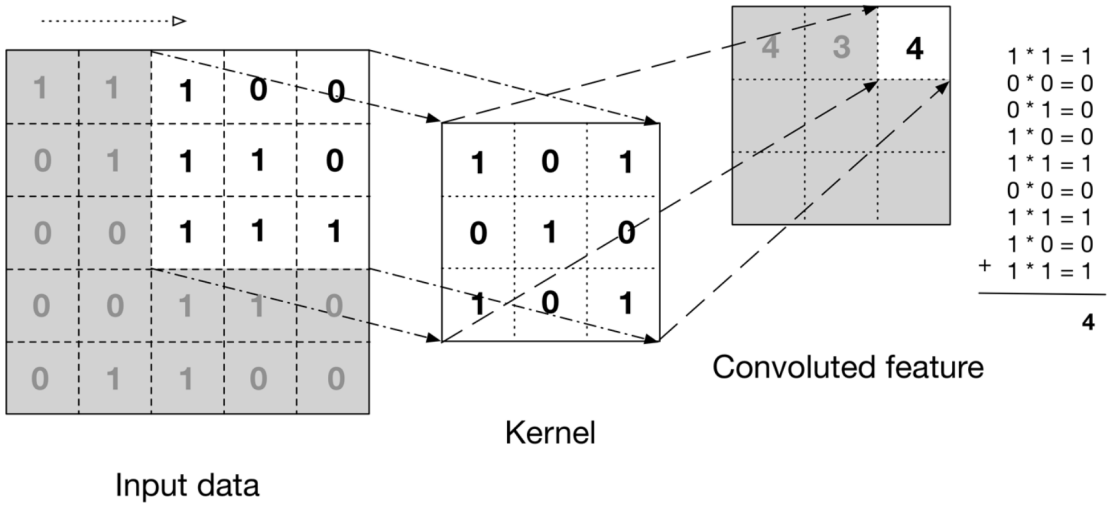

Figure 2.2: Representation of a convolution layer. Image from [2].

Hence, the values of the filter play an important role in the performance of the DNN, as a change in the kernel unit values represent a change in the importance of the pixel. So, these values are adjusted in the process of training the DNN, as any other parameter.

The other type of layer present in the convolutional part of the network is pooling layers. These are placed after the convolution layers and downsample the image from the previous layer, compacting the information of the image into a smaller one, hence reducing the dimensionality of the problem. The downsampling is done by sliding a window through the feature map (convoluted image) and replacing the pixels in its receptive field with a single value given by either the average (average pooling) or maximum value (max pooling) of the latter [2]. An example of a pooling layers is represented in Figure 2.3.

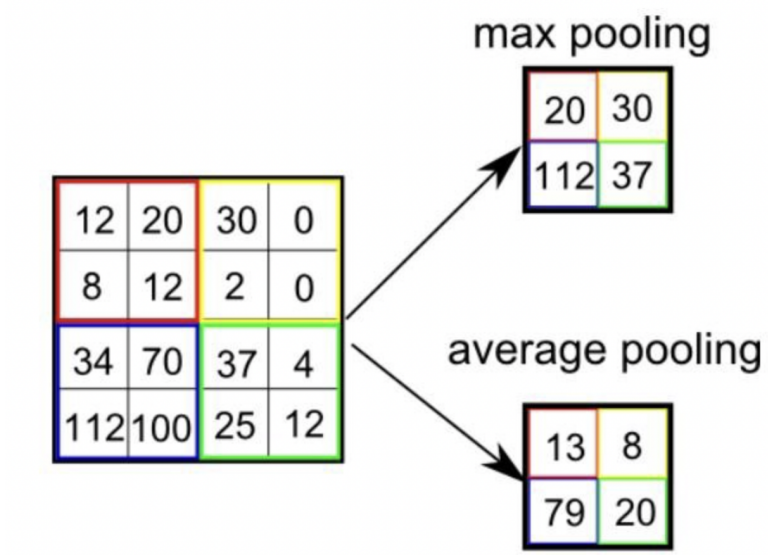

Figure 2.3: Representation of a pooling layer. Image from [2].

To note that maximum pooling is used more often than average pooling, as the this are computationally less expensive without any loss in model capacity.

So, for the complete model, it is common to stack a multitude of CNNs and then finish the model architecture with a Fully Connected Layer, to perform the classification, as shown in Figure 2.4.

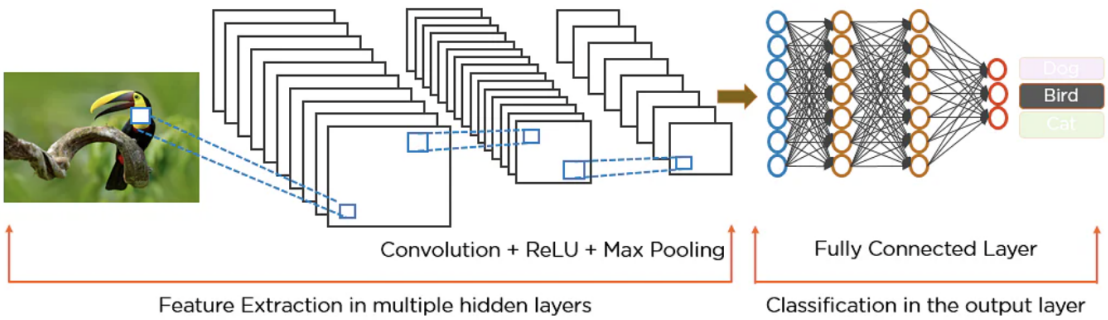

Figure 2.4: Representation of a common architecture for a computer vision DNN model. Image from [3].

In the case of object detection, the used model can be divided into two categories: two-stage detectors and one-stage detectors [4].

- Two-stage detectors: As the name indicates, the two-stage detectors divide the object detection task into two stages: extraction of the Region of interest (ROI), then regressing

and classification of the ROIs.

For the region proposal stage, a multitude of strategies can be used. However, the two main ones are using tradicional Computer Vision methods to select the ROIs, and the second is to train a specific neural network to do so [23]. After selecting the ROIs, a convolution neural network is used to make a classification of every ROI and select the best one.

The two-stage detector was the first methodology successfully used to solve object detection problems. However, these methods tend to be slow, as it is necessary to predict every ROI, so multiple forward passes are required. Despite this, these methods ate still used as they obtain great accuracy [20].

There is a large variety of models that apply the two-stage detection methodology. However, the most representative is the R-CNN model family. Within this family, the most commonly used model is the Faster R-CNN [6].

- Singel-stage detectors: This methodology was conceived with the intention of removing the first extraction of regions of interest from the two-stage detectors. This way reduces the computation cost and fastens the classification process. For this, single-stage detectors use a single network to detect objects. With this, single-stage detectors directly classify and regress the candidate anchor boxes.

  The main advantage of single-stage is that those algorithms are generally faster than multi-stage detectors and structurally simpler without losing predictive power [20].

  There are not as many single-stage detectors as there are two-stage detectors. For single-stage detectors, there basically only exists one family of models, the YOLO (You Only Look Once) [24]. Within this family, there are currently seven models, with YOLOv7 being the last and best one [25, 26].

2.2 Motivation/goal for interpretability

First of all, it is important to notice that model interpretability is not fundamental in every application. In some cases, users might only care about getting the highest predictive performance possible, regardless of the model's reasoning [27]. An example of such a case could be the problem of predicting the cost of prices in the stock market. In this case, the user only cares if the model has the capability of maximizing the investor returns and does not take into account the reasons the model took in such decisions. In these cases, interpretability is not useful.

Model interpretability becomes useful when the application has a bigger impact on human life, areas such as medicine, judicial systems, autonomous driving, and many more.

For these applications, and independently of the problem type (classification, regression, etc.), the ability to interpret the model is crucial for many reasons. However, according to [27, 28], the five main reasons are the following: assessing correctness, assessing fairness, knowledge discovery, justification, and user acceptance.

- The first reason to study the arguments behind a model is to assess its correctness. That is, to assess if the reasons why a model made a certain decision match the decision in itself.

  Let us take as an example a model that aims to identify whether a dog is present in an image. On a certain correct classification made by the model, it is possible to observe that the pixels considered as most important by the model do not contain the dog. Hence, this indicates that, although the model is making correct decisions in this database, it may not be able to generalize well to other examples. Thus, an adjustment is necessary.

- The second reason is assessing the fairness of the model. Any model developed is only able to learn from the data that it was provided. Hence if there are any discriminations in the data (due to the data quality, over/under-representation of a subpopulation, for example), this could lead the model to capture this unwanted behavior.

  A famous example is one of the COMPAS score. The COMPAS score uses an algorithm to assess the potential recidivism risk of criminal felons in the United States of America. However, in the study [29], it was proven that this model was racially motivated, and it was harder for African Americans to have a lower recidivism risk when compared to white Americans.

  In conclusion, the fairness of the model is important to ensure no sub-population present in the data is discriminated against.

- The third reason is knowledge discovery. Domain experts can use machine learning to discover correlations in the data. Even though correlation is not causation, machine learning models can still reveal relationships that can be used to formulate new statistical hypothesis about the real world. For obvious reasons, interpretability is the key for knowledge discovery.

- The fourth reason is a justification for the model decision. Given recent regulations in data protection such as the General Data Protection Regulation of the European Union,

interpretability can be necessary to comply with the right to an explanation of algorithmic decisions [30].

An example of such is when asking for credit at a bank. As it is mandatory for the bank to provide a reason for declining to its clients.

- The fifth reason is user acceptance. In areas where every decision has a serious consequence, a metric saying how good a model performs is not enough for the model deployment. This is because every domain expert will be reluctant of the model every time it goes against the user's opinion. A way of giving confidence to the users of the model is by showing how the model has come to the decision presented. This way, it is possible for the user to rethink his opinion and gain confidence in the model.

2.3 Techniques for Interpretability

The explainability approach refers to the mechanism in which the explanation is generated from the AI-based system. Multiple prior surveys in the literature have tried to generate different taxonomy of explainability approaches, and two groups of approaches have been created: ante-hoc and post-hoc [31].

The ante-hoc methodology is the interpretability technique that is pre-built in the model considered. That is, this methodology is defined by the usage of white box models, such as linear regression models, decision trees, k-nearest neighbor models, rule-based learners, generalized additive models, and Bayesian learners. Despite the easy interpretability of these models, they are not always used because it is commonly assumed that they lack predictive power when compared to black box model. This is especially true when considering non-structured problems, like computer vision problems. For these problems, a black box model is usually considered, such as DNNs. For that, a post-hoc methodology is needed.

The post-hoc methodology tries to generate explanations of already trained models. For post-hoc explainability methods, there is often a further distinction between model-agnostic methods and model-specific ones. This distinction is about whether the method works for all types of models (model-agnostic) or only specific ones, such as DNNs, support-vector machines (SVMs), or random forests (model-specific).

Figure 2.5 shows a diagram with the different sub-categories for explainability algorithms that can be considered for object detection models.

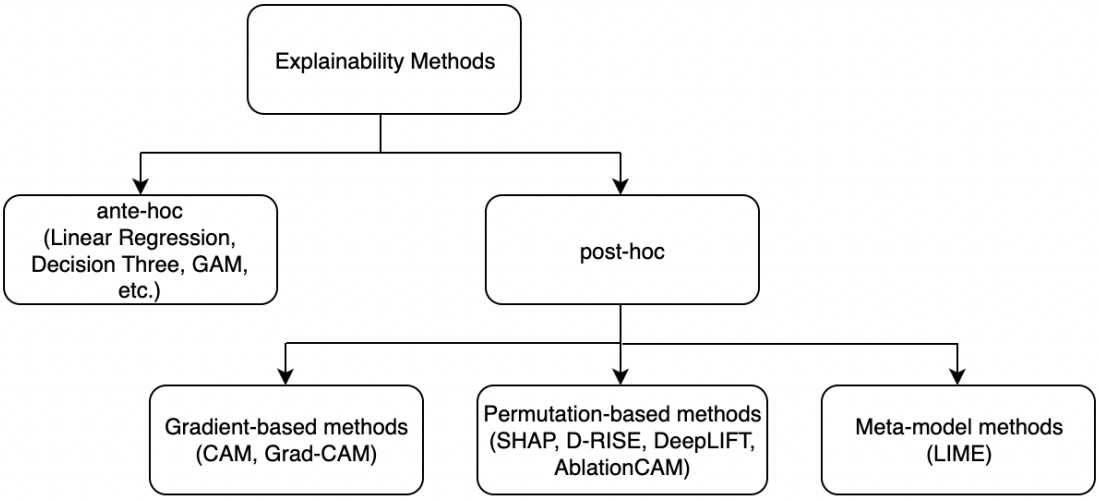

Figure 2.5: Representation of overarching taxonomies of explainable artificial intelligence methods.

Since the goal of the present work is to study explainability methods for object detection methods, that are usually solved with DNN, only post-hoc methods for these models will be presented and studied.

2.3.1 Gradient-based methods

As the name indicates, Gradient-based methods rely on the calculation of the gradient of the model's prediction to compute the importance of each pixel. The first popular algorithm of this type was Class Activation Mapping (CAM) [32]. The CAM algorithm backpropagates the gradient of the model output from the last convolution layer of the DNN back to the input layer. Then a weighted sum of each feature (pixel) activation value is done in order to identify the importance of each one.

With the popularity of the CAM algorithm, many variants were developed, being Grad-CAM the most popular one [33]. In Grad-CAM, the general process is the same as the one used for the CAM method, this is, the algorithm also backpropagates the model output gradient back to the input layer. However, the difference is that in Grad-CAM, the importance of every pixel is calculated by weighing the feature activation values at every location with the average gradient of the class score instead of just the average gradient values.

Gradient-based methods tend to give good results and be computationally efficient as they only require a single forward and backward pass through the model. However, they are not widely used in object detection problems. The reason behind it is the fact that it is common

to use some post-processing of the model output with techniques like Greedy Non-Maximum Suppression, Soft-NMS, etc [34], which makes the outcome non-differentiable [35].

2.3.2 Permutation-based methods

Permutation-based methods, also called Gradient free methods, are methods that make perturbations, such as adding noise, inpainting, and blurring to individual parts of the input image, to observe the impact it has on the outcome. The methods estimates the importance of each region by seeing the impact of each permutation on the model outcome.

These types of methods are not specific to DNN, as they are also used for standard machine learning models. Two of the most popular methods are SHapley Additive exPlanation (SHAP) [36]. The SHAP, like any permutation-based method, starts by applying perturbations of the input. However, this method used the permutation to compute the Shapley Values. Given a reference (or baseline) input, a random permutation of the input features is added one by one. After each step, the network is evaluated. The output difference after adding each feature corresponds to its attribution. But the ordering in which the features are added is important for nonlinear functions. Thus, these differences are averaged when repeating this process several times, each time choosing a new random permutation of the input features [37]. In the end, the highest the Shapley Values the more important the feature/pixel is.

On the other hand, there are DNN-specific methods that apply this technique. D-RISE [4] is a standard method used to do so.

Lastly, there are also variants of the gradient-based methods, that use the same methodology but are gradient-free. These methods follow the same procedure as the gradient-based methods presented, but instead of backpropagating the gradient, these methods backpropagate a different metric. Examples of such are DeepLIFT, AblationCAM, and EigenCAM [38–40], where, for example, EigenCAM uses the eigenvalues of the first principle component of the activations, instead of the gradient to compute the importance of each pixel in the image [35].

2.3.3 Meta-model methods

The last big category of explainers is the meta-model explainers, or simply called meta-explainers. This technique tackles the challenge of producing model explainability by building surrogate models to try to approximate the original model. However, the key idea is that the models used to approximate the original model are always ante-hoc models. Hence, if the approximation is good the ante-hoc model will allow us to interpret the results.

Surrogate models can be created in many ways, for instance, by probing the original model

via local perturbations or by leveraging its structure. Accordingly, surrogate models can be the result of most ways of functioning [31].

The most commonly used algorithm that uses this technique is Local Interpretable Model-agnostic (LIME) [41]. By inception, LIME uses a simple linear model as its surrogate model to approximate the original model. However, the algorithm allows for different choices, such as Decision Tree.

Chapter 3

Theoretical Background

In this chapter, a comprehensive description of explainability methods theoretical background is presented, as well as, two metrics used to evaluate the explanations produced by each algorithm. This way the objective is to establish a solid foundation and understanding of the concepts, techniques, and algorithms that enable interpretability and transparency in visual recognition tasks. Specifically, this chapter focuses on four prominent explainability methods: GradCAM, EigenCAM, D-RISE, and LIME.

The selected methods represent a diverse range of approaches for interpreting and explaining the decisions made by computer vision models. This selection ensures that at least one technique from each sub-category of algorithms is presented in 2.3, is tested. For Gradient-based methods, Section 2.3.1, the most recent algorithm in the literature will be tested, that is, Grad-CAM. Grad-CAM utilizes gradient information to identify the crucial regions in an image that contribute to the model's predictions.For permutation-based methods, Section 2.3.2, two algorithms will be tested, D-RISE and EigenCAM. D-RISE focuses on generating counterfactual explanations by perturbing the input and observing changes in the model's output. On the other hand, EigenCAM is a gradient free version of the Grad-CAM algorithm, taking leverages the eigenvalues of the convolutional layers to highlight salient features in an input image. Lastly, the meta-model algorithm, Section 2.3.3, considered is going to be LIME. LIME adopts a local interpretability approach by approximating the model's behavior with an interpretable surrogate model.

Besides the algorithms in itself, this chapter will also present two techniques proposed in [4] to evaluate the explanations given by each algorithm. These metrics are Deletion and Insertion, where both metrics play with the saliency map produced by the algorithm to analyze the model prediction and this way to evaluate the quality of the explanation.

In summary, the insights gained from this chapter will lay the groundwork for the next chap-

ter, where practical implementations and evaluations of the explainability methods will be discussed.

3.1 Algorithms for model interpretability

3.1.1 Grad-CAM

Gradient Class Activation Mapping or Grad-CAM for short [33], is a gradient-based explainability approach that is mainly used for image data predictions [42]. Grad-CAM generates an output that involves localized class-specific areas in an image created within a single forward and backward pass [43].

For the computation of the saliency map, Grad-CAM requires the gradient of the model on the specific image, hence, the model predictions are required. Therefore, the initial step of the algorithm is the forward propagation of the image through the model in order to obtain the predicted score of each class, y^c. After such, it is possible to compute the gradient of the output, with respect to the feature map activation, A^k, of each convolutional layer in the neural network:

$$\frac{\partial y^c}{\partial A^k}. \tag{3.1}$$

This way, the computed gradient will be specific for each input image, as the feature activation map A^k is going to change for each image. It is also important to notice that the computed gradient will have the same shape as the feature map selected. Hence if we set h as the height and w the width of the feature map, then the computed gradient will have the shape of (k, h, w).

Next, the gradients are globally average-pooled over the width and the height (indexed by i and j, respectively) and this will lead to the computation of the neuron importance for a feature map k of class c, α_k^c:

$$\alpha_k^c = \frac{1}{h \times w} \sum_i \sum_j \frac{\partial y^c}{\partial A_{i,j}^k} \tag{3.2}$$

After pooling over the width and height, the shape of the alpha values is (k, 1, 1), which is the same as a single vector of dimension k.

Finally, the α values are taken as weights for each corresponding feature map and a weighted sum of these feature maps will give the final value of Grad-CAM. The ReLU operation is applied over the sum to keep only those values that are positive.

$$L^c_{\text{Grad-CAM}} = ReLU\left(\sum_k \alpha^c_k A^k\right) \tag{3.3}$$

The application of the ReLU operator to the linear combination of maps is because we are only interested in the features that have a positive influence on the class of interest, i.e. pixels whose intensity should be increased in order to increase y^c. Negative pixels are likely to belong to other categories in the image, so they shouldn't affect the computation [33].

The output of the Grad-CAM has shape of (h, w), which is also the shape of the selected convolution layer feature activation map. Since the goal of convolution layers is to downsize the image dimension, then the dimension of the Grad-CAM saliency map is also smaller than the input image. Subsequently, it is necessary to upsample the saliency map to the original size of the image.

Lastly, the selection of the convolution layer, and consequently the feature activation map, is an important step for the algorithm explanation. While any convolution layer can be used, it is common to select the feature activation map of the last convolution layer in the neural network. This selection is made because the last convolutional layer will contain only high-level information, that is, the most critical regions/features of the image [44].

3.1.2 Eigen-CAM

EigenCAM [40] is an algorithm that generates a visual explanation of the principal components derived from the learned representations of the convolutional layers of the network. This method had its origin based on three main observations about the traditional gradient-based algorithms:

1. Methods that only consider the backpropagation of the gradient assume that the neural network is 100% accurate. Hence, such a method would only work when the neural network is correct.

2. Gradients are noisy by nature. Additionally, methods such as Grad-CAM only consider positive gradients regardless of redundancy in the feature space.

3. Convolutional Neural Network layers map images into different classes, and the final decision is only made by the fully connected layers of the neural network.

So the authors of the method considered the following assumption for the EigenCAM algorithm: "What features go through all local linear transformations and stay relevant in the same

direction of maximum variation? In other words, what salience features will be in the direction of the principal component of the learned representation." Paraphrasing, the method assumes that the hierarchical representation of the features during the propagation process stays the same in the direction of the principal components that could be learned as an explanation purging redundant dependencies [45].

Let I represent the input image of size (i, j), $I \in R^{i,j}$, and let $W_{L=n}$ represent the combined weight matrix of the first n^{th} convolutional layers of size (h, w). Then, the class activated output is the image I projected onto the last convolution layer L=k and is given by:

$$O_{L=k} = W_{L=k}^{T} I \tag{3.4}$$

In other words, the matrix $O_{L=k}$ represents the feature activation map of the last convolution layer in the neural network. Using singular value decomposition, principal components are computed:

$$O_{L=k} = U \Sigma V^{T}, \tag{3.5}$$

where U is an $h \times h$ orthogonal matrix and the columns of U are the left singular vectors, Σ is a diagonal matrix of size (h, w) with singular values along the diagonal and V is an (w, w) orthogonal matrix and the columns of V are the right singular vectors.

Finally, the class activation map, $L_{\text{Eigen-CAM}}$ is calculated as a multiplication of this projection $O_{L=k}$ and the first eigenvector V_1 of the matrix of the right singular vectors V:

$$L_{\text{Eigen-CAM}} = O_{L=k} V_1 \tag{3.6}$$

Lastly, since the Eigen-CAM algorithm does not consider the backpropagation of the gradient or any maximum activation locations, this makes this method less prone to classification errors when compared to methods such as Grad-CAM. This way, this method becomes more robust than the previous method mentioned.

3.1.3 D-RISE

The Detector Randomized Input Sampling for Explanation (D-RISE) method [4] is a permutation-based method for object detectors. Hence, just like any permutation-based algorithm, the XAI method evaluates the importance of each pixel in the image for the object detector model by performing small modifications to the input image and observing how the model reacts. This process is then repeated multiple times to get a confident conclusion.

Thus D-RISE algorithm can be divided into three main stages: mask generation, similarity metric, and saliency inference.

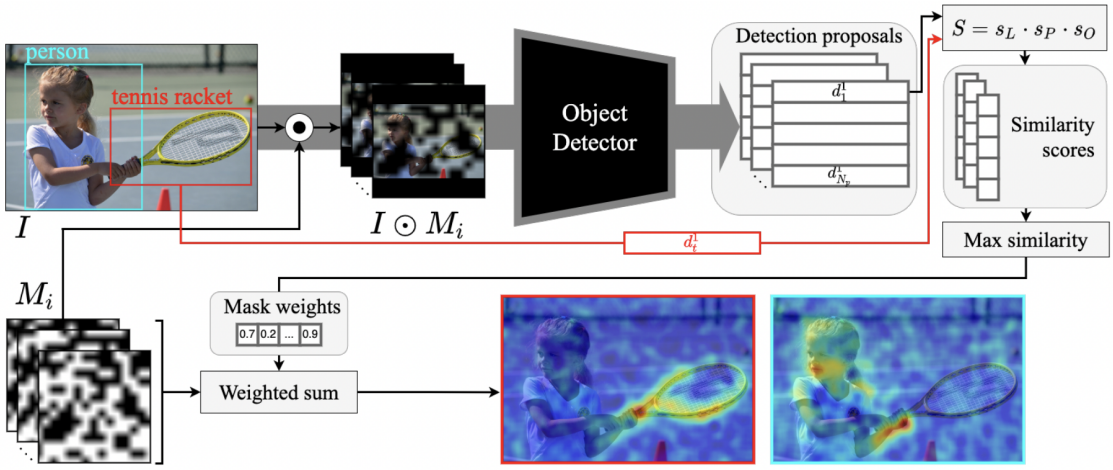

Figure 3.1: Representation of an overview of the D-RISE algorithm. Image from [4]

3.1.3.A Mask generation

The first step of the algorithm is to generate the mask, which will modify and identify the areas of the image to nullify and observe its importance for the object detection model. For mask generation, the authors use a technique previously proposed on [46], which intends to create a sparse mask of the same size as the input image. This way, the mask generation process goes as follows:

1. randomly sample a binary mask of size (h, w) (smaller than image size (H, W)) by setting each element independently to 1 with probability p and to 0 with the remaining probability;
2. upsample the mask to size $((h+1)C_H, (w+1)C_W)$ using bilinear interpolation, where $(C_H, C_W) = (\lfloor H/h \rfloor, \lfloor W/w \rfloor)$ is the size of the cell in the upsampled mask;
3. crop the mask with uniformly random offsets ranging from $(0, 0)$ up to (C_H, C_W). In order to get a mask of the same shape as the input image.

For this process, there are two key parameters: the mask size and the probability of setting the mask pixel value to 1. The following image shows the effect of both parameters in the final mask.

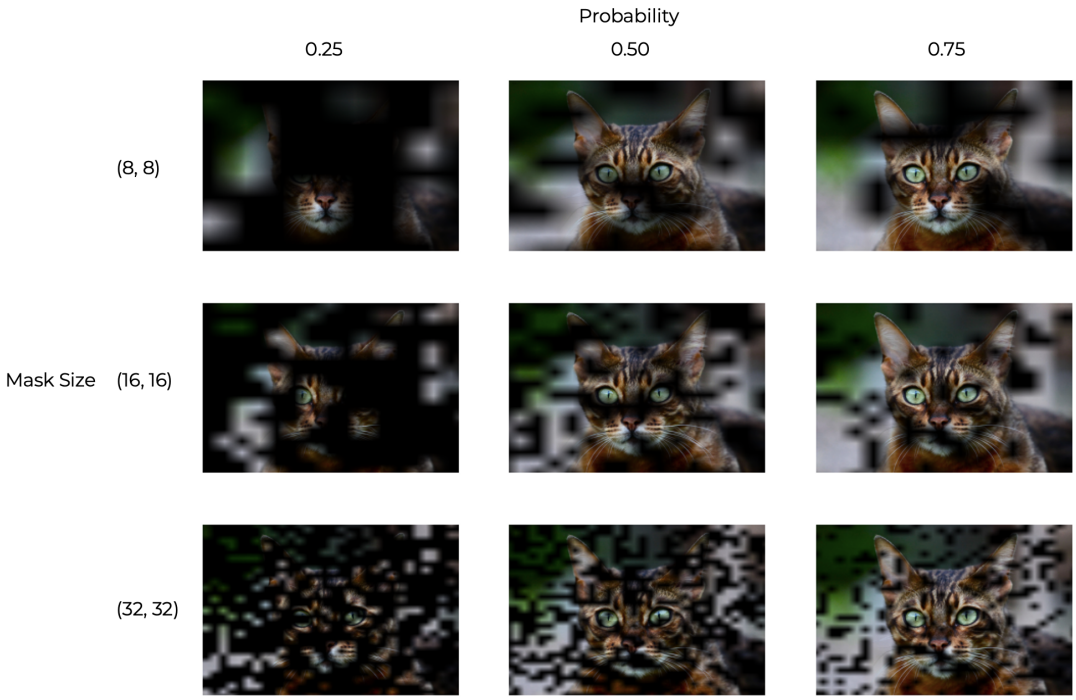

Figure 3.2: Representation effect of the initial mask size and the probability p in the final mask.

As it is possible to observe from Figure 3.2, as the probability p increases the number of disturbed pixels decreases. Hence, smaller probabilities allow us to guarantee the importance of each region. However, since a smaller p also tests fewer pixels in each iteration, it means that the algorithm will need more iterations to converge to a final solution.

On the other hand, the initial mask size controls the size of each covered region. Small initial mask sizes will lead to big patches of perturbed pixels, inversely, bigger initial mask sizes will lead to small patches of perturbed pixels. In-between initial mask size is the optimal value, as big patches of perturbed pixels will make it hard for the algorithm to converge into a solution. At the same time, small patches of perturbed pixels will make it hard to identify the full reason for the model decision.

For this reason, the authors recommend an initial mask size of 16×16 and a probability p of 0.5. The figure below shows a par of masks created with these parameters.

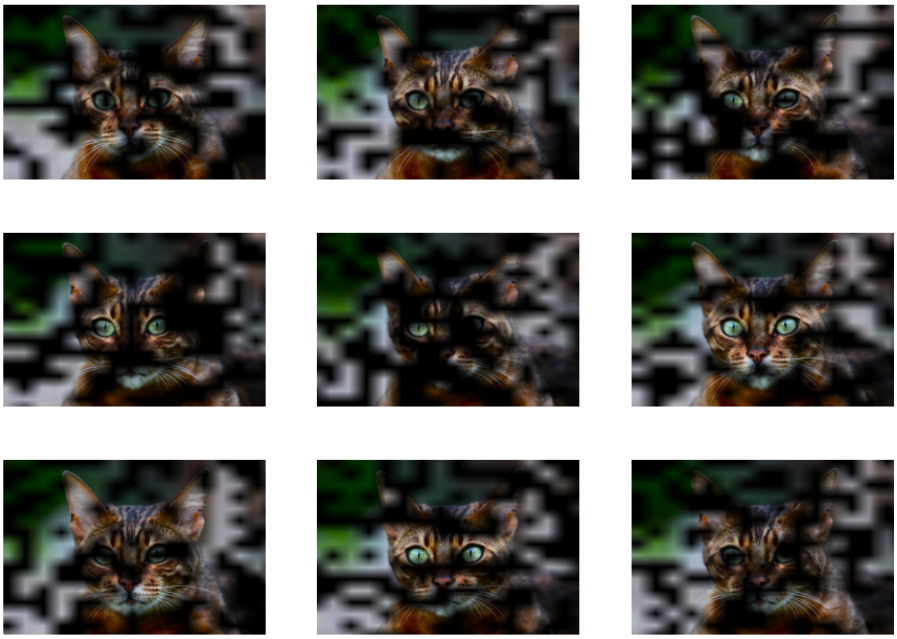

Figure 3.3: Representation of multiple mask with an initial mask size of 16×16 and a probability p of 0.5.

3.1.3.B Similarity metric

After applying a mask to the input image, it is necessary to observe the effect of such in the object detection model output. Hence it is necessary to compare the output of the model on the masked image with the actual target. For such, the authors propose a metric with three components.

The first component of the metric is the Intersection over Union (IoU) to measure the spatial proximity between the predicted and expected bounding boxes. As the name indicates, the Intersection over Union metric is the area of the intersection of the two bounding boxes divided by the union of the same objects:

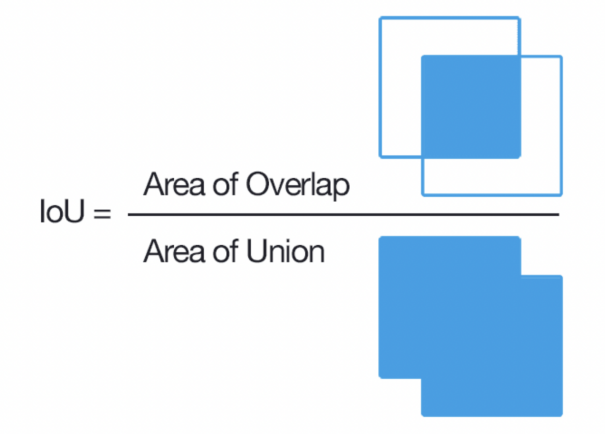

Figure 3.4: Representation of Intersection over Union. Image from [5]

Hence, IoU values close to 1 mean that two objects are very similar, in other words, the model made a good prediction. Therefore, it means that the regions affected by the mask do not seem to influence the model decision.

The second component of the metric is the cosine similarity of the predicted probabilities and the targets, to evaluate how similar two regions are. If the network only outputs the probability of the class it believes to be present in the image, then this component simply becomes the single probability outputted by the model.

Lastly, the third component of the metric is the objectness score. However, this score is only produced by some network architectures like the YOLO [25] family. For networks that do not output such values, like the Faster R-CNN [6], the objectness term can be simply omitted.

Thus, the similarity score between two detection vectors is the product of the three components:

$$s(d_t, d_j) = s_L(d_t, d_j) \cdot s_P(d_t, d_j) \cdot s_O(d_t, d_j), \tag{3.7}$$

in which

$$s_L(d_t, d_j) = IoU(L_t, L_j), \tag{3.8}$$

$$s_P(d_t, d_j) = \frac{P_t \cdot P_j}{||P_t|| ||P_j||}, \tag{3.9}$$

$$s_O(d_t, d_j) = O_j, \tag{3.10}$$

where L represents the bounding box, P a vector of probabilities for each class, and $O \in [0, 1]$ is the objectness score for the target and the predicted output by the network.

3.1.3.C Saliency inference

Finally, it is possible to combine all the components of the algorithm. Thus, D-RISE generates the object detection model saliency map in the following way:

1. Generate N masks, $M = \{M_i, 1 \leqslant i \leqslant N\}$.
2. Apply each mask only once to the input image, I, obtaining the masked images, $D = \{D_i = I \odot M_i, 1 \leqslant i \leqslant N\}$. Also, keep the target of the model, E_t.
3. Run the object detection model, f, on the masked images, producing N_p proposals for each image, $E_p = \{E_p^i, 1 \leqslant i \leqslant N\} = \{f(D_i), 1 \leqslant i \leqslant N\}$.
4. Compute pairwise similarities between two sets of detection vectors E_t and E_p and take maximum score per each masked image per each target vector, $w_i^t = max_{1 \leqslant i \leqslant N_p} s(E_t, E_i^t)$.
5. Compute a weighted sum of masks M_i with respect to computed weights w_i^t to get saliency maps $H_t = \sum_{i=1}^{N} w_i^t M_i$

3.1.4 LIME

The local interpretable model-agnostic explanations (LIME) [41] is a method to produce explanations for predictions of classifier and regression models based on the approximation of ante-hoc models.

Contrary to the previous methods described, LIME is not a method specific to object detection problems, nor to neural networks [47]. In fact, LIME is able to produce explanations for models solving a large variety of problems, classification; regression; image; text; etc. For this, LIME is an extremely popular algorithm in practice.

The method works by locally approximating the model under study with a simpler surrogate model (e.g. a linear model) to be more easily interpretable. More precisely, for image problems, LIME can be divided into three main components: Superpixels, Sampling, and Surrogate Model [48].

3.1.4.A Superpixels

One of the main difficulties with image-based problems is the high-dimensionality, as each pixel in the image will represent one axis of dimensionality. For neural networks, this problem is avoided with CNN, however, for a traditional model, like the ones used as surrogate models

by LIME, it becomes impossible to get good results, and consequently good approximations. Thus, the first step of LIME is to split the input image into superpixels.

Superpixels are a combination of pixels such that LIME considers all the pixels as a single, in order to reduce the dimensionality of the problem, and consequently reduce unnecessary complexity. There are multiple ways of combining pixels to create superpixels, for example, it is possible to simply combine continuous patches of the image in squares, as shown in Figure 3.5. It is also possible to consider more complex techniques, such as combining continuous patches of the image based on color and/or brightness similarities [48].

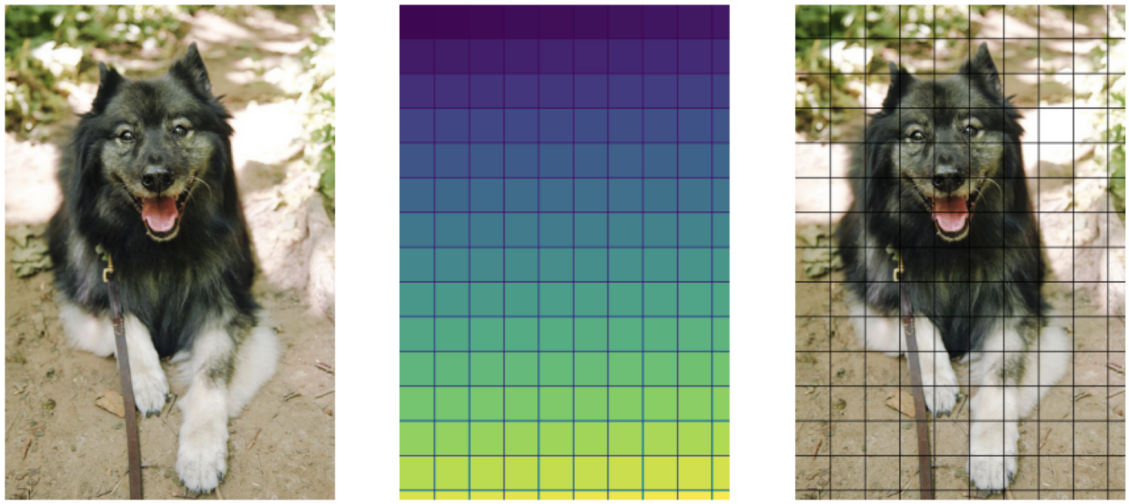

Figure 3.5: Representation of superpixels. Original image is on the left, the superpixels considered in the middle, and the application of the superpixels to the image is on the right.

3.1.4.B Sampling

Just like permutation-based algorithms, in order to produce explanations, LIME creates new examples that are slight modifications from the base input. These new examples are images where the superpixels are randomly modified. There are multiple ways of modifying the superpixels, however, the two more common techniques are: zero padding and mean padding.

Zero padding consist in the random replacement of a superpixel by zeros, creating a black mask in the image. On the other hand, mean padding is when each channel of the superpixel is randomly replaced by its mean, creating a blur mask in the image. Figure 3.6 shows an example of both sampling techniques.

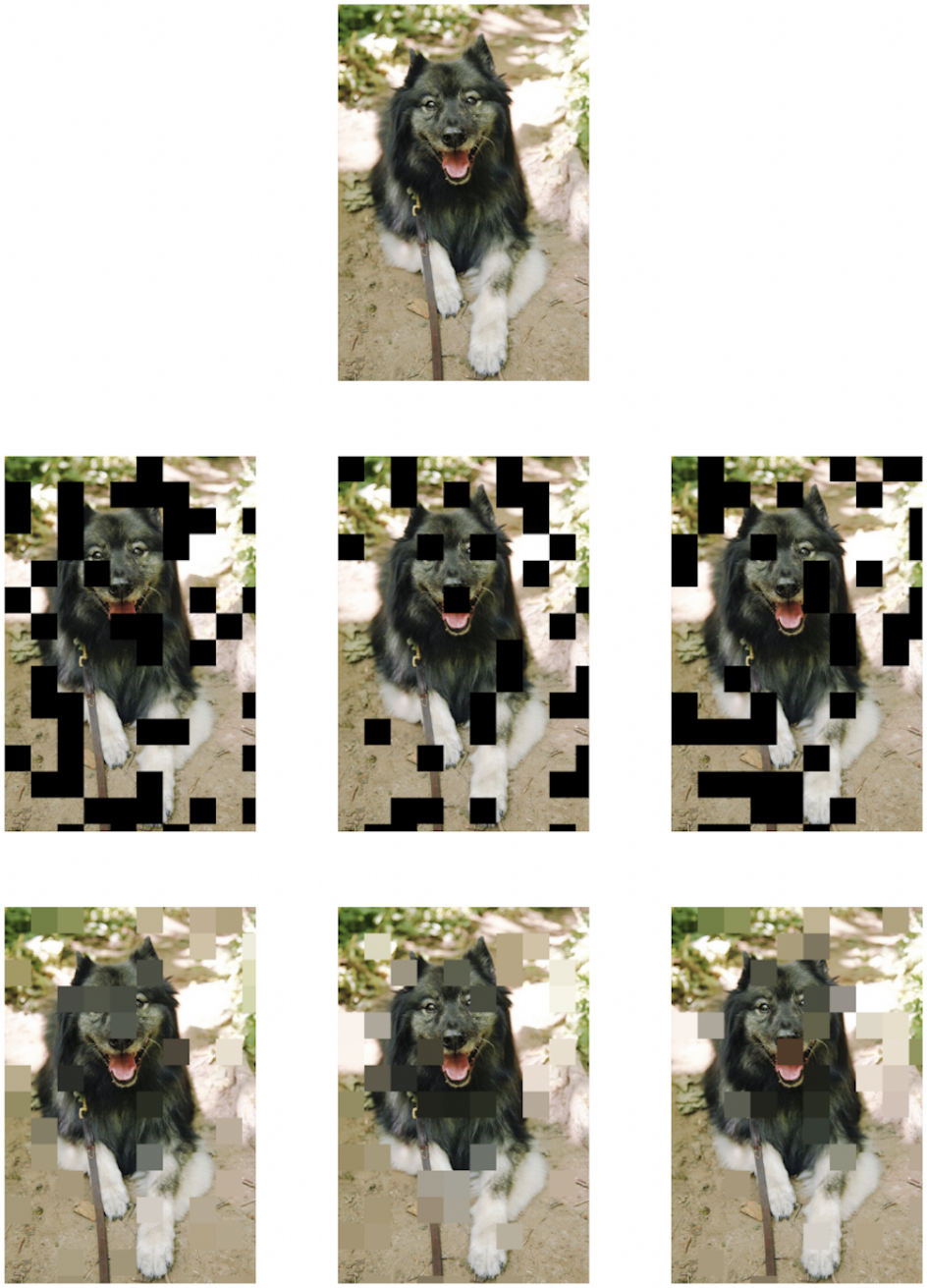

Figure 3.6: Representation of the sampling process. Original image is on the first row, zero padding sampling in the middle row, and mean padding in the third row.

3.1.4.C Surrogate Model

Finally, it is possible to combine all the components of the algorithm. Thus, LIME generates the object detection model saliency map in the following way:

1. Generate superpixels of the input image.

2. Generate N masked images with the zero or mean padding techniques.

3. For each masked image fit an ante-hoc model. Commonly, the ante-hoc model used is a simple Linear Regression for regression problems and Logistic Regression for classification problems. So:

$$\hat{\beta} \in \underset{\beta \in \mathbb{R}^{d+1}}{\arg\min} \left\{ \sum_{i=1}^{n} (y_i - \beta^T x_i)^2 + \lambda ||\beta|| \right\} \tag{3.11}$$

4. Compute the average of each superpixel coefficient for all the trained models, $w_p^n = \frac{1}{N} \sum_{n=1}^{N} \beta_p^n$.

5. The superpixels with the highest positive weights are the most important regions of the image.

3.2 Evaluation of saliency maps

Although there has been considerable research in the development of XAI methods for explaining the decisions of prediction models, the evaluation of these methods remains a key challenge [49].

A common approach is to assume that our model is not biased, that is, our model is only using the object itself for the prediction and not its context. Hence, one possible metric to evaluate the explainability algorithms is the correlation between the explanation and the human-labeled ground truth mask of an object [4]. However, such a method has a multitude of problems.

1. The first problem is the cost of production of the ground truth explanations, as for each object, a group of experts must identify the reason for the observed outcome. Such a process can be expensive and extremely time-consuming [49].

2. The second major problem is the assumption made, that is, the assumption that our model is not biased. The goal of any explanation algorithm is to understand the reasons behind any model, whether it is biased or not. Hence the assumption made is not pertinent for the question at hand.

To solve this problem, the authors of [4, 46] came up with two techniques to evaluate any explanation based on saliency maps, that is, any explanation for computer vision problems: Deletion and Insertion.

3.2.1 Deletion

The underlying concept behind the Deletion technique revolves around the hypothesis that eliminating pixels considered crucial would significantly affect the predictions made by the base model [46]. To be more specific, this approach quantifies the reduction in accuracy for the predicted class as pixels are progressively removed. The determination of which pixels to remove is guided by their importance in the saliency map, with higher-priority pixels being removed first. Figure 3.7 provides an illustrative example of this process.

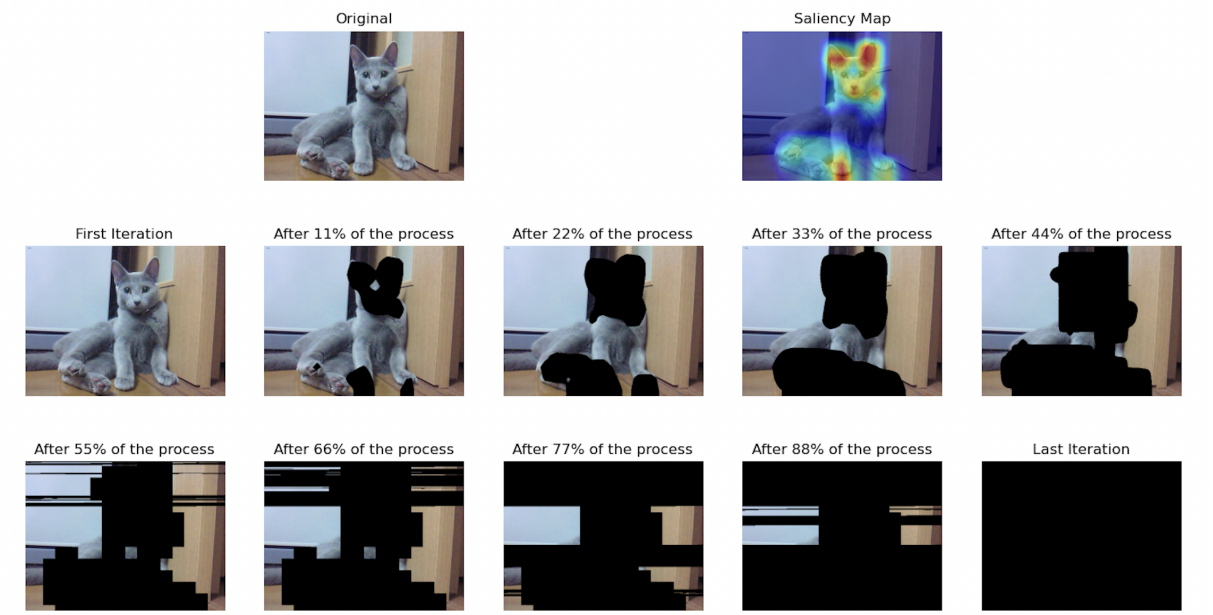

Figure 3.7: Representation selection of pixels to remove for the deletion metric.

It is important to notice that it is not mandatory to remove only one pixel in each iteration, as such a process would be extremely computationally expensive. Hence a group of pixels can be removed in each iteration to simplify the process.

So, in each iteration, the base model predicts the current image, and the effect of the removal of pixels is evaluated. In order to make this evaluation, the accuracy of the prediction is computed. For such, equation 3.7 is considered, which allows to evaluate the model's confidence in the prediction, as well as the quality of the bounding box. Hence it allows us to evaluate both components of the object detection process.

After this process, it is possible to observe the impact of each set of pixies on the base model. A quick and sharp drop in the accuracy of the model indicates that the explanations obtained are a good representation of the underlying reason for the model.

The interpretation of the metric can be summarised by the area under the curve produced by the base model score and the percentage of the image already removed. The smaller the area under, the curve the better the explanation is according to the deletion metric.

Figure 3.8 shows an example of a good and bad explanation according to the deletion metric.

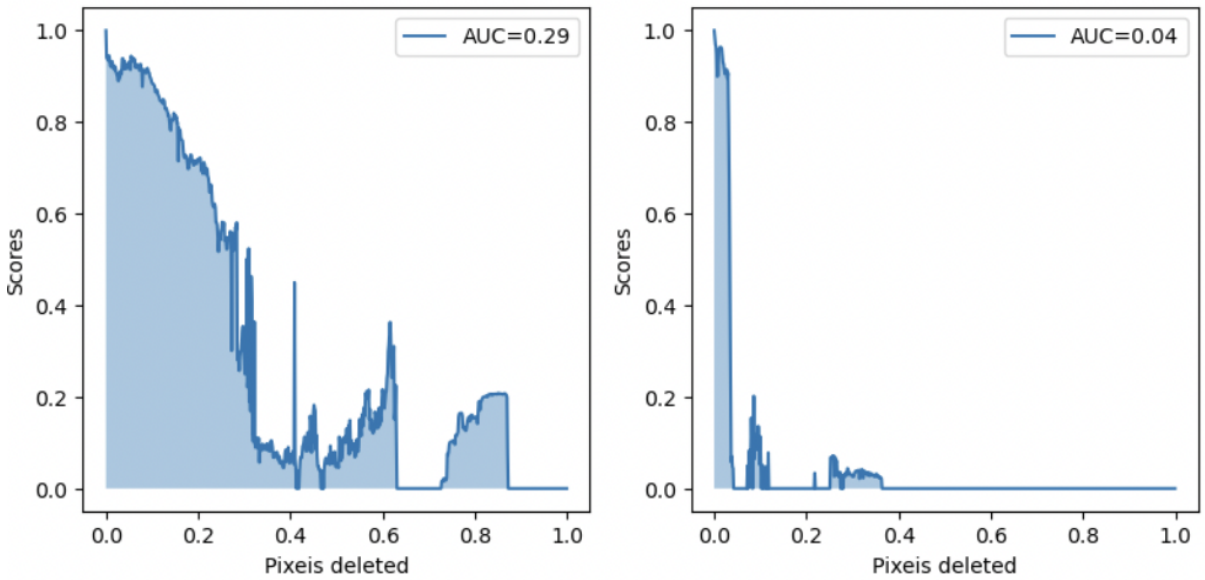

Figure 3.8: Representation of a bad explanation (left), and good explanation (right), according to the deletion metric.

3.2.2 Insertion

The insertion metric closely resembles the deletion metric in terms of its underlying process. However, the key distinction lies in their respective approaches. While the deletion metric begins with the complete image and progressively removes pixels in each iteration, the insertion metric follows the opposite procedure. That is, the insertion metric initiates with an empty image and, in each iteration, adds pixels back to the image. The selection of which pixels to add is guided by their significance in the saliency map, prioritizing the inclusion of higher-priority pixels first. This process is demonstrated in Figure 3.9.

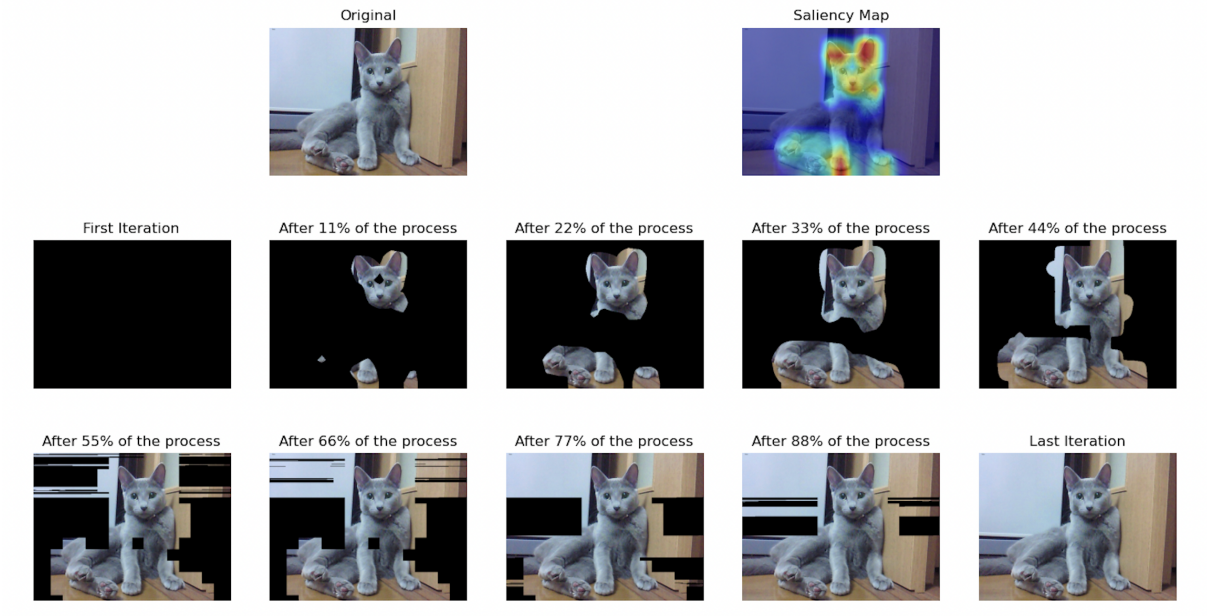

Figure 3.9: Representation selection of pixels to add for the insertion metric.

The following steps of the process are the same as for deletion, that is, in each iteration the base model is evaluated on the current image with the equation 3.7.

For an explanation to be considered good, there must be a rapid increase in the model's score. Contrarily, an unsatisfactory explanation exhibits the opposite behavior. That is, a slow and gradual rise in the score in each iteration.

Once again, the interpretation of the metric can be summarised by the area under the curve produced by the base model score and the percentage of the image already added. The greater the area under the curve, the better the explanation is according to the insertion metric.

Figure 3.10 shows an example of a good and bad explanation according to the insertion metric.

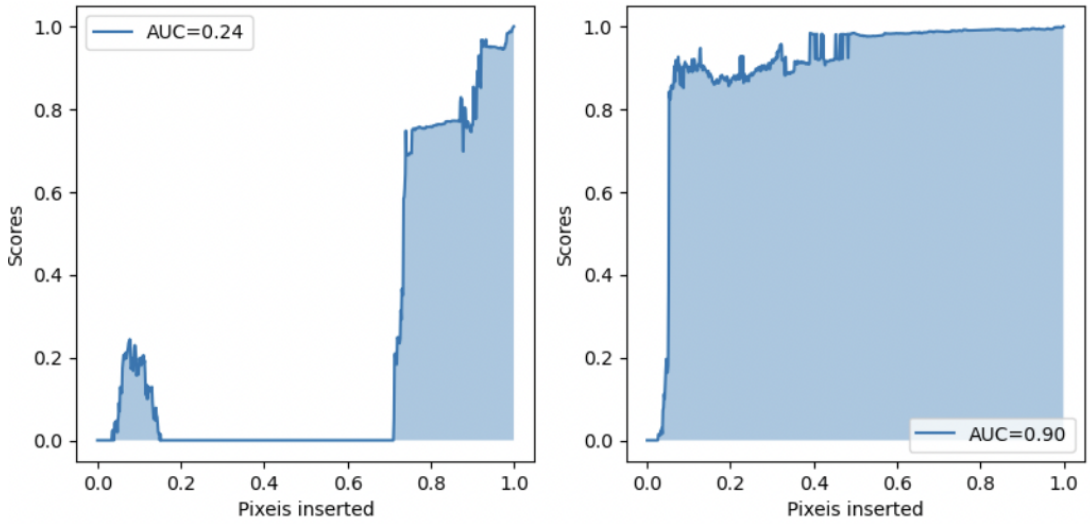

Figure 3.10: Representation of a bad explanation (left), and good explanation (right), according to the insertion metric.

Chapter 4

Experiments and Analysis

This chapter presents the detailed experimental setup, methodologies, and results obtained in the study. The primary objective of this chapter is to provide a comprehensive overview of the used dataset, the model architecture employed, and the findings obtained from the experiments conducted. Hence, the chapter is divided into three main sub-divisions: dataset, model architecture, and results.

The chapter begins by introducing the dataset used in the study. A thorough description of the datasets is provided with the objective of providing a clear understanding of the data used for all future experiments.

Following this introduction, the chapter delves into the model architecture employed for the experiments. A detailed overview of the chosen model, including its underlying principles, and architectural design is presented.

Finally, the chapter presents the results of the conducted experiments. That is, it is presented a comparison of all the explainability algorithms studied in the previous chapter, indicating the advantages and disadvantages of each when compared between themselves.

In summary, this chapter aims to provide a thorough account of the experimental setup, including the dataset, model architecture, and the comparative results obtained from the application of different explainability algorithms. By doing so, it enables a comprehensive evaluation and critical analysis of the proposed approach, contributing to the overall understanding of the study's outcomes.

4.1 Dataset

All the experiments presented in this thesis were performed on the same dataset, the ”Dog and Cat Dataset” [50]. This dataset consists of 3686 different-sized images of dogs and cats for train and test. Each image within the dataset is accompanied by corresponding labels, which include the accurate class designation and the coordinates of the bounding box. In other words, the label provides the correct object class along with the x and y coordinates of the top-left corner, and the bottom-right corner of the bounding box.

Figure 4.1 shows some examples of the images present in the dataset.

Figure 4.1: Exemplification of the type of images present in the ”Dog and Cat Dataset”.

To ensure the accuracy and reliability of the results, the dataset was divided into training and test samples. The training sample, used to fit the neural network model, contained 80% of the data. More precisely, the training sample contains 2949 images, and among these, 2001 images depicted dogs, while the remaining 948 featured cats. On the other hand, the test sample, used to test all the explainability algorithms, consisted of 738 images, with 497 showcasing dogs and 241 featuring cats.

It is worth noting that the only preprocessing technique applied to the images was pixel normalization. In other words, the pixel values of all the images were transformed to fall within the range of [0, 1].

4.2 Model architecture

The neural network architecture used as a baseline to provide the base model prediction was a Faster RCNN [6], more precisely, the architecture considered is a Faster RCNN with a MobileNet V3 as the backbone of the model.

The Faster RCNN architecture can be divided into two parts: the Region Proposal Network and the Classifier (as shown in Figure 4.2)

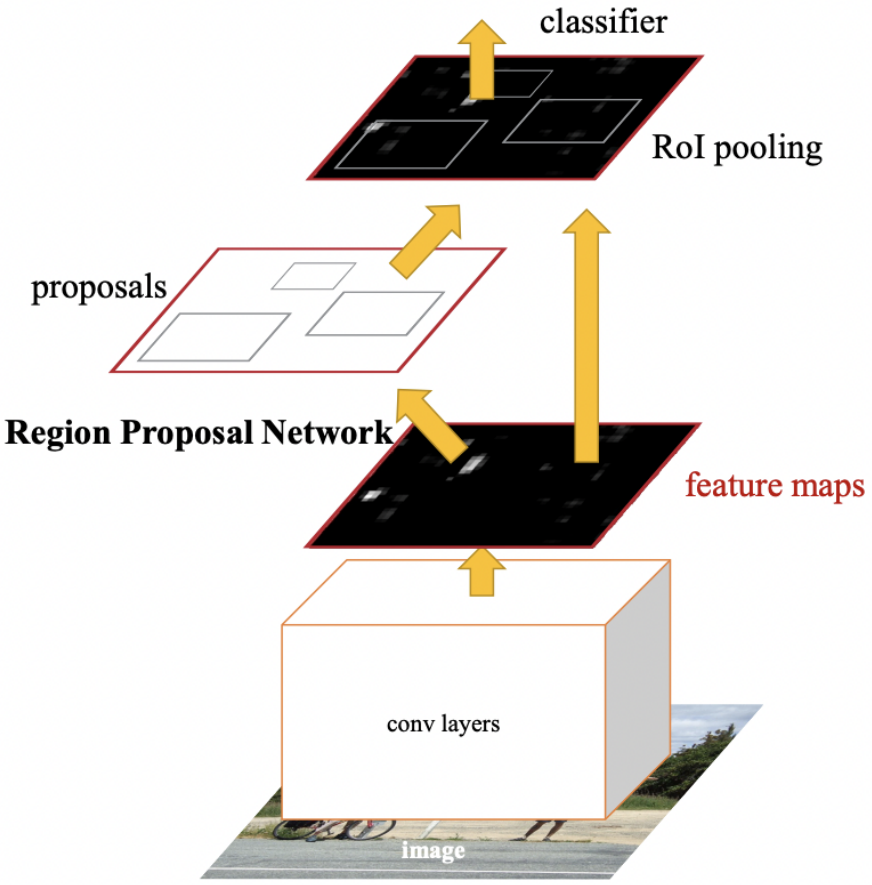

Figure 4.2: Representation of the overall structure of a Faster RCNN architecture. Source [6].

The goal of the Region Proposal Network is to identify potential regions of interest, that is, regions where it is likely to contain an object. In order to achive that, a specific and independent neural network is used, in this case, a MobileNet V3 [7].

MobileNet V3 is a standard CNN that takes advantage of convolutional layers to propose regions of interest for the classifier. However, the architecture of MobileNet V3 came from an automatic optimization process intending to determine the optimal architecture configuration, which makes it more suitable for these type of problems.

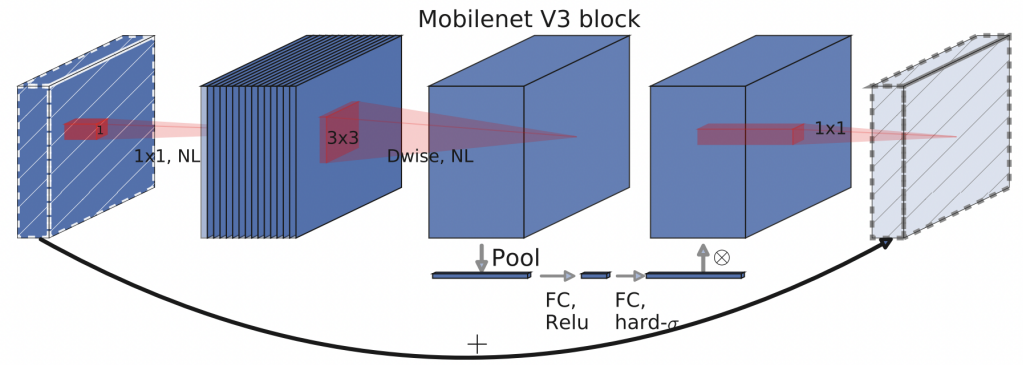

Figure 4.3: Representation of the block structure of a MobileNet V3 block. Source [7].

After the regions of interest proposal, a simple fully connected neural network is used to classify the region, that is, to indentify which object is present, and to predict the object's bounding boxes.

A scheme of the entire process of the architecture is represented in Figure 4.4

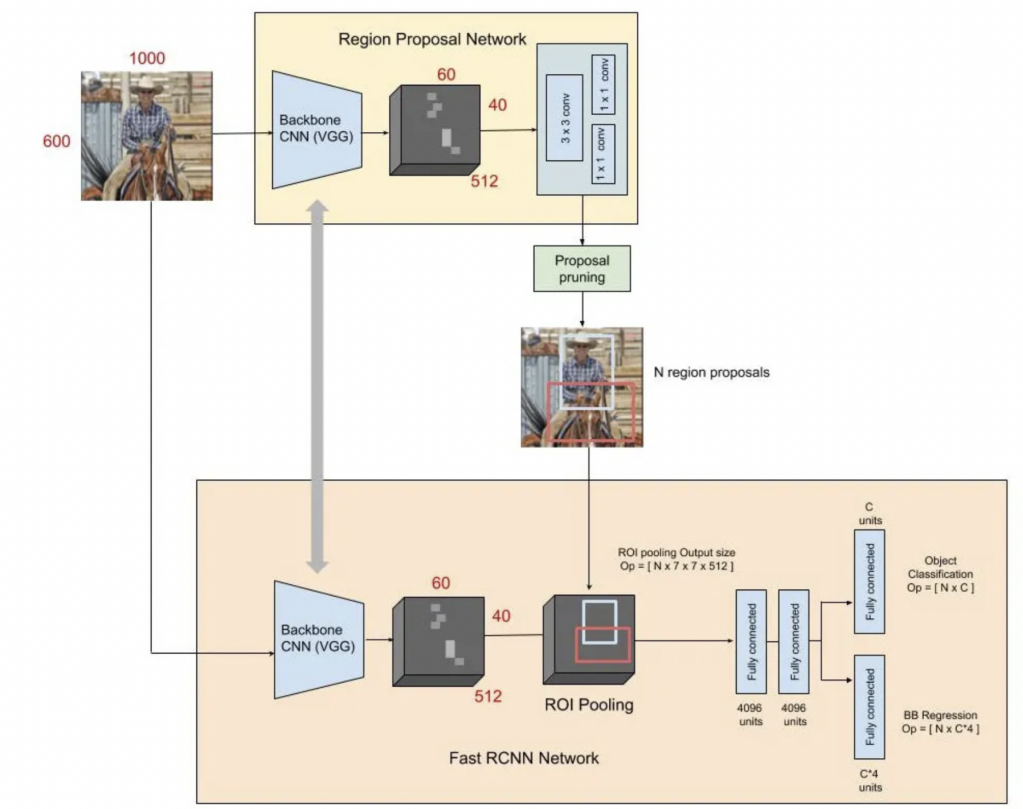

Figure 4.4: Representation of the entire process behind a Faster RCNN's prediction. Source [8].

The chosen implementation of the architecture is the PyTorch version [51]. Furthermore, a pre-trained version of the architecture is utilized to accelerate the model conversion during the training process. In essence, this entails using a model with pre-adjusted parameters based on the COCO dataset, followed by readjusting the network parameters to our specific dataset to optimize its performance.

4.3 Results

In this subsection, we present our study's results, which evaluate the performance of the four different methods previously presented for model interpretation.

All methods were implemented based on Pytorch, [51], which is a Python package for deep learning. On top of that, pytorch-gradcam [52] and Captum [53] packages were used for the implementation of the EigenCAM and LIME algorithms, respectively. The remaining tested algorithms, Grad-CAM and D-RISE, as well as the two metrics Deletion and Insertion, were implemented by me. All the code considered in this study can be found in the corresponding article Git repository[1].

All the tests conducted had the goal of providing insights into the effectiveness and comparative performance of these methods, hence achieving the research objectives.

4.3.1 Algorithms comparison

In order to evaluate the quality of the explanations generated by each algorithm, a comprehensive comparison was conducted among all four methods. During this evaluation, the base model generated predictions for all the test samples, which were then used by each interpretability algorithm. By following this approach, consistency was maintained between the predictions and the base model, enabling a direct and meaningful comparison across all the algorithms.

Consequently, after obtaining predictions from the base model, explanations were computed using each algorithm in the study. Since the test sample size is 738 images, each interpretability algorithm generated 738 explanations, one for each image-prediction pair, in the form of saliency maps.

Figure 4.5 illustrates the saliency maps provided by each tested algorithm for a couple of examples.

[1] Project Git repository: https://github.com/DiogoCarvalh0/computer_vision_XAI_comparison.git

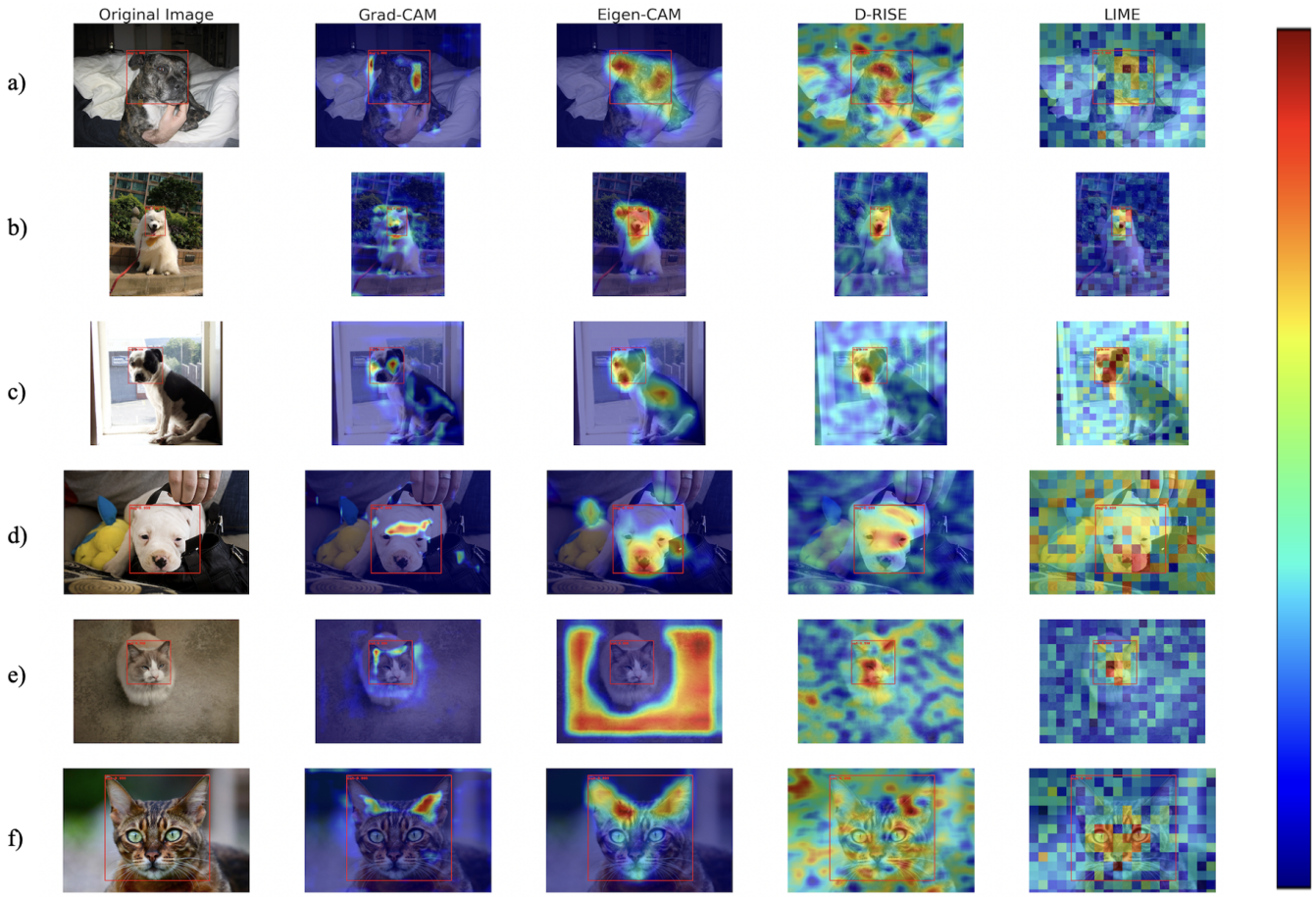

Figure 4.5: Comparison between the explanation provided by Grad-CAM, Eigen-CAM, D-RISE, and LIME. The most important pixels are in red, and the least important pixels in blue.

As it is possible to observe all the algorithms appear to provide logical explanations, as all the highlighted pixels are always on top of the animal. The only exception to this is image e) for the Eigen-Cam, where the saliency map highlights are inverted. That is the algorithm considered the cat as the least important region of the image, while the background as the most crucial. Either way, the algorithm was still able to distinguish the cat, however, in an incorrect way.

From Figure 4.5, it is also possible to observe that the four algorithms are not always concordant in which region the base model considers crucial. For example, in the image d) Grad-CAM and D-RISE indicate that the crucial region for the neural network is the puppy's eyes, on the other hand, Eigen-CAM and Lime consider the mouth and nose of the puppy as the most important region for the base model.

With all the differences it becomes hard to identify the algorithm that provides the best explanation, hence the next step is to compute the metric deletion and insertion for all the 738

test sample explanations created by the explanation algorithms.

Figure 4.6 shows an example of the entire process for a single image. That is, on the figure, it is possible to observe the initial image alongside the prediction made from the base model. Afterward, the explanations created by each algorithm are shown. Finally, the deletion and insertion metrics curves are plotted with the area under the curve for each in order to compare each saliency map.

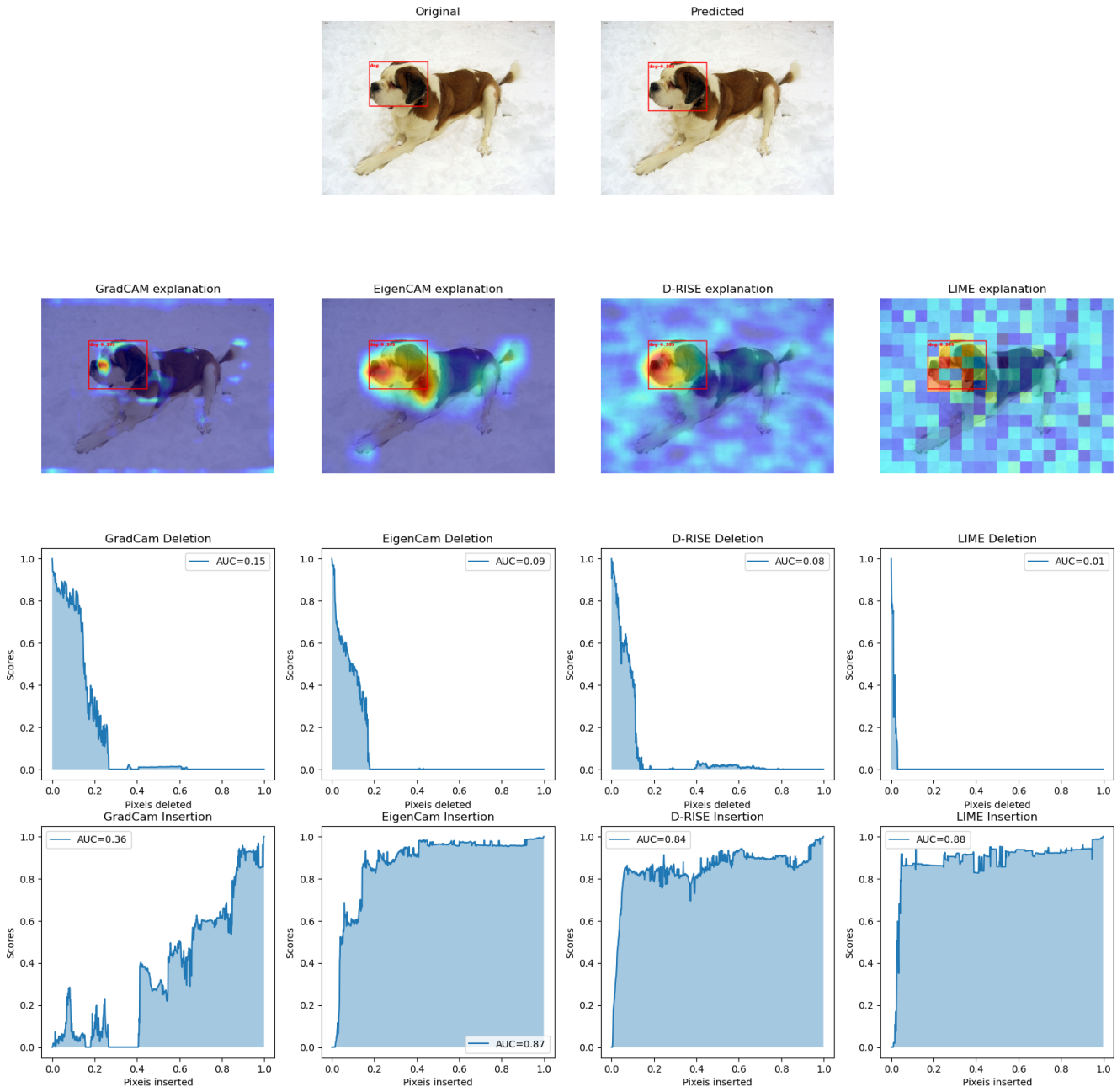

Figure 4.6: Representation of the entire process, i.e, model prediction, explanation based on Grad-CAM, Eigen-CAM, D-RISE, and LIME algorithms, and the computation of the deletion and insertion metrics.

This process is then repeated for all images in the test sample, in order to obtain the deletion and insertion metrics for each observation. After such, it is possible to compute the average area under the curve for deletion and insertion over all the images in the test sample. The results of such computation are represented in Table 4.1.

Method	Deletion (↓)	Insertion (↑)
Grad-CAM	0.350	0.370
Eigen-CAM	0.381	0.529
D-RISE	0.143	**0.687**
LIME	**0.061**	0.677

Table 4.1: Average of the deletion and insertion metric over all the test sample observations for Grad-CAM, Eigen-CAM, D-RISE, and LIME algorithms. In bold are highlighted the best algorithms for each metric.

As it is possible to observe, the two metrics do not concur with which explanation algorithm is the best. While the deletion metric considers LIME as the best algorithm for this dataset, insertion indicates D-RISE as the best algorithm.

Regardless of such, both metrics show a clear discrepancy between LIME, D-RISE, and Grad-CAM, Eigen-CAM. The latter two algorithms are substantially worst than the first two.

However, it is important to notice that by the nature of the algorithms LIME and D-RISE are much more computationally expensive, as both of these require thousands of forward passes through the neural network. On the other hand, Grad-CAM and Eigen-CAM are extremely fast to compute as only a single forward and backward propagation is necessary. Hence, it is evident that the quality of the explanation improves with increased computational resources dedicated to the algorithm.

4.3.2 Artifact data

The production of model explanations can be more useful than just the interpretation of the backbone reason of the neural network. One example of such scenario is the identification of a problem on the dataset. That is, the dataset used for training could contain an unwanted artifact that helps the model in the task at hand. However, when putting the model into production this artifact will not be present, hence the model will not be able to perform with the same confidence as in the modeling stage.

Therefore, to test the ability of the explainability algorithms to capture such a scenario, a new dataset will be simulated. This simulated dataset will contain the exact same images as the dataset previously considered, with the addition of two circles on the top left and bottom right of the object bounding box. On top of that, the circles will be red if the object present is a dog and blue if the object is a cat.

Figure 4.7 shows a couple of examples of the images in the simulated data with the added artifacts.

Figure 4.7: Exemplification of the type of images present in the ”Dog and Cat Dataset”, with the addiction of the artifacts.

This way, a new Faster RCNN model is trained on the simulated dataset. The expected behavior is that the model simply learns to locate the circle and its corresponding color.

After training the model, and obtain the predictions on the test sample, the explanations according to the four algorithms are computed.

Figure 4.8 shows a multitude of examples of the outcomes of the explanations for each algorithm. As it is possible to observe all the algorithms are in agreement that the model only learned to identify the circles present in the dataset and not the underlying characteristics of the animals. Hence, with the help of the explainability algorithm, it was possible to identify the underlying problems present in the dataset.

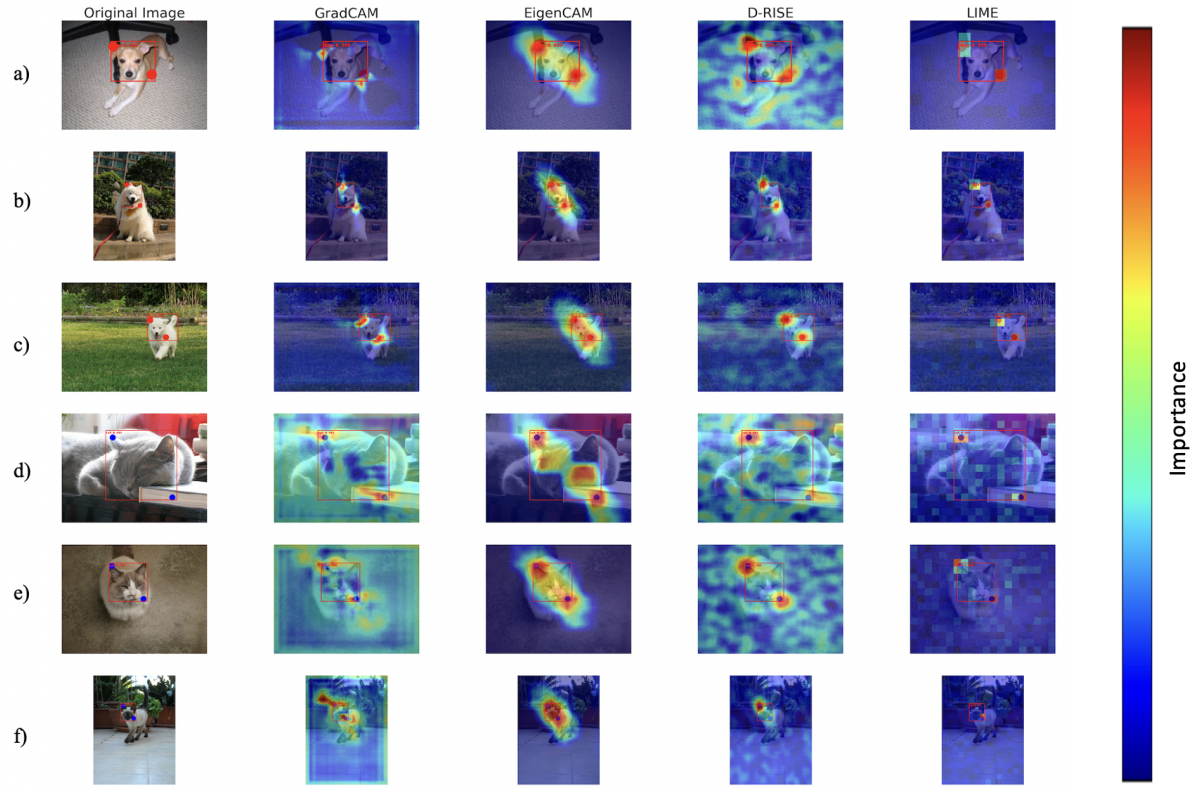

Figure 4.8: Comparison between the explanation provided by Grad-CAM, Eigen-CAM, D-RISE, and LIME, on the dataset with artifacts. The most important pixels are in red, and the least important pixels in blue.

Chapter 5

Conclusion

Given the increasing usage of neural networks in the development of applications with significant repercussions in our lives, such as automated driving and medical diagnosis, the fact that they are still largely black boxes remains an issue. This has led to the growth of research in the domain of neural network interpretability.

Currently, there exists a multitude of techniques that try to provide an explanation for any computer vision DNN model. However, with such a diversity in techniques, it becomes hard to identify which are the best or which are the most adequate for a given scenario. Hence, in this study, a comparison between a multitude of popular explainability techniques is presented.

For such, four explainability techniques, Grad-CAM, Eigen-CAM, D-RISE, and LIME were implemented in Pytorch, a deep-learning Python package. On top of that, Grad-CAM and D-RISE were implemented from scratch while Eigen-CAM and LIME were implemented from pre-existing packages.

5.1 Discussion of results

All the results presented were based on the Faster RCNN model architecture and its performance on the ”Dog and Cat Dataset”. To obtain these results, all the explainability algorithms were applied and tested on the model's predictions for the test samples, allowing for direct and meaningful comparison among the algorithms.

Upon conducting the tests, it became evident that D-RISE and LIME are the algorithms that excel in capturing the underlying reasons behind a computer vision model. The difference between these two algorithms and Grad-CAM and Eigen-CAM is substantial, with a gap greater than 0.15 when considering deletion and insertion metrics.

However, it should be noted that this improved explanation provided by D-RISE and LIME comes at a cost. These algorithms are significantly more computationally intensive compared to the other two. While Grad-CAM and Eigen-CAM only require a single forward and backward pass through the neural network, D-RISE and LIME need thousands of forward passes, resulting in slower processing times. Hence, in scenarios where computational resources are limited, Grad-CAM and Eigen-CAM may be more favorable options.

Furthermore, it was observed that producing model explanations not only enhances user trust but also contributes to ensuring data quality. This was demonstrated by introducing artifacts into the images and analyzing which characteristics the model would capture.

In conclusion, this study highlights the indispensability of model explanations when working with neural networks, as their applications extend beyond providing insights into the model's inner workings.

5.2 Future work

This work opens several options for further studies, which were not possible to explore in the time devoted to this thesis. For instance, it would be interesting to observe the results of the explainability algorithms on a more challenging task, one where the neural network would not always be correct. An example of such could be the application of a model in a real medical scenario, that is, to have the model giving diagnoses to patients. Besides that, it would also be interesting to create a new loss function that incorporates interpretability. Hence, the training of any neural network would automatically have into consideration the model interpretability. However, this would be more challenging as the computation cost of a loss function must be low, and, as observed in this work, the current ways of evaluating the interpretability ability can be quite expensive. Hence, a new way of computing an interpretability metric would need to be studied.

Bibliography

[1] Ritesh Kanjee. Object segmentation vs. object detection - which one should you use?, 2022. Accessed on 5, February, 2023. URL: https://www.linkedin.com/pulse/object-segmentation-vs-detection-which-one-should-you-ritesh-kanjee/?trk=pulse-article_more-articles_related-content-card.

[2] Manav Mandal. Introduction to convolutional neural networks (cnn), 2021. Accessed on 5, February, 2023. URL: https://www.analyticsvidhya.com/blog/2021/05/convolutional-neural-networks-cnn/.

[3] Avijeet Biswal. Convolutional neural network tutorial, 2022. Accessed on 5, February, 2023. URL: https://www.simplilearn.com/tutorials/deep-learning-tutorial/convolutional-neural-network.

[4] 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, 2020. URL: https://arxiv.org/abs/2006.03204, doi:10.48550/ARXIV.2006.03204.

[5] Adrian Rosebrock. Intersection over union (iou) for object detection, 2016. Accessed on 20, May, 2023. URL: https://pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/.

[6] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks, 2015. URL: https://arxiv.org/abs/1506.01497, doi:10.48550/ARXIV.1506.01497.

[7] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, Quoc V. Le, and Hartwig Adam. Searching for mobilenetv3, 2019. arXiv:1905.02244.

[8] Shilpa Ananth. Faster r-cnn for object detection, 2019. Accessed on 5, Jun, 2023. URL: https://towardsdatascience.com/faster-r-cnn-for-object-detection-a-technical-summary-474c5b857b46.

[9] Keiron O'Shea and Ryan Nash. An introduction to convolutional neural networks, 2015. URL: https://arxiv.org/abs/1511.08458, doi:10.48550/ARXIV.1511.08458.

[10] Alex Sherstinsky. Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network. Physica D: Nonlinear Phenomena, 404:132306, mar 2020. URL: https://doi.org/10.1016%2Fj.physd.2019.132306, doi:10.1016/j.physd.2019.132306.

[11] Jeffrey De Fauw, Joseph R Ledsam, Bernardino Romera-Paredes, Stanislav Nikolov, Nenad Tomasev, Sam Blackwell, Harry Askham, Xavier Glorot, Brendan O'Donoghue, Daniel Visentin, George van den Driessche, Balaji Lakshminarayanan, Clemens Meyer, Faith Mackinder, Simon Bouton, Kareem Ayoub, Reena Chopra, Dominic King, Alan Karthikesalingam, Cían O Hughes, Rosalind Raine, Julian Hughes, Dawn A Sim, Catherine Egan, Adnan Tufail, Hugh Montgomery, Demis Hassabis, Geraint Rees, Trevor Back, Peng T Khaw, Mustafa Suleyman, Julien Cornebise, Pearse A Keane, and Olaf Ronneberger. Clinically applicable deep learning for diagnosis and referral in retinal disease. Nat. Med., 24(9):1342–1350, September 2018.

[12] Justin Sirignano, Apaar Sadhwani, and Kay Giesecke. Deep learning for mortgage risk. SSRN Electron. J., 2018.

[13] Chenyi Chen, Ari Seff, Alain Kornhauser, and Jianxiong Xiao. DeepDriving: Learning affordance for direct perception in autonomous driving. In 2015 IEEE International Conference on Computer Vision (ICCV). IEEE, December 2015.

[14] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger, editors, Advances in Neural Information Processing Systems, volume 25. Curran Associates, Inc., 2012. URL: https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

[15] Geoffrey Hinton, Li Deng, Dong Yu, George E. Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N. Sainath, and Brian Kingsbury. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. IEEE Signal Processing Magazine, 29(6):82–97, 2012. doi:10.1109/MSP.2012.2205597.

[16] Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. On using very large target vocabulary for neural machine translation, 2014. URL: https://arxiv.org/abs/1412.2007, doi:10.48550/ARXIV.1412.2007.

[17] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks, 2013. URL: https://arxiv.org/abs/1312.6199, doi:10.48550/ARXIV.1312.6199.

[18] SuperAnnotate. Introduction to object detection with deep learning, 2021. Accessed on 5, February, 2023. URL: https://www.superannotate.com/blog/object-detection-with-deep-learning.

[19] FRITZ LABS INCORPORATED. Object detection guide, 2021. Accessed on 5, February, 2023. URL: https://www.fritz.ai/object-detection/.

[20] Gaudenz Boesch. Object detection in 2023: The definitive guide, 2023. Accessed on 5, February, 2023. URL: https://viso.ai/deep-learning/object-detection/.

[21] Zijie J. Wang, Robert Turko, Omar Shaikh, Haekyu Park, Nilaksh Das, Fred Hohman, Minsuk Kahng, and Duen Horng Polo Chau. CNN explainer: Learning convolutional neural networks with interactive visualization. IEEE Transactions on Visualization and Computer Graphics, 27(2):1396–1406, feb 2021. URL: https://doi.org/10.1109%2Ftvcg.2020.3030418, doi:10.1109/tvcg.2020.3030418.

[22] Rahul Awati. convolutional neural network (cnn), 2022. Accessed on 5, February, 2023. URL: https://www.techtarget.com/searchenterpriseai/definition/convolutional-neural-network.

[23] Alberto Rizzoli. The ultimate guide to object detection, 2023. Accessed on 5, February, 2023. URL: https://www.v7labs.com/blog/object-detection-guide#h3.

[24] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection, 2015. URL: https://arxiv.org/abs/1506.02640, doi:10.48550/ARXIV.1506.02640.

[25] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors, 2022. URL: https://arxiv.org/abs/2207.02696, doi:10.48550/ARXIV.2207.02696.

[26] Gaudenz Boesch. Yolov7: The most powerful object detection algorithm (2023 guide), 2023. Accessed on 5, February, 2023. URL: https://viso.ai/deep-learning/yolov7-guide/.

[27] H. J. P. Weerts. Interpretable machine learning as decision support for processing fraud alerts. Master's thesis, Eindhoven University of Technology, 2019.

[28] M Naiseh. C-XAI: Design Method for Explainable AI Interfaces to Enhance Trust Calibration. PhD thesis, Bournemouth University, 2021.

[29] J. Angwin and J. Larson. Machine bias, 2016. Accessed on 29, January, 2023. URL: https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing.

[30] General data protection regulation, gdpr, 27 April 2016. Accessed on 29, January, 2023. URL: https://gdpr-info.eu/.

[31] Timo Speith. A review of taxonomies of explainable artificial intelligence (XAI) methods. In 2022 ACM Conference on Fairness, Accountability, and Transparency, New York, NY, USA, June 2022. ACM.

[32] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization, 2015. URL: https://arxiv.org/abs/1512.04150, doi:10.48550/ARXIV.1512.04150.

[33] Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-CAM: Visual explanations from deep networks via gradient-based localization. International Journal of Computer Vision, 128(2):336–359, oct 2019. URL: https://doi.org/10.1007%2Fs11263-019-01228-7, doi:10.1007/s11263-019-01228-7.

[34] Jan Hosang, Rodrigo Benenson, and Bernt Schiele. Learning non-maximum suppression, 2017. URL: https://arxiv.org/abs/1705.02950, doi:10.48550/ARXIV.1705.02950.

[35] Jacob Gildenblat and contributors. Pytorch library for cam methods. https://github.com/jacobgil/pytorch-grad-cam, 2021.

[36] Scott Lundberg and Su-In Lee. A unified approach to interpreting model predictions, 2017. URL: https://arxiv.org/abs/1705.07874, doi:10.48550/ARXIV.1705.07874.

[37] Ma Leticia Jose C. Basilan, https://orcid.org/0000-0003-3105-2252, Maycee Padilla, and https://orchid.org/0000-0001-5025-12872, maleticiajose.basilan@deped.gov.ph, maycee.padilla@deped.gov.ph, Department of Education- SDO Batangas Province, Batangas, Philippines. Assessment of teaching english language skills: Input to digitized activities for campus journalism advisers. International Multidisciplinary Research Journal, 4(4), January 2023.

[38] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. 2017. URL: https://arxiv.org/abs/1704.02685, doi:10.48550/ARXIV.1704.02685.

[39] Saurabh Desai and Harish G. Ramaswamy. Ablation-cam: Visual explanations for deep convolutional network via gradient-free localization. In 2020 IEEE Winter Conference on Applications of Computer Vision (WACV), pages 972–980, 2020. doi:10.1109/WACV45572.2020.9093360.

[40] Mohammed Bany Muhammad and Mohammed Yeasin. Eigen-CAM: Class activation map using principal components. In 2020 International Joint Conference on Neural Networks (IJCNN). IEEE, jul 2020. URL: https://doi.org/10.1109%2Fijcnn48605.2020.9206626, doi:10.1109/ijcnn48605.2020.9206626.

[41] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "why should i trust you?": Explaining the predictions of any classifier, 2016. URL: https://arxiv.org/abs/1602.04938, doi:10.48550/ARXIV.1602.04938.

[42] Arun Das and Paul Rad. Opportunities and challenges in explainable artificial intelligence (xai): A survey, 2020. arXiv:2006.11371.

[43] Adrian Zbiciak and Tymon Markiewicz. A new extraordinary means of appeal in the polish criminal procedure: the basic principles of a fair trial and a complaint against a cassatory judgment. Access to Justice in Eastern Europe, 6(2):1–18, March 2023.

[44] Adrian Zbiciak and Tymon Markiewicz. A new extraordinary means of appeal in the polish criminal procedure: the basic principles of a fair trial and a complaint against a cassatory judgment. Access to Justice in Eastern Europe, 6(2):1–18, March 2023.

[45] Kseniya Sahatova and Ksenia Balabaeva. An overview and comparison of xai methods for object detection in computer tomography. Procedia Computer Science, 212:209–219, 2022. 11th International Young Scientist Conference on Computational Science. URL:

https://www.sciencedirect.com/science/article/pii/S1877050922016969, doi:https://doi.org/10.1016/j.procs.2022.11.005.

[46] Vitali Petsiuk, Abir Das, and Kate Saenko. Rise: Randomized input sampling for explanation of black-box models, 2018. arXiv:1806.07421.

[47] Joana Luís Martins. Interpretability of deep neural networks at the model level. Master's thesis, Faculdade de Ciências e Tecnologia, Universidade Nova de Lisboa, 2021.

[48] Damien Garreau and Dina Mardaoui. What does lime really see in images?, 2021. arXiv:2102.06307.

[49] Kumar Abhishek and Deeksha Kamath. Attribution-based xai methods in computer vision: A review, 2022. arXiv:2211.14736.

[50] Dog and cat dataset. URL: https://www.kaggle.com/datasets/andrewmvd/dog-and-cat-detection?resource=download.

[51] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In Advances in Neural Information Processing Systems 32, pages 8024–8035. Curran Associates, Inc., 2019. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[52] Jacob Gildenblat and contributors. Pytorch library for cam methods. https://github.com/jacobgil/pytorch-grad-cam, 2021.

[53] Captum: Model interpretability for pytorch. https://captum.ai/, 2023.

[54] Avijeet Biswal. Convolutional neural network tutorial, 2022. Accessed on 5, February, 2023. URL: https://www.simplilearn.com/tutorials/deep-learning-tutorial/convolutional-neural-network.